**UNIVERSIDADE FEDERAL DE CAMPINA GRANDE**

**CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA**

**COORDENAÇÃO DE PÓS-GRADUAÇÃO EM INFORMÁTICA**

# DISSERTAÇÃO DE MESTRADO

# PERSONÆ:
## UM MODELO DE MEDIAÇÃO SEMÂNTICA PARA ARQUITETURAS ORIENTADAS A SERVIÇOS

**MESTRANDO**

**PATRÍCIO DE ALENCAR SILVA**

**ORIENTADORES**

**ULRICH SCHIEL, DR. RER. NAT.**

**CLÁUDIA MARIA FERNANDES ARAÚJO RIBEIRO, DRA.**

**CAMPINA GRANDE**

**MAIO – 2007**

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA

COORDENAÇÃO DE PÓS-GRADUAÇÃO EM INFORMÁTICA

# PERSONÆ:
# UM MODELO DE MEDIAÇÃO SEMÂNTICA PARA ARQUITETURAS ORIENTADAS A SERVIÇOS

**PATRÍCIO DE ALENCAR SILVA**

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Informática do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande – Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação (M. Sc).

ÁREA DE CONCENTRAÇÃO: CIÊNCIA DA COMPUTAÇÃO

LINHA DE PESQUISA: SISTEMAS DE INFORMAÇÃO E BANCO DE DADOS

**ORIENTADORES**

**ULRICH SCHIEL, DR. RER. NAT.**

**CLÁUDIA MARIA FERNANDES ARAÚJO RIBEIRO, DRA.**

**CAMPINA GRANDE**

**MAIO – 2007**

**FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG**

S586p
2007 Silva, Patrício de Alencar.
Personae: um modelo de mediação semântica para arquiteturas orientadas a serviços / Patrício de Alencar Silva.─ Campina Grande, 2007.
157f.: il.

Dissertação (Mestrado em Ciências da Computação) - Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.
Referências.
Orientadores: Dr. Ulrich Schiel e Dra Cláudia Maria Fernandes Araújo Ribeiro.

1. Métodos Formais. 2. Arquiteturas Orientadas a Serviços. 3. Reconciliação de Ontologias. 4. Web Semântica. 5. Sistemas de Informação I. Título.

CDU 004.414.28

*Aos meus pais,*
*José de Alencar e Elza Tiago,*
*Com amor e gratidão por tudo.*

*Aos meus orientadores,*
*Ulrich Schiel e Cláudia Ribeiro,*
*Com respeito e admiração.*

*À Eshley,*
*Com zelo e carinho.*

# AGRADECIMENTOS

A Deus, o princípio da vida e de todas as coisas que existem, por me fazer entender a cada dia que "nEle posso todas as coisas".

Aos meus pais, José de Alencar e Elza Tiago, pelo amor incondicional, por suportarem pacientemente a dor que significa a distância e por me guiarem no caminho da honestidade. Pai e mãe, obrigado por tudo!

Aos meus orientadores Prof. Dr. Ulrich Schiel e Profª. Drª. Cláudia Ribeiro, pelo exemplo de paciência, simpatia, ética e profissionalismo e por terem me dado o mais importante no momento certo: uma oportunidade.

Ao Prof. Dr. Eustáquio Rangel, pela orientação no estágio docência, pela simpatia e pela contribuição neste trabalho.

Ao professor Dr. Marcus Sampaio, por ter acreditado em meu potencial.

Às minhas primas Vanda, Keliane e Íris, pelo apoio e carinho nos gestos mais simples.

A toda a minha família, pelo apoio e torcida, mesmo à distância.

Aos meus amigos, Prof. Dr. Raimundo Júnior e Flávia Batista, por me acolherem sempre tão bem, durante as reuniões de trabalho na cidade de Natal.

Aos excelentes amigos que fiz no decorrer deste curso: Elvis, Fábio Leite, Damião, Milena, Ana Cristina, Ana Emília, Hellen e Isabella, pelas ótimas gargalhadas e ótimos momentos vividos em dois anos de estudo.

Aos meus amigos de apartamento, Lucas Hartmman e José Pedrosa, por suportarem meu estresse fora dos padrões aceitos como normais.

Às secretárias do curso, Ana Sauvè e Vera Lúcia, pela simpatia e prontidão em me atender sempre, simplificando minhas pendências burocráticas.

Ao pessoal da cantina da Dona Inês, pela simpatia, incentivo e pelas boas gargalhadas nos horários de almoço.

À CAPES pelo apoio financeiro.

À COPIN pela oportunidade.

A todos os que contribuíram com a realização desta conquista. Muito obrigado!

# SUMÁRIO

# LISTA DE ABREVIATURAS

| | |
|---|---|
| BPEL | BUSINESS PROCESS EXECUTION LANGUAGE |
| DRS | DYNAMIC RESOURCE SCHEDULING |
| FLOWS | FIRST-ORDER LOGIC ONTOLOGY FOR WEB SERVICES |
| IFITT | INTERNATIONAL FEDERATION FOR IT AND TRAVEL & TOURISM |
| IHMO | INTEROPERABILITY MINIMUM HARMONIZATION ONTOLOGY |
| IRS-III | INTERNET REASONING SERVICE |
| MOF | META OBJECT FACILITY |
| NAICS | NORTH AMERICAN INDUSTRIAL CLASSIFICATION SYSTEM |
| OCML | OPERATIONAL CONCEPTUAL MODELLING LANGUAGE |
| OTA | OPEN TRAVEL ALLIANCE |
| OWL-S | WEB ONTOLOGY LANGUAGE FOR SERVICES |
| PSL | PROCESS SPECIFICATION LANGUAGE |
| RDF | RESOURCE DESCRIPTION FRAMEWORK |
| ROWS | RULE ONTOLOGY FOR WEB SERVICES |
| SEI-TUR | SISTEMA DE INFORMAÇÕES TURISTICAS |
| SLA | SERVICE LEVEL AGREEMENT |
| SOA | SERVICE ORIENTED ARCHITECTURE |
| SOAP | SIMPLE OBJECT ACCES PROTOCOL |
| SUMO | SUGGESTED UPPER LEVEL ONTOLOGY |
| SWRL | SEMANTIC WEB RULE LANGUAGE |
| SWSA | SEMANTIC WEB SERVICE ARCHITECTURE |
| SWSF | SEMANTIC WEB SERVICES FRAMEWORK |
| SWSL | SEMANTIC WEB SERVICES LANGUAGE |
| SWSO | SEMANTIC WEB SERVICES ONTOLOGY |
| TTI | TRAVEL TECHNOLOGY INITIATIVE |
| UDDI | UNIVERSAL DESCRIPTION DISCOVERY AND INTEGRATION |
| W3C | WORLD WIDE WEB CONSORTIUM |
| WSDL | WEB SERVICE DESCRIPTION LANGUAGE |
| WSDL-S | WEB SERVICE SEMANTICS |
| WSMO | WEB SERVICES MODELING ONTOLOGY |
| WSMX | WEB SERVICE MODELING EXECUTION ENVIRONMENT |
| WTO | WORLD TRAVEL ORGANIZATION |
| XML | EXTENDED MARKUP LANGUAGE |

# LISTA DE FIGURAS

# LISTA DE TABELAS

# RESUMO

Atualmente, a personalização é uma forte tendência na provisão de serviços em geral. Esta tendência tem sido observada também na provisão de serviços na Web, onde a automatização dos processos de busca e seleção da informação tem revelado uma preocupação crescente quanto ao oferecimento de serviços diferenciados. Entretanto, nem sempre provedores e clientes compartilham uma visão comum sobre um mesmo serviço. Estas visões podem divergir em duas dimensões: na disputa de interesses em questão e na forma como esses interesses são representados.

No plano da representação, ontologias têm sido utilizadas como vocabulários para descrição de vários domínios de serviços na Web. Entretanto, não é realística a possibilidade de determinação de um único vocabulário para a definição desses interesses. Desta forma, torna-se evidente a necessidade de um tratamento centrado na mediação destes vocabulários, visando um estreitamento de relações entre clientes e provedores de serviços. Isto implica a inclusão de um elemento mediador entre as partes, que identifique similaridades entre os vocabulários utilizados. A transparência neste processo pode ser traduzida em termos de vantagens para ambos os lados. No lado do cliente, ampliam-se as possibilidades na descoberta e escolha dos serviços que melhor atendam às suas reais necessidades. No lado do provedor, o termo "padrão" na descrição de seus serviços pode passar a significar uma "opção" a mais. Isto leva à possibilidade de comunicação com a preservação das diferenças.

Para o tratamento deste aspecto, esta dissertação de mestrado define um modelo de mediação para Arquiteturas Orientadas a Serviços, denominado PERSONÆ, baseado em reconciliação de ontologias. Diferentemente das abordagens tradicionais, PERSONÆ representa um modelo formal para a especificação precisa e não ambígua de aspectos comportamentais de mediadores para este contexto. Este modelo permitiu um entendimento mais aprofundado sobre as técnicas de fusão, integração e alinhamento de ontologias, tornando mais claro como estas técnicas podem ser combinadas para resolução de problemas de heterogeneidade semântica, mais especificamente nas fases de descoberta, seleção e estabelecimento de contratos de serviços na Web.

# ABSTRACT

Nowadays, personalization is a strong trend in service provisioning. This trend can also be noticed on the Web, where some degree of automation in processes such as information discovery and selection points to a concern on differentiated services provisioning. However, clients and providers do not share the same perspective about the same service very often. These perspectives diverge through two dimensions: interest conflicts and how these interests can be represented.

Ontologies are shared vocabularies used to define different knowledge domains on the Web. Nevertheless, it is unrealistic to expect a global consensus between people and organizations onto a common, shared ontology. Therefore, the need of a mediation-centered approach for reconciliation of these different vocabularies becomes clear, in order to close the gap between the sides involved towards a better communication. This implies the inclusion of a mediator between them, in order to identify similarities on the vocabularies used. The transparency in this process brings benefits to both of them. On the client side, it can be essential in allowing discovery and selection of the most appropriate services. On the provider side, the meaning of the "standard" can be just one more "option". It brings the possibility of communication despite the differences.

To deal with this aspect, this master degree thesis defines a mediation model for Service Oriented Architectures, named PERSONÆ, based on ontology reconciliation. Differently from traditional approaches, PERSONÆ represents a formal model for describing behavioral aspects of the mediator for this context, in a precisely and unambiguously manner. This model provided a deep understanding about ontology reconciliation techniques like merging, integration and alignment, making clear how these can be combined in order to minimize semantic heterogeneity problems, more specifically in service discovery, service selection and service level agreements establishment on the Web.

# CAPÍTULO 1

# INTRODUÇÃO

*"O princípio é a metade do todo."*
[Platão]

*Em um cenário de prestação de serviços, nem sempre provedores e clientes compartilham uma visão comum sobre um mesmo serviço. Se por um lado o cliente busca por maior qualidade de serviço com menores custos, provedores buscam vantagens de lucros com menor consumo de recursos. Este tipo de conflito pode se tornar ainda mais pronunciado quando ambas as partes utilizam diferentes termos para expressar seus interesses. Este último aspecto consiste no foco desta dissertação de mestrado, cujo objetivo é propor um modelo de mediação entre clientes e provedores de serviços.*

*As próximas seções apresentam o contexto e as razões que motivaram a concepção de um modelo de mediação para arquiteturas orientadas a serviço. O problema específico tratado por esta dissertação é descrito e uma visão geral da solução proposta é apresentada. As premissas consideradas no contexto deste trabalho também são apresentadas. Finalmente, a organização dos próximos capítulos é mostrada.*

## 1.1 CONTEXTO

A Web tem se tornado o maior repositório de informações de que se tem conhecimento atualmente. Neste ambiente, a liberdade de publicação e expressão tem permitido uma rápida disseminação de páginas que disponibilizam o acesso a todo tipo de recurso. Estes recursos variam de textos, vídeos, anúncios, até serviços mais sofisticados, como motores de busca para recuperação de outros recursos. Isto tem provocado mudanças na forma da prestação de serviços em geral.

Um cenário comum de prestação de serviços envolve clientes, provedores e meio de divulgação de serviço. Normalmente, o cliente utiliza este meio de divulgação como referência para encontrar um provedor de um serviço específico. Exemplos comuns de divulgação de serviços incluem classificados ou páginas amarelas, em listas telefônicas. Obviamente, isto não impede que um cliente tenha acesso direto a um provedor, sem o

auxílio de uma propaganda. Porém, quando a oferta é grande e a necessidade do cliente é específica, a utilização de um classificado, por exemplo, pode ser de grande utilidade.

Este modelo de prestação de serviços também é utilizado na Web atualmente. Neste sentido, a Arquitetura Orientada a Serviços (ou SOA, do inglês *Service Oriented Architecture*) representa a implementação computacional deste modelo para a Web. A provisão dos serviços ocorre segundo um processo semelhante: provedores publicam descrições dos seus serviços em meios de divulgação chamados de registros; clientes (que podem ser páginas Web que capturam as requisições dos usuários) realizam uma busca nestes registros por serviços que atendam aos usuários; após encontrar as informações de localização de um provedor de interesse, o cliente interage com este provedor diretamente, para ter acesso ao serviço específico (BOOTH, 2004).

Esses serviços na realidade são informações recuperadas por componentes de software de funcionalidade bem definida. Estes componentes são chamados de *Web services*. Em termos mais técnicos, um *Web service* pode ser definido como uma unidade de software encapsulada que provê acesso a informações de serviços físicos (ALONSO, 2004). Desta forma, um mesmo *Web service* pode recuperar informações de vários serviços e pode ser utilizado por várias aplicações. A possibilidade de reuso destes componentes confere maior flexibilidade e rapidez ao desenvolvimento de aplicações para a Web. O comércio eletrônico representa uma das possibilidades de utilização deste tipo de tecnologia. Para fins de simplicidade, será adotado o termo serviço no texto adiante neste capítulo.

Por parte do cliente, a busca por serviços é realizada com base na descrição destes serviços. Estas descrições são publicadas nos registros e fornecem uma especificação técnica dos parâmetros necessários para utilização destes serviços. Entretanto, estas descrições permitem apenas que as buscas sejam realizadas por meio de palavras-chave. Isto significa que uma busca baseada nestas descrições pode retornar serviços irrelevantes para o usuário. O aumento da oferta de serviços pode constituir um problema adicional na recuperação daqueles que atendam de fato à necessidade do usuário. Este problema tem sido identificado não apenas na Arquitetura Orientada a Serviços, mas na recuperação de informação na Web em geral.

Para entender o problema, considere-se a estrutura de um texto comum. A análise das palavras separadas é chamada de análise léxica ou morfológica, classificando os componentes da linguagem em nível mais elementar. A análise sintática permite verificar como as palavras podem ser articuladas na formação de orações, que expressam algum significado. Porém, é na análise semântica que se verifica o relacionamento de orações

dentro de um contexto maior. Na Web, a informação é descrita de forma semelhante. Uma busca por palavras-chave é análoga a uma análise léxica, retornando todo tipo de resultados sem considerar o contexto da informação. A construção de tecnologias que permitam descrever a informação da Web, mediante o estabelecimento de relações entre os dados e informações, é chamada de Web semântica.

A Web semântica representa a nova fase na evolução da Web como é conhecida hoje, onde a informação será disponibilizada com um significado bem definido (BERNERS-LEE; HENDLER; LASSILA, 2001). Este significado representa justamente o contexto da informação, determinado pelas relações com outras informações. Neste sentido, um tipo específico de estrutura de dados tem sido utilizado para a especificação destas relações. Estas estruturas são chamadas de *ontologias*. O termo *ontologia* é de origem filosófica, definido por princípio como a classificação dos relacionamentos entre as coisas que existem. Este termo tem sido utilizado na ciência da computação para definir os relacionamentos que existem entre os conceitos de um determinado domínio de aplicação. Na Web, as ontologias podem ser utilizadas para estabelecer ligações entre os dados, enriquecendo a descrição das informações. Isto permite a descoberta de informações que estejam relacionadas, de acordo com o contexto de interesse do usuário.

Estas estruturas também têm sido utilizadas na Arquitetura Orientada a Serviços. O objetivo é enriquecer a descrição dos serviços, mediante o estabelecimento de relações entre serviços de um mesmo domínio de aplicação. Desta forma, o processo de busca que antes era realizado apenas com base em palavras-chave pode ser realizado com base em conceitos. Isto permite a recuperação de serviços que estejam relacionados de acordo com o domínio de aplicação, aumentando as possibilidades de atendimento das necessidades dos usuários. A aplicação de tecnologias da Web semântica na Arquitetura Orientada a Serviços deu origem à Arquitetura Orientada a Serviços Semânticos.

## 1.2 MOTIVAÇÃO

A partir das considerações anteriormente citadas, é possível verificar que existe uma crescente preocupação em melhor atender aos usuários de serviços na Web. Este melhor atendimento está relacionado não apenas à automatização do processo de descoberta de serviços, mas também à possibilidade do usuário selecionar o serviço mais adequado às suas necessidades. Isto representa uma tendência na provisão de serviços em geral, e fator vital na garantia de espaço num mercado cada vez mais competitivo. Esta tendência é chamada de personalização.

A personalização traz benefícios para ambos os lados envolvidos na provisão de serviços. No lado do cliente, existe maior possibilidade de atendimento de suas necessidades, pela consideração de suas preferências ou prioridades. No lado do provedor, aumentam-se as chances de conquista de uma maior parcela de mercado, pelo atendimento diferenciado. Isto pode implicar o surgimento de relações de fidelidade entre clientes que optam por determinados provedores que estão mais preocupados em atender a estas necessidades. Esta é a principal motivação deste trabalho.

Embora a personalização seja de vital importância para a provisão de serviços diferenciados, a operacionalização deste conceito não é trivial. Isto se deve principalmente ao fato de que a personalização depende da visão que um determinado usuário possui sobre requisitos de qualidade de serviço. Estes requisitos são bastante subjetivos, variando de acordo com o nível de exigência de cada usuário. Desta forma, a personalização depende da aproximação das visões do usuário e do provedor sobre o nível de qualidade de um mesmo serviço.

## 1.3 DEFINIÇÃO DO PROBLEMA

Existe normalmente uma dificuldade em aproximar clientes e provedores de serviços em um processo de fidelização. Isto ocorre em grande parte pelo fato de que nem sempre o cliente ocupa uma posição ativa nesta relação. Em outras palavras, manter este tipo de relação quase sempre implica na aceitação das condições impostas pelo provedor por parte do cliente.

Esta situação se repete no contexto da Arquitetura Orientada a Serviços. O distanciamento entre clientes e provedores é causado por dois grandes motivos: existem disputas de interesses das partes envolvidas e diferenças nos vocabulários utilizados na descrição dos serviços.

O primeiro motivo é de natureza mais geral, e consiste no fato de que, normalmente, clientes e provedores disputam por maiores benefícios. Em termos mais técnicos, enquanto o provedor busca por maiores lucros com alocação reduzida de recursos computacionais, o cliente procura por maior nível de qualidade de serviço com menor custo. Esta é uma situação bastante comum na provisão de serviços em geral, na qual o resultado final da negociação depende do poder de barganha das entidades envolvidas.

O segundo problema é de natureza mais específica, e consiste no fato de que nem sempre são utilizados os mesmos termos para descrever serviços. Em termos mais técnicos, isto quer dizer que não existe consenso sobre qual ontologia utilizar para descrever um serviço. Conforme é detalhado no capítulo 2, existem diferentes abordagens de ontologias

para descrição de serviços. Este é um problema típico da Web semântica, também observado na Arquitetura Orientada a Serviços Semânticos. Embora uma ontologia represente um consenso sobre os termos que descrevem determinado domínio de conhecimento, não é realística a imposição de padrões neste sentido para quaisquer domínios.

O segundo aspecto constitui um agravante em relação ao primeiro e representa o problema específico tratado nesta dissertação de mestrado. Pode-se prever que um processo de negociação pode se tornar mais difícil se as partes envolvidas utilizarem termos diferentes para definir seus interesses. Considerando que este aspecto não é tratado explicitamente na Arquitetura Orientada a Serviços Semânticos, torna-se evidente a necessidade de inclusão de elementos de apoio que possibilitem uma mediação de termos utilizados pelas entidades envolvidas. Neste caso, o objetivo principal da mediação é proporcionar a comunicação entre as partes envolvidas sem a imposição de padrões sobre os termos utilizados na descrição dos serviços.

## 1.4 SOLUÇÃO PROPOSTA

A estratégia adotada para o segundo problema exposto anteriormente (i.e. a existência de diferenças entre os termos utilizados na provisão de serviços) consistiu na criação de um modelo conceitual de mediação semântica. Este modelo, denominado PERSONÆ, representa uma proposta de extensão da Arquitetura Orientada a Serviços, pela inclusão de um quarto elemento de primeiro nível, denominado *mediador*. A função do mediador é atuar especificamente na aproximação de diferentes termos utilizados pelas entidades envolvidas.

Uma vez identificado "qual" elemento acrescentar, é necessário definir precisamente "como" ele deve operar. Isto implica a definição de técnicas específicas de aproximação que podem ser utilizadas em diferentes contextos. Considerando que a aproximação aqui tratada se refere à tradução de termos definidos em diferentes ontologias, torna-se necessário entender como reconciliar estas diferentes visões conceituais. Este processo é denominado *reconciliação ontológica* e consiste basicamente na combinação de técnicas de mapeamento de conceitos de diferentes ontologias (HAMEED; PREECE; SLEEMAN, 2003). Este mapeamento pode ocorrer em duas dimensões: vertical ou horizontal.

A dimensão vertical para mapeamento de ontologias inclui basicamente duas técnicas: *fusão* e *integração*. Na técnica de *fusão*, ocorre a geração de uma terceira ontologia a partir de duas ontologias originais. A ontologia resultante agrega conceitos em comum entre as duas ontologias de origem, representando uma visão mais generalizada dos

conceitos originais. Na técnica de *integração*, uma nova ontologia é gerada a partir de fragmentos das outras ontologias, sem considerar o que existe em comum. Estas ontologias geradas são chamadas de *ontologias-núcleo*.

A dimensão horizontal inclui a técnica de *alinhamento*. Nesta técnica, não ocorre a geração de uma ontologia mais genérica. Em vez disso é gerado um conjunto de mapeamentos que relacionam o que existe em comum entre as ontologias originais. Esta técnica consiste de um processo semi-automático, onde o usuário final define quais os mapeamentos que lhe interessam.

O desenvolvimento de modelos e algoritmos para a implementação destas técnicas têm sido o foco de trabalhos recentes (ABELS; HAAK; HAHN, 2005), (SCHORLEMMER; KALFOGLOU, 2005), (EHRIG, 2006), (HUANG, 2006). Entretanto ainda existem fortes motivos que justificam considerações adicionais neste campo. Em primeiro lugar, não existe consenso sobre "quais" são os mecanismos específicos utilizados nestas técnicas. Em grande parte dos casos, todas as técnicas são tratadas como um único mecanismo, denominado "mapeamento" de ontologias. Isto evidencia a necessidade de definições mais precisas e formais sobre cada procedimento. Em segundo lugar, a imprecisão na especificação destes mecanismos torna obscura a aplicação destes em contextos específicos. Em outras palavras, este problema poderia ser traduzido nas seguintes questões: "que técnica de reconciliação poderia ser utilizada na descoberta de serviços com diferentes descrições?", ou "por que determinado tipo de técnica é melhor que outra na seleção de serviços?". A resposta a estas perguntas pode partir de uma investigação mais aprofundada sobre cada uma destas técnicas. Em terceiro lugar, o aspecto da combinação destas técnicas em contextos de tarefas mais complexas ainda é pouco discutido, ou seja, no geral uma técnica é aplicada em detrimento de outra.

Estes três aspectos foram considerados na concepção do modelo PERSONÆ. A estratégia adotada envolveu primeiramente a especificação formal de cada técnica existente para reconciliação de ontologias. Com base no entendimento das propriedades de cada uma destas técnicas, foram identificadas algumas das possibilidades de aplicação no contexto da Arquitetura Orientada a Serviços Semânticos. Finalmente, foram identificadas possibilidades de combinação destas técnicas mais especificamente nas tarefas de descoberta, seleção e estabelecimento de contratos de serviços.

O volume e a natureza das informações tratadas, bem como a complexidade das funções que compõem o modelo PERSONÆ, torna naturalmente difícil a sua compreensão e uso. Nesse sentido, a formalização permite esclarecer quais as informações manipuladas e

quais as atividades desempenhadas por cada elemento. Um benefício direto advindo de uma compreensão mais aprofundada sobre o modelo e seus elementos é a simplificação de sua implementação.

O modelo PERSONÆ foi formalizado através da notação Z (SPIVEY, 1992). Esta notação é adequada tanto para a especificação de estruturas de dados que representam o estado do sistema quanto para a especificação, manutenção e verificação de restrições sobre estes dados durante a execução de operações. A existência de ferramentas que atuam sobre a especificação formal, dentre elas os provadores de teoremas (SAALTINK, 1997) permite a validação e o raciocínio sobre as propriedades do modelo.

## 1.5 PREMISSAS

Esta seção apresenta as principais premissas consideradas durante o desenvolvimento deste trabalho. Além de ajudar na delimitação do escopo da dissertação, estas premissas permitem esclarecer os principais pontos que foram considerados na especificação do modelo proposto. As premissas são:

- O problema da reconciliação ontológica transcende diversos modelos arquiteturais baseados em técnicas da Web Semântica. Entretanto, esta dissertação está focada na verificação da aplicabilidade de um modelo formal para reconciliação de ontologias em Arquiteturas Orientadas a Serviço.
- O ciclo de vida básico de um serviço compreende as fases de publicação, descoberta, seleção, composição, negociação, estabelecimento de contratos de serviços, execução, monitoramento e reconfiguração. O modelo PERSONÆ está focado na aplicação das técnicas de reconciliação mais especificamente nas tarefas de descoberta e seleção de serviços. Um modelo simplificado para estabelecimento de contratos é fornecido.
- Um modelo de reconciliação ontológica deve ser independente de tecnologias específicas de implementação. A abstração do termo ontologia pode ser concretizada em diversas linguagens de representação. Neste sentido, a utilização de modelos matemáticos para o tratamento do problema em questão permite um nível de abstração adequado e independência de aspectos específicos de implementação.
- Duas razões principais determinaram a utilização da notação Z para a modelagem do problema em questão. Em primeiro lugar, existe uma forte

relação de morfismo[1] entre a notação Z e os fundamentos das linguagens mais utilizadas para especificação de ontologias para a Web, como OWL, por exemplo. Este relacionamento permite a tradução e verificação de qualidade da ontologias mediante mapeamento de representações OWL para Z e vice-versa. Em segundo lugar, a existência de ferramentas para verificação automática de tipos de dados e provadores de teoremas para a notação Z permite semi-automatização das tarefas de verificação de tipos.

O estabelecimento destas premissas confere maior liberdade ao modelo, permitindo que maior ênfase seja dada em responder por que a mediação é uma tarefa necessária em um cenário orientado à provisão de serviços personalizados, sem considerar diretamente aspectos técnicos de implementação.

## 1.6 ESTRUTURA DO DOCUMENTO

Este documento está organizado em mais cinco capítulos, além desta introdução. O capítulo 2 é dedicado à apresentação dos conceitos e propriedades de sistemas de mediação para arquiteturas orientadas a serviços. O objetivo deste capítulo é apresentar os conceitos básicos adotados neste trabalho e fornecer uma visão geral sobre a área, incluindo os mecanismos atuais para mediação entre clientes e provedores de serviços. Inicialmente, são introduzidos conceitos relacionados com a tarefa de mediação, princípios gerais da mediação e funcionalidade de uma entidade mediadora. Em seguida, são apresentados os principais tipos de conflitos em Arquiteturas Orientadas a Serviços, os quais justificam a necessidade da inserção de mediadores como entidades de primeiro nível neste cenário, em adição a clientes, provedores e registros.

No capítulo 3, introduz-se o modelo PERSONÆ. Além da descrição das entidades e relacionamentos do modelo proposto, este capítulo explica como os elementos de PERSONÆ estão relacionados aos processos de descoberta, seleção e estabelecimento de contratos de serviços. Também são descritas as abstrações utilizadas na definição de cada entidade do modelo. Finalmente, é descrito como a mediação pode complementar as tarefas mencionadas.

No capítulo 4, apresenta-se a formalização do modelo PERSONÆ. A notação Z foi utilizada para a formalização dos conceitos propostos. Neste capítulo são especificadas as

[1] Segundo o trabalho de Lucanu et al. (LUCANU; LI; DONG, 2005), a aplicação de co-morfismos de instituições na verificação de relacionamentos entre os fundamentos lógicos de OWL e Z provou que existe uma relação semântica forte entre essas linguagens. O modelo completo da verificação dessas propriedades encontra-se em http://www.comp.nus.edu.sg/~liyf/OWL2Z.tex

técnicas de reconciliação e a relação destas técnicas com as fases de descoberta, seleção e estabelecimento de contratos de serviços. Por fim, são discutidas as estratégias existentes para refinamento de especificações funcionais até o nível de implementação.

No capítulo 5, apresenta-se um cenário de uso que demonstra a aplicação do modelo PERSONÆ no contexto da provisão de serviços com problemas de heterogeneidade semântica. O cenário de uso, baseado em uma aplicação de serviços de pacotes turísticos na Web, ilustra como a mediação se aplica mais especificamente nas tarefas de descoberta e seleção de serviços.

Por fim, no capítulo 6 resumem-se as principais contribuições deste trabalho, sendo apontandas as perspectivas futuras e encaminhamento de outras pesquisas. Esta dissertação é complementada por um apêndice, que contém a especificação completa em notação Z do modelo PERSONÆ.

# CAPÍTULO 2

# SISTEMAS DE MEDIAÇÃO E ARQUITETURAS ORIENTADAS A SERVIÇOS SEMÂNTICOS

*"O Universo é uma harmonia de contrários."*
[Platão]

*De uma forma geral, um processo de mediação consiste na reconciliação de perspectivas. Frequentemente, a divergência de perspectivas resulta em conflitos que dificultam processos diretos de negociação. Neste sentido, uma entidade mediadora imparcial pode ser utilizada para facilitar a aproximação das partes envolvidas, mediante a identificação de similaridades implícitas entre os interesses envolvidos.*

*No contexto da prestação de serviços na Web, são comuns os conflitos entre clientes e provedores. Tais conflitos partem das diferenças entre vocabulários utilizados na descrição dos serviços e se estendem até a possibilidade de impasses na negociação sobre qualidade de serviço. O oferecimento de serviços diferenciados sob o invólucro da personalização depende da aproximação entre clientes e provedores, e consiste em um dos principais desafios na consolidação da Arquitetura Orientada a Serviços Semânticos na Web. Tal aproximação está diretamente associada à adoção de um modelo estruturado para a mediação, a qual permita uma negociação efetiva.*

*A identificação de trabalhos que tratam deste aspecto é objeto de estudo neste capítulo. Inicialmente, conceitos básicos de mediação são apresentados. Em seguida, são discutidos alguns dos trabalhos que tratam sobre mediação no contexto de arquiteturas orientadas a serviços semânticos.*

## 2.1 FILOSOFIA DA MEDIAÇÃO

A mediação engloba um processo de reconciliação entre duas ou mais entidades que podem ou não compartilhar uma mesma perspectiva dentro de um processo de negociação direta (WIEDERHOLD, 1996), (NADLER, 2001). No caso da impossibilidade de alcançar um acordo em primeira instância, uma entidade mediadora é inserida no contexto com o objetivo de facilitar a aproximação das partes e o processo de negociação.

Embora sejam frequentemente utilizados no mesmo contexto, os termos *mediação* e *reconciliação* denotam algumas semelhanças e diferenças. O termo mediação representa uma generalização que inclui o processo de reconciliação e facilitação dos processos de resolução de disputas (NADLER, 2001). Ambos os processos envolvem a utilização de uma "terceira entidade" neutra aos interesses das partes envolvidas, utilizando determinadas

técnicas em um contexto bem definido e tendo como objetivo a identificação de similaridades e diferenças entre as partes. Na reconciliação, esta entidade atua simplesmente na exploração das similaridades, gerando uma intersecção de conhecimento que pode ajudar na tomada de decisão entre as partes envolvidas. Entretanto, esta não assume comportamento pró-ativo. Na mediação, esta entidade que será denominada *mediador,* dispõe de conhecimento especializado no seu domínio de atuação, podendo sugerir soluções com base nas similaridades e diferenças detectadas, assumindo comportamento pró-ativo dentro na negociação.

Um importante aspecto a ser considerado é que a mediação também pode ser utilizada na prevenção de conflitos. No caso do estabelecimento de acordos que incluem a determinação de várias premissas após um processo de negociação, a utilização de um mediador em um contexto prévio pode minimizar conflitos futuros envolvendo disputas acirradas. Alguns aspectos comuns na tarefa de mediação incluem os seguintes estágios:

- Uma situação de controvérsia, disputa ou diferença de posições entre pessoas, ou a necessidade de tomada de decisão ou resolução de problemas;
- Um processo de tomada de decisão final, realizado pelas partes envolvidas e não pelo mediador;
- A exposição de requisitos ou preferências das partes envolvidas;
- A identificação de similaridades, facilitada pelo mediador;
- A conversão de interesses mais específicos em interesses mais gerais, por parte do mediador;
- A análise dos possíveis efeitos das várias soluções propostas;
- O ajuste ou refinamento da solução escolhida;
- O registro ou formalização de um possível acordo em contrato.

A mediação *online*, uma subcategoria da disputa de resoluções, é a aplicação das tecnologias da Web no processo de mediação (BONNET, 2002), (SHNEIDERMAN, 2000). A mediação *online* consiste na implementação computacional da função de um mediador para disputas entre pessoas que estão geograficamente distantes e impossibilitadas de firmar um acordo de negócios face a face. Um cenário apropriado para a aplicação de mediadores *online* é o comércio eletrônico (NADLER, 2001).

O papel do mediador no comércio eletrônico consiste em ajudar as partes envolvidas a alcançar um objetivo final de compra e venda de um determinado serviço (uma contraposição genérica que inclui todas as possíveis variedades de troca ou venda de

recursos). Isto deve implicar benefícios para ambos os lados (visando tipicamente o estabelecimento de um acordo bilateral), aproximando harmonicamente as partes envolvidas rumo a um equilíbrio balanceado no processo de negociação (WIEDERHOLD, 1997). Neste sentido, o mediador busca encontrar um acordo positivo entre as partes, mediante a composição de todos os aspectos que concernem à melhor possibilidade de agradar as preferências de ambos os lados em questão (CIMPIAN; MOCAN; STOLLBERG, 2006).

Um importante aspecto a ser considerado é a confiabilidade da mediação. Considerando que um mediador representa uma entidade neutra no que concerne a concessão de privilégios a uma das partes envolvidas em detrimento da outra, este deve ser suficientemente qualificado para atuação em um determinado escopo e mediante a utilização de técnicas pertinentes, de forma a evitar injustiças durante a negociação (WIEDERHOLD, 2002). Tendo em vista que o resultado esperado de uma tarefa de mediação é o estabelecimento de um contrato entre as partes, falhas na mediação podem gerar futuras rupturas de contrato, com base em processos de monitoramento sobre termos de qualidade de serviço acordados.

A escolha do mediador adequado para uma dada tarefa é de importância crucial. Esta tarefa está diretamente relacionada com o modelo de mediação associado, o tipo de técnica usada pelo mediador e o contexto no qual este atua (YERNENI, 1999). O termo "escolha do mediador" implica um processo de deliberação e tomada de decisão. Vale salientar que a própria mediação é um tipo especial de serviço, o qual está sujeito à avaliação quantitativa (funcional) e qualitativa (diferencial). Esta última tem um aspecto mais subjetivo, incluindo requisitos como confiabilidade e segurança do serviço de mediação oferecido. Na prática, não existem ainda mecanismos formais para seleção *a priori* de um dado tipo de mediador, visto que o contexto da negociação pode ser modificado ou expandido, saindo do escopo da funcionalidade do mediador. Neste caso, uma composição de mediações minoritárias pode ser utilizada para a resolução ou atenuação do conflito como um todo. O próprio caráter da mediação é uma função direta dos interesses das partes envolvidas, da natureza do conflito e do contexto em que ocorre (WIEDERHOLD, 1996), (TZITZIKAS; SPYRATOS; CONSTANTOPOULOS, 2001).

O estabelecimento de regras bem definidas e de um vocabulário de base para a negociação representa um importante aspecto a ser observado na mediação. São estas "leis" que governam o processo de negociação e permitem a tradução adequada por parte do mediador. Considerando ainda a possibilidade de extensão de um conflito além do domínio

do mediador, são estas regras que determinam o critério base para a seleção de um mediador adequado ao contexto, ou ainda, a inclusão de um outro mediador para complementar a tarefa geral de negociação (WIEDERHOLD, 1997).

De uma forma geral, dado que a mediação atenta por produzir acordos que reforçam pactos entre as partes envolvidas em uma negociação, uma mediação efetiva deve considerar compensações ou penalidades no caso da ruptura de um contrato. O poder "concreto" de um contrato reside no equilíbrio do pacto, na reconciliação dos respectivos interesses e na inclusão de métricas que tornem muito pouco conveniente a ruptura do contrato por quaisquer das entidades envolvidas.

Alguns benefícios justificam a inserção de mediadores em um determinado contexto de negociação. Entre eles podem-se destacar: (1) a descoberta de restrições, preferências e prioridades das partes envolvidas; (2) a facilitação do processo de negociação e estabelecimento de contratos; (3) a identificação de benefícios ou penalidades determinados pelo cumprimento ou violação de premissas contratuais; e (4) o acúmulo de conhecimento gerado pela resolução de conflitos, o qual pode ser reutilizado em futuras negociações (WIEDERHOLD, 1992).

## 2.2 MEDIAÇÃO E WEB SEMÂNTICA

Na Web semântica, os dados são descritos mediante o uso de ontologias, as quais representam vocabulários de comunicação entre aplicações (GÓMEZ-PEREZ; FERNÁNDES-LÓPEZ; CORCHO, 2004), (VISSER, 2004), (USCHOLD, 2003). Ontologias fornecem um nível mais detalhado de descrição sobre domínios de conhecimento, enriquecendo a descrição dos dados e recursos, tornando explícito o contexto da informação na Web. Segundo Tim Berners-Lee (BERNERS-LEE; HENDLER; LASSILA, 2001), o objetivo principal nesta nova fase da Web é permitir a cooperação entre humanos e máquinas, no sentido de facilitar a descoberta da informação relevante de acordo com o contexto.

Uma mesma ontologia pode ser utilizada para a anotação semântica de múltiplos recursos, desde páginas Web até documentos XML, bancos de dados relacionais, etc. O uso de tais terminologias permite a interoperabilidade entre esses recursos de dados e entre aplicações que acessam estes recursos. Entretanto, isto não resolve completamente o problema da integração da informação, visto que não é realístico prever que todos os indivíduos e organizações utilizem uma ontologia comum (VISSER, 2004), (USCHOLD, 2003). Portanto, uma previsão mais realista inclui o surgimento e utilização de múltiplas ontologias na Web (HORROCKS, 2005). Desta forma, a garantia de interoperabilidade

entre aplicações depende de uma abordagem centrada na resolução de diferenças entre estes vocabulários. A mediação dessas diferentes visões da informação recebe o nome de *reconciliação ontológica*.

A reconciliação ontológica permite o reuso de dados por diferentes aplicações na Web semântica e a cooperação entre diferentes organizações, considerando o uso de várias ontologias (DOAN, 2003), (STUMME; MAEDCHE, 2001). No contexto do gerenciamento de conhecimento, a reconciliação ontológica é especialmente importante no compartilhamento de dados entre diferentes bases de conhecimento, permitindo que aplicações reutilizem dados provenientes de diferentes bases (YAN; ÖZSU; LIU, 1997). Outra importante área de aplicação para a reconciliação ontológica são os *Web services semânticos* (BUSSLER; FENSEL; SADEH, 2005), (DAVIS; FENSEL; RICHARSON, 2004). Em geral, não se pode assumir que clientes e provedores de serviços utilizem a mesma terminologia para comunicação de seus interesses e, portanto, a reconciliação ontológica pode permitir uma mediação efetiva entre diferentes provedores e clientes. A investigação da aplicabilidade desta tarefa em um cenário orientado à prestação de serviços na Web consiste no cerne desta dissertação de mestrado.

## 2.3 TÉCNICAS DE RECONCILIAÇÃO ONTOLÓGICA

Podem-se distinguir dois mecanismos gerais para reconciliação ontológica: mapeamento e aglutinação de ontologias. No primeiro mecanismo, as correspondências entre duas ontologias são armazenadas separadamente das ontologias originais. Estas correspondências podem ser utilizadas, por exemplo, para a consulta de dados sobre diferentes bases de conhecimento, mediante a utilização de uma interface comum ou mecanismos de transformação de dados em diferentes representações ou visões (DOAN, 2003), (EHRIG, 2006), (SCHORLEMMER; KALFOGLOU, 2005).

No segundo mecanismo, uma nova ontologia é criada, a qual representa a união das ontologias originais. A nova ontologia captura todo o conhecimento das ontologias originais. Existem dois mecanismos para a implementação desta abordagem: a *integração* e a fusão de ontologias. A integração de ontologias consiste basicamente em reutilizar fragmentos de ontologias na formação de uma ontologia composta (PINTO; GÓMEZ-PEREZ; MARTINS, 1999), (PINTO; MARTINS, 2001). Na fusão de ontologias o desafio é a garantia da preservação das propriedades das ontologias originais na ontologia gerada, visto que a união considera um nível mais detalhado de verificação de similaridades e diferenças entre as ontologias originais (BRUIJN, 2003), (GIUNCHIGLIA; SHVAIKO, 2004).

Nesta seção, é fornecida uma visão geral sobre as principais técnicas para reconciliação ontológica, com ênfase na integração, fusão e alinhamento de ontologias. Um aspecto transversal à aplicação destas técnicas refere-se à localização e especificação de sobreposições e diferenças entre conceitos, propriedades e instâncias de diferentes ontologias. Para um melhor entendimento das diferenças entre estas técnicas, noções sobre os tipos de diferenças ontológicas que podem ocorrer nestes processos são apresentadas na subseção seguinte, com base no trabalho de Klein (KLEIN, 2001). As técnicas de reconciliação de ontologias são apresentadas nas subseções subseqüentes.

### 2.3.1 TIPOS DE DIFERENÇAS ONTOLÓGICAS

Os dois tipos básicos de diferenças ontológicas são: (1) *conceituais*, as quais consistem na heterogeneidade de conceitualizações para um mesmo domínio e (2) *contextuais*, as quais se referem à forma como os conceitos são especificados em domínios distintos.

Diferenças *conceituais* classificam-se em duas categorias: (1) diferenças de *escopo*, quando duas classes possuem alguma intersecção em suas extensões (os conjuntos de instâncias correspondentes), mas as extensões não são exatamente as mesmas (por exemplo, os conceitos de Estudante e Passageiro); e (2) diferenças na *cobertura e granularidade* do modelo, quando existe uma diferença em (a) a parte do domínio que é descrita por ambas as ontologias (e.g. ontologias para estudantes e funcionários de universidades) ou (b) o nível de detalhe através do qual o modelo é descrito (e.g. uma ontologia pode ter um conceito `Pessoa`, enquanto outra ontologia distingue entre `PessoaJovem`, `PessoaAdulta` e `PessoaIdosa`).

Diferenças de *contexto* classificam-se em três categorias: (1) diferenças de *estilo de modelagem*, quando (a) o paradigma utilizado para especificar um determinado conceito (e.g. tempo) é diferente (e.g. intervalos ou funções pontuais do tempo) ou (b) existem diferenças na forma como o conceito é descrito (e.g. uso de subclasses ou atributos na distinção de grupos de instâncias); (2) diferenças *terminológicas,* quando dois conceitos são equivalentes, mas são representados por nomes diferentes (sinonímia) ou quando o mesmo nome é usado para diferentes conceitos (homonímia); e (3) diferenças de *codificação,* quando existem diferentes representações de medida de valores em diferentes ontologias (e.g. quilômetros ou milhas, para a representação de uma medida de distância).

### 2.3.2 FUSÃO

A operação de fusão de ontologias consiste na criação de uma terceira ontologia a partir de duas ontologias originais. A nova ontologia representa uma visão unificada dos conceitos contidos nas ontologias originais.

Podem-se distinguir duas abordagens para o processo de fusão de ontologias. Na primeira abordagem, a entrada do processo é uma coleção de ontologias, e a saída é uma nova ontologia que captura as características das ontologias originais. Um exemplo de implementação desta abordagem é a ferramenta PROMPT (NOY; MUSEN, 2000), uma ferramenta baseada em um conjunto de algoritmos para fusão interativa de ontologias. Na segunda abordagem, as ontologias originais não são substituídas. Em vez disso, é criada uma "visão" mais abstrata das ontologias originais, chamada "ontologia-ponte" (*bridge ontology*), a qual importa as ontologias originais e estabelece novas relações entre os conceitos de ambas mediante a utilização de axiomas pré-definidos. Um exemplo de implementação desta abordagem é a ferramenta OntoMerge (DOU; MCDERMOTT; QI, 2002), que permite a criação de uma ontologia-ponte mediante a importação das ontologias originais, relacionando os conceitos através de um conjunto de funções ou axiomas.

A ferramenta PROMPT (NOY; MUSEN, 2000), (NOY, 2004) implementa um conjunto de algoritmos para fusão interativa de duas ontologias. O elemento central da ferramenta PROMPT é o algoritmo que define um conjunto de passos para o processo de fusão interativa: (1) identificação de conceitos candidatos à fusão, baseada em similaridades de nomes de classes. O resultado é a apresentado ao usuário como uma lista de operações fusão em potencial; (2) o usuário escolhe uma das operações sugeridas pela lista ou especifica uma operação de fusão diretamente; (3) o sistema realiza a operação solicitada e executa as modificações adicionais derivadas da ação sugerida; (4) o sistema cria uma nova lista de ações sugeridas para o usuário, com base na nova estrutura da ontologia, determinando conflitos introduzidos pela última operação solicitada, encontrando possíveis soluções para os conflitos e fornecendo os resultados para o usuário.

A ferramenta PROMPT identifica um conjunto de operações de fusão (para classes e propriedades) e um conjunto de possíveis conflitos introduzidos pela aplicação de operações (conflitos de nomes, referências de *namespaces*, redundâncias na hierarquia de classes ou violações de restrições impostas pelas propriedades).

A ferramenta OntoMerge (DOU; MCDERMOTT; QI, 2002) constitui uma abordagem *online*, na qual as ontologias originais são mantidas depois da operação de

fusão, enquanto que na ferramenta PROMPT, as ontologias originais são substituídas. A saída gerada pela ferramenta OntoMerge não é uma ontologia completamente mesclada, mas sim, uma ontologia ponte, a qual importa as ontologias originais, contendo um conjunto de axiomas-ponte (*bridging axioms*). Os axiomas-ponte representam basicamente um conjunto de regras de tradução usado para conectar as sobreposições conceituais existentes. As duas ontologias originais, juntamente com o conjunto de axiomas-ponte, são submetidas na forma de um modelo teórico a um provador de teoremas, para a execução de três tarefas principais:

1. *Tradução do conjunto de dados* (conforme transformação de instâncias proposta por de Bruijn et al. (BRUIJN, 2003): a tradução do conjunto de dados consiste transformação da representação de um conjunto de dados (instâncias) em outro tipo de representação.

2. *Geração da extensão da ontologia*: o problema da geração de uma extensão ontológica consiste em gerar conjuntos de extensão $O_{1ex}$ e $O_{2ex}$ (instâncias de conceitos) a partir de duas ontologias originais $O_1$ e $O_2$, respectivamente. Um exemplo da operação seria a geração de uma extensão de um documento WSDL baseado em uma descrição OWL-S de um *Web service*.

3. *Consulta por diferentes ontologias*: a reescrita de consultas é uma técnica para resolução de problemas envolvendo consultas a múltiplas ontologias, sendo um problema tratado superficialmente por Dou et al. (DOU; MCDERMOTT; QI, 2002).

### 2.3.3 INTEGRAÇÃO

A operação de integração de ontologias consiste na criação de uma nova ontologia mediante a composição de partes de outras ontologias. Assim como a fusão, este processo gera uma nova ontologia. A diferença entre esta abordagem e a fusão consiste em que apenas partes das ontologias originais são integradas, permitindo reuso. Portanto, o objetivo final não é alcançar uma fusão completa, mas apenas reunir os conceitos e propriedades que possuem algum nível de similaridade (JANNINK, 1998).

O projeto SUMO[2] (*Suggested Upper Merged Ontology*) foi iniciado como parte dos projetos do *IEEE Standard Upper Ontology Working Group*. O objetivo era o estabelecimento de uma ontologia-núcleo[3] padrão, com vistas a promover

[2] Vide www.ontologyportal.com e http://suo.ieee.org

[3] Ontologias-núcleo são mais comumente referenciadas como *Core Ontologies* ou *Upper Level Ontologies,* as quais são originadas a partir de processos de integração ou fusão.

interoperabilidade de dados, recuperação inteligente da informação, inferências automáticas e processamento de linguagem natural. SUMO é um padrão aberto, o qual pode ser reutilizado para fins acadêmicos ou comerciais. Desde 2004, SUMO e suas ontologias de domínio compõem a maior ontologia pública formal que existe, e a única mapeada por todo o dicionário léxico WordNet[4]. SUMO é composta por cerca de 20.000 termos e 60.000 axiomas, utilizados para fins de busca, estudos sobre lingüística e motores de inferências automáticos. Entretanto, uma ontologia-núcleo está limitada a termos genéricos, abstratos ou filosóficos.

O ponto principal consiste na disponibilização de uma ontologia genérica que possibilite a descrição de um grande domínio de áreas. Portanto, conceitos específicos de cada domínio não podem ser incluídos nesta ontologia. Uma ontologia-núcleo pode prover uma estrutura de base, a partir da qual extensões podem ser derivadas na forma de ontologias de domínio para a instanciação de vocabulários para áreas diversas, tais como Medicina, Finanças ou Serviços.

Ontologias de domínios específicos são definidas em SUMO para diferentes áreas. As ontologias catalogadas em 2005 incluem exemplos para áreas de Comunicações, Países e Regiões, Computação Distribuída, Economia, Finanças, Componentes de Engenharia, Geografia, Governo, Departamento Militar, *North American Industrial Classification System* (NAICS), Elementos Químicos (tabela periódica), Transportes e Aeroportos. Ontologias adicionais de nomenclatura para terrorismo e armas de destruição em massa estão disponíveis em subseções especiais.

A ontologia SUMO é modular e permite a extensão de informação mediante sub-ontologias auto-contidas. O termo *mereotopology* é um termo incomum até mesmo no contexto da Web semântica, e se refere à teoria formal do *parte-e-todo*, a qual faz uso de propriedades topológicas dos conceitos para a derivação de leis e relações de contato entre conceitos. Aspectos de corretude na definição destes, além de propriedades como superfície, ponto e vizinhança, relacionados ao grafo de conceitos de uma ontologia também são definidos.

Outras abordagens existentes para ontologias-núcleo incluem *Cyc*[5] e *OpenCyc*[6]. A primeira delas foi desenvolvida durante 15 anos pela Cycorp Company, sendo uma opção

[4] Maior dicionário léxico de termos da Web. Vide www.wordnet.org
[5] Vide www.cyc.com
[6] Vide www.opencyc.org
[9] Um *matchmaker* é um módulo de software que realiza um processo de associação de parâmetros, sejam sintáticos ou semânticos.

proprietária. Consequentemente, o conteúdo desta ontologia não foi submetido a processos abertos de revisão de conceitos. A ontologia *Cyc*, considerada como a maior e mais completa base de conhecimento geral do mundo, juntamente com seu motor de inferência de vocabulários, tem sido utilizada por uma grande quantidade de organizações privadas.

Como resposta à proposta SUMO, a Cycorp Company disponibilizou uma versão aberta de sua ontologia, denominada *OpenCyc*. Esta versão pode ser utilizada como base para uma grande variedade de aplicações inteligentes, mas inclui apenas uma pequena parte da base de conhecimento original. Um subconjunto maior, denominado *ResearchCyc*, é oferecido sob licença restrita a entidades "qualificadas". Embora limitada em seu escopo, a *OpenCyc* é considerada apropriada para a implementação de aplicações como interpretação automática de discurso, integração de bancos de dados, desenvolvimento rápido de ontologias em sentido "vertical", aplicações de roteamento e funções de anotação semântica.

### 2.3.4 ALINHAMENTO

O alinhamento de ontologias consiste no processo de descoberta de similaridades entre duas ontologias. Esta operação é geralmente descrita como a aplicação do operador de associação ou *Match operator* (RAHM; BERNSTEIN, 2001). A entrada do operador é um determinado número de ontologias e a saída é a especificação das correspondências entre estas ontologias.

Existem diferentes algoritmos que implementam o operador de associação. Tais algoritmos podem ser classificados em torno de duas dimensões. Por um lado existe uma distinção entre alinhamento em nível de esquema e em nível de instância (*schema-based* e *instance-based*, respectivamente). O alinhamento baseado em esquema é baseado na análise de estruturas de conceitos e propriedades contidos nas ontologias e utiliza determinadas medidas de similaridade para determinar os níveis de correspondência (NOY; MUSEN, 2000). O alinhamento em nível de instância consiste na verificação de quais instâncias pertencem a diferentes classes nas diferentes ontologias, comparando estas instâncias com as similaridades encontradas entre conceitos (DOAN, 2003).

Existe uma distinção entre associações em nível de elementos e em nível de estruturas. A associação em nível de elemento compara as propriedades de um conceito ou relação em particular (e.g. nome) e as utiliza para encontrar similaridades (NOY, 2004). A associação em nível de estrutura compara a estrutura (e.g. a hierarquia de conceitos) das ontologias para encontrar similaridades (GIUNCHIGLIA; SHVAIKO, 2004). Estes tipos de abordagens para associação podem ser combinados (EHRIG, 2006). Por exemplo, na

ferramenta Anchor-PROMPT (NOY, 2004), um *matchmaker*[9] que opera em nível de estrutura, recebe como entrada uma lista inicial de similaridades entre conceitos. O algoritmo é então utilizado para identificação de similaridades adicionais, baseado nas similaridades iniciais e na estrutura das ontologias.

O algoritmo Anchor-PROMPT visa melhorar os resultados de métodos para associação (*matchmaking*), que somente analisam o contexto local das estruturas das ontologias, tais como o PROMPT, mediante a descoberta de possíveis pontos adicionais de similaridades estruturais das ontologias. O algoritmo recebe como entrada dois pares de termos relacionados e analisa os elementos que estão incluídos no caminho que conecta os elementos da mesma ontologia com os elementos do caminho equivalente da outra ontologia. Desta forma, têm-se dois caminhos (um para cada ontologia) e os termos que constituem estes caminhos. O algoritmo procura por termos ao longo dos caminhos que possam ser similares aos termos do outro caminho, que pertencem à outra ontologia, assumindo que existe uma similaridade entre os conceitos destes caminhos. Estes novos termos são associados a um critério de peso de similaridade, o qual pode ser modificado durante a descoberta de outros caminhos nos quais estes termos possam aparecer. Termos com pesos de similaridades equivalentes são apresentados para que o usuário apresente suas sugestões de auxílio ao processo de fusão no PROMPT.

Outra abordagem prática é oferecida pelo GLUE (DOAN, 2003), que emprega técnicas de aprendizagem de máquina para a criação semi-automática de mapeamentos entre ontologias heterogêneas, baseados em dados de instância, onde uma ontologia é vista como uma taxonomia de conceitos. O sistema GLUE tem foco na descoberta de mapeamentos *um-para-um* entre os conceitos das taxonomias, embora os autores anunciem a extensão do mecanismo de *matchmaking* para relações *muitos-para-muitos* (tais como relações *um-para-N* e *N-para-um*) para trabalhos futuros.

A similaridade entre dois conceitos *A* e *B* em duas ontologias $O_1$ e $O_2$, respectivamente, é baseada no conjunto de instâncias que se sobrepõem entre dois conceitos. Para determinar se uma determinada instância do conceito *B* é também uma instância do conceito *A*, primeiro um *classificador*[10] é construído, utilizando as instâncias do conceito *A* como conjunto de dados para treinamento. Este classificador é então utilizado para classificar as instâncias do conceito *B*. O classificador então decide para cada instância de *B*, se esta é também uma instância de *A* ou não.

[10] Classificadores são elementos arquiteturais utilizados para treinamento de algoritmos, em sistemas baseados em mineração de dados e aprendizagem de máquina.

Baseadas nestas classificações, quatro probabilidades são computadas, a saber, *P(A.B)*, P(*A*,*B*), *P(A,B)* e *P(A,B),* onde, por exemplo, *P(A,B)* é a probabilidade de que uma instância no domínio pertença a *A*, mas não a *B*. Estas quatro probabilidades podem ser utilizadas para computar a probabilidade conjunta de distribuição de instâncias para os conceitos *A* e *B*, a qual é uma função suprida pelo usuário, tendo estas quatro probabilidades como parâmetros.

A abordagem do *Semantic Matching* (GIUNCHIGLIA; SHVAIKO, 2004) é um outro exemplo baseado na classificação de hierarquias. Os autores implementam um operador de associação que recebe duas estruturas de grafos (e.g. esquemas de bancos de dados ou ontologias) como entrada e produz um mapeamento entre os elementos dos dois grafos que apresentam algum tipo de correspondência semântica. Segundo esta abordagem, a maioria das novas abordagens para *matchmaking* de esquemas e ontologias tem sido mais sintática do que semântica. No *matchmaking* sintático, os rótulos e as estruturas sintáticas dos grafos são associados e algum coeficiente de similaridade [0,1] é obtido, o qual indica a similaridade entre dois nós. O *Semantic Matching* verifica relações de conjuntos entre nós do grafo gerado, levando em consideração o significado de cada nó; a semântica de um nó é determinada pelo seu rótulo e pela semântica de todos os nós associados a níveis mais altos na hierarquia. As possíveis relações retornadas pelo algoritmo do *Semantic Matching* são de igualdade (=), sobreposição (∩), disjunção (⊥), generalização (⊆) e especialização (⊇). A correspondência dos símbolos com a teoria de conjuntos não é uma simples coincidência, visto que cada conceito na classificação de hierarquias representa um conjunto de instâncias.

Outra importante abordagem para alinhamento de ontologias inclui o *Quick Ontology Mapping* (QOM)(EHRIG; STAAB, 2004), que foi projetado para a detecção e criação de mapeamentos *on-the-fly* de ontologias na Web. Visando aumentar a velocidade de processamento na identificação de similaridades entre duas ontologias, o QOM não compara todas as entidades da primeira ontologia com todas as entidades da segunda, utilizando heurísticas (e.g. identificação de rótulos similares) para a redução do número de comparações na detecção de mapeamentos. A computação de similaridades é realizada mediante a combinação de várias funções de verificação de similaridades, tais como similaridades de *strings*. Estas métricas são combinadas por uma função composta, que agrega as medidas individuais de similaridades. O QOM aplica uma função de sigmóide, que confere maior ênfase a altas similaridades de instâncias e menor ênfase às diferenças. As correspondências entre as entidades nas ontologias são extraídas mediante a aplicação de

limites para a agregação de medidas de similaridade. A saída de uma iteração pode ser usada como parte da entrada para a iteração subseqüente, com vistas a produzir um resultado mais refinado. Após um determinado número de iterações, obtém-se uma tabela de correspondências entre as ontologias de entrada.

### 2.3.5 INTEROPERABILIDADE VERTICAL *VERSUS* HORIZONTAL

Considerando o aspecto de interoperabilidade entre aplicações, as abordagens citadas para reconciliação ontológica podem ser classificadas ao longo de duas dimensões: *vertical* e *horizontal* (vide Figura 2.1).

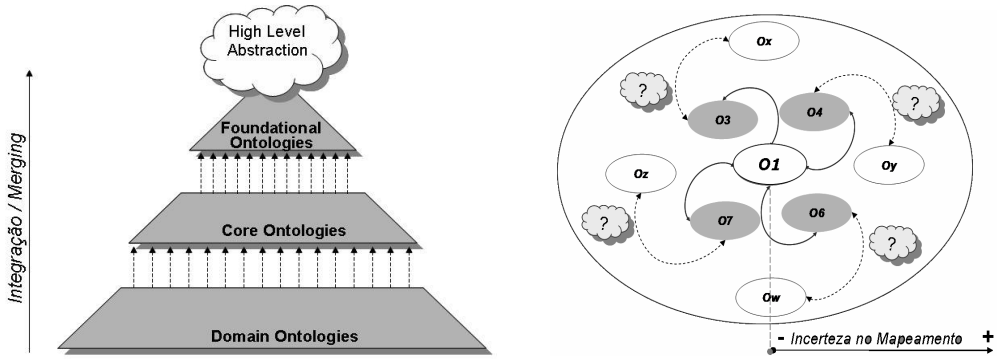



**Figura 2.1 – Dimensões nos processos de reconciliação ontológica**

A interoperabilidade *vertical* consiste na aplicação de técnicas de aglutinação conceitual, como a fusão e a integração. O produto final da aplicação de tais técnicas consiste na formação de ontologias-núcleo (*upper-level*, *foundational* ou *core ontologies*) (GANGEMI; PISANELLI; STEVE, 1998), (MITRA; WIEDERHOLD, 2004), (NILES; PEASE, 2001), (STEPHENS; GANGAM; HUHNS, 2004). O reuso de conhecimento na integração e formação de visões organizacionais, na aplicação da fusão, representam algumas das vantagens desta abordagem. Entretanto, aglutinações sucessivas podem gerar conceitos por demais abstratos e arbitrários (COHEN; RAVIKUMAR; FIENBERG, 2003). Neste caso a perda da especificidade dos conceitos e propriedades pode implicar na adoção de um número restrito de operações verticais, para evitar a perda da utilidade das ontologias geradas.

Por outro lado, a interoperabilidade *horizontal* consiste na aplicação de técnicas de mapeamento, como o alinhamento. O produto final do alinhamento é um conjunto de mapeamentos entre propriedades e conceitos das ontologias originais (HUANG, 2006), (WANG; GASSER, 2002). Entre as vantagens, pode-se incluir a preservação da especificidade conceitual em cada uma das ontologias mapeadas, além da menor granularidade dos mapeamentos que permite análise de determinados aspectos em comum, e não a aglutinação

dos conceitos das ontologias originais. Como desvantagem, pode-se citar que a aplicação sucessiva de alinhamentos também pode sofrer perdas conceituais no caso de correspondência semântica mais fraca (ou seja, quando a intersecção conceitual é diferente de vazio, existe incerteza quando ao grau da intersecção, com base em teoria de conjuntos). Ou seja, no caso de múltiplas ontologias, nada pode garantir a transitividade dos mapeamentos gerados.

Discussões em torno da escolha da técnica mais adequada representam em si um tipo de conflito de preferências. Desta forma, uma abordagem híbrida para reconciliação envolve a própria combinação das técnicas, de acordo com o contexto. Por exemplo, no caso da descoberta e composição de serviços, o alinhamento pode ser a técnica mais adequada, visto que o objetivo é o mapeamento de parâmetros. No caso do estabelecimento de contratos após uma negociação (considerando também como utópica a adoção de um padrão para contrato de serviço na Web), a integração ou fusão podem ser utilizadas para o estabelecimento de um contrato de representação híbrida.

## 2.4 MEDIAÇÃO E ARQUITETURAS ORIENTADAS A SERVIÇOS SEMÂNTICOS

Segundo a definição de Alonso et al. (ALONSO, 2004), *Web services* constituem fragmentos de funcionalidade encapsulada que podem ser acessados pela Web. Esta tecnologia tem conferido um novo nível de funcionalidade para a Web atual, proporcionando interoperabilidade entre aplicações mediante o uso de componentes de software que operam segundo padrões de comunicação bem definidos.

Apesar das expectativas, as tecnologias atuais para operação de *Web services*, SOAP, WSDL e UDDI operam em nível sintático da informação, ou seja, embora forneçam suporte à interoperabilidade (i.e., interoperabilidade entre plataformas existentes para o desenvolvimento de aplicações orientadas a serviços), estas tecnologias ainda exigem um nível elevado de intervenção humana (WANG, 2004). Desta forma, recaem sobre o programador as tarefas de descobrir, selecionar ou mesmo compor funcionalidade para o atendimento de requisitos específicos de uma dada aplicação, o que significa perda de escalabilidade e redução de lucros na corrida pela garantia de espaço no comércio eletrônico (BUSSLER; MAEDCHE; FENSEL, 2003).

A automatização de tarefas, tais como descoberta, seleção, composição e execução, pressupõe a existência de descrições semânticas dos serviços (SYCARA, 2003), (KELLER, 2005). Uma união da tecnologia de *Web services* com tecnologias da Web semântica abre possibilidades neste sentido. Mais especificamente, a utilização de ontologias tem sido considerada a alternativa mais plausível para enriquecimento das descrições de serviços

(BURSTEIN, 2005). A marcação semântica pode ser explorada para a automatização de tarefas como descoberta ou seleção de serviços, por exemplo, permitindo interoperabilidade entre aplicações com um mínimo de intervenção humana (SRINIVASAN; PAOLUCCI; SYCARA, 2006). A adoção desta abordagem tem gerado grandes expectativas na automatização de processos no comércio eletrônico e na integração de aplicações de larga escala. *Web services* oferecidos por diversas organizações podem ser localizados com base em necessidades específicas de outras aplicações, podendo ser compostos na construção de serviços mais complexos, de funcionalidade mais ampla.

A concretização deste cenário depende da estruturação de plataformas e linguagens que permitam o desenvolvimento de tais aplicações. As próximas subseções tratam das principais abordagens existentes para a implementação destes aspectos.

### 2.4.1 ABORDAGENS PARA DESCRIÇÕES DE SERVIÇOS

O modelo SWSA (*Semantic Web Service Architecture*) consiste na extensão do modelo SOA pela inserção de ontologias para enriquecimento das descrições dos serviços (SYCARA, 2003), (PREECE; DECKER, 2002). O objetivo é a automatização inteligente de tarefas como descoberta, seleção ou composição de serviço (BURSTEIN, 2005), (KELLER, 2005). O estabelecimento deste modelo envolve a construção de *frameworks* para o desenvolvimento de aplicações. Estes devem prover: (1) um modelo conceitual bem definido para descrição dos serviços, (2) uma linguagem formal para provisão da base sintática e semântica (com base em diferentes lógicas, em diferentes níveis de expressividade) para o modelo conceitual, e (3) um ambiente de execução, que reúna todos os componentes que utilizam a linguagem definida para a automatização das tarefas relacionadas aos *Web services*.

As subseções seguintes fornecem uma visão geral sobre os *frameworks* propostos para o desenvolvimento de *Web services* semânticos. Será dada ênfase especial a aspectos relacionados à fundamentação de linguagens e mecanismos de mediação pertinentes às abordagens mencionadas.

### 2.4.2 WSMO – *WEB SERVICES MODELING ONTOLOGY*

WSMO (WSMO, 2005) constitui a principal iniciativa no desenvolvimento de *Web services* semânticos na Europa, integrando parte do projeto SDK Cluster[11]. O objetivo principal é a

[11] Vide www.sdk-cluster.org

provisão de um *framework* para suporte à modelagem conceitual de Web services, representação formal da descrição dos serviços e um ambiente de execução. Os elementos principais desta abordagem são: (1) o modelo conceitual WSMO (*Web Service Modeling Ontology*); (2) a linguagem WSML (*Web Service Modeling Language*); e (3) o ambiente de execução WSMX (*Web Service Modeling Execution Environment*).

Alguns dos princípios que fundamentam a abordagem WSMO incluem: (1) compatibilidade com padrões da Web, mediante utilização de Identificadores de Recursos Universais (URI) e *namespaces*, para delimitação de escopo de validade das marcações semânticas, (2) utilização de ontologias para descrição de requisitos funcionais, requisitos não-funcionais, *Web services* e mediadores e (3) centralização na mediação, para resolução de problemas terminológicos.

O conceito de mediação em WSMO foca a possibilidade de heterogeneidade em nível terminológico (nível de dados), em nível de comunicação (protocolos) e em nível de processos executados. Mediadores em WSMO conectam todos os demais elementos arquiteturais, garantindo fraco acoplamento e interoperabilidade entre os componentes, nos níveis citados. WSMO define quatro tipos de mediadores para conexão de seus elementos arquiteturais (vide Figura 2.2): *OOMediators*, para mediação de ontologias heterogêneas; *GGMediators*, para conexão de requisitos funcionais (*goals*); *WGMediators*, para associação de *Web services* a requisitos funcionais (*goals*); e *WWMediators*, para interoperabilidade entre *Web services*, com vistas a atingir um determinado requisito funcional (MOCAN, 2005).

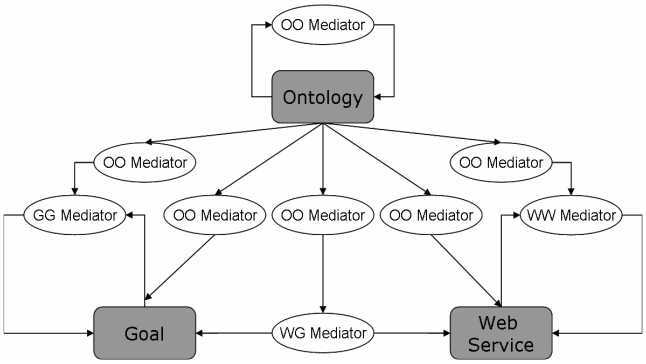



**Figura 2.2 – Relacionamentos entre mediadores em WSMO (adaptado de MOCAN ET AL, 2005)**

Em relação aos fundamentos lingüísticos de WSMO, seus elementos são definidos pela linguagem MOF (*Meta Object Facility*) (OMG, 2002), uma linguagem para especificação de *meta-modelos* (vide Figura 2.3). MOF define uma linguagem abstrata e um *framework* para especificação, construção e gerenciamento de *meta-modelos* neutros de

aspectos específicos de implementação. Tendo em vista que WSMO é considerado como um *meta-modelo* para Web services semânticos, MOF representou a escolha para a definição dos elementos arquiteturais de WSMO. Em relação aos níveis de abstração de MOF (*meta-meta-modelo, meta-modelo, modelo* e *informação*), a linguagem que define WSMO corresponde ao *meta-meta-modelo*; WSMO em si constitui a camada de *meta-modelo*; os elementos definidos por WSMO (incluindo *goals*, *Web services*, *ontologias* e *mediadores*) representam a camada de *modelo*; e os dados descritos pelas ontologias e compartilhados pelos *Web services* constituem a camada de *informação*. Além disso, a descrição dos elementos de WSMO é complementada mediante o uso de terminologia para requisitos não-funcionais, provida pelo padrão Dublin Core (WEIBEL, 1998), para a consideração de aspectos relacionados a qualidade de serviço.

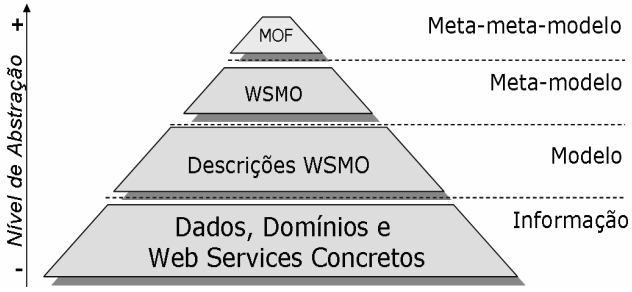



**Figura 2.3 – Fundamentos lingüísticos de WSMO (adaptado de MOCAN ET AL, 2005)**

### 2.4.3 OWL-S – *WEB ONTOLOGY LANGUAGE FOR SERVICES*

A abordagem OWL-S (*Web Ontology Language for Services*)(OWL-S, 2004), que constitui um dos projetos do programa DAML[12], é uma ontologia para descrição de serviços baseada na linguagem OWL (OWL, 2004). Esta representação visa prover blocos de codificação para enriquecimento das descrições de serviços, tendo como base a expressividade da linguagem OWL. A ontologia OWL-S é comumente referenciada como uma linguagem para descrição de Web services, provendo um vocabulário básico que pode ser utilizado juntamente com outros aspectos de OWL para a criação de uma descrição para serviços.

OWL-S consiste de três sub-ontologias relacionadas: (1) *Service Profile*, que é utilizada para expressar "o que o serviço faz", para fins de publicação, construção de consultas e *matchmaking* de aspectos funcionais do serviço, na tarefa de descoberta; (2) *Service Model*, que descreve "como o serviço trabalha", que serve como base para as

[12] Vide www.daml.org

tarefas de invocação, composição, monitoramento e reconfiguração de serviços compostos, com base em modelos de *workflows*; e (3) *Service Grounding*, que mapeia descrições mais abstratas do *process model* em parâmetros concretos de formatos de mensagens e protocolos de comunicação (normalmente descritos na forma de documentos WSDL).

Estas três sub-ontologias são ligadas pelo conceito abstrato de serviço (*Service*), o qual serve como um ponto de referência organizacional para a declaração de um *Web service*; sempre que um serviço é declarado, uma instância do conceito *Service* é criada. Conforme mostradas na Figura 2.4, as propriedades *presents*, *describedBy* e *supports*, são propriedades de um serviço. As classes *ServiceProfile* (a qual identifica a sub-ontologia *profile*), *ServiceModel* (que identifica a sub-ontologia *Process*) e *ServiceGrounding* (que identifica a sub-ontologia *Grounding*) são as respectivas imagens das funções representadas por estas propriedades.

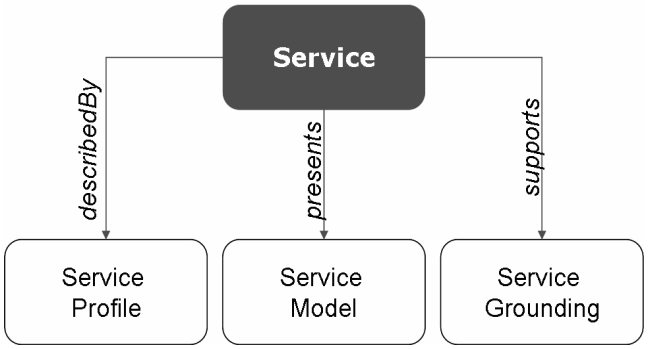



**Figura 2.4 – Esquema básico da ontologia OWL-S (adaptado de (OWL-S, 2004))**

Embora exista uma separação entre os aspectos que descrevem um serviço em OWL-S (aspectos funcionais, modelos de *workflows* e mapeamentos com WSDL), esta abordagem não considera explicitamente o aspecto da mediação. A composição de serviços exemplifica a necessidade de mediação entre serviços em nível de orquestração ou gerenciamento de *workflows*. Em OWL-S também não existe uma separação nítida de visões de clientes e provedores sobre um mesmo serviço. A representação do serviço é bilateral e depende da aplicação determinar qual ponto de vista adotar. Uma descrição mais específica para requisitos não-funcionais (indicadores de parâmetros de qualidade de serviço) também não está incluída em OWL-S. Existem propostas de extensão da sub-ontologia *ServiceProfile* mediante a utilização de ontologias de QoS (BIANCHINI; ANTONELLIS; MELCHIORI, 2004), (RIBEIRO; ROSA; CUNHA, 2004c) e ontologias de domínio (MENA, 1998). Quanto ao aspecto da mediação, algumas abordagens tais como

incluem a especificação ou materialização de mediadores WSMO para OWL-S (PAOLUCCI; SRINIVASAN; SYCARA, 2004).

Em relação à fundamentação lingüística de OWL-S, a linguagem OWL apresenta alguns problemas relacionados à estruturação de seus construtores. Os fundamentos da linguagem incluem combinações de regras em SWRL (*Semantic Web Rule Language*, que consiste numa combinação de OWL com RuleML, cujas regras são baseadas em cláusulas de Horn) e aspectos sintáticos do *framework* DRS (*Dynamic Resource Scheduling*, que constitui um conjunto de algoritmos pra escalonamento de requisitos de QoS relacionados com a provisão de serviços multimídia), o que constitui um ponto de geração de indecidibilidade ou semântica aberta (*Open World Assumption*[13]). A separação das camadas de *meta-modelos* também não é bem definida.

### 2.4.4 SWSF – *SEMANTIC WEB SERVICES FRAMEWORK*

A abordagem SWSF (*Semantic Web Services Framework*), é uma das mais novas abordagens para *Web services* semânticos, proposta pelo Comitê de Linguagens para *Web services*[14] (iniciativa para desenvolvimento de *Web services* semânticos, do W3C). Esta abordagem é baseada em dois componentes principais: uma ontologia e um modelo conceitual correspondente para descrição dos *Web services*, denominada SWSO (*Semantic Web Services Ontology*) e uma linguagem para especificação de caracterizações formais dos conceitos e descrições de *Web services*, denominada SWSL (*Semantic Web Services Language*).

O modelo conceitual SWSO (SWSO, 2005) tem sua caracterização formal baseada em duas variantes de SWSL: SWSL-FOL, baseada em lógica de primeira ordem, e ROWS (*Rule Ontology for Web Services*), fundamentada na semântica de programação lógica. As ontologias resultantes são chamadas: FLOWS (*First-Order Logic Ontology for Web Services*), com base em lógica de primeira ordem e ROWS (*Rule Ontology for Web Services*), baseada em regras. Considerando o foco destas seções em aspectos ontológicos, ênfase especial será dada à análise dos fundamentos de FLOWS.

O desenvolvimento de FLOWS teve influência direta da abordagem OWL-S e da verificação de resultados provenientes da aplicação desta abordagem em vários estudos de caso. FLOWS provê um rico modelo para descrição de fluxos de processos e aspectos

[13] Considerar como falso tudo aquilo que não é reconhecido como verdadeiro, a priori, é chamado de *Closed World Assumption*. A pressuposição inversa ou *Open World Assumption*, estabelece que, a ausência de conhecimento sobre algo não implica necessariamente em falsidade, ou seja, a interpretação é vaga ou imprecisa.
[14] Ver www.swsi.org

comportamentais de *Web services*, com base em PSL (*Process Specification Language*), (GRUNINGER, 2003). Desta forma, FLOWS pode ser considerada como uma extensão ou aperfeiçoamento da abordagem OWL-S, com ênfase na provisão de interoperabilidade com padrões existentes em *Web services* (e.g. BPEL, WSDL, etc.).

Embora exista uma similaridade entre FLOWS e OWL-S, uma diferença importante entre as duas abordagens reside na expressividade de cada linguagem de base. FLOWS é baseada em lógica de primeira ordem, enquanto que OWL-S é baseada em OWL-DL. Desta forma, é possível realizar cálculos de situações relacionadas com estados abstratos de um sistema baseado em *Web services*, com FLOWS, tais como cálculo de predicados. Predicados na forma de *invariantes* são modelados como relações em SWSO.

Assim como OWL-S, FLOWS consiste de três componentes para descrição de serviços: *Service Descriptors*, *Process Model* e *Grounding*. A primeira ontologia componente é usada para prover informações descritivas básicas sobre um serviço. A segunda componente é usada para descrever como o serviço opera. A terceira componente é usada para a ligação das descrições abstratas dos serviços com especificações mais concretas de mensagens, protocolos, providas por WSDL.

Em relação ao embasamento lingüístico da abordagem, SWSL-FOL e SWSL-Rules (sub-linguagens de SWSL) são compatíveis com padrões Web amplamente aceitos, como o uso de URIs, integração com XML, *namespaces* e mecanismos de importação. Ambas as linguagens possuem organização em níveis de expressividade, onde cada camada acrescenta novos conceitos e propriedades à camada inferior.

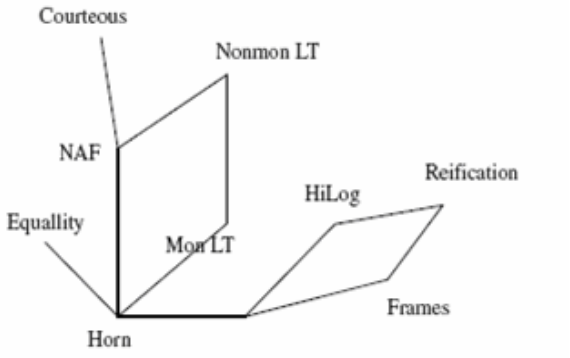



**Figura 2.5 – Estrutura de linguagens de SWSL-Rules (SWSF, 2005)**

SWLS-Rules incluem características de *Couteous Logic Programming* (GROSOF, 1999), *HiLog* (CHEN; KIFER; WARREN, 2003) e *F-Logic* (KIEFER, 2004), e podem ser consideradas como linguagens de especificação e de implementação. As regras de SWLS-

Rules provêem suporte a tarefas automatizadas, como descoberta e invocação de Web services. O relacionamento entre as linguagens de embasamento da abordagem SWSF são mostrados na Figura 2.5. Assim como OWL-S, SWSF não provê tratamento explícito a aspectos de mediação quanto a conflitos terminológicos ou modelos de *workflows* de serviços.

### 2.4.5 IRS-III – *INTERNET REASONING SERVICE*

A abordagem IRS-III (DOMINGUE, 2004) consiste em um *framework* de implementação que atua como mediador entre os objetivos do cliente (requisitos funcionais) e os serviços disponíveis. IRS-III utiliza WSMO como ontologia de base. Entre alguns dos princípios de projeto desta abordagem podemos destacar: (1) o suporte à invocação de serviços com base na descrição de requisitos funcionais (*capabilities*, no lado do provedor), permitindo a invocação de um serviço mediante o fornecimento de um requisito funcional que expresse um objetivo esperado; e (2) independência de plataformas específicas de implementação (embora seja recomendada a utilização de padrões derivados de XML, os quais já são amplamente aceitos na Web).

O modelo arquitetural de IRS-III foi criado para interligar descrições baseadas em ontologias com todos os componentes que oferecem suporte à automatização de tarefas de serviços. Os três principais componentes são: *IRS-III Server*, *IRS-III Publisher* e *IRS-III Client*, os quais se comunicam via protocolo SOAP (vide Figura 2.6).

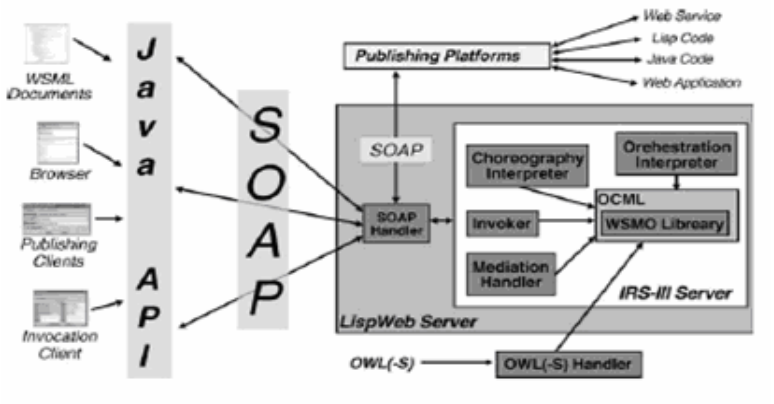



**Figura 2.6 – Arquitetura IRS-III (DOMINGUE ET AL, 2005)**

No cerne do servidor, se encontra uma biblioteca WSMO, na qual as definições são armazenadas com uso da linguagem de representação OCML (*Operation Conceptual Modelling Language*) (MOTTA, 1998). A biblioteca é estruturada em modelos de

conhecimento para requisitos funcionais (*goals*), *Web services* e mediadores. Em WSMO, um *Web service* é associado a uma interface, a qual contém detalhes sobre orquestração e coreografia. A orquestração especifica os modelos de fluxo de dados e processos entre *Web services*, em uma composição. A coreografia especifica aspectos de comunicação na tarefa de invocação dos serviços, tais como a geração de mensagens em formato SOAP.

O elemento gerenciador da tarefa de mediação na arquitetura (*Mediation Handler*) provê a funcionalidade de interpretar descrições de mediadores WSMO, incluindo regras de mediação de dados, invocação de serviços de mediação e conexão de mediadores. A ontologia IRS-III é baseada no modelo conceitual WSMO, com a diferença de que provê suporte à invocação direta de serviços com base na funcionalidade do serviço (*capability*). Para atender a esta propriedade, a descrição dos *Web services* deve estar relacionada a regras de entrada e saída aceitas (*input/output roles*). Os objetivos (*goal*s) são associados à funcionalidade dos serviços (*capabilities*) com o uso de *WGMediators*. Quando um objetivo é requisitado, o módulo *IRS Broker* cria um conjunto de possíveis serviços que possam atender essa funcionalidade, mediante o uso de um *WGMediator*. Um *Web service* é então selecionado mediante o uso de uma função de aplicabilidade, que filtra os serviços previamente descobertos.

### 2.4.6 WSDL-S – *WEB SERVICE SEMANTICS*

A abordagem WSDL-S (AKKIRAJU, 2005), proposta pelo METEOR-S Group[15], consiste em um mecanismo para enriquecimento das descrições funcionais de um *Web service*, conforme a representação base de WSDL. Partindo do pressuposto da existência de um modelo semântico para *Web services*, WSDL-S descreve um mecanismo para associar o referido modelo semântico a especificações concretas em WSDL. Utilizando da extensibilidade dos elementos de WSDL, um conjunto de anotações pode ser criado para descrever entradas, saídas, precondições e efeitos de operações automatizadas para *Web services* (vide Figura 2.7). Pelo fato de este modelo semântico ser independente da sintaxe de WSDL, qualquer linguagem para definição de ontologias pode ser utilizada para a extensão do modelo.

WSDL-S opera sob um conjunto de princípios que incluem: (1) utilização de padrões existentes para *Web services*, considerando que a adoção de padrões facilita aspectos de integração, promovendo compatibilidade de camada adicional de

[15] Vide http://lsdis.cs.uga.edu/

semântica com documentos WSDL comuns; (2) anotação semântica independente de linguagens de representação, visto que provedores podem utilizar diferentes representações para descrição de seus serviços; e (3) suporte a anotações compatíveis com os tipos de dados nativos de XML Schema, o qual constitui uma das principais definições para tipos de dados comuns na Web.

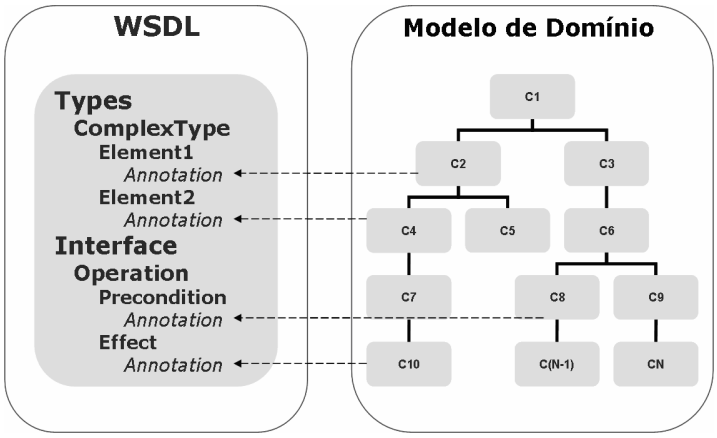



**Figura 2.7 – Associação de semântica aos elementos de WSDL (AKKIRAJU ET AL, 2005)**

## 2.5 MECANISMOS DE MEDIAÇÃO EM SOA

A tarefa da mediação no suporte a outras tarefas de SOA, como descoberta, seleção e composição de serviços tem sido considerada mais recentemente por várias abordagens. No sentido prático, mediadores têm sido implementados como programas que realizam traduções *ponto-a-ponto*, em operações de baixo nível (GARCIA-MOLINA, 1997), (STOLLBERG; CIMPIAN; FENSEL, 2005), (TUNG; LIN, 2005). Embora estes mediadores atendam aos objetivos de curto prazo, permitindo que dois sistemas se comuniquem, aspectos como manutenibilidade e escalabilidade constituem problemas ainda não resolvidos quanto ao uso desses programas. Portanto, uma abordagem para mediação para arquiteturas de sistemas distribuídos, como SOA, deve incluir baixo acoplamento com aspectos específicos de implementação.

A mediação semântica permite uma abordagem mais dinâmica através do uso de ontologias. Neste sentido, mediadores podem ser utilizados para a conversão de parâmetros entre implementações distintas (SILVA, 2006), (SILVA, 2007). A modelagem de dados e processos relacionados aos serviços na Web, mediante o uso de ontologias, permite a definição de equivalências, similaridades ou diferenças entre as interfaces que descrevem os serviços. Um mediador pode realizar a associação destes parâmetros para garantir um determinado nível de interoperabilidade entre clientes e provedores de serviços. Em geral,

as abordagens atuais para mediação em SOA são classificadas em dois níveis: *mediação de dados* e *mediação de processos*.

A *mediação de dados* é necessária quando, dadas as descrições do tipo de mensagens enviadas por um sistema e recebidas por outro, estas são semelhantes em nível semântico, mas diferem na representação sintática de parâmetros (YAN; ÖZSU; LIU; 1997). Isto ocorre devido à existência de diferentes convenções para formatação de tipos de parâmetros por parte de diferentes organizações. Para contornar estas diferenças, ferramentas de mapeamento podem ser utilizadas em tempo de projeto para a resolução de diferenças terminológicas. Esta abordagem pode ser utilizada para o mapeamento de relações *um-para-um* entre parâmetros de diferentes descrições de mensagens e tipos de dados. Para mapeamentos mais complexos, do tipo *muitos-para-muitos* ou *um-para-muitos*, linguagens baseadas em regras podem ser necessárias para a descrição dos relacionamentos gerados. Considerando o desenvolvimento de um mediador para conversão de tipos de dados, sua funcionalidade também pode ser descrita (e.g. o tipo das aplicações de origem e destino entre as quais este opera) e, desta forma, as partes interessadas em estabelecer uma comunicação (sejam estas humanas ou aplicações), podem utilizar estas descrições com o propósito de escolha do mediador adequado. Sendo assim, é conveniente pensar no mediador como um serviço em si, o qual também possui uma interface bem definida que descreve sua funcionalidade, técnicas utilizadas e contexto de atuação.

A *mediação de processos* é necessária quando existe equivalência semântica entre processos compartilhados por duas aplicações. Entretanto, as mensagens trocadas durante a execução do processo diferem quanto aos padrões utilizados (MEDJAHED; BOUGUETTAYA; ELMAGARMID, 2003). Um mediador de processos deve garantir a troca de mensagens entre processos gerados pelas aplicações envolvidas. Como resultado, o mediador precisa criar novas mensagens que simulem uma comunicação direta entre a aplicação de origem e a aplicação de destino. O conteúdo de tais mensagens é traduzido em tempo de execução pelo mediador, o qual permite comunicação transparente entre as aplicações.

Segundo Wiederhold (WIEDERHOLD, 1997), um serviço de mediação é caracterizado pela autonomia na realização de inferências sobre a resolução de um determinado conflito. Tal serviço também deve ser escolhido por parte das entidades envolvidas na negociação. No modelo SOA, *Web services* representam uma das alternativas para implementação deste conceito. Entretanto, abordagens atuais como WSMO e IRS-III, encontram-se em estágio inicial no que concerne à especificação concreta de mecanismos de operação para *Web services* de mediação. No contexto da mediação de dados e de

processos, alternativas tais como *matchmakers* e *composers* têm apresentado soluções parciais para mediação, respectivamente. Vale salientar que a mediação provida por estes mecanismos é de caráter estático, visto que estes componentes não podem ser selecionados *a priori* pelo cliente ou provedor, como o seria no caso de *Web services*. As próximas subseções tratam de aspectos gerais dessas abordagens.

### 2.5.1 *MATCHMAKERS* E REGISTROS SEMÂNTICOS

Os mecanismos atuais para busca de serviços em UDDI são limitados a buscas por palavras-chave, além de que as taxonomias *TModels* não oferecem suporte a inferências automáticas (CONSTANTINESCU; FALTINGS, 2003). Como conseqüência, as buscas produzem resultados em sua maioria irrelevantes, sem considerações sobre o contexto ou domínio de um determinado serviço de interesse. Tal problema de associação semântica pode ser contornado mediante o uso de ontologias em OWL ou RDF, ao invés do uso de parâmetros sintáticos em XML (AKKIRAJU, 2003), (JAEGER, 2005). Para produzir resultados precisos na busca por serviços, ontologias em OWL-S podem ser associadas aos parâmetros dos *TModels*, conferindo maior especificidade à busca e associação de inferências automáticas para mineração de serviços candidatos ao atendimento da funcionalidade desejada.

Entre as principais abordagens relacionadas ao enriquecimento de descrições UDDI com ontologias OWL-S, encontram-se os trabalhos de Paolucci et al. (PAOLUCCI, 2002a), (PAOLUCCI, 2002b) e Akkiraju et al. (AKKIRAJU, 2003). Para atingir esta simbiose, descrições OWL-S armazenadas em um registro UDDI. Desta forma, o registro UDDI é acrescido de um módulo de *matchmaking* interno, o qual processa descrições em OWL-S, e realiza as conversões para a representação UDDI. O componente *matchmaker* é completamente embutido no registro UDDI.

A arquitetura proposta por Paolucci et al. prevê a integração do módulo de *matchmaking* no registro UDDI, mediante a utilização de uma API que permite consultas específicas pela funcionalidade do serviço desejado (vide Figura 2.8). O componente *matchmaker* é fortemente acoplado à estrutura de UDDI, mediante o uso das portas (*publish* e *inquiry*) para a realização de suas operações. Maior detalhamento do algoritmo proposto pode ser verificado em (SRINIVASAN; PAOLUCCI; SYCARA, 2004). A combinação de OWL-S com UDDI também está associada à extensão das taxonomias *TModels* com a sub-ontologia *ServiceProfile*, de OWL-S.

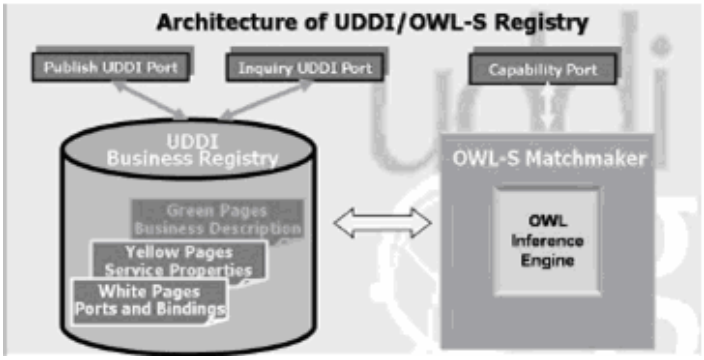



**Figura 2.8 – Arquitetura do registro UDDI enriquecido com OWL-S (PAOLUCCI ET AL, 2002a)**

A atuação do módulo *matchmaker* consiste basicamente na tradução de consultas. A descoberta de serviços é realizada mediante o mapeamento de parâmetros de busca na forma de conceitos OWL-S para representações internas de UDDI em *TModels*. O algoritmo de *matchmaking* proposto define mecanismos de associação mais flexíveis do que a busca baseada em palavras-chave, com base em critérios de associação descritos em (SRINIVASAN; PAOLUCCI; SYCARA, 2006).

Quando uma consulta é submetida, o algoritmo procura primeiramente por associações de parâmetros de saída (*outputs*). Caso a busca forneça algum resultado, uma nova busca é feita sobre o conjunto de serviços retornados, dessa vez com base em parâmetros de entrada (*inputs*). O nível de associação varia de acordo com as relações semânticas dos conceitos e suas propriedades nas ontologias, ampliando a busca sintática simples. As inferências sobre as estruturas de conceitos e propriedades podem ser realizadas mediante o uso de motores de inferência como o RACER (HAARSLEV; MÖLLER, 2003). As desvantagens desta proposta consistem principalmente: (1) no aspecto arquitetural fortemente acoplado do módulo de *matchmaking* e o registro UDDI, pelas limitações dos *TModels*; (2) no aspecto estático da mediação do *matchmaker* (onde clientes e provedores não têm escolha sobre o tipo de mediação, confiando apenas nos resultados de um único módulo); e (3) na cobertura de tarefas resumidas à descoberta de serviços, sem considerar possibilidades de composição ou seleção de serviços com base em parâmetros de qualidade de serviço.

### 2.5.2 *COMPOSERS*

Várias técnicas têm sido propostas para a composição de serviços (CASATI, 2000), (MEDJAHED; BOUGUETTAYA; ELMAGARMID, 2003), (AGGARWAL, 2004). Entretanto, a maioria destas propostas envolve a manipulação de detalhes de programação em baixo nível, tornando este tipo de operação demasiadamente complexo para o usuário.

Neste sentido, *composers* representam aplicações que automatizam a conexão de operações, invocação de serviços e mapeamento de mensagens em cadeias de serviços compostos.

Enquanto a fase de descoberta de serviços consiste em responder "quais" serviços podem atender uma determinada funcionalidade desejada, a fase de composição determina "como" estes serviços podem ser relacionados para atingir o objetivo desejado. Para tal, *composers* são usados na coordenação de serviços mais simples na construção de um serviço mais complexo. O processo de composição envolve uma seleção prévia de serviços (BERNSTEIN; KLEIN, 2004), (WAGNER; KELLERER, 2004), interconexão de operações, mapeamento de mensagens e de tipos de dados, de forma transparente ao usuário. A semântica da descrição dos *Web services* é crucial na verificação de compatibilidade de tipos de dados, mensagens, entradas e saídas na formação de um *workflow* de serviços.

Entre várias abordagens para construção de *composers*, Medjahed et al. (MEDJAHED; BOUGUETTAYA; ELMAGARMID, 2003) propõe um modelo baseado em níveis progressivos de composição (vide Figura 2.9). Em sua abordagem, são consideradas três entidades participantes: provedores, clientes e *composers*. Os provedores são responsáveis pela descrição dos serviços através de ontologias. Os *composers* realizam a montagem de *workflows* dos serviços com base nas descrições das ontologias. Desta forma, uma composição de serviços pode ser publicada como um serviço simples, em um registro, permitindo a descoberta por parte dos clientes. Estes últimos podem invocar o *Web service*, seja este simples ou composto.

Segundo Medjahed et al. (MEDJAHED; BOUGUETTAYA; ELMAGARMID, 2003), o principal fator na definição de um serviço composto consiste em determinar se os serviços constituintes são compatíveis. Por exemplo, seria difícil invocar uma operação sem um mapeamento prévio entre os parâmetros fornecidos pela aplicação cliente e os parâmetros desta operação. Neste sentido, os autores propõem um modelo de composição de *Web services* baseado em regras para verificação de compatibilidade de serviços em nível sintático (operacional) e semântico (qualitativo). As regras sintáticas incluem: (1) compatibilidade de modos de operação (*mode composability*), para comparação de modos de operações; e (2) compatibilidade de protocolos (*binding composability*), para comparação de detalhes de protocolos utilizados. As regras semânticas incluem: (1) compatibilidade de mensagens (*message composability*), para comparação do número de parâmetros de mensagens trocadas, tipos de dados e regras de negócios; (2) compatibilidade de semântica das operações (*operation semantics composability*), para comparação da semântica das operações; (3)

compatibilidade em nível de QoS (*qualitative composability*), para verificação de convergências ou conflitos entre requisitos não-funcionais de serviços; e (4) verificação de valor agregado na composição (*composition soundness*), dada a possibilidade de uma determinada composição agregar maior valor do que a simples soma das partes.

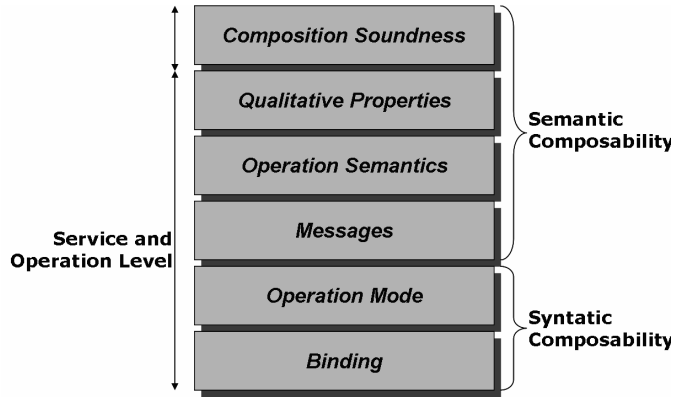



**Figura 2.9 – Níveis de composição de serviços (adaptado de MEDJAHED ET AL, 2003)**

O aspecto da agregação de valor em serviços compostos (*composition soundness*), de caráter bem mais complexo e subjetivo que os anteriores, está diretamente relacionado à análise de convergência de requisitos não-funcionais dos serviços, para determinação do impacto geral sobre a qualidade oferecida. O trabalho referenciado não aprofunda questões sobre negociação de serviços, diferenças ontológicas sob o ponto de vista da composição ou reconfiguração de serviços.

Outros trabalhos relevantes em composição de serviços incluem (ANDRADE, 2006), (MILANOVIC; MALEK, 2004), (PATIL, 2004). Embora os *composers* representem módulos indispensáveis na provisão de funcionalidade mais complexa, a própria tarefa de composição depende de atividades intermediárias de mediação. Em nível mais elementar, existe a necessidade de mediação no caso de diferenças terminológicas na descrição de modelos de processos de serviços. Em nível de negociação de serviços, falhas em serviços elementares de uma composição podem implicar a reconfiguração de serviços ou renegociação de contratos. Neste ponto, a necessidade de uma abordagem focada na mediação se torna ainda mais proeminente.

## 2.6 CONSIDERAÇÕES FINAIS

As abordagens mencionadas para o desenvolvimento de *Web services* semânticos apresentam semelhanças e diferenças que precisam ser consideradas. As diferenças apenas

confirmam o fato de que o estabelecimento de um padrão único neste sentido é uma alternativa inviável. Entretanto, é justamente nas semelhanças que um modelo centrado na mediação pode minimizar divergências.

Com relação às ontologias de descrição de serviços, WSMO e OWL-S fornecem as ontologias de base para os *frameworks* IRS-III e SWSF, respectivamente. O aprimoramento destes *frameworks* ocorreu, em grande parte, devido às experiências obtidas mediante o uso de OWL-S e WSMO em estudos de caso. Ambas as ontologias permitem a descrição de aspectos funcionais de serviços (perfil do serviço), modelos para montagem de *workflows* (orquestração ou modelo de serviços) e interoperabilidade com padrões de interfaces de especificação de protocolos, como WSDL-S, que acrescenta semântica às descrições em WSDL (vide Figura 2.10).

Em relação a mecanismos de mediação em nível de dados e processos, *matchmakers* permitem automatização no processo de descoberta de serviços, enquanto que os *composers* fornecem suporte à tarefa de composição. Vale destacar que ambos operam sob os fundamentos de uma linguagem em detrimento de outra. Além disso, são mecanismos estáticos de mediação, no sentido de que não podem ser selecionados ou trocados por clientes ou provedores em tempo de execução.

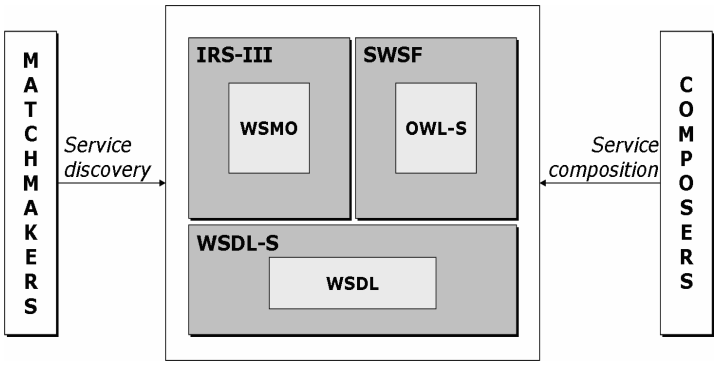



**Figura 2.10 – Relacionamento entre WSDL-S, WSMO e OWL-S**

Apesar dos trabalhos pesquisados representarem um avanço significativo na automatização de tarefas de serviços em SWSA, alguns aspectos ainda carecem de melhorias no que concerne à tarefa de mediação, a qual precede uma negociação de serviço e estabelecimento de contrato.

O primeiro aspecto refere-se à ausência de uma abordagem orientada a serviços para a própria tarefa de mediação. Um mediador representa, teoricamente, um tipo especial de serviço, que também pode ser descoberto ou selecionado, dinamicamente, por clientes e provedores envolvidos em uma negociação. O uso de *matchmaker* sobrecarrega a estrutura

interna de um registro. Semelhantemente, composers, sobrecarregam as aplicações na sincronização dos serviços compostos. A implementação da tarefa de mediação por *Web services* pode representar uma solução de fraco acoplamento e que permita maior liberdade de escolha por clientes e provedores.

O segundo aspecto refere-se à ausência de especificações precisas sobre a tarefa de mediação. A confiabilidade é um importante requisito na tarefa de mediação, ou seja, um mediador precisa dispor de regras bem definidas e de técnicas para manipulação destas regras, de forma a garantir resultados com o mínimo de perdas ou ambigüidade.

É precisamente neste sentido que está direcionada a principal contribuição deste trabalho de dissertação de mestrado: um modelo que inclui um elemento mediador como a quarta entidade de primeiro nível em SWSA.

O modelo proposto, denominado PERSONÆ[16], está baseada em um modelo conceitual que trata explicitamente os aspectos funcionais de uma entidade mediadora para conflitos terminológicos entre provedores e clientes. Este modelo está fundamentado na premissa de que as tarefas de descoberta, seleção e estabelecimento de contratos de serviços partem do pressuposto do estabelecimento de um consenso sobre a terminologia utilizada na descrição destes serviços. Portanto, PERSONÆ consiste no fornecimento de um modelo formal para mediação semântica em SWSA, com base em reconciliação ontológica.

No capítulo 3, detalha-se o modelo de reconciliação ontológica e o modelo proposto, ora documentado.

[16] O nome do modelo provém do termo original em latim, que quer dizer "relacionado ou centrado na pessoa". A personalização, conforme indicada por este trabalho, denota o aspecto de liberdade na escolha do mediador, por parte de clientes, nas tarefas de descoberta e seleção de serviços de um dado provedor.

# CAPÍTULO 3

# O MODELO PERSONÆ

*"Os conceitos mais simples são os mais abstratos."*
[Friedrich Wilhelm Ostwald]

*"A simplicidade é o último grau de sofisticação."*
[Leonardo da Vinci]

*PERSONÆ é um modelo definido no contexto da provisão de serviços personalizados, cuja ênfase é a tarefa de mediação de vocabulários com base no processo de reconciliação ontológica. As tarefas cobertas pelo modelo incluem a descoberta e seleção de serviços, com auxílio da mediação.*

*Este capítulo está dedicado ao detalhamento do modelo PERSONÆ. Inicialmente, é detalhado um modelo conceitual para tratamento da reconciliação ontológica, sobre o qual está baseado o modelo. Uma visão da estrutura básica do modelo é introduzida, seguida pelo detalhamento dos componentes principais de PERSONÆ. Este detalhamento fornece uma visão dos relacionamentos entre os componentes e da funcionalidade do elemento mediador no modelo. Por fim, é detalhado como a mediação está associada com as tarefas de descoberta, seleção e estabelecimento de contrato de serviços.*

## 3.1 DEFINIÇÃO DO MODELO CONCEITUAL

O foco do modelo proposto, denominado PERSONÆ, é a tarefa de mediação semântica. Conforme mencionado anteriormente, dois principais tipos de conflitos surgem durante o processo de interação entre clientes e provedores em SWSA. O primeiro problema consiste na diferença de vocabulários utilizados na descrição de serviços e na descrição de requisitos do usuário (sejam funcionais ou não-funcionais), que pode dificultar a tarefa de descoberta de serviços. O segundo problema consiste nas disputas envolvidas no processo de negociação de qualidade de serviço (envolvendo requisitos não-funcionais), afetando diretamente os processos de seleção e negociação de serviços. O modelo PERSONÆ representa uma contribuição no primeiro aspecto.

Por mediação semântica em SWSA, entenda-se a verificação de similaridades e diferenças que podem existir entre diferentes representações para descrição de serviços. A realização desta tarefa por um serviço de mediação parte do pressuposto da existência de um modelo que determine explicitamente seu comportamento e funcionalidade. A

determinação de qual tipo de mediador utilizar ocorre em função de três fatores principais: (1) das preferências do usuário; (2) do tipo de tarefa solicitada (e.g. descoberta ou seleção); e (3) do tipo de técnica de reconciliação ontológica mais adequada para cada tarefa.

O primeiro aspecto será discutido na próxima seção, o qual representa o fator de personalização do modelo. O segundo aspecto consiste na combinação das técnicas existentes para interoperabilidade de ontologias: fusão, integração e alinhamento. Conforme mencionado no capítulo anterior, a aplicação de cada uma dessas técnicas ocorre em função do contexto. O terceiro aspecto consiste na aplicação dessas técnicas, na execução de tarefas mais complexas, consistindo na reconciliação ontológica. O modelo PERSONÆ associa a reconciliação ontológica a três tarefas principais: descoberta, seleção e estabelecimento de contratos de serviços.

Na descoberta de serviços, o objetivo é a associação de requisitos funcionais de usuários e provedores de um determinado serviço. Uma descoberta semântica consiste não somente na associação direta de parâmetros, mas também na descoberta de similaridades implícitas entre os conceitos relacionados aos requisitos, os quais podem estar representados em diferentes ontologias. Neste caso, a aplicação da técnica de alinhamento pode ser mais adequada.

No caso da multiplicidade da oferta, a seleção se respalda na especificidade da procura (BALKE; WAGNER, 2003), (MAAMAR; MOSTEFAOUI; MAHMOUD, 2005). Neste caso, requisitos não-funcionais representam a base para a seleção dos serviços que melhor atendem às necessidades dos usuários. Alguns trabalhos propostos por Ribeiro et al. (RIBEIRO; ROSA; CUNHA, 2004a), (RIBEIRO; ROSA; CUNHA, 2006) indicam que requisitos não-funcionais e propriedades de correlação e níveis de prioridades podem ser especificados mediante o uso de ontologias. Desta forma, uma seleção semântica consistiria na verificação de requisitos funcionais em termos de qualidade desejada (pelo usuário) e qualidade oferecida (pelo provedor). A técnica de alinhamento de ontologias representa uma boa alternativa ainda neste caso.

Com relação ao estabelecimento de contrato de serviço ou SLA (*Service Level Agreements*), vários modelos de representação têm sido propostos neste sentido (BELLUCCI; ZELEZNIKOW, 2001), (RIBEIRO; ROSA; CUNHA, 2004b), (RIBEIRO; ROSA; CUNHA, 2004c), (TUNG; LIN, 2005). O uso de ontologias para representação de SLA também mostra que o estabelecimento de um padrão neste sentido não é uma abordagem realista. Neste caso, as técnicas de fusão ou integração podem ser utilizadas na formação de um modelo de contrato mais genérico, mediante aglutinação de representações utilizadas pelo cliente e pelo provedor.

Além da especificação técnicas de reconciliação ontológica, o modelo proposto prevê a integração destes mecanismos nos processos de descoberta, seleção e estabelecimento de contrato de serviço, partindo de uma configuração inicial de requisitos funcionais e não-funcionais desejados pelo usuário.

Esta integração é especificada pelo modelo, denominado PERSONÆ, que visa orientar o processo de desenvolvimento de serviços e aplicações, que estão diretamente relacionados à provisão de serviços personalizados na Web, com auxílio da tarefa de mediação apoiada sobre a reconciliação de ontologias.

## 3.2 ESTRUTURA BÁSICA DO MODELO

A arquitetura SOA utiliza o modelo *find-bind-execute* (BOOTH, 2004), (vide Figura 3.1). Primeiramente, provedores publicam seus serviços em um registro. Este registro é então utilizado por clientes, que buscam por serviços que atendam a determinados critérios. Se o registro contém a descrição de tal serviço, este provê informações sobre o provedor deste serviço, formas de acesso e, em alguns casos, informações sobre contratos. Como discutido no capítulo anterior, no modelo SWSA, os registros podem ser enriquecidos com descrições de serviços em OWL-S, facilitando a tarefa de descoberta por meio de *matchmakers*. No caso de atendimento parcial da funcionalidade desejada, por parte de um único serviço, *composers* podem ser utilizados no lado do cliente para a composição de serviços.

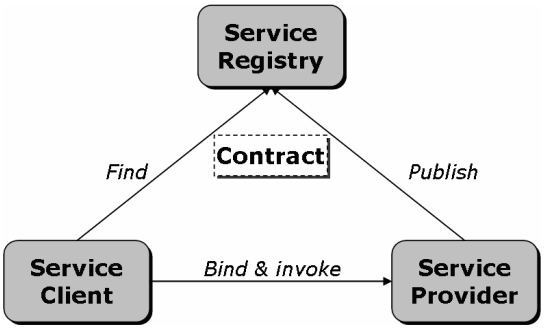



**Figura 3.1 – Modelo SOA (BOOTH ET AL, 2004)**

PERSONÆ estende o modelo SOA (vide Figura 3.2). Desta forma os elementos originais de SOA são preservados, com o acréscimo do elemento mediador. Obviamente, a inserção deste novo elemento implica na modelagem de novos relacionamentos, processos e funções, que redefinem as tarefas relacionadas a serviços, sob a perspectiva da mediação (SILVA ET AL, 2006), (SILVA ET AL, 2007).

Em PERSONÆ, o termo "mediador" abstrai vários processos intermediários de mediação. Portanto, não se trata de apenas um único serviço de mediação, mas, possivelmente, uma composição de mediadores, de acordo com a tarefa (seja descoberta ou seleção). Neste caso, o objetivo é o desacoplamento da tarefa de *matchmaking* do registro, mediante a transferência desta funcionalidade para um serviço de mediação. A próxima seção fornece um maior detalhamento dos elementos que compõem o modelo.

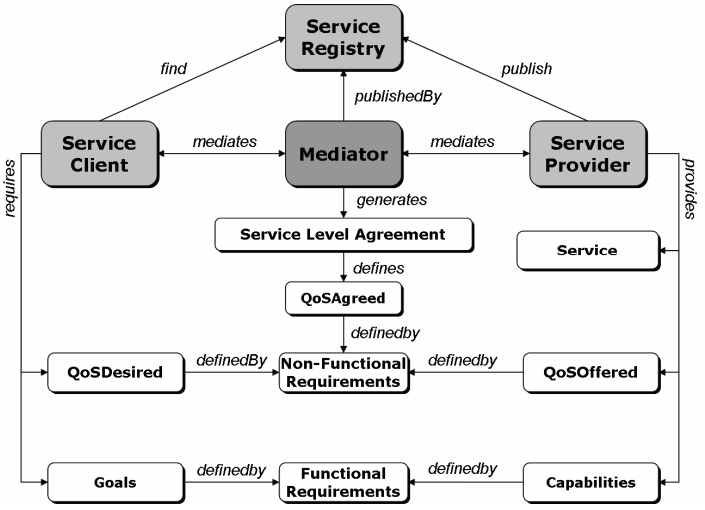



**Figura 3.2 – Modelo PERSONÆ (Visão Geral)**

## 3.3 ELEMENTOS

PERSONÆ é estruturada segundo um modelo de entidades e relacionamentos. Os relacionamentos permitem a caracterização dos elementos ativos (e.g. *Service Client*) e conceitos abstratos (e.g. *Goals*). A associação dos conceitos abstratos com descrições ontológicas foi suprimida do modelo mostrado na Figura 3.2**,** por motivos de simplificação. Os relacionamentos ainda permitem a visualização de quais elementos são envolvidos em cada tarefa do modelo. As subseções seguintes detalham como o conceito de ontologia está associado com as entidades e com os relacionamentos do modelo.

### 3.3.1 ENTIDADES

As entidades de primeiro nível no modelo incluem: (1) clientes (*Service Client*), (2) provedores (*Service Provider*), (3) registro (*Service Registry*) e (4) mediador (*Mediator*).

A entidade *Service Client* é caracterizada por dois conceitos mais abstratos: *goals* e *QoSDesired*. O primeiro conceito refere-se ao requisito funcional desejado (*functional requirements*) e são tipicamente considerados na tarefa de descoberta de serviços. O segundo conceito refere-se às propriedades que qualificam um determinado serviço, ou seja, requisitos não-funcionais (*non-functional requirements*). Estes requisitos estabelecem filtros a serem aplicados no caso de descoberta de múltiplos serviços que atendam à mesma funcionalidade (*goal*). O conceito de *QoSDesired* se refere a outros conceitos mais específicos: (1) *QoSConstraints*, (2) *QoSPreferences* e (3) *QoS Priorities* (vide Figura 3.3).

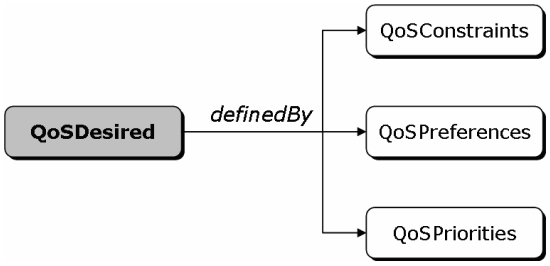



**Figura 3.3 – Elementos de classificação de QoS Desired**

O primeiro filtro seletivo (*QoSConstraints*) especifica "o que exclui" determinado serviço, segundo o nível de qualidade desejada por um usuário. Por exemplo, um determinado usuário pode impor determinadas condições de tempo de execução de um serviço. Neste caso, este critério de seleção exclui *a priori* serviços que interessam ao cliente, restringindo os parâmetros de busca e tornando mais específica a tarefa de descoberta. O segundo filtro seletivo (*QoSPreferences*) especifica "o que distingue" determinado serviço. Por exemplo, um cliente de serviços de hotelaria pode ter preferências específicas sobre tipo de hospedagem e tipo de entretenimento. O segundo filtro seletivo (*QoSPriorities*) especifica "o que é mais importante" entre dois requisitos que não podem ser fornecidos simultaneamente. Em ESCHER (RIBEIRO, 2004), (RIBEIRO; ROSA; CUNHA, 2004b), as propriedades de correlação (convergências e conflitos) e prioridades são estabelecidas entre requisitos não-funcionais com vistas a estabelecer as possibilidades de um processo de negociação. Requisitos como custo e desempenho exemplificam uma correlação de conflito, onde um é fornecido em detrimento do outro.

Vale destacar que a noção de requisitos não-funcionais é considerada como equivalente a requisitos de QoS neste trabalho. As restrições impostas pelos requisitos de

QoS podem ser relacionadas ao processo de desenvolvimento ou podem ser restrições a serem consideradas durante o tempo de execução (KOTONYA; SOMMERVILLE, 1998). No contexto do trabalho, somente são considerados os requisitos não-funcionais perceptíveis em tempo de execução do sistema. O tratamento de requisitos não-funcionais relacionados ao processo de desenvolvimento do software, como manutenibilidade, está fora do escopo do modelo PERSONÆ. Exemplos de requisitos de QoS considerados neste trabalho e específicos para a caracterização de serviços incluem a classificação de propriedades formais de requisitos não-funcionais proposta por O'Sulivan (O'SULLIVAN; EDMOND; HOFSTEDE, 2005).

A entidade *Service Provider* disponibiliza os serviços a serem publicados (*Service*). Um serviço é caracterizado por dois conceitos mais abstratos: *capabilities* e *QoSOffered*. O primeiro conceito refere-se a requisitos funcionais (*functional requirements*) e são associados com os requisitos funcionais do cliente (*goals*) na tarefa de descoberta de serviços. O segundo conceito refere-se ao diferencial do provedor, traduzido em termos da qualidade oferecida. Este conceito é associado com os requisitos de qualidade do cliente (*non-functional requirements*) na tarefa de seleção de serviços.

A entidade de registro de serviços (*Service Registry*) representa um apontador para descrições dos serviços. Estas descrições são representadas por ontologias de serviços, referenciadas por identificadores universais de recursos (URIs) na Web. Esta abordagem simplifica o projeto de registros, dado que estes representam meramente o papel de páginas amarelas ou classificados, ou seja, devem conter o mínimo de informação necessária apenas para tarefas de localização de um determinado provedor.

Finalmente, a entidade mediadora (*mediator*) representa o ponto de apoio entre clientes e provedores no modelo. Um mediador é um tipo especial de serviço, o qual também pode ser publicado em um determinado registro. Isto significa que um mediador também é passível de caracterização em termos de requisitos funcionais (e.g. mediador para descoberta de serviços, tipo de técnica de reconciliação empregada) e requisitos não-funcionais (e.g. confiabilidade, imparcialidade, custo de mediação). Vale salientar que os objetivos do mediador incluem, além da reconciliação de vocabulários, a geração de um contrato de serviço (SLA – *Service Level Agreement*), estabelecido com base no nível de qualidade acordada (*QoSAgreed*). Este último conceito refere-se à verificação de intersecções entre a qualidade desejada pelo cliente (*QoSConstraints, QoSPreferences* e *QoSPriorities*) e a qualidade oferecida pelo provedor (*QoSOffered*).

### 3.3.2 RELACIONAMENTOS

O modelo PERSONÆ define dois tipos de relacionamentos: relacionamentos entre entidades e relacionamentos entre processos. O primeiro tipo inclui relações como *find, publish* (típicas de SOA), *mediates* (entre mediadores, clientes e provedores), *generates* (entre mediadores e SLAs), *definedBy* (entre *goals/capabilities*, *QoSDesired/QoSOffered* e *functional* e *non-functional requirements*, respectivamente). Entretanto, relacionamentos sobre estes conjuntos de conceitos devem ser formalmente estabelecidos mediante o uso de alguma estrutura de dados, que estabeleça relações entre estes e permitindo raciocínio sobre propriedades não explicitamente declaradas. Neste sentido, os elementos abstratos definidos em PERSONÆ são descritos por ontologias, que expressam o vocabulário de base para a mediação semântica nas tarefas do modelo (vide Figura 3.4).

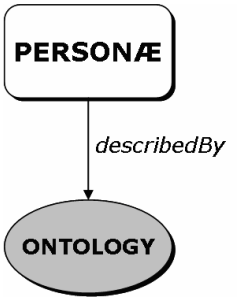



**Figura 3.4 – Abstração na descrição de elementos de PERSONÆ**

A anatomia de uma ontologia inclui cinco elementos: classes, propriedades, indivíduos, axiomas e restrições (vide Figura 3.5) As classes assemelham-se à noção de conjuntos, especificando a gênese de um determinado conceito. As propriedades podem ser classificadas em dois tipos: (1) *Object Properties*, as quais estabelecem funções ou relações entre as classes (de acordo com os conjuntos de classes domínio e imagem); e (2) *Data Type Properties*, as quais especificam a relação de uma classe com algum tipo de dado determinado por alguma especificação de tipos nativos (e.g. *XMLSchema*, para especificação de ontologias com linguagens da Web semântica).

Os indivíduos (*individuals*) representam as instâncias de uma determinada classe e, desta forma, qualquer propriedade incidente sobre uma classe modifica a caracterização de suas instâncias, de forma semelhante ao paradigma orientado a objetos. Os axiomas representam um conjunto de premissas que estabelecem relações entre classes, propriedades e indivíduos. Por exemplo, os axiomas `subClassOf` e `disjointWith` da linguagem OWL estabelecem postulados de relações entre classes; os axiomas `sameAs` e

`differentFrom` postulam relações entre indivíduos, enquanto que `subPropertyOf` e `equivalentProperty` o fazem para as propriedades. Por fim, um conjunto de restrições refina a caracterização das propriedades quanto à cardinalidade e quantificação universal e existencial (e.g. `maxCardinality`, `minCardinality`, `exists` e `forall`, em OWL, respectivamente).

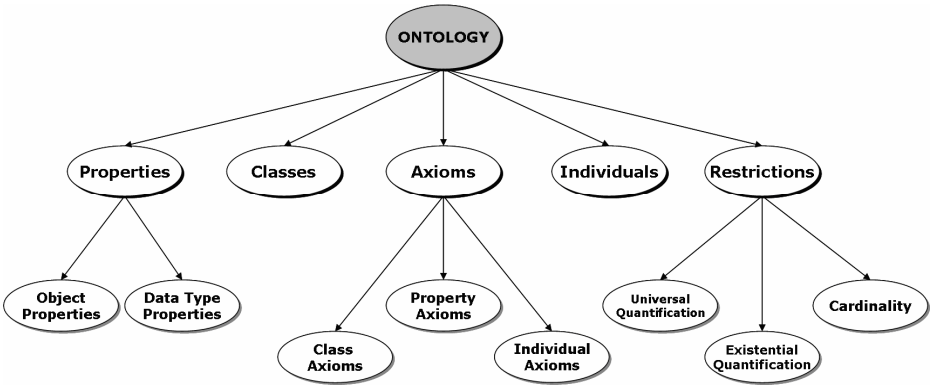



**Figura 3.5 – Elementos estruturais de uma ontologia**

Vale destacar que as ontologias admitem herança múltipla, em nível de classes e de propriedades. A determinação de similaridades, intersecções ou diferenças são explicitadas pelas propriedades, axiomas e restrições. Quanto maior o número de propriedades e restrições que incidem sobre uma classe, maior será a especificidade do conceito. Segundo Gómez-Pérez (GÓMEZ-PEREZ; FERNÁNDES-LÓPEZ; CORCHO, 2004), várias metodologias podem ser utilizadas na especificação, validação e atualização de ontologias. O diferencial no uso deste tipo de estrutura de dados consiste na possibilidade de automatização do processo de inferência de novos conceitos. Este processo envolve a construção de módulos de software denominados motores de inferência, ou *reasoners*. Além disso, *reasoners* também podem ser invocados remotamente por serviços de mediação, para realização de inferências em tempo de execução e verificação de consistência de ontologias submetidas a processos de integração, fusão ou alinhamento.

Considerando a premissa de que os elementos em PERSONÆ são descritos por ontologias, e que as diferenças são contornadas no processo de reconciliação, existe certa liberdade na definição dos conceitos. Por exemplo, o perfil de um determinado cliente pode ser representado por uma ontologia pessoal, (ou *personal ontology*)(HUHNS; STEPHENS, 1999). Isto pode permitir que um determinado provedor possa verificar *a priori* que tipo de cliente ele pretende atender e principalmente, como atendê-lo. O mesmo pode ocorrer em relação ao provedor, onde o cliente pode realizar buscas com base características específicas

do provedor e não do serviço. A fidelização consiste justamente em um estágio mais avançado de personalização, onde clientes e provedores estabelecem relações de prioridade reciprocamente.

Em relação à descrição de requisitos funcionais ou não-funcionais, vários modelos têm sido propostos (BIANCHINI; ANTONELLIS; MELCHIORI, 2004), (BALKE; WAGNER, 2004), (O'SULLIVAN; EDMOND; HOFSTEDE, 2005). Em Ribeiro et al. (RIBEIRO; ROSA; CUNHA, 2006) é proposta uma ontologia de QoS para descrição de requisitos não-funcionais relacionados a serviços, como é mostrado na Figura 3.6. Desta forma, propriedades tais como correlação ou prioridades entre requisitos podem ser facilmente descritas na forma de propriedades nas ontologias, permitindo inferências no processo de negociação de serviços por parte de serviços de mediação.


**Figura 3.6 – Ontologia de QoS para definição de requisitos não-funcionais (RIBEIRO; ROSA; CUNHA, 2006)**

O segundo tipo de relacionamento em PERSONÆ consiste na associação de processos mais complexos. Neste sentido, a mediação ocupa um papel central no auxílio de tarefas como descoberta, seleção e estabelecimento de contratos de serviço. Conforme mencionado no capítulo anterior, uma mediação semântica em SOA pode ocorrer em dois momentos: (1) na reconciliação de vocabulários e (2) na negociação de serviços. PERSONÆ está focada no primeiro momento. Portanto, a tarefa de mediação é baseada na reconciliação ontológica. Esta última representa uma combinação dos processos de integração, fusão e alinhamento (vide Figura 3.7).

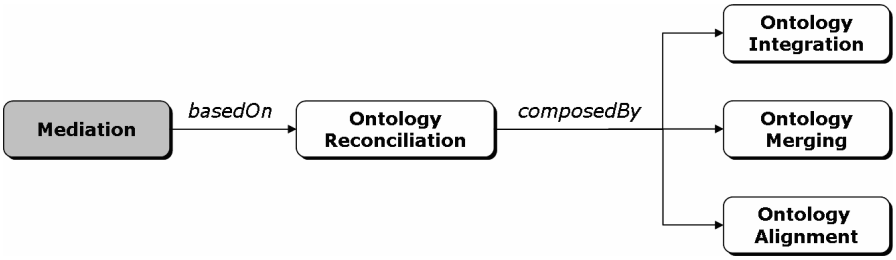



**Figura 3.7 – Fluxo de processos de reconciliação ontológica**

Em PERSONÆ também são especificadas as relações entre o processo de mediação e os processos de descoberta, seleção e estabelecimento de contratos (vide Figura 3.8). Por exemplo, como o objetivo na tarefa de descoberta é encontrar o maior número possível de serviços candidatos, a técnica de fusão pode se utilizada. No caso da detecção de ontologias importadas, a fusão pode retornar melhores resultados. Na tarefa de seleção, como o objetivo é o refinamento das intersecções encontradas no processo de descoberta, a técnica de alinhamento pode ser usada para que o cliente identifique somente os mapeamentos convergentes com seus interesses. Na tarefa de estabelecimento de contratos, como o objetivo é a agregação de interesses do cliente e do provedor, a integração pode ser a abordagem mais simples, mediante a composição de uma ontologia-núcleo (*core ontology*) para representação de um SLA.

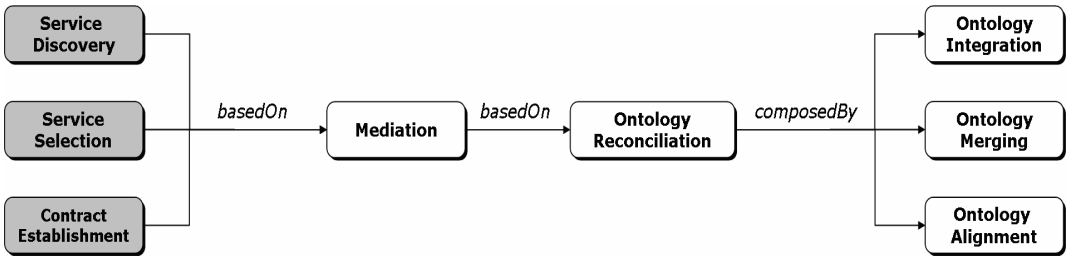



**Figura 3.8 – Relacionamento entre tarefas de serviços e mecanismos de reconciliação de ontologias**

A próxima seção trata da associação da tarefa de mediação com os processos de descoberta, seleção e estabelecimento de contrato de serviço. Melhor detalhamento é fornecido quanto às razões que motivam a aplicação de cada método de reconciliação ontológica com a tarefa de mediação, de acordo com o contexto.

## 3.4 ATIVIDADES SOBRE SERVIÇOS

O ciclo de básico de provisão de serviços inclui as tarefas de publicação, descoberta, seleção, composição, negociação, estabelecimento de contratos, monitoramento e reconfiguração dinâmica de serviços. Neste trabalho, é descrito como a tarefa de mediação

pode ser aplicada especificamente na descoberta, seleção e estabelecimento de contratos. O objetivo de cada tarefa determina a técnica de reconciliação utilizada.

### 3.4.1 MEDIAÇÃO E DESCOBERTA DE SERVIÇOS

Em geral, a descoberta de serviços consiste na associação dos requisitos funcionais do cliente (*goals*) com a funcionalidade oferecida pelo serviço (*capabilities*). A semântica destes conceitos pode ser representada como um conjunto de classes em uma ontologia para requisitos funcionais (RIBEIRO; ROSA; CUNHA, 2006) , (vide Figura 3.9). O mediador verifica as similaridades entre *goals* e *capabilities* em nível conceitual. A verificação dessas similaridades pressupõe que estes elementos estejam relacionados de alguma forma. Precisamente, algumas relações entre estes conceitos devem ser determinadas. Os relacionamentos mais básicos incluem os seguintes níveis de associação: (1) *exact match* (e.g. *goal* = *capability*); (2) *subsumes match* (e.g. *goal* ⊂ *capability*); (3) *plugin match* (e.g. *capability* ⊂ goal); (4) *intersection match* (e.g. *goal* ∩ *capability* ≠ ∅); e (5) *impasse* ou *non-match* (e.g. *goal* ∩ *capability* = ∅).

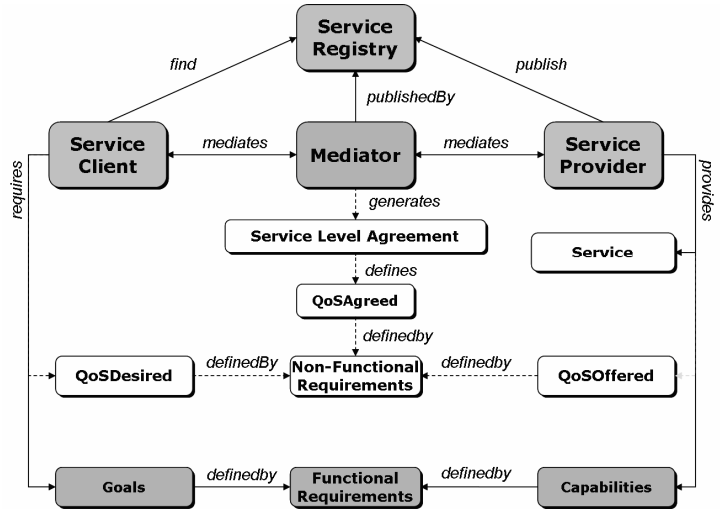



**Figura 3.9 – Modelo PERSONÆ e a perspectiva de descoberta de serviços**

Estas relações baseadas em conjuntos provêm a base para a formalização de um modelo intuitivo para associação entre *goals* e *capabilities* de *Web services* no mundo real. Estas relações são comuns em outros trabalhos que tratam sobre descoberta de serviços (MANDELL; MCILLRAITH, 2003), (PAOLUCCI, 2002b), (STOLLBERG; KELLER; FENSEL, 2005), (vide Figura 3.10). A situação ideal é quando ocorre um *exact match* entre requisitos funcionais do usuário e do provedor. Quando a funcionalidade oferecida forma

um superconjunto dos requisitos desejados pelo cliente, ocorre um *subsumes match*. Nesta situação, o *Web service* atende completamente à funcionalidade desejada, entretanto, pode deliberar resultados que não interessam ao cliente. O *plugin match* corresponde à situação contrária, e neste caso, o serviço atende somente a uma parte da funcionalidade desejada, justificando uma composição de serviços de forma que a funcionalidade desejada seja completa. Quando um *intersection match* ocorre, uma composição de serviços também é necessária. Caso não exista nenhum tipo de intersecção entre os conceitos, ocorre um *impasse* e nenhum mapeamento é possível neste caso.

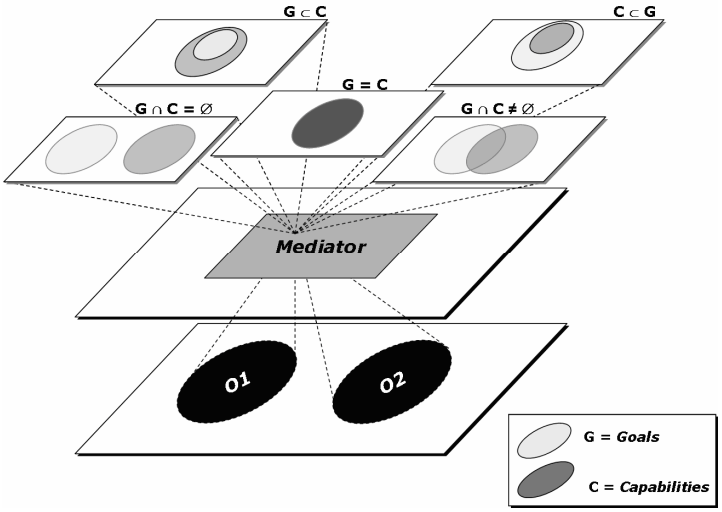



**Figura 3.10 – Relacionamentos entre classes de requisitos funcionais**

No caso da descoberta de serviços, o objetivo é alcançar o maior número de intersecções entre conceitos e propriedades das ontologias de descrição de serviços. A técnica de fusão é adequada a este propósito. Quanto maior for o número de similaridades, mais serviços poderão atender à funcionalidade desejada. A técnica de fusão é completamente automática, ou seja, as similaridades são verificadas sem o fornecimento de um relacionamento inicial entre conceitos das ontologias de origem. Portanto, a técnica de fusão pressupõe a existência de um ancestral em comum nas ontologias de origem, ou seja, uma ontologia importada (BENATALLAH, 2003), (BENBERNOU; HACID, 2005), (COLUCCI, 2004), (SHETH; THACKER; PATEL, 2003).

Considerando as ontologias de origem $O_1$ e $O_2$, a ontologia importada $O_3$, o conjunto de relacionamentos propagados a partir do relacionamento de classes, propriedades e indivíduos da ontologia $O_3$ em $O_1$ e $O_2$, denominados $P1_{propagation}$ e $P2_{propagation}$, respectivamente, então a ontologia $O_4$ gerada pela aplicação da fusão será igual a:

(1) $O_3$, se e somente se $O_1 = O_3$ e $O_2 = O_3$ forem verdadeiros;

(2) $O_1$, se e somente se $O_3 \subseteq O_1$ e $O_2 = O_3$ forem verdadeiros;

(3) $O_2$, se e somente se $O_1 = O_3$ e $O_3 \subseteq O_2$ forem verdadeiros;

(4) $O_3 \cup (P1_{propagation} \cap P2_{propagation})$, se e somente se $O_3 \subset O_1$ e $O_3 \subset O_2$ forem verdadeiros.

A Figura 3.11 ilustra estes casos (suprimido o caso 2, que é comutativo com o caso 3).

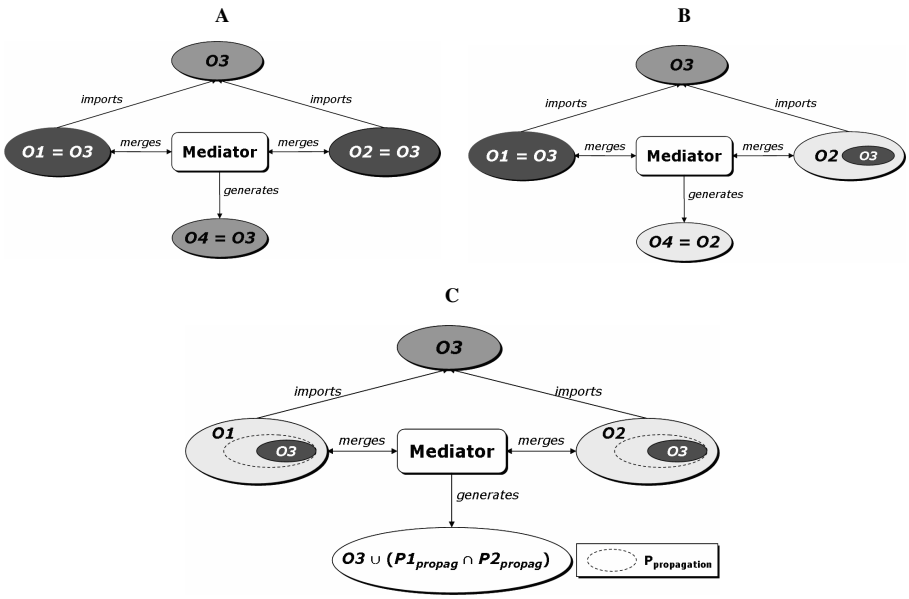



**Figura 3.11 – Fusão de ontologias em importação do tipo: A) total em ambas as ontologias; B) total em uma ontologia e parcial na outra; C) parcial em ambas as ontologias.**

No primeiro caso, em que a importação de $O_3$ é total em $O_1$ e $O_2$, a ontologia gerada $O_4$ é equivalente à ontologia $O_3$. Considerando que a fusão difere da integração pelo fato de que a fusão de conceitos elimina a duplicação ou redundância, neste caso a fusão é total, sem perdas. No segundo caso (o terceiro caso é semelhante, diferindo apenas na inversão de domínio e imagem da operação de fusão), uma das ontologias importa $O_3$ completamente, enquanto a outra tem importação parcial. Como $O_3$ é um subconjunto próprio de $O_2$, o resultado agrega $O_3$ e a extensão de $O_2$. No quarto e último caso, tem-se que as duas ontologias de origem têm importação parcial de $O_3$. A propagação de relações entre conceitos e propriedades de $O_3$ nas ontologias $O_1$ e $O_2$ gera novas ramificações semânticas, as quais também podem apresentar diferenças e semelhanças. O conjunto de conceitos e propriedades de $O_1$ e $O_2$ os quais estão conectados de alguma forma com o núcleo de conceitos importados, foi denominado no contexto deste trabalho como $P1_{propagation}$ e $P2_{propagation}$, respectivamente. Portanto, a ontologia final $O_4$ será formada pelo núcleo

comum $O_3$ e pela intersecção das redes semânticas propagadas. Este é o caso de maior complexidade na operação de fusão de ontologias.

A formalização das precondições envolvidas na análise de semelhanças entre as estruturas ontológicas é fornecida no capítulo 4.

### 3.4.2 MEDIAÇÃO E SELEÇÃO DE SERVIÇOS

A seleção de serviços refina os resultados produzidos na fase de descoberta. Ou seja, diante da multiplicidade da oferta, um serviço personalizado deve atender à especificidade da procura (MAAMAR; MOSTEFAOUI; MAHMOUD, 2005). Quando vários serviços atendem à funcionalidade desejada, a diferenciação acontece em nível de requisitos de QoS. Ontologias para descrição de requisitos não-funcionais podem ser utilizadas para extensão de modelos atuais que descrevem o perfil de um serviço. Portanto, também é possível a realização de inferências automáticas no processo de seleção, desde que o conhecimento sobre níveis de qualidade fornecidos esteja especificado em uma ontologia. Um mediador pode ser utilizado para a associação do nível de qualidade desejada (*QoSDesired*, ou *QoSConstraints, QoSPreferences* e *QoSPriorities*) com o nível de qualidade oferecida por um serviço de um determinado provedor (*QoSOffered*), como é mostrado na Figura 3.12

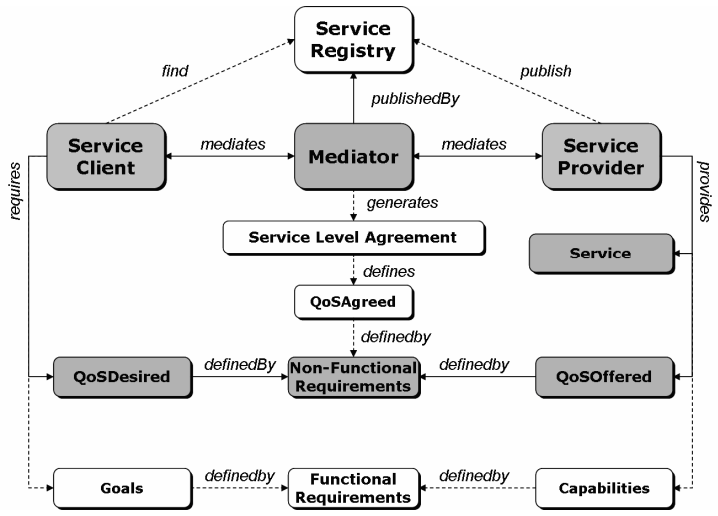



**Figura 3.12 – Modelo PERSONÆ e a perspectiva de seleção de serviços**

O conceito de nível de qualidade de serviço é demasiadamente relativo, constituindo o aspecto mais subjetivo da personalização. O uso de variáveis lingüísticas, tais como "alto" e "baixo", variam de acordo com o nível de exigência de cada usuário. Entretanto, é possível determinar intervalos que determinam o conjunto imagem de uma função que relaciona um

determinado requisito não-funcional com níveis oferecidos. Tais propriedades podem ser modeladas por *datatype properties*, em uma ontologia. Estas propriedades associam um conceito com um tipo de dado comum, definido em algum esquema de tipos na Web. Por exemplo, *XML Schema* fornece um conjunto de tipos (e.g. `string`, `integer`, `float` e `datetime`) que permite a modelagem de intervalos de valores.

A classificação de níveis de qualidade oferecida, tais como "alto" ou "baixo", está associada ao elemento de interação humana, e, portanto, é um aspecto de difícil operacionalização. A instanciação de valores nas ontologias visa somente prover meios de raciocínio por parte do serviço de mediação, cuja função neste caso é facilitar a identificação das similaridades. Contudo, a decisão final sobre a seleção é tarefa do usuário.

Existe uma forte relação entre a tarefa de seleção de serviços e o processo de alinhamento de ontologias. Ambos consistem em processos semi-automáticos, que auxiliam na tomada de decisão final quanto às preferências do usuário. O alinhamento de ontologias não estabelece como premissa básica a existência de ontologias importadas nas ontologias de origem (isto ocorre na operação de fusão). Portanto, não existem relações entre classes, propriedades ou instâncias determinadas a priori entre as diferentes ontologias. Neste sentido, é necessária a intervenção humana na tarefa de inicialização do processo de alinhamento. Esta inicialização consiste basicamente na associação de um conceito ou propriedade de uma ontologia com algum conceito ou propriedade da outra. Esta associação é dada pelo estabelecimento de axiomas de classes ou propriedades (e.g. `equivalentFrom` ou `subClassOf`, para classes, e `subPropertyOf`, `inversePropertyOf`, para propriedades). Os axiomas representam propriedades de ligação que podem ser utilizados dentro de uma mesma ontologia ou como pontes que ligam ontologias diferentes.

Por exemplo, considere-se as ontologias de origem $O_1$ e $O_2$ e as classes de conceitos $C_1$, $C_2$ e $C_3$, tal que $C_1 \subset O_1$, $C_2 \subset O_2$ e $C_3 \subset O_2$ são relações válidas. Considere ainda que uma relação do tipo `equivalentFrom` é inicializada entre $C_1$ e $C_2$. Neste caso, o objetivo é encontrar o maior nível de similaridades possíveis, visto que o axioma `equivalentFrom` determina o mais alto grau de similaridade entre classes. Se $C_1$ é equivalente a $C_2$, e $C_2$ está relacionado com $C_3$ através de outro axioma, novas relações poderão ser estabelecidas entre $C_1$ e $C_3$. O nível de propagação dessas relações varia de acordo com os axiomas usados na inicialização e dos axiomas que ligam os conceitos internos das ontologias alinhadas (e.g. `subClassOf`, `superClassOf`,

`disjointWith`). Na Figura 3.13, é mostrada a propagação do alinhamento de $C_1$, $C_2$ e $C_3$, de acordo com os axiomas de ligação.

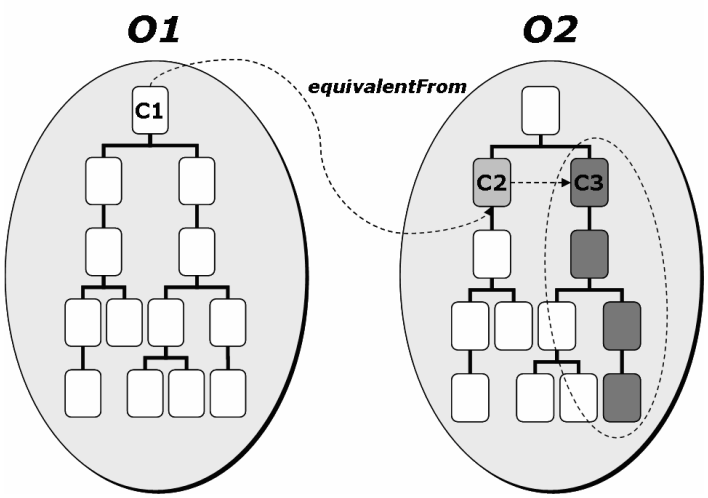



**Figura 3.13 – Alinhamento de ontologias com axioma de inicialização**

Conforme mencionado no capítulo 2, o resultado do alinhamento é um conjunto de mapeamentos entre as ontologias originais, permitindo que o usuário selecione quais os mapeamentos de seu interesse. É nesse ponto que reside a principal razão da aplicabilidade do processo de alinhamento na tarefa de seleção de serviços. O objetivo da seleção é o refinamento das similaridades entre requisitos de QoS desejados e oferecidos (*QoSDesired* e *QoSOffered*). O cliente pode então escolher somente os mapeamentos consoantes com seus interesses.

O alinhamento produz associações em diferentes níveis de similaridade semântica. Por exemplo, os mapeamentos realizados sobre a propriedade `equivalentFrom` retornam conceitos e propriedades mais próximos. Por outro lado, mapeamentos sobre a propriedade `disjointWith` não produzem outro resultado a não ser o impasse. No caso de um alinhamento não retornar resultados de equivalência, em primeira instância seletiva, um relaxamento das restrições pode ser feito em nível de conceitos que tenham relação de parentesco (e.g. `subClassOf` ou `superClassOf`). Desta forma, quanto maior for a especificidade das relações semânticas de interesse, maior refinamento poderá ser aplicado ao processo de seleção de serviços.

Vale salientar que, conforme as premissas estabelecidas no capítulo 1, o modelo PERSONÆ não trata da composição de serviços. Desta forma, o modelo de seleção aqui descrito se aplica ao conjunto de serviços retornados na fase de descoberta, os quais

atendam totalmente à funcionalidade desejada. Considerar serviços compostos implica a modelagem de um mecanismo de seleção que avalie a qualidade da composição por completo, não apenas um serviço em particular.

### 3.4.3 MEDIAÇÃO E ESTABELECIMENTO DE CONTRATO DE SERVIÇO

O estabelecimento de um contrato de serviço (SLA) registra um acordo entre clientes e provedores de serviço sobre níveis de qualidade de serviço desejada e oferecida (vide Figura 3.14). Teoricamente um SLA representa um registro formal que regulamenta a provisão de serviços, para fins de monitoramento e renegociação. O estabelecimento deste registro pode incluir ou não um processo de negociação prévia. No caso de negociação prévia, clientes e provedores podem buscar a maximização de ganhos com base no poder de barganha. Isto implica na verificação de propriedades de correlação (conflitos e convergências) e prioridades entre requisitos não-funcionais. Na ausência de conflitos no processo de negociação (concordância pacífica), um SLA pode ser estabelecido após o processo de seleção de serviços.

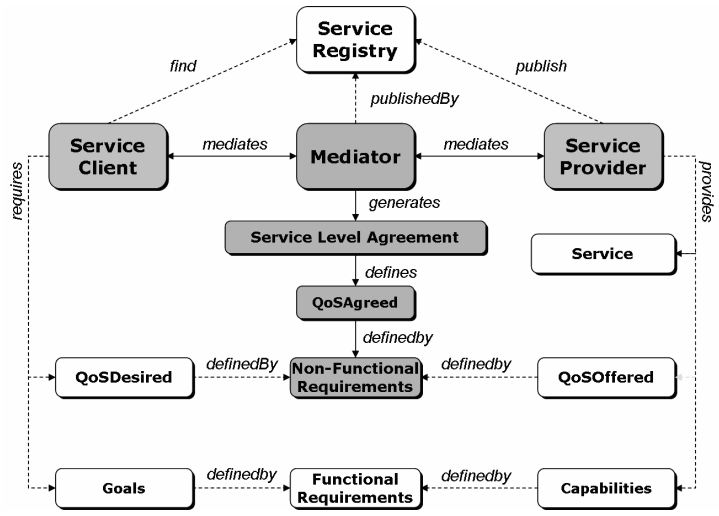



**Figura 3.14 – PERSONÆ e a perspectiva de estabelecimento de SLA**

Diversos modelos têm sido propostos para a representação de SLAs (BELLUCCI; ZELEZNIKOW, 2001), (RIBEIRO, 2004), (TUNG; LIN, 2005). A utilização de ontologias para a representação e instanciação de SLAs também representa um aspecto tratado recentemente em SOA (RIBEIRO; ROSA; CUNHA, 2006). Considerando que um contrato inclui requisitos de QoS, ontologias para modelagem de propriedades de correlações entre requisitos não-funcionais têm sido propostas (BIANCHINI; ANTONELLIS; MELCHIORI,

2004), (RIBEIRO; ROSA; CUNHA, 2006), as quais podem estender ontologias para SLAs, de forma análoga a ontologias de domínio que estendem OWL-S.

Ontologias para requisitos podem ser demasiadamente genéricas, incluindo desde requisitos não-funcionais para desenvolvimento de software até requisitos mais específicos como a classificação proposta por O'Sullivan et al. (O'SULLIVAN; EDMOND; HOFSTEDE, 2005), que apresenta relações formais entre requisitos específicos para Web services. Além disso, o domínio do serviço oferecido por um determinado provedor pode delimitar requisitos ainda mais específicos relacionados com regras de negócios. Obviamente, a determinação de padrões neste sentido também não representa uma abordagem realista.

Por exemplo, em ESCHER (RIBEIRO, 2004) é estabelecido um modelo de SLA ou contrato de QoS, que inclui especificações abstratas e concretas sobre níveis de QoS (RIBEIRO; ROSA; CUNHA, 2004c). As especificações abstratas são definidas pelos requisitos de qualidade do usuário (QoS desejada) e pelas restrições impostas pelo tipo da aplicação (QoS requerida). As especificações concretas dizem respeito aos parâmetros técnicos a partir dos quais um serviço é configurado (QoS acordada), atendido (QoS fornecida) e monitorado (QoS medida). A presença dessas diversas especificações em um mesmo documento torna o SLA um elemento vital na prestação de serviços, uma vez que o usuário pode acompanhar a prestação do serviço através da seção mais abstrata do SLA, e o provedor o faz através da seção mais concreta. A estrutura interna do contrato proposto por esta abordagem pode ser visualizada na Figura 3.15. A estrutura do contrato de QoS (*Contract*) é definida por um tipo (*ContractType*) composto dos requisitos de QoS do usuário (*UserQoSspec*), dos requisitos de QoS da aplicação (*ApplicationQoSspec*) e requisitos mais concretos relacionados ao próprio serviço (*ServiceQoSspec*).

Zhou et al. (ZHOU; CHIA; LEE, 2004) propõem um outro modelo de ontologia que estabelece métricas sobre requisitos de QoS. A ontologia contém três camadas: *QoS profile*, que pode ser utilizada para fins descoberta de requisitos não-funcionais; *QoS property*, para a modelagem de restrições sobre requisitos, que pode ser utilizada para a tarefa de seleção; e *QoS metrics*, que pode ser utilizada para estabelecimento de métricas de qualidade no estabelecimento de contratos e monitoramento de serviços.

Os dois modelos referenciados apresentam aspectos que podem ser integrados na instanciação de uma ontologia para SLA. Outros abordagens têm sido propostas para extensão de modelos de contrato com base em descrições de requisitos de QoS (PAPAIOANNOU, 2006), (LIN, 2005). O aspecto em comum refere-se ao fato de que um

contrato deve explicitar as condições do cliente e do provedor sobre o nível de qualidade acordada (*QoSAgreed*). A forma ou estrutura destas seções pode variar de acordo com as políticas de contrato ou modelo de negócios.

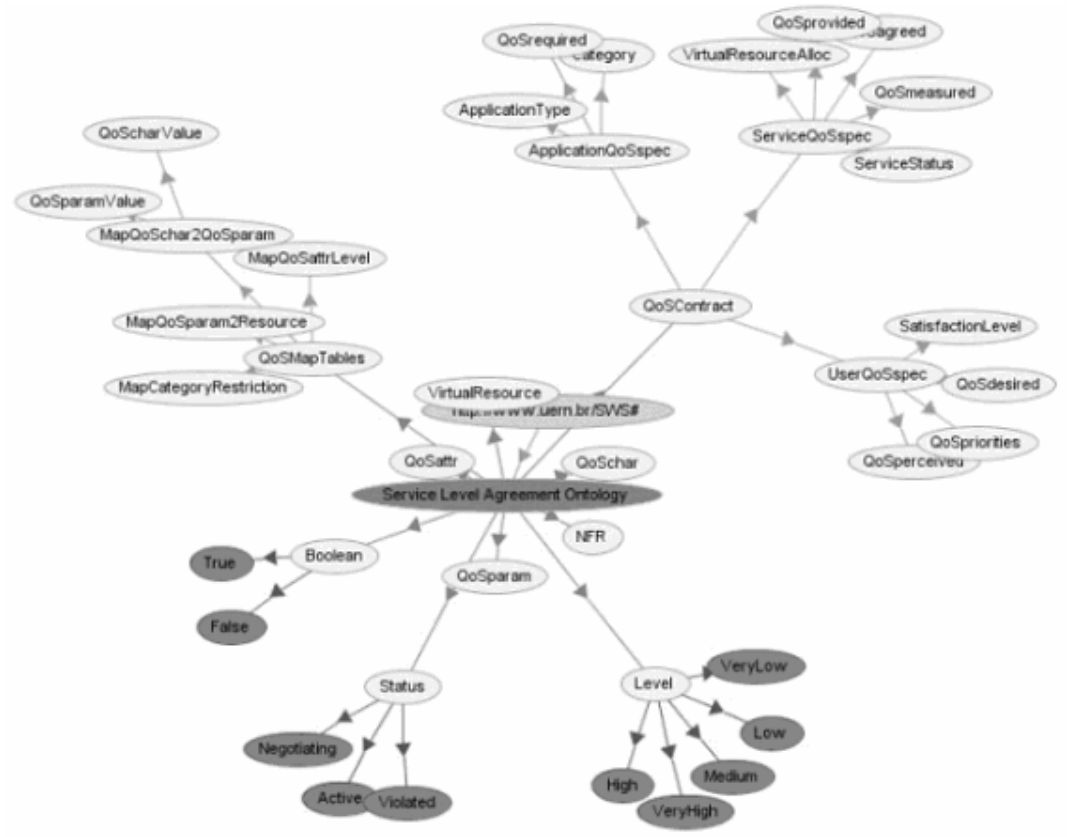



**Figura 3.15 – Estrutura de SLA segundo a abordagem ESCHER (RIBEIRO, 2004)**

A integração de ontologias consiste basicamente no reuso de sub-ontologias na formação de uma ontologia-núcleo (*core ontology*), sem alinhamento ou fusão de conceitos (vide Figura 3.16). É uma abordagem rápida para estabelecimento de consenso. A ontologia OWL-S exemplifica uma ontologia-núcleo integrada, contendo três sub-ontologias que podem ser estendidas por ontologias de QoS ou de domínio. Considerando que uma instância de contrato apenas registra um acordo final, não existe a necessidade de alinhamento de conceitos, visto que tal processo ocorre em tempo de seleção. A integração de ontologias ainda pode permitir a instanciação rápida de SLAs para provisão de serviços compostos, onde múltiplos provedores podem estar envolvidos na provisão de serviços. Entretanto, neste caso é preciso estabelecer consenso sobre a terminologia de QoS e as métricas utilizadas na determinação de níveis de qualidade, garantindo igualdade de privilégios no contrato.

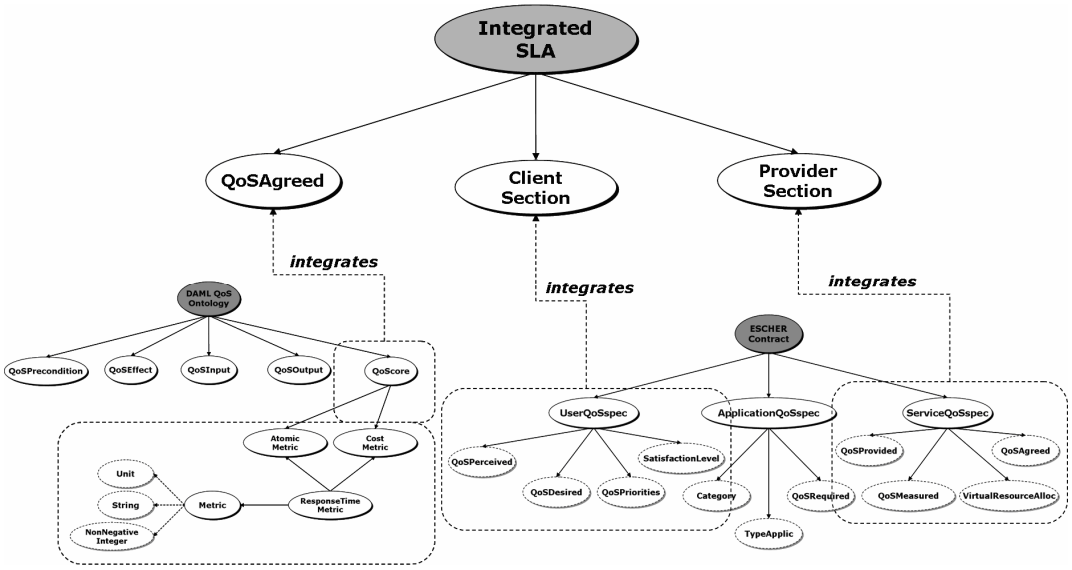



**Figura 3.16 – Integração de representações para SLA**

Vale salientar que, dependendo do nível de especificidade na determinação de requisitos de QoS, a integração de representações na formação de uma instância de ontologia para SLA pode exigir alinhamento prévio sobre as ontologias de QoS utilizadas. Isto evita a ambigüidade na determinação nos requisitos de QoS que compõem a base do contrato.

## 3.5 CONSIDERAÇÕES FINAIS

Neste capítulo, foram descritos os elementos do modelo PERSONÆ. Inicialmente, foram definidas as entidades de primeiro nível do modelo, com ênfase no papel do mediador como entidade facilitadora na aproximação de clientes e provedores de serviços. O papel da mediação baseada em reconciliação ontológica nos processos de descoberta, seleção e estabelecimento de contratos de serviços também foi detalhado. No capítulo 4, detalha-se a especificação formal em notação Z, a qual descreve os relacionamentos entre os elementos e processos de PERSONÆ.

# CAPÍTULO 4

# FORMALIZAÇÃO DO MODELO PERSONÆ

*"Uma verdade matemática não é simples nem complicada por si mesma. É uma verdade."*
[Emile Lemoine]

*Durante a formalização do modelo proposto foi dada ênfase na descrição dos processos de reconciliação ontológica, os quais constituem os fundamentos da mediação semântica em SOA, segundo a perspectiva que orienta este trabalho. Por conseguinte, a mediação semântica representa o fator comum nas tarefas de descoberta, seleção e estabelecimento de contratos de serviços. Os principais objetivos da especificação proposta incluem a disponibilização de um modelo de apoio ao raciocínio sobre a funcionalidade do modelo PERSONÆ, bem como a descrição precisa dos aspectos conceituais propostos.*

*Este capítulo é dedicado ao detalhamento da formalização do modelo PERSONÆ mediante o uso da notação Z. A escolha por este formalismo seguiu alguns critérios, dentre eles a existência de relações de morfismo entre Z e linguagens para definição de ontologias para a Web, como OWL, além do poder de expressividade da notação e a existência de ferramentas de apoio às tarefas de especificação e verificação. As próximas seções estão estruturadas da seguinte forma: inicialmente, são detalhadas algumas decisões tomadas durante o processo de formalização; em seguida, é detalhada a especificação dos elementos que compõem o estado abstrato do modelo incluindo a formalização das operações de descoberta e seleção de serviços.*

## 4.1 CONSIDERAÇÕES INICIAIS

No contexto deste trabalho, o objetivo da formalização é aprofundar o conhecimento sobre o modelo PERSONÆ e explicitar o processo de mediação semântica durante as tarefas de descoberta, seleção e estabelecimento de contratos de serviços.

Métodos formais, baseados em conceitos e operações matemáticas elementares, podem ser utilizados na especificação de aspectos comportamentais relacionados com a funcionalidade de sistemas computacionais de funcionalidade complexa (HARMELEN; FENSEL, 1995). Dentre os principais objetivos desta prática podemos destacar: (1) redução da ambigüidade na descrição de elementos e operações que compõem um determinado sistema computacional, para fins de projeto e implementação; (2) redução de erros em

tempo de execução do sistema; e (3) aumento da precisão na modelagem de aspectos comportamentais relacionados à funcionalidade desejada do software, permitindo maior ênfase em aspectos críticos do sistema (COHEN, 1989).

Dentre várias aplicações de métodos formais na prototipação de sistemas computacionais complexos pode-se destacar: (1) o uso da teoria da probabilidade na verificação de *performance*; (2) projeto de compiladores mediante uso de gramáticas livres de contexto; e (3) uso de cálculo relacional para especificação de algoritmos de otimização de consultas em bancos de dados (WOODCOCK; DAVIES, 1996). No contexto deste trabalho, métodos formais são aplicados na especificação de processos específicos de reconciliação ontológica no contexto de uma Arquitetura Orientada a Serviços Semânticos.

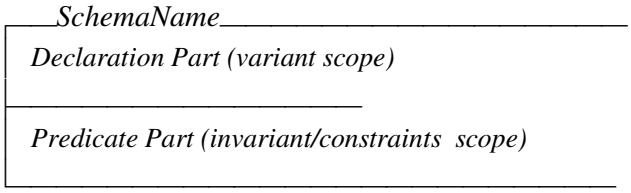
Em geral, três critérios orientam a escolha pelo formalismo: (1) o poder de expressão do formalismo; (2) a possibilidade de verificação de propriedades específicas do domínio de aplicação; e (3) a aplicabilidade do método formal escolhido na especificação de soluções para o problema analisado (POTTER; SINCLAIR; TILL, 1996). Além desses critérios, outra característica desejável é a disponibilidade de ferramentas para verificação automática dessas propriedades.

A escolha de Z (WOODCOCK; DAVIES, 1996) para formalização do modelo PERSONÆ seguiu estes critérios. Além de ser bem conhecida, a notação Z possui ferramentas para verificação de tipos e verificação de propriedades (SAALTINK, 1997). Durante o processo de formalização, as informações tratadas pelos componentes de PERSONÆ (que representam o estado), bem como as operações que incidem sobre estas informações, foram formalizadas através de esquemas, que representam o principal elemento de uma especificação formal em Z.

A notação Z é baseada em teoria de conjuntos e lógica de primeira ordem (SPIVEY, 1992). A teoria de conjuntos utilizada inclui operadores-padrão para conjuntos, compreensões de conjuntos, produtos cartesianos e conjuntos potência (*power sets*). A lógica de primeira ordem é baseada em cálculos de predicados. Considere-se ainda que as estruturas matemáticas em Z podem ser agrupadas em especificações de tipos complexos, chamadas de esquemas. A linguagem de esquemas pode ser utilizada para descrição do estado de um sistema e as operações que modificam este estado. Além disso, esta linguagem ainda permite o raciocínio sobre refinamento progressivo de operações, até a produção de código executável (WOODCOCK; DAVIES, 1996), (WOOD, 1993).

Em um esquema Z, a parte superior é conhecida como declarativa, sendo utilizada para declarar as variáveis e seus tipos. A segunda parte, chamada de predicado ou *invariante* é

dedicada à descrição de relacionamentos e restrições entre estas variáveis (vide representação abaixo). A especificação formal foi dividida em duas partes. A primeira refere-se às informações que representam o estado do modelo. A segunda refere-se às operações realizadas pelos componentes do modelo e que incidem sobre o estado. Durante a especificação formal do modelo PERSONÆ foi utilizada a ferramenta Z-EVES (SAALTINK, 1997). A utilização desta ferramenta possibilitou a verificação automática da sintaxe de Z, verificação de tipos e verificação de domínio. A especificação de um esquema Z segue o formato abaixo:




A correlação entre a linguagem Z e as linguagens para definição de ontologias para Web (como OWL) têm sua gênese nos fundamentos da lógica de primeira ordem com base em cálculos de predicados. Segundo Lucanu et al. (LUCANU; LI; DONG, 2005), a aplicação de co-morfismos de instituições na verificação de relacionamentos entre os fundamentos lógicos de OWL e Z provou que existe uma relação semântica forte entre essas linguagens. Além disso, segundo Dong et al. (DONG; SUN; WANG, 2002), (DONG, 2004), tecnologias atuais para verificação de ontologias na Web, tais como os motores de inferência (*reasoners*) ainda se encontram em estágio de aprimoramento, fator que se justifica pela sedimentação recente de linguagens para definição de ontologias. Neste caso, a notação Z pode ser utilizada como abordagem complementar no desenvolvimento de aplicações neste campo. Em PERSONÆ, as propriedades especificadas representam uma intersecção entre os axiomas definidos por OWL e WSMF. Isto permite que a análise seja focada no problema, em detrimento de especificidades de implementação típicas das linguagens mencionadas.

## 4.2 PADRÕES DE PROJETO EM Z

Padrões têm sido introduzidos na engenharia de software para fins de documentação e promoção de melhores práticas e reuso de conhecimento entre projetistas e desenvolvedores de software. Um padrão provê uma solução abstrata para um determinado problema em um determinado contexto. Os padrões existentes se aplicam a uma grande variedade de

contextos, desde padrões de desenvolvimento de código (BECK, 1997), padrões de projeto (GAMMA, 1995), análise de domínio (FOWLER, 1997), padrões ontológicos (NOY; MUSEN, 2000) e meta-propriedades, tais como gerenciamento de projetos e estruturas de desenvolvimento de software (COPLIEN, 1995).

No trabalho proposto por Stepney et al. (STEPNEY; POLACK; TOYN, 2003), são identificados padrões de especificação para uso da notação Z. A motivação principal é de facilitar o uso e interpretação desta notação, permitindo maior intercâmbio entre projetistas e desenvolvedores de software. Neste caso, os padrões propostos e técnicas de refatoramento são aplicados para melhorar a "estrutura semântica" de Z, facilitando os processos de escrita, leitura e apresentação de Z. Outros padrões para refinamento, implementação e prova encontram-se em fase de aprimoramento (VALENTINE; STEPNEY; TOYN, 2004).

A nomenclatura proposta para os padrões em Z provê um vocabulário para a descrição dos problemas e o projeto de soluções. Tipicamente, um padrão consiste de um algoritmo e uma especificação de sua aplicabilidade. Em PERSONÆ, a escolha de padrões simples não compromete a expressividade do modelo proposto.

A aplicação de padrões depende do contexto. Desta forma, os detalhes de um determinado padrão de apresentação podem ser afetados pelo modelo arquitetural utilizado, pelo estilo de codificação, pelo domínio de aplicação ou pelo propósito da especificação (e.g. direcionamento à prova ou implementação).

Segundo Stepney et al. (STEPNEY; POLACK; TOYN, 2003), os padrões de especificação em Z (*specification patterns*) incluem: (1) padrões de apresentação, para formatação e organização de documentos em Z; (2) padrões de idioma, que definem estilos de escrita para frases e axiomas; (3) padrões estruturais, que definem possibilidades de concatenação de pequenos fragmentos de axiomas e esquemas; (4) padrões arquiteturais, que definem modelos genéricos para especificação de arquiteturas; (5) padrões de domínio, que fornecem diretrizes para aplicação de determinadas estruturas conforme a aplicação; e (6) padrões de desenvolvimento, que provêm diretrizes para o processo de engenharia de requisitos e funcionalidade desejada, incluindo a determinação de níveis de formalidade e rigor apropriados, de acordo com o problema.

O padrão arquitetural escolhido foi o Delta/Xi (**Δ/Ξ**). O propósito é a especificação de um sistema caracterizado por elementos de estado abstrato e operações que alteram este estado, com preservação de *invariantes* (restrições de predicados nos esquemas). Este

padrão, também denominado de "estilo de Estados e Operações", aparece em grande parte dos modelos e textos propostos em notação Z, por exemplo (BARDEN; STEPNEY; COOPER, 1994), (POTTER; SINCLAIR; TILL, 1996), (WOODCOCK; DAVIES, 1996), também chamado de "estratégia clássica para definição de arquiteturas".

**Tabela 4.1 – Resumo dos padrões de especificação em Z utilizados em PERSONÆ**

| CATEGORIA | SUBCATEGORIA | |
|---|---|---|
| | ESTILO | USO |
| APRESENTAÇÃO | Consistência de nomes de variáveis | Definição de terminologia da Web semântica |
| IDIOMA | Representação de mapeamentos *muitos-para-um* (funcional) e *muitos-para-muitos* (relacional)<br>*Schema binding*<br>*Schema hinding*<br>Declarações locais<br>*Bemused mu*<br>*Bemused lambda* | Definição de predicados restritivos nos esquemas de estado abstrato<br>Ênfase na simplificação do modelo e apresentação em detrimento da utilização de encapsuladores complexos |
| ESTRUTURA | *Name meaningful chunks* | Modelagem de produtos cartesianos<br>Pré-condições parciais |
| ARQUITETURA | Orientação a objetos<br>Estilo algébrico | Delta/Xi (**Δ/Ξ**) |
| DOMÍNIO | *Application oriented theory* | Reconciliação ontológica e morfismo OWL/Z |
| DESENVOLVIMENTO | Especificação de requisitos funcionais<br>Ênfase na expressividade | Refinamento<br>Checagem de tipos e sintaxe |

As chamadas "convenções Delta/Xi" não são apenas convenções de nomenclatura, visto que definem uma semântica para caracterização de variáveis. A variável decorada com um apóstrofo (variável tipo *after-state*) possui uma semântica que define o processo de composição seqüencial; os símbolos de decoração "**?**" e "**!**" (para variáveis de entrada e saída, respectivamente), são definidos segundo a norma ISO-Z (ISO, 2002) para a utilização dos símbolos de encapsulamento de estados Δ e Ξ. Um caso de uso freqüente da notação Delta/Xi consiste na modificação de parte do estado abstrato do sistema. Outro padrão utilizado com freqüência em PERSONÆ é a compreensão de conjuntos (*set comprehension*), que consiste na concatenação de predicados restritivos que definem os elementos que pertencem a um determinado conjunto. Este padrão é utilizado nas tarefas de descoberta e seleção de serviços. Os padrões utilizados em PERSONÆ são referenciados na Tabela 4.1. Maiores detalhes sobre a estrutura de cada padrão pode ser encontrados em (STEPNEY; POLACK; TOYN, 2003), (VALENTINE; STEPNEY; TOYN, 2004).

## 4.3 ESTADO ABSTRATO DO MODELO

O estado abstrato do modelo PERSONÆ é composto por vários esquemas que descrevem precisamente todas as informações tratadas pelos diversos componentes arquiteturais. Na Figura 4.1, é mostrada a hierarquia de esquemas de estado do modelo PERSONÆ. A notação utilizada para representação gráfica é bastante simples: os retângulos representam esquemas e as linhas direcionadas representam ligações entre esquemas.

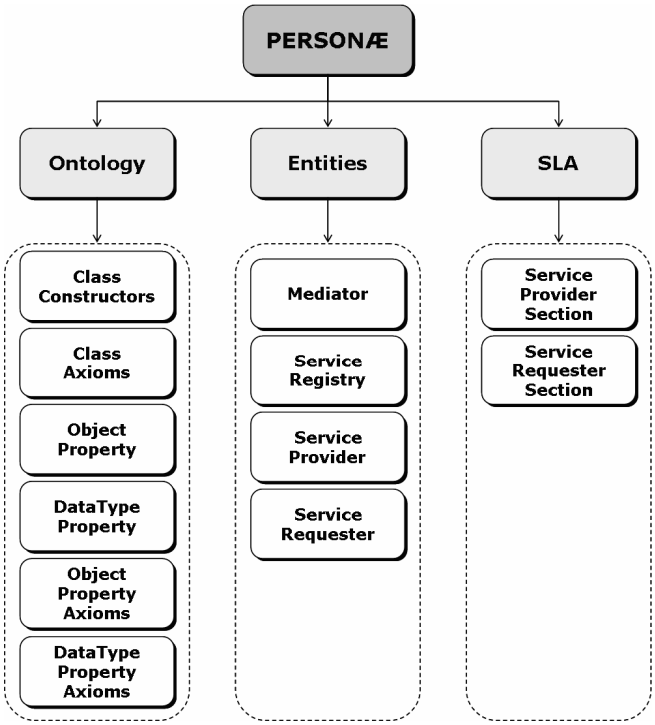



**Figura 4.1 – Hierarquia de esquemas de estados de PERSONÆ**

As ligações entre esquemas de estado representam a dependência hierárquica funcional, situação em que atributos de um esquema complexo são descritos através de instâncias de esquemas mais simples, que indicam seu tipo. No topo da hierarquia de esquemas de estado está o esquema que representa o modelo (*PERSONÆ*), composto pelos três elementos: (1) ontologias (*Ontologies*); (2) entidades (*Entities*) e (3) contrato (*SLA – ServiceLevelAgreement*).

Os esquemas que representam os componentes estruturais do esquema *Ontology* são denominados respectivamente: *ClassConstructors, ObjectProperty, DataTypeProperty, ClassAxioms, ObjectPropertyAxioms, DataTypePropertyAxioms e IndividualAxioms*. Os esquemas que representam as quatro entidades de PERSONÆ são denominados

respectivamente: *ServiceRequester* (Cliente), *ServiceProvider* (Provedor de Serviços), *ServiceRegistry* (Registro de Serviços) e *Mediator* (Mediador). O esquema que representa a estrutura básica do SLA utiliza duas abstrações para a determinação do nível de qualidade acordada (*QoSAgreed*), representadas por *ServiceRequesterSection* e *ServiceProviderSection.*

Para fins de simplificação, o detalhamento da especificação formal segue uma abordagem *top-down* (dos esquemas mais abstratos aos mais concretos). O esquema que especifica o modelo PERSONÆ em nível mais abstrato (PERSONÆ) é definido pela inclusão dos quatro esquemas que especificam os principais elementos: (1) entidades (*Entities*); (2) a estrutura genérica para ontologia (*Ontology*); (3) o contrato de serviço (*SLA*) e (4) relacionamentos (*Relationships*). A ausência de predicado neste esquema não compromete a precisão do modelo, uma vez que outros esquemas pertencentes à hierarquia de PERSONÆ, estabelecem tais restrições.

*PERSONAE*

*Ontologies*
*Entities*
*SLA*

A definição de esquemas mais complexos parte do pressuposto da definição de estruturas mais simples (BAUMEISTER, 2000). Em Z, a semântica de tipos de variáveis é definida por conjuntos básicos ou *given sets* (ver abaixo *given sets* definidos para PERSONÆ). Este elemento abstrato, juntamente com outros poucos tipos (e.g. o tipo $\mathbb{Z}$), formam a base de tipos da especificação, e representam conjuntos de elementos cuja estrutura interna não postula maior nível de detalhamento. Ou seja, *given sets* representam a unidade básica de especificação em Z. A especialização de um conceito em estruturas internas mais detalhadas consiste na transformação de um *given set* em um esquema abstrato de dados, o qual pode reunir variáveis de tipos heterogêneos, representando um tipo de dado complexo.

[*CLASS*, *DATATYPE*, *ID*, *INSTANCE*, *MESSAGE*, *OPERATION*, *PROTOCOL*, *RESOURCE*, *URI*]

O primeiro elemento arquitetural de PERSONÆ é representado pelas ontologias. O esquema que especifica uma ontologia (*Ontology*) possui nove variáveis que identificam cada um dos nove componentes. A variável *Classes* é a única definida por um *given set*

simples. A variável *Individuals*, que representa um conjunto de instâncias de uma determinada classe, é representada por uma função total de *CLASS* em *INSTANCE* (tipos primitivos especificados nos *given sets*). As demais variáveis são especificadas por tipos mais complexos representados por esquemas, constituindo basicamente os tipos de relacionamentos da estrutura interna de uma ontologia.

Os tipos *ObjectProperties* e *DataTypeProperties* constituem os relacionamentos entre classes e entre classes e tipos de dados, respectivamente. Tais propriedades possuem um identificador único para diferenciação em uma mesma ontologia. O nome de uma propriedade é um exemplo de identificador. Outros tipos de identificadores podem ser utilizados. Este aspecto foi abstraído pelo uso do given set *ID*, que representa o conjunto de todos os tipos de identificadores utilizados na especificação. A determinação de tipos específicos de identificadores depende da estratégia de implementação. Vale destacar que a noção de instanciação também se aplica às propriedades. As funções compostas *DtPropInstantiation* e *ObjPropInstantiation* relacionam indivíduos de uma mesma classe com tipos de dados (*DATATYPE*) e outros indivíduos, respectivamente. A especificação completa de *Ontology* segue conforme as definições a seguir:

$$Individual: CLASS \rightarrow \mathbb{P}\ INSTANCE$$

*ObjectProperty*

$$Id: ID$$
$$Relationship: CLASS \leftrightarrow CLASS$$

*DataTypeProperty*

$$Id: ID$$
$$Relationship: CLASS \nrightarrow DATATYPE$$

$$ObjPropInstantiation: ObjectProperty \rightarrow Individual \leftrightarrow Individual$$
$$DtPropInstantiation: DataTypeProperty \rightarrow Individual \nrightarrow DATATYPE$$

*Ontology*

$$Classes: \mathbb{P}\ CLASS$$
$$ObjectProperties: \mathbb{P}\ ObjectProperty$$
$$DataTypeProperties: \mathbb{P}\ DataTypeProperty$$
$$Individuals: \mathbb{P}\ Individual$$
$$ClassConstructAxioms: \mathbb{P}\ ClassConstructors$$
$$OntologicalClassAxioms: \mathbb{P}\ ClassAxioms$$
$$OntologicalObjectPropertyAxioms: \mathbb{P}\ ObjectPropertyAxioms$$
$$OntologicalDataTypePropertyAxioms: \mathbb{P}\ DataTypePropertyAxioms$$
$$OntologicalIndividualAxioms: \mathbb{P}\ IndividualAxioms$$

---

$$\forall c1, c2: CLASS; objp1, objp2: ObjectProperty; dtp1, dtp2: DataTypeProperty;$$
$$i1, i2: Individual \mid c1 \in Classes \land c2 \in Classes$$
$$\land\ objp1 \in ObjectProperties \land objp2 \in ObjectProperties$$
$$\land\ dtp1 \in DataTypeProperties \land dtp2 \in DataTypeProperties$$
$$\land\ i1 \in Individuals \land i2 \in Individuals$$
$$\bullet\ c1 \neq c2 \land objp1 \neq objp2 \land dtp1 \neq dtp2 \land i1 \neq i2$$

O conceito de classe provê um mecanismo de abstração para o agrupamento de recursos com características similares. Em OWL, uma classe está associada a um conjunto de indivíduos, denominado extensão de classe (*class extension*)(OWL, 2004). Os indivíduos pertencentes a uma extensão de classe são chamados de instâncias de uma classe. Uma classe possui um significado intencional, o qual está relacionado (mas não é igual, necessariamente) à sua extensão de classe. Portanto, duas classes podem ter a mesma extensão de classe, mas ainda assim, podem ser classes diferentes.

Em PERSONÆ, são especificados três tipos de descrições avançadas para construtores de classes, os quais são utilizados em Description Logic. Esses construtores podem ser interpretados como operadores AND, OR e NOT, para classes. Os três operadores são: *IntersectionOf*, *UnionOf* e *ComplementOf*.

O construtor *IntersectionOf* mapeia uma classe em uma lista de descrições de classe. Desta forma, esta função descreve uma classe para a qual a extensão de classe contém precisamente os indivíduos que são membros da extensão de classe de todas as descrições de classe na lista. O construtor *UnionOf* é especificado de forma semelhante, diferindo nas restrições predicativas indicadas no esquema. Esta função descreve uma classe anônima para a qual a extensão de classe contém os indivíduos que ocorrem em pelo menos uma das extensões de classe da descrição de classe na lista. Esta propriedade é análoga a uma disjunção lógica. O construtor *ComplementOf* mapeia uma classe, precisamente, em uma descrição de classe. Esta propriedade descreve uma classe para a qual a extensão de classe contém exatamente os indivíduos que não pertencem à extensão de classe da descrição correspondente, ou seja, esta propriedade é análoga a uma negação lógica: a extensão de classe consiste nos indivíduos que não são membros da extensão de classe do complemento de classe.

*ClassConstructors*

$$\begin{aligned}
&\mathit{IntersectionOf}: \text{seq } \mathit{CLASS} \rightarrow \mathit{CLASS} \\
&\mathit{UnionOf}: \text{seq } \mathit{CLASS} \rightarrow \mathit{CLASS} \\
&\mathit{ComplementOf}: \mathit{CLASS} \leftrightarrow \mathit{CLASS}
\end{aligned}$$

---

$$\begin{aligned}
&\forall \mathit{c1}, \mathit{c2}: \mathit{CLASS}; \mathit{cn}: \text{seq } \mathit{CLASS} \\
&\quad \bullet\ \mathit{IntersectionOf}\ \mathit{cn} = \mathit{c1} \\
&\qquad \Leftrightarrow \mathit{Individual}\ \mathit{c1} = \cap \{\ x: \text{ran } \mathit{cn} \bullet\ (\mathit{Individual}\ x)\ \} \\
&\qquad\quad \wedge (\mathit{UnionOf}\ \mathit{cn} = \mathit{c1} \Leftrightarrow \mathit{Individual}\ \mathit{c1} = \cup \{\ x: \text{ran } \mathit{cn} \bullet\ (\mathit{Individual}\ x)\ \}) \\
&\qquad\quad \wedge (\mathit{c1}\ \mathit{ComplementOf}\ \mathit{c2} \Leftrightarrow \mathit{INSTANCE} \setminus \mathit{Individual}\ \mathit{c1} = \mathit{Individual}\ \mathit{c2})
\end{aligned}$$

Tipicamente, axiomas de classe contêm componentes adicionais que determinam características necessárias e/ou suficientes de uma classe. Em OWL, existem três construtores para combinação de descrições de classes em axiomas. Em PERSONÆ, foram derivados mais cinco axiomas deste tipo para fins de clareza e tarefas de mapeamento de ontologias. Os axiomas são *isSubClassOf*, *isSuperClassOf*, *equivalentFrom*, *isSiblingClassOf* e *disjointWith*.

O construtor *isSubClassOf* é definido como parte da especificação de RDF Schema. A semântica do construtor é idêntica em OWL: se a descrição de classe $C_1$ é definida como uma subclasse da descrição $C_2$, então o conjunto de indivíduos na extensão $C_1$ pode ser considerado como subconjunto dos indivíduos da extensão $C_2$. Por definição, uma classe pode ser considerada como subconjunto dela própria. O construtor *equivalentFrom* relaciona uma descrição de classe em outra, estabelecendo que as duas descrições possuem a mesma extensão de classe (ou seja, contêm o mesmo conjunto de indivíduos). O construtor *disjointWith* estabelece que duas descrições de classe não possuem qualquer indivíduo em comum. De forma análoga aos axiomas *isSubClassOf* e *isSuperClassOf*, a operação de disjunção de classes é uma definição parcial, ou seja, esta impõe condições necessárias, mas não suficientes, sobre uma classe. O esquema abaixo estabelece as relações e restrições mencionadas.

*ClassAxioms*

$$
\begin{array}{l}
isSuperClassOf: CLASS \leftrightarrow CLASS \\
isSubClassOf: CLASS \leftrightarrow CLASS \\
isSiblingClassOf: CLASS \leftrightarrow CLASS \\
EquivalentClass: CLASS \leftrightarrow CLASS \\
DisjointWith: CLASS \leftrightarrow CLASS \\
\hline
\forall c1, c2, c3: CLASS \\
\quad \bullet\ c1\ isSuperClassOf\ c2 \\
\quad\ \Leftrightarrow Individual\ c2 \subseteq Individual\ c1 \\
\quad\ \wedge (c1\ isSubClassOf\ c2 \Leftrightarrow Individual\ c1 \subseteq Individual\ c2) \\
\quad\ \wedge (c1\ isSiblingClassOf\ c2 \\
\quad\quad \Leftrightarrow c3\ isSuperClassOf\ c1 \wedge c3\ isSuperClassOf\ c2 \\
\quad\quad\ \vee c1\ isSubClassOf\ c3 \wedge c2\ isSubClassOf\ c3) \\
\quad\ \wedge (c1\ EquivalentClass\ c2 \Leftrightarrow Individual\ c1 = Individual\ c2) \\
\quad\ \wedge (c1\ DisjointWith\ c2 \Leftrightarrow Individual\ c1 \cap Individual\ c2 = \varnothing)
\end{array}
$$

Os axiomas de propriedades são classificados em dois tipos: axiomas para relações entre propriedades distintas (*SubObjectPropertyOf*, *SuperObjectPropertyOf* e

*EquivalentObjectProperty*) e axiomas de definição lógica (*TransitiveProperty*, *SymmetricProperty* e *InversePropertyOf*). Para fins de simplicidade, estes axiomas foram agrupados em um mesmo esquema na especificação.

O axioma *SubObjectPropertyOf* determina que uma propriedade é sub-propriedade de outra propriedade. Formalmente, isto implica que, se $P_1$ e $P_2$ pertencem à relação *SubObjectPropertyOf*, então a extensão de propriedade de $P_1$ (um conjunto de pares que instanciam a propriedade) pode ser considerada como subconjunto da extensão de propriedade de $P_2$. A propriedade inversa é representada pelo axioma *SuperObjectPropertyOf*.

O axioma *EquivalentObjectProperty* pode ser utilizado quando duas propriedades têm a mesma extensão de propriedades. Sintaticamente, este axioma representa uma relação cujo domínio e imagem são do tipo *ObjectProperty*. Vale destacar que a propriedade de equivalência difere da propriedade de igualdade (*SameAs*). Propriedades equivalentes possuem o mesmo conjunto de "valores" (ou seja, a mesma extensão de propriedade), mas podem ter significados diferentes (ou seja, denotar conceitos diferentes). A propriedade de igualdade se aplica apenas a indivíduos, por definição do axioma (*SameAs*).

Propriedades possuem uma direção (noção de função matemática), de um conjunto domínio a um conjunto de imagem. Na prática, é muito comum utilizar propriedades que definem relações em ambas as direções (e.g. pessoas possuem carros, carros são propriedades de pessoas). O axioma *InversePropertyOf* pode ser utilizado para definir tal relação entre propriedades. Sintaticamente, o axioma *InversePropertyOf* possui conjuntos do tipo *ObjectProperty* no domínio e na imagem. Formalmente, um par ordenado ($P_1$, $P_2$) está contido na relação *InversePropertyOf*, se para cada par (x,y) na extensão de propriedade de $P_1$, existe um par (y,x) na extensão de propriedade de v, e vice-versa. Portanto, a propriedade *InversePropertyOf* também é simétrica.

O axioma *TransitiveProperty* estabelece que, se uma determinada propriedade *P* é transitiva, isto implica que se um par (x,y) é uma instância de *P*, e um par (y,z) também é uma instância de *P*, então se pode inferir que o par (x,z) também é uma instância de *P*. O axioma *SymmetricProperty* estabelece que, se um par (x,y) é uma instância de *P*, então o par (y,x) também é uma instância de *P*.

Os axiomas correspondentes para relações do tipo *DataTypeProperty* são especificados de forma análoga no esquema *DataTypePropertyAxioms*. As definições de transitividade, simetria e inversão não se aplicam neste tipo de propriedade, visto que os conjuntos de domínio e imagem são diferentes.

*ObjectPropertyAxioms*

$SubObjectPropertyOf: ObjectProperty \leftrightarrow ObjectProperty$
$SuperObjectPropertyOf: ObjectProperty \leftrightarrow ObjectProperty$
$EquivalentObjectProperty: ObjectProperty \leftrightarrow ObjectProperty$
$TransitiveProperty: \mathbb{P}\ ObjectProperty$
$SymmetricProperty: \mathbb{P}\ ObjectProperty$
$InversePropertyOf: ObjectProperty \leftrightarrow ObjectProperty$

---

$\forall objP1, objP2, objPn: ObjectProperty; i1, i2, i3: Individual$
$\bullet\ objP1\ SubObjectPropertyOf\ objP2$
$\Leftrightarrow objP1 \in ObjectProperty \land objP2 \in ObjectProperty$
$\Rightarrow ObjPropInstantiation\ objP1 \subseteq ObjPropInstantiation\ objP2$
$\land (objP1\ SuperObjectPropertyOf\ objP2$
$\Leftrightarrow objP1 \in ObjectProperty \land objP2 \in ObjectProperty$
$\Rightarrow ObjPropInstantiation\ objP2 \subseteq ObjPropInstantiation\ objP1)$
$\land (objP1\ EquivalentObjectProperty\ objP2$
$\Leftrightarrow objP1 \in ObjectProperty \land objP2 \in ObjectProperty$
$\Rightarrow ObjPropInstantiation\ objP1 = ObjPropInstantiation\ objP2)$
$\land (objPn \in TransitiveProperty$
$\Leftrightarrow (i1, i2) \in ObjPropInstantiation\ objPn)$
$\land ((i2, i3) \in ObjPropInstantiation\ objPn$
$\Rightarrow (i1, i3) \in ObjPropInstantiation\ objPn)$
$\land (objPn \in SymmetricProperty$
$\Leftrightarrow (i1, i2) \in ObjPropInstantiation\ objPn$
$\Rightarrow (i2, i1) \in ObjPropInstantiation\ objPn)$
$\land (objP1\ InversePropertyOf\ objP2$
$\Leftrightarrow ObjPropInstantiation\ objP1 = (ObjPropInstantiation\ objP2)^{\sim})$

*DataTypePropertyAxioms*

$SuperDataTypePropertyOf: DataTypeProperty \leftrightarrow DataTypeProperty$
$SubDataTypePropertyOf: DataTypeProperty \leftrightarrow DataTypeProperty$
$EquivalentDataTypeProperty: DataTypeProperty \leftrightarrow DataTypeProperty$

---

$\forall dtP1, dtP2: DataTypeProperty$
$\bullet\ dtP1\ SuperDataTypePropertyOf\ dtP2$
$\Leftrightarrow dtP1 \in DataTypeProperty \land dtP2 \in DataTypeProperty$
$\Rightarrow DtPropInstantiation\ dtP2 \subseteq DtPropInstantiation\ dtP1$
$\land dtP1\ SubDataTypePropertyOf\ dtP2$
$\Leftrightarrow dtP1 \in DataTypeProperty \land dtP2 \in DataTypeProperty$
$\Rightarrow DtPropInstantiation\ dtP1 \subseteq DtPropInstantiation\ dtP2$
$\land (dtP1\ EquivalentDataTypeProperty\ dtP2$
$\Leftrightarrow dtP1 \in DataTypeProperty \land dtP2 \in DataTypeProperty$
$\Rightarrow DtPropInstantiation\ dtP1 = DtPropInstantiation\ dtP2)$

Muitas linguagens atribuem vocábulos diferentes para distinção de diferentes conceitos no mundo real. Na Web, tal suposição não é possível. Por exemplo, uma mesma pessoa pode ser referenciada em diferentes formas (ou seja, por diferentes URIs). Para o estabelecimento de relações entre indivíduos, dois axiomas podem ser utilizados: *sameAs* e *differentFrom.*

O axioma *sameAs* relaciona um indivíduo a outro. Desta forma, esta propriedade indica que duas URIs se referem ao mesmo conceito, ou seja, os indivíduos possuem a mesma identidade. Esta propriedade também é frequentemente utilizada para definição de mapeamentos entre ontologias. A pressuposição de que um mesmo vocábulo seja utilizado para nomear uma mesma instância de conceito é demasiadamente arbitrária. Para ontologias em que é válida a propriedade de atribuição de nomes únicos às instâncias, o axioma *differentFrom* pode ser utilizado para declaração de relações de disjunção completa entre instâncias.

*IndividualAxioms*

$$SameAs: Individual \leftrightarrow Individual$$
$$DifferentFrom: Individual \leftrightarrow Individual$$

$$\forall i1, i2: Individual \bullet i1\ SameAs\ i2 \Leftrightarrow i1 = i2 \land i1\ DifferentFrom\ i2 \Leftrightarrow i1 \neq i2$$

O esquema *ServiceRequester* define o tipo da especificação de requisitos funcionais e não-funcionais desejados pelo usuário. Este esquema é composto por quatro elementos: (1) perfil de usuário (*Profile*), definido por uma ontologia que pode ser utilizada para recuperação de determinadas classes de usuários com base nas preferências; (2) objetivos do usuário (*Goal*), para definição da funcionalidade desejada; (3) restrições de QoS (*QoSConstraints*) para especificação de requisitos não desejados pelo usuário; (4) preferências de QoS (*QoSPreferences*), que indicam os requisitos desejados pelo usuário, em primeira instância; (5) prioridades de QoS (*QoSPriorities*), para diferenciação de requisitos não-funcionais; e (6) qualidade de serviço desejada (*QoSDesired*), que representa a união de *QoSConstraints* e *QoSPreferences*.

O nível associado a um requisito não-funcional (ver esquema abaixo) é definido por um tipo de dado (*LEVEL*). A definição da variável nível (*LEVEL*) através de *DATATYPE* abstrai detalhes de implementação específicos neste sentido, visto que várias coleções de tipos podem ser utilizadas em uma mesma aplicação. Ainda que seja considerada a base de tipos fornecida por XML Schema, nem todos os tipos de dados são utilizados em uma mesma ontologia. A instanciação dos intervalos de níveis de qualidade também varia de uma ontologia para outra.

A opção por definir as prioridades através de relações, em vez de, por exemplo, um conjunto de prioridades, refere-se ao fato de existirem situações em que esta informação somente é insuficiente. Por exemplo, o usuário pode definir que entre custo e desempenho, custo é prioritário, mas também pode definir que entre usabilidade e custo, o primeiro é prioritário. Dessa forma, se o custo for afetado por alterações nas condições de oferta, a relação permite uma decisão mais consistente. O predicado do esquema *ServiceRequester* inclui essencialmente restrições sobre requisitos não-funcionais. Se dois requisitos não-funcionais, denominados *nfr1* e *nfr2*, pertencem ao conjunto de prioridades do usuário (*QoSPriorities*), então somente um (*nfr1* ou *nfr2*) pode ser definido como prioritário ou mais restritivo. Por exemplo, se "tempo de resposta" é definido como mais restritivo do que "custo", então esta restrição deve ser respeitada na definição de prioridades, ou seja, é vetada a inclusão de prioridade definida pela inversão na ordem destes requisitos.

*NonFunctionalRequirement*

$$
\begin{array}{l}
Id: ID \\
Class: CLASS \\
Domain: Ontology \\
\hline
Class \in Domain\ .\ Classes
\end{array}
$$

*ServiceRequester*

$$
\begin{array}{l}
Profile: Ontology \\
Goal: FunctionalRequirement \\
QoSConstraints: NonFunctionalRequirement \pfun LEVEL \\
QoSPreferences: NonFunctionalRequirement \pfun LEVEL \\
QoSPriorities: NonFunctionalRequirement \times NonFunctionalRequirement \\
\qquad \pfun NonFunctionalRequirement \\
QoSDesired: NonFunctionalRequirement \pfun LEVEL \\
\hline
\forall nfr1, nfr2: NonFunctionalRequirement \\
\quad \mid (nfr1, nfr2) \in \mathrm{dom}\ QoSPriorities \land (nfr2, nfr1) \in \mathrm{dom}\ QoSPriorities \\
\quad \bullet\ nfr1\ .\ Domain = nfr2\ .\ Domain \\
\qquad \land (QoSPriorities\ (nfr1, nfr2) = nfr1 \land QoSPriorities\ (nfr2, nfr1) = nfr1 \\
\qquad\quad \lor QoSPriorities\ (nfr1, nfr2) = nfr2 \land QoSPriorities\ (nfr2, nfr1) = nfr2) \\
\qquad \land \mathrm{ran}\ QoSPriorities \subseteq \mathrm{dom}\ QoSPreferences \\
\qquad \land \mathrm{dom}\ QoSConstraints \cap \mathrm{dom}\ QoSPreferences = \varnothing \\
\qquad \land QoSDesired = QoSConstraints \cup QoSPreferences
\end{array}
$$

Conforme mencionado na seção 2.4 do capítulo 2, as principais alternativas de operacionalização da descrição de *Web services* incluem WSMO e OWL-S, que são utilizadas pelos *frameworks* IRS-III e SWSO, respectivamente. Ambas as abordagens definem estruturas que descrevem basicamente "o que" o serviço faz e "como" ele opera (e.g. *Service Profile* e *Process Model*, em OWL-S; *Service Description* e *Interface*, em WSMO). As informações representadas nestas estruturas acrescentam níveis de abstração que complementam descrições operacionais em WSDL, para fins de automatização dos processos de descoberta, seleção e composição de serviços.

A forma como estas informações estão organizadas diferem para cada abordagem citada. Em PERSONÆ, estas informações são generalizadas nos esquemas *ServiceDescription*, *ServiceInterface* e *Service*, que representam o núcleo comum dos parâmetros descritos nas abordagens existentes. Neste sentido, a generalização permite o estabelecimento de um esquema global que facilita o mapeamento de descrições nas tarefas de descoberta e seleção de serviços. Além disso, permite que detalhes específicos de organização de parâmetros somente sejam considerados durante a fase de implementação, onde técnicas específicas de refinamento progressivo podem conduzir à geração de código pertinente a um determinado paradigma de programação. O esquema *ServiceDescription* define a configuração básica de descrição de um serviço, sendo composto por cinco elementos: (1) nome do serviço (*ServiceName*), definido de acordo com um tipo de dado apropriado (que pode ser representado com tipos definidos em XML Schema); (2) domínio de aplicação do serviço (*Domain*), definido em uma ontologia; (3) entradas e saídas (*Input* e *Output*), que também correspondem aos tipos de dados aceitos e fornecidos pelo serviço; (4) capacidade ou funcionalidade (*Capability*) que define o requisito funcional provido pelo serviço; e (5) qualidade de serviço provida (*QoSProvided*), que define o nível de qualidade do serviço.

O esquema *ServiceDescription* contém informações utilizadas nas tarefas de descoberta e seleção de serviços. O esquema *ServiceInterface* define informações relacionadas com as tarefas de invocação e execução do serviço (*binding*), sendo composto por cinco elementos: (1) tipos de dados aceitos (*Types*); (2) protocolos (*Bindings*); (3) tipos de portas (*PortType*), que definem um conjunto de operações e abstratas e mensagens; (4) mensagens trocadas na comunicação de serviços (*Messages*) que consistem nas primitivas de transmissão (e.g. *one-way*, *request-response*, *solicit-response* e *notification*); (5) a porta de comunicação (*Port*), que define um ponto único de acesso ao serviço, mediante especificação de um endereço para a tarefa de invocação.

O esquema *Service* integra os dois esquemas anteriores pela definição de dois níveis de abstração para caracterização de um serviço: descrição e implementação. O esquema *ServiceRegistry* define uma estrutura simplificada para registro, contendo dois elementos: (1) um identificador de registro (*RegistryID*) e (2) um conjunto de referências de serviços (*Services*). Esta simplificação reduz a complexidade do registro, visto que a tarefa de reconciliação ontológica é realizada pelo serviço de mediação. Neste caso o registro funciona apenas como um apontador para serviços, cuja descrição mais detalhada encontra-se no provedor, de acordo com o esquema *ServiceProvider*.

O esquema *ServiceProvider* é composto por quatro elementos: (1) um perfil de provedor (*Profile*), definido por uma ontologia, o qual pode ser utilizado em um processo de busca considere uma relação de fidelização entre clientes e provedores; (2) um conjunto de serviços (*Services*); (3) um conjunto de recursos (*RESOURCE*), que representam uma abstração de recursos computacionais utilizados na provisão do serviço; e (4) qualidade de serviço oferecida (*QoSOffered*), que representa o nível de qualidade que um provedor oferece sobre cada serviço.

*ServiceDescription*

$$
\begin{array}{l}
ServiceName: DATATYPE \\
Domain: Ontology \\
Input, Output: \mathbb{P}\ DataTypeProperty \\
Capability: FunctionalRequirement \\
QoSProvided: NonFunctionalRequirement \pfun LEVEL \\
\hline
\exists nfr: NonFunctionalRequirement \\
\quad \bullet\ nfr \in \mathrm{dom}\ QoSProvided \\
\qquad \land\ Capability\ .\ Class \in Domain\ .\ Classes \\
\qquad \land\ nfr\ .\ Class \in Domain\ .\ Classes \\
\qquad \land\ Input \subseteq Domain\ .\ DataTypeProperties \\
\qquad \land\ Output \subseteq Domain\ .\ DataTypeProperties
\end{array}
$$

*ServiceInterface*

$$
\begin{array}{l}
Types: \mathbb{P}\ DATATYPE \\
Bindings: \mathbb{P}\ PROTOCOL \\
PortType: \mathbb{P}\ OPERATION \\
Messages: \mathbb{P}\ MESSAGE \\
Port: DATATYPE \pfun URI \\
\hline
\mathrm{dom}\ Port \subseteq Types
\end{array}
$$

*Service*

*Description: ServiceDescription*
*ServiceImplementation: ServiceInterface*

*ServiceRegistry*

*RegistryID: ID*
*Services: ID ⇸ ServiceDescription*

*ServiceProvider*

*Profile: Ontology*
*Services: ℙ Service*
*Resource: ℙ RESOURCE*
*QoSOffered: NonFunctionalRequirement ⇸ LEVEL*

---

*∀S: Service | S ∈ Services*
• *QoSOffered ⊆ S . Description . QoSProvided*

O contrato de QoS (*SLA - Service Level Agreement*) constitui um registro do nível de qualidade na provisão do serviço. Idealmente, um processo de negociação precede o estabelecimento das premissas de um contrato. Em PERSONÆ, esta fase ocorre após a seleção de serviços. Conforme será detalhado adiante, uma combinação das técnicas de alinhamento e integração de ontologias pode ser utilizada para o estabelecimento do nível de qualidade acordada (*QoSAgreed*) e para registro dos perfis das entidades envolvidas (*ServiceRequester* e *ServiceProvider*), respectivamente.

O esquema *ServiceLevelAgreement* é constituído de três variáveis: (1) *ServiceRequesterSection,* que representa a qualidade de serviço, segundo a perspectiva do usuário; (2) *ServiceProviderSection,* que representa a qualidade, segundo a perspectiva do provedor; e (3) *QoSAgreed,* que define a qualidade de serviço pela intersecção das duas perspectivas (*QoSDesired* e *QoSOffered*). No nível inferior da hierarquia de esquemas de estado estão as abstrações de requisitos funcionais (*FunctionalRequirement*) e requisitos não-funcionais (*NonFunctionalRequirement*), para definição de *Goals* e requisitos de QoS, respectivamente. A definição do nível de QoS acordado é funcional com domínio em um serviço específico.

*ServiceLevelAgreement*

*ServiceRequesterSection, ServiceProviderSection: Ontology*
*SLAOntology: ℙ Ontology*
*QoSAgreed: NonFunctionalRequirement ⇸ LEVEL*

---

*ServiceRequesterSection ∈ SLAOntology*
*ServiceProviderSection ∈ SLAOntology*

## 4.4 OPERAÇÕES NO MODELO PERSONÆ

Operações possivelmente afetam o estado de um sistema. Esquemas de operação, descritos em Z, relacionam o estado anterior e posterior à execução da operação. Um esquema de operação genérico possui: estado anterior, estado posterior, variáveis de entrada e de saída, e pré-condição para a aplicação da operação. As variáveis de entrada são precedidas pelo símbolo (**?**) e as variáveis de saída pelo símbolo (**!**). Variáveis de uso interno do esquema são identificadas somente pelo nome.

Um esquema de operação pode ser executado a partir de outro esquema (chamada interna) ou de elementos externos como o usuário ou outros sistemas (chamada externa). Operadores especiais de Z possibilitam a execução de esquemas em várias formas, dentre elas, a composição, que caracteriza a execução seqüencial de dois ou mais esquemas.

A formalização das operações executadas pelos elementos que compõem o modelo PERSONÆ é objeto das duas seções seguintes. A primeira apresenta as operações que fazem parte do processo de reconciliação ontológica. A segunda seção detalha as operações das tarefas de descoberta, seleção e estabelecimento de contrato de serviços.

### 4.4.1 OPERAÇÕES DE RECONCILIAÇÃO ONTOLÓGICA

As operações de reconciliação ontológica representam a base do processo de mediação semântica em PERSONÆ. A verificação de similaridades ou diferenças entre conceitos e propriedades determina o estabelecimento de pré-condições que devem ser satisfeitas. As principais operações são: fusão, integração e alinhamento.

#### *A FUSÃO*

Três esquemas compõem a operação de fusão: (1) *MergingInTotalImpor*t, que especifica a fusão de duas ontologias quando ambas são constituídas unicamente por um fator comum, que constitui uma ontologia importada; (2) *MergingInPluginAndSubsumptionImport*, para definição das condições envolvidas quando a importação é total em uma ontologia e parcial na outra; e (3) *MergingInIntersectionImport*, quando ambas as ontologias importam parcialmente uma terceira ontologia. Cada um destes esquemas é composto por outros quatro esquemas intermediários, para fusão de classes, *object properties*, *datatype properties* e indivíduos (para fins de simplificação, será mostrado um esquema componente para cada caso).

Formalmente, o processo de fusão pode ser definido através do operador de composição seqüencial, definido pelo símbolo "⨾", conforme representação a seguir.

$$OntologyMerging \mathrel{\widehat{=}} MergingInTotalImport \fatsemi MergingInPluginAndSubsumptionImport \fatsemi MergingInIntersectionImport$$

***ObjectPropertyMergingInTotalImport***

O esquema *ObjectPropertyMergingInTotalImport* utiliza duas variáveis de entrada (*O1?* e *O2?*), uma variável de saída (*CoreOntology!*) e importa um conjunto de ontologias para manipulação de axiomas (*ΔOntologies*). O predicado consiste em concatenar pré-condições que determinam se a ontologia importada (i.e. um ancestral comum) representa a totalidade das ontologias de entrada. Neste caso, a ontologia resultante é constituída exatamente pela ontologia importada, visto que esta última representa a intersecção entre as ontologias de origem. As operações correspondentes para classes, *datatype properties* e indivíduos são semelhantes (vide Apêndice). O processo completo é definido pelo esquema composto *MergingInTotalImport.*

*ObjectPropertyMergingInTotalImport*

$$\Delta Ontologies$$
$$O1?, O2?, CoreOntology!: Ontology$$

---

$$\exists ImportedOntology: Ontology \mid O1? \in OntologySet \land O2? \in OntologySet \land ImportedOntology \in OntologySet \land CoreOntology! \notin OntologySet \land O1?.ObjectProperties = ImportedOntology.ObjectProperties \land O2?.ObjectProperties = ImportedOntology.ObjectProperties \bullet CoreOntology!.ObjectProperties = ImportedOntology.ObjectProperties \land OntologySet' = OntologySet \cup \{CoreOntology!\}$$

$$MergingInTotalImport \mathrel{\widehat{=}} ClassMergingInTotalImport \fatsemi ObjectPropertyMergingInTotalImport \fatsemi DataTypePropertyMergingInTotalImport \fatsemi IndividualMergingInTotalImport$$

***ObjectPropertyMergingInPluginAndSubsumptionImport***

O esquema abaixo define as pré-condições necessárias à formação de uma ontologia-núcleo (*core ontology*) quando a importação é total em uma das ontologias de entrada e parcial na

outra. Neste caso, a ontologia gerada na fusão será igual à ontologia que contém estende a ontologia importada, conforme detalhado no capítulo 3. As operações de fusão em importação parcial para classes, *datatype properties* e indivíduos são semelhantes (vide Apêndice). O processo completo é definido pelo esquema composto *MergingInPluginAndSubsumptionImport.*

*ObjectPropertyMergingInPluginAndSubsumptionImport*

$\Delta Ontologies$
$O1?, O2?, CoreOntology!: Ontology$

---

$\exists ImportedOntology: Ontology$
$\mid O1? \in OntologySet$
$\wedge O2? \in OntologySet$
$\wedge ImportedOntology \in OntologySet$
$\wedge CoreOntology! \notin OntologySet$
$\wedge O1?.ObjectProperties = ImportedOntology.ObjectProperties$
$\wedge ImportedOntology.ObjectProperties \subseteq O2?.ObjectProperties$
$\Rightarrow CoreOntology!.ObjectProperties$
$= CoreOntology!.ObjectProperties \cup O2?.ObjectProperties$
$\wedge ImportedOntology.ObjectProperties \subseteq O1?.ObjectProperties$
$\wedge O2?.ObjectProperties = ImportedOntology.ObjectProperties$
$\Rightarrow CoreOntology!.ObjectProperties$
$= CoreOntology!.ObjectProperties \cup O1?.ObjectProperties$
$\bullet OntologySet' = OntologySet \cup \{CoreOntology!\}$

*MergingInPluginAndSubsumptionImport* $\widehat{=}$
*ClassMergingInPluginAndSubsumptionImport*
$\S$ *ObjectPropertyMergingInPluginAndSubsumptionImport*
$\S$ *DataTypePropertyMergingInPluginAndSubsumptionImport*
$\S$ *IndividualMergingInPluginAndSubsumptionImport*

***IndividualMergingInIntersectionImport***

O esquema a seguir define as pré-condições necessárias à formação de uma ontologia-núcleo (*core ontology*) quando a importação é parcial em ambas as ontologias de entrada. Neste caso, a ontologia gerada na fusão será composta pela ontologia importada e pela intersecção na propagação de propriedades, de acordo com os axiomas de ligação de conceitos e propriedades. Este é o caso de maior nível de complexidade no processo de fusão. Foram utilizadas compreensões de conjuntos para a definição de novas ligações entre os conceitos. As operações de fusão para intersecção de ontologias importadas são semelhantes para classes, *datatype properties* e indivíduos. A diferença consiste

basicamente nos axiomas utilizados, que difere para cada um dos quatro casos (vide Apêndice). O processo completo é definido pelo esquema composto *MergingInIntersectionImport.*

*IndividualMergingInIntersectionImport*

$\Delta Ontologies$
$\Xi IndividualAxioms$
$O1?, O2?, CoreOntology!: Ontology$

---

$\exists IndividualAxiomSet: IndividualAxioms; ImportedOntology: Ontology$
$\mid IndividualAxiomSet \in CoreOntology!\ .\ OntologicalIndividualAxioms$
$\wedge\ O1? \in OntologySet$
$\wedge\ O2? \in OntologySet$
$\wedge\ ImportedOntology \in OntologySet$
$\wedge\ CoreOntology! \notin OntologySet$
$\wedge\ ImportedOntology\ .\ Individuals \subseteq O1?\ .\ Individuals$
$\wedge\ ImportedOntology\ .\ Individuals \subseteq O2?\ .\ Individuals$
$\bullet\ IndividualAxiomSet\ .\ SameAs$
$= \{\ i1, i2, i3: Individual$
$\mid i1 \in ImportedOntology\ .\ Individuals$
$\wedge\ i2 \in O1?\ .\ Individuals$
$\wedge\ i3 \in O2?\ .\ Individuals$
$\wedge\ (i1\ SameAs\ i2 \wedge i1\ SameAs\ i3) \bullet (i2, i3)\ \}$
$\wedge\ IndividualAxiomSet\ .\ DifferentFrom$
$= \{\ i1, i2, i3: Individual$
$\mid i1 \in ImportedOntology\ .\ Individuals$
$\wedge\ i2 \in O1?\ .\ Individuals$
$\wedge\ i3 \in O2?\ .\ Individuals$
$\wedge\ (i1\ SameAs\ i2 \wedge i1\ DifferentFrom\ i3)$
$\vee\ i1\ DifferentFrom\ i2 \wedge i1\ DifferentFrom\ i3 \bullet (i2, i3)\ \}$
$\wedge\ CoreOntology!\ .\ OntologicalIndividualAxioms$
$= CoreOntology!\ .\ OntologicalIndividualAxioms \cup \{IndividualAxiomSet\}$
$\wedge\ OntologySet' = OntologySet \cup \{CoreOntology!\}$

---

$MergingInIntersectionImport \mathrel{\widehat{=}} ClassMergingInIntersectionImport$
$\S\ ObjectPropertyMergingInIntersectionImport$
$\S\ DataTypePropertyMergingInIntersectionImport$
$\S\ IndividualMergingInIntersectionImport$

## *B INTEGRAÇÃO*

A integração constitui o processo mais simples de formação de ontologias-núcleo. Neste caso, ocorre apenas uma união generalizada de conceitos, propriedades e instâncias, sem análise detalhada das intersecções possíveis. Em outras palavras, ocorre o reuso de

fragmentos de ontologias, ainda que os conceitos sejam divergentes. O processo completo de integração é representado pelo esquema composto *OntologyIntegration*. Para descrição detalhada dos outros processos, vide Apêndice.

*ObjectPropertyIntegration*

$\Delta Ontologies$
$O1?, O2?, CoreOntology!: Ontology$

---

$O1? \in OntologySet$
$O2? \in OntologySet$
$CoreOntology! \notin OntologySet$
$CoreOntology!.ObjectProperties$
$= O1?.ObjectProperties \cup O2?.ObjectProperties$
$OntologySet' = OntologySet \cup \{CoreOntology!\}$

$OntologyIntegration \mathrel{\widehat{=}}$
$ClassIntegration \fatsemi ObjectPropertyIntegration \fatsemi DataTypePropertyIntegration$
$\fatsemi IndividualIntegration$

### *C ALINHAMENTO*

A operação de alinhamento gera um conjunto de mapeamentos entre os conceitos e propriedades das ontologias de origem. Estes mapeamentos permitem uma análise mais detalhadas das intersecções de interesse, constituindo um processo semi-automático, em termos operacionais. O esquema *MappingSet* define um conjunto de mapeamentos na forma de relações. O alinhamento difere da fusão no sentido de que não pressupõe o compartilhamento de uma ontologia importada entre as ontologias de origem. O processo é iniciado com a definição de uma relação de equivalência entre dois conceitos ou propriedades quaisquer das ontologias. O axioma de equivalência é utilizado preferencialmente visto que representa a relação mais forte no caso de semelhança entre classes e propriedades (no caso de indivíduos é utilizado o axioma *SameAs*). O esquema composto *OntologyAlignment,* engloba o alinhamento de classes, propriedades e indivíduos (para verificação do detalhamento dos esquemas relacionados, vide Apêndice).

*MappingSet*

$O1?, O2?: Ontology$
$ClassMappings: CLASS \leftrightarrow CLASS$
$ObjectPropertyMappings: ObjectProperty \leftrightarrow ObjectProperty$
$DataTypePropertyMappings: DataTypeProperty \leftrightarrow DataTypeProperty$
$IndividualMappings: Individual \leftrightarrow Individual$

---

$\forall c1, c2: CLASS; objp1, objp2: ObjectProperty; dtp1, dtp2: DataTypeProperty;$
$i1, i2: Individual$
$\bullet\ c1\ ClassMappings\ c2$
$\Leftrightarrow (c1 \in O1?\ .\ Classes \land c2 \in O2?\ .\ Classes$
$\lor c2 \in O1?\ .\ Classes \land c1 \in O2?\ .\ Classes)$
$\land\ objp1\ ObjectPropertyMappings\ objp2$
$\Leftrightarrow (objp1 \in O1?\ .\ ObjectProperties \land objp2 \in O2?\ .\ ObjectProperties$
$\lor objp2 \in O1?\ .\ ObjectProperties \land objp1 \in O2?\ .\ ObjectProperties)$
$\land\ dtp1\ DataTypePropertyMappings\ dtp2$
$\Leftrightarrow (dtp1 \in O1?\ .\ DataTypeProperties \land dtp2 \in O2?\ .\ DataTypeProperties$
$\lor dtp2 \in O1?\ .\ DataTypeProperties \land dtp1 \in O2?\ .\ DataTypeProperties)$
$\land\ i1\ IndividualMappings\ i2$
$\Leftrightarrow i1 \in O1?\ .\ Individuals \land i2 \in O2?\ .\ Individuals$
$\lor i2 \in O1?\ .\ Individuals \land i1 \in O2?\ .\ Individuals$

*ClassAlignment*

$\Xi Ontologies$
$\Xi ClassAxioms$
$O1?, O2?: Ontology$
$MapSet!: MappingSet$
$Report!: REPORT$

---

$MapSet!\ .\ ClassMappings$
$= \{\ c1, c2, c3: CLASS; EquivalentClass: CLASS \leftrightarrow CLASS$
$\mid\ O1? \in OntologySet \land O2? \in OntologySet$
$\land\ c1 \in \text{dom}\ EquivalentClass$
$\land\ c2 \in \text{ran}\ EquivalentClass$
$\land\ c3 \in \text{ran}\ EquivalentClass$
$\land\ \text{dom}\ EquivalentClass \subseteq O1?\ .\ Classes$
$\land\ \text{ran}\ EquivalentClass \subseteq O2?\ .\ Classes$
$\land (c2, c3) \in isSuperClassOf \lor (c3, c2) \in isSuperClassOf$
$\lor (c2, c3) \in isSubClassOf\ \lor (c3, c2) \in isSubClassOf$
$\lor (c2, c3) \in EquivalentClass \lor (c2, c3) \in isSiblingClassOf \bullet\ (c1, c3)\ \}$
$MapSet!\ .\ ClassMappings = \{\} \Rightarrow Report! = Impasse$
$MapSet!\ .\ ClassMappings \neq \{\} \Rightarrow Report! = Match$

---

$OntologyAlignment\ \widehat{=}$
$ClassAlignment \fatsemi ObjectPropertyAlignment \fatsemi DataTypePropertyAlignment$
$\fatsemi\ IndividualAlignment$

### 4.4.2 OPERAÇÕES SOBRE SERVIÇOS

As operações sobre serviços em PERSONÆ incluem a descoberta e seleção de serviços. Idealmente, a fase de estabelecimento de contrato de serviço pressupõe uma negociação prévia sobre níveis de qualidade desejada. Esta tarefa consiste na análise de relações de convergência ou divergência entre níveis de QoS desejados e oferecidos. Portanto, o modelo aqui definido para a fase de estabelecimento de contrato visa tão somente propor o mecanismo de integração para o estabelecimento de uma representação integrada para contrato.

#### *A DESCOBERTA SINTÁTICA*

A descoberta sintática de serviços consiste basicamente na associação simples de parâmetros, sem considerações sobre relações de similaridades. Este é o mecanismo mais comum de busca de serviços. O esquema *SyntacticalServiceDiscovery* sumariza este processo, onde um tipo de dado comum (e.g. o nome do serviço) pode ser fornecido para a realização da busca. O resultado é um conjunto de referências possíveis retornadas do registro, representada por uma compreensão de conjuntos. Conforme poderá ser observado nos esquemas que se seguem, uma variável de saída é utilizada para indicar o retorno dos processos de busca e seleção. Considerando que o tratamento de exceções constitui uma preocupação de nível operacional, tal aspecto foi suprimido desta especificação.

*SyntacticalServiceDiscovery*

$\Xi PERSONÆ$
$ServiceName?: DATATYPE$
$ServiceDiscoveredSet!: ID \pfun ServiceDescription$
$Response!: REPORT$

---

$ServiceDiscoveredSet!$
$= \{\ Registry: ServiceRegistry;\ id: ID;\ SD: ServiceDescription$
$\quad \mid Registry \in Registries$
$\quad\quad \land (id, SD) \in Registry\ .\ Services$
$\quad\quad \lor ServiceName? = SD\ .\ ServiceName \bullet (id, SD)\ \}$
$ServiceDiscoveredSet! = \{\} \Rightarrow Response! = NoServiceDiscovered$
$ServiceDiscoveredSet! \neq \{\} \Rightarrow Response! = ServiceDiscovered$

#### *B DESCOBERTA SEMÂNTICA*

A descoberta sintática é baseada na associação exata de parâmetros e, portanto, não considera outros níveis de similaridades entre conceitos, no caso de associação parcial. Neste sentido a descoberta semântica amplia o potencial da descoberta de serviços,

mediante a associação de termos relacionados com a funcionalidade desejada (*goals*) e a funcionalidade oferecida (*capabilities*). A busca semântica ainda pode ser refinada pela verificação de compatibilidade entre os tipos de dados de entrada e saída fornecidos pela aplicação cliente e aqueles disponibilizados pelo serviço.

Os esquemas *GoalOrientedServiceDiscovery* e *InputOrientedServiceDiscovery* definem o início do procedimento de descoberta semântica de serviços. Conforme mencionado anteriormente, requisitos funcionais podem ser definidos segundo uma ontologia. Neste caso, uma composição seqüencial pode ser utilizada para associação do primeiro esquema com operações de fusão de ontologias. A composição seqüencial é representada neste caso pelo operador "⨟" (ou *pipeline*). O objetivo é recuperar o máximo número possível de similaridades entre os conceitos. No caso da busca com base em parâmetros de entrada ou saída (e.g. *InputOrientedServiceDiscovery*), o fusão pode ocorrer apenas em nível de propriedades, visto que entradas e saídas são representadas por propriedades de tipos de dados.

*GoalOrientedServiceDiscovery*

$$
\begin{array}{l}
\Xi PERSONÆ \\
Goals?: FunctionalRequirement \\
ServiceSet!: ID \rightarrow\!\!\!\!\!\!\bullet\ \ ServiceDescription \\
O1!, O2!: Ontology \\
\hline
\exists GoalClasses, CapabilityClasses: \mathbb{P}\ CLASS \\
\quad \mid GoalClasses = \{\ fr: FunctionalRequirement \mid fr = Goals? \bullet fr\ .\ Class\ \} \\
\quad \wedge\ CapabilityClasses \\
\qquad = \{\ Registry: ServiceRegistry;\ id: ID;\ SD: ServiceDescription; \\
\qquad\quad fr: FunctionalRequirement \mid Registry \in Registries \wedge (id, SD) \in Registry\ .\ Services \\
\qquad\qquad \wedge\ fr = SD\ .\ Capability \bullet fr\ .\ Class\ \} \\
\quad \wedge\ O1! \in OntologySet \wedge O2! \in OntologySet \\
\quad \bullet\ O1!\ .\ Classes = GoalClasses \wedge O2!\ .\ Classes = CapabilityClasses
\end{array}
$$

*InputOrientedServiceDiscovery*

$$
\begin{array}{l}
\Xi PERSONÆ \\
Input?: DataTypeProperty \\
ServicesSet!: ID \rightarrow\!\!\!\!\!\!\bullet\ \ ServiceDescription \\
O1!, O2!: Ontology \\
\hline
\exists DtPropertySet: \mathbb{P}\ DataTypeProperty \\
\quad \mid DtPropertySet = \{\ Registry: ServiceRegistry;\ id: ID;\ SD: ServiceDescription; \\
\qquad input: DataTypeProperty \\
\qquad\quad \mid Registry \in Registries \\
\qquad\quad \wedge\ (id, SD) \in Registry\ .\ Services \wedge input \in SD\ .\ Input \bullet input\ \} \\
\quad \wedge\ O1! \in OntologySet \wedge O2! \in OntologySet \\
\quad \bullet\ O1!\ .\ DataTypeProperties = O1!\ .\ DataTypeProperties \cup \{Input?\} \\
\quad \wedge\ O2!\ .\ DataTypeProperties = DtPropertySet
\end{array}
$$

O processo completo de descoberta (vide Figura 4.2) é formalizado pelo esquema composto *ServiceDiscovery*, mediante o uso combinado de disjunção dos esquemas de descoberta baseados em goals, entradas e saídas (*Input/OutputOrientedServiceDiscovery*). As disjunções são compostas com uma operação completa de fusão de ontologias, para a integração das similaridades que podem ser recuperadas na descoberta de serviços associados com a requisição do cliente.

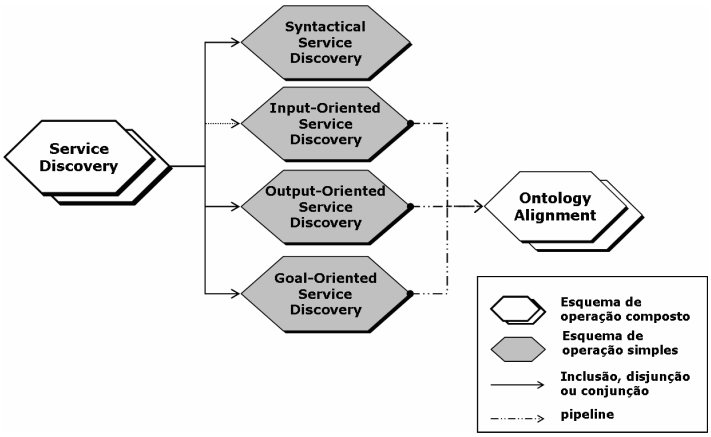



**Figura 4.2 – Fluxo de operações do esquema composto *ServiceDiscovery***

$$
\begin{aligned}
&\mathit{ServiceDiscovery} \mathrel{\widehat{=}} \\
&\quad \mathit{SyntacticalServiceDiscovery} \\
&\quad \vee (\mathit{GoalOrientedServiceDiscovery} \vee \mathit{InputOrientedServiceDiscovery} \\
&\qquad \vee \mathit{OutputOrientedServiceDiscovery}) \gg \mathit{OntologyMerging}
\end{aligned}
$$

### *C SELEÇÃO SINTÁTICA*

De forma análoga à descoberta sintática, a seleção sintática está baseada na comparação simples de parâmetros sintáticos. No esquema *SyntacticalServiceSelection*, são realizadas duas compreensões de conjuntos. A primeira é similar à descoberta sintática, a qual representa a recuperação de descrições de um determinado serviço que atenda à funcionalidade desejada, com base nos requisitos funcionais desejados (pela variável *goals?*). A segunda toma como base os resultados retornados da primeira e representa a associação de parâmetros de requisitos funcionais (*QoSDesired?* e *QoSProvided?*). O retorno do processo é representado pela variável de saída *ServiceSelectedSet!*, a qual constitui o conjunto dos identificadores de serviço selecionados.

*SyntacticalServiceSelection*

$\Xi PERSONÆ$
$Goals?: FunctionalRequirement$
$QoSDesired?: NonFunctionalRequirement \rightarrow\!\!\!\!\!\!\bullet\ LEVEL$
$ServiceSelectedSet!: \mathbb{P}\ ID$
$Response!: REPORT$

---

$\exists ServiceSet: ID \rightarrow\!\!\!\!\!\!\bullet\ ServiceDescription$
$\mid ServiceSet$
$= \{\ Registry: ServiceRegistry;\ id: ID;\ SD: ServiceDescription$
$\mid Registry \in Registries$
$\wedge\ (id, SD) \in Registry\ .\ Services \wedge Goals? = SD\ .\ Capability \bullet\ (id, SD)\ \}$
$\wedge\ ServiceSelectedSet!$
$= \{\ id: ID;\ SD: ServiceDescription$
$\mid (id, SD) \in ServiceSet \wedge QoSDesired? = SD\ .\ QoSProvided \bullet\ id\ \}$
$\bullet\ ServiceSelectedSet! = \{\} \Rightarrow Response! = NoServiceSelected$
$\wedge\ ServiceSelectedSet! \neq \{\} \Rightarrow Response! = ServiceSelected$

## *D SELEÇÃO SEMÂNTICA*

A fase de seleção semântica caracteriza o refinamento da busca por serviços com base nos requisitos de QoS definidos pelo usuário. Em PERSONÆ, a entidade *ServiceRequester* representa uma aplicação cliente que recebe como entrada estes requisitos pessoais. Desta forma, o processo de seleção está diretamente relacionado com a provisão de serviços personalizados. Isto implica a projeção de múltiplas percepções de qualidade de serviço sobre os níveis de qualidade associados com requisitos não-funcionais que caracterizam um determinado serviço e seu provedor.

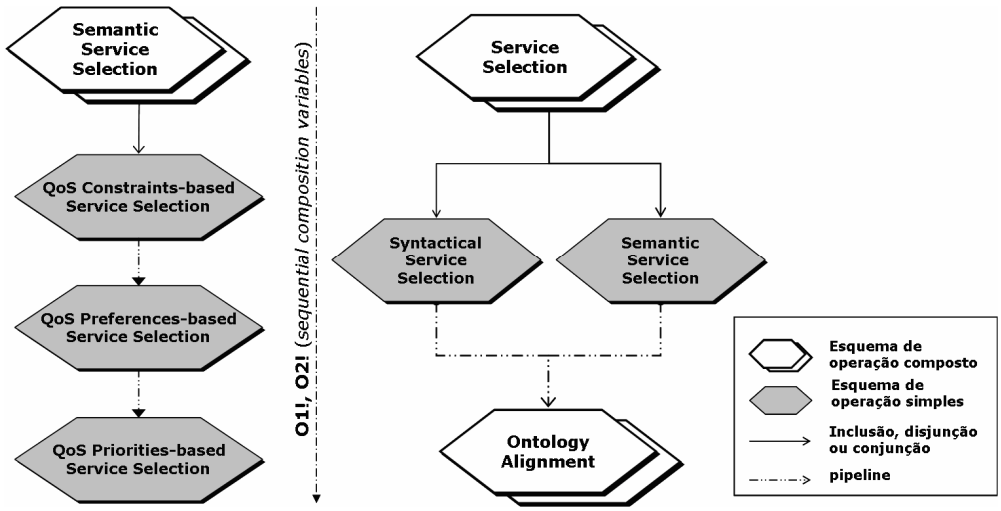



**Figura 4.3 – Fluxo de operações dos esquemas *SemanticServiceSelection* e *ServiceSelection***

A seleção semântica (vide Figura 4.3) é representada pelo esquema composto *SemanticServiceSelection*, que representa a composição seqüencial de três esquemas: (1) *QoSConstraintsBasedSelection*, que representa o primeiro filtro de QoS, excluindo requisitos não-funcionais indesejados pelo usuário; (2) *QoSPreferencesBasedSelection*, que considera as preferências do usuário; e (3) *QoSPrioritiesBasedSelection*, que define uma relação de prioridade entre dois requisitos não-funcionais, no caso de impossibilidade de oferecimento desses requisitos simultaneamente.

O esquema *QoSConstraintsBasedSelection* é apresentado a seguir, cujo predicado consiste na aplicação de duas compreensões de conjuntos. A primeira identifica precisamente a classe do requisito funcional que representa uma determinada restrição. A segunda identifica esta mesma restrição nas descrições de serviços apontadas pelo registro. Conforme dito anteriormente, o modelo aqui definido parte do pressuposto de que as relações entre requisitos não-funcionais e a especificação de níveis de qualidade oferecidos estão representados em uma ontologia. Desta forma, as compreensões de conjuntos precedem uma subtração das classes de requisitos que foram restringidos pelo usuário.

*QoSConstraintsBasedServiceSelection*

$$
\begin{array}{l}
\Xi PERSONÆ \\
QoSConstraints?: NonFunctionalRequirement \rightarrow\!\!\!\!\!\!\bullet\ \ LEVEL \\
ServiceSet!: ID \rightarrow\!\!\!\!\!\!\bullet\ \ ServiceDescription \\
O1!: Ontology \\
\hline
\exists NFRConstrainedClasses, NFROfferedClasses: \mathbb{P}\ CLASS \\
\quad \mid NFRConstrainedClasses \\
\qquad = \{\ nfr: NonFunctionalRequirement \mid nfr \in \mathrm{dom}\ QoSConstraints? \\
\qquad\quad \bullet\ nfr\ .\ Class\ \} \\
\quad \wedge NFROfferedClasses \\
\qquad = \{\ Registry: ServiceRegistry;\ id: ID;\ SD: ServiceDescription; \\
\qquad\quad nfr: NonFunctionalRequirement \\
\qquad\qquad \mid Registry \in Registries \\
\qquad\qquad\quad \wedge (id, SD) \in Registry\ .\ Services \\
\qquad\qquad\quad \wedge nfr \in \mathrm{dom}\ SD\ .\ QoSProvided \bullet\ nfr\ .\ Class\ \} \\
\quad \wedge O1! \in OntologySet \\
\quad \wedge NFRConstrainedClasses \subseteq NFROfferedClasses \\
\quad \bullet\ O1!\ .\ Classes = NFROfferedClasses \setminus NFRConstrainedClasses
\end{array}
$$

O esquema *QoSPreferencesBasedServiceSelection* é definido de forma semelhante (vide Apêndice). Após a aplicação deste segundo filtro, as relações de prioridade são consideradas. No esquema *QoSPrioritiesBasedServiceSelection*, as compreensões de

conjuntos precedem a atribuição de relações de igualdade entre os requisitos selecionados (*QoSPriorities*) e os requisitos oferecidos pelo provedor (*QoSOffered*). As variáveis de saída *O1!* e *O2!*, representam as ontologias que contém os requisitos a serem alinhados. O esquema finaliza com atribuição dos resultados das compreensões de conjuntos nas ontologias de saída.

O esquema composto *ServiceSelection* representa o processo de seleção concatenado com um alinhamento de ontologias. Sendo assim, os filtros de QoS refinam as classes de requisitos não-funcionais para simplificação do processo de alinhamento de ontologias. Desta forma, somente serão considerados os alinhamentos encontrados nos requisitos que definem as prioridades. O alinhamento permite que a seleção seja realizada com base apenas nas similaridades de interesse do usuário.

*QoSPrioritiesBasedServiceSelection*

$$
\begin{array}{l}
\Xi PERSONÆ \\
QoSPriorities?: NonFunctionalRequirement \times NonFunctionalRequirement \\
\qquad \pfun NonFunctionalRequirement \\
ServiceSet!: ID \pfun ServiceDescription \\
O1!, O2!: Ontology \\
\hline
\exists NFRPriorityClasses, NFROfferedClasses: \mathbb{P}\ CLASS \\
\quad \mid NFRPriorityClasses \\
\qquad = \{\ nfr: NonFunctionalRequirement \mid nfr \in \mathrm{ran}\ QoSPriorities? \\
\qquad\quad \bullet\ nfr\ .\ Class\ \} \\
\quad \land NFROfferedClasses \\
\qquad = \{\ Registry: ServiceRegistry; id: ID; SD: ServiceDescription; \\
\qquad\quad nfr: NonFunctionalRequirement \\
\qquad\quad\quad \mid Registry \in Registries \\
\qquad\quad\quad\quad \land (id, SD) \in Registry\ .\ Services \\
\qquad\quad\quad\quad \land nfr \in \mathrm{dom}\ SD\ .\ QoSProvided \bullet\ nfr\ .\ Class\ \} \\
\quad \land O1! \in OntologySet \\
\quad \land O2! \in OntologySet \\
\quad \land NFRPriorityClasses \subseteq NFROfferedClasses \\
\quad \bullet\ O1!\ .\ Classes = NFRPriorityClasses \land O2!\ .\ Classes = NFROfferedClasses
\end{array}
$$

$$
\begin{array}{l}
SemanticServiceSelection \mathrel{\widehat=} QoSConstraintsBasedServiceSelection \\
\fatsemi QoSPreferencesBasedServiceSelection \\
\fatsemi QoSPrioritiesBasedServiceSelection
\end{array}
$$

$$
\begin{array}{l}
ServiceSelection \mathrel{\widehat=} \\
SyntacticalServiceSelection \lor (SemanticServiceSelection \gg OntologyAlignment)
\end{array}
$$

### *E ESTABELECIMENTO DE CONTRATO*

Um contrato de serviço registra o nível de qualidade de serviço acordada (*QoSAgreed*), e os perfis das entidades envolvidas (vide esquemas *ServiceRequester* e *ServiceProvider*). Durante a fase de seleção, os filtros representados pelos esquemas *QoSConstraintsBasedServiceSelection*, *QoSPreferencesBasedSelection* e *QoSPrioritiesBasedSelection* constituem basicamente um refinamento dos requisitos de qualidade definidos pelo cliente sobre o nível de qualidade oferecida por um serviço.

Vale salientar que o nível de qualidade oferecida pelo serviço (*QoSOffered*) pode ser diferente do nível de qualidade realmente provida (*QoSProvided)*. O poder de barganha no cliente junto ao provedor pode depender da exploração desta diferença. Esta fase consiste na negociação de serviços, a qual não será abordada neste trabalho. Entretanto, a combinação das técnicas de integração e alinhamento de ontologias pode ser utilizada como abordagem simplificada no estabelecimento de contratos, após o processo de seleção, o que também poderia ser chamado de estabelecimento de contrato "pacífico".

O esquema *QoSAlignment* representa a primeira fase do processo de estabelecimento de contrato. A parte predicativa analisa requisitos não-funcionais dois a dois (*nfr1* e *nfr2*), pertencentes ao conjunto de prioridades do cliente (*QoSPriorities*, de *Requester?*) e do conjunto de requisitos de qualidade do serviço (*QoSProvided*). As classes destes requisitos são atribuídas às variáveis de *pipeline* (*O1!* e *O2!*). Conforme o esquema composto *ContractEstablishment*, este esquema é concatenado com um esquema de alinhamento (*OntologyAlignment*), para verificação das similaridades conceituais. É importante destacar que esta abordagem consiste numa alternativa ao estabelecimento de contrato, somente aplicado ao caso de aceitação pacífica, por parte de provedores e clientes, da intersecção encontrada entre *QoSPriorities* e *QoSProvided*.

*QoSAlignment*

$\Delta PERSONÆ$
*Requester?: ServiceRequester*
*Provider?: ServiceProvider*
*O1!, O2!: Ontology*

---

$\exists s: Service; nfr1, nfr2: NonFunctionalRequirement$
| $Requester? \in ServiceRequesters \land Provider? \in ServiceProviders$
$\land\ s \in Provider?\ .\ Services \land nfr1 \in \text{ran}\ Requester?\ .\ QoSPriorities$
$\land\ nfr2 \in \text{dom}\ s\ .\ Description\ .\ QoSProvided$
$\land\ O1! \in OntologySet \land O2! \in OntologySet$
• $O1!\ .\ Classes = O1!\ .\ Classes \cup \{nfr1\ .\ Class\}$
$\land\ O2!\ .\ Classes = O2!\ .\ Classes \cup \{nfr2\ .\ Class\}$

O alinhamento de ontologias precede uma composição seqüencial com o esquema *ProfileIntegration*. Nesta fase, uma variável auxiliar (representada por *nfr3*) é utilizada para receber os valores dos mapeamentos das classes de *nfr1* e *nfr2* (que estão contidas em *QoSPriorities* e *QoSProvided*, por generalização no esquema anterior). A variável *nfr3* define o nível de qualidade acordada (*QoSAgreed*). Em seguida, a variável *Contract!* (que representa uma instância de SLA) é atualizada pela atribuição do nível de qualidade acordada (*QoSAgreed!*). As variáveis de *pipeline O1!* e *O2!* também são atualizadas na atribuição dos perfis do cliente e do provedor. A próxima fase consiste justamente na concatenação com um processo de integração de ontologias, que integra os perfis das entidades envolvidas.

*ProfileIntegration*

$\Delta PERSONÆ$
*Requester?: ServiceRequester*
*Provider?: ServiceProvider*
*O1!, O2!: Ontology*
*QoSAgreed!: NonFunctionalRequirement* $\rightarrow\!\!\!\!\!\!\rightarrow$ *LEVEL*
*Contract!: ServiceLevelAgreement*
*MapSet?: MappingSet*

---

$\exists$ *nfr1, nfr2, nfr3: NonFunctionalRequirement; c1, c2: CLASS*
| *MapSet? . ClassMappings* $\neq \emptyset$
$\wedge$ *nfr3* $\in$ dom *QoSAgreed!*
$\wedge$ (*c1*, *c2*) $\in$ *MapSet? . ClassMappings*
$\wedge$ *Contract!* $\notin$ *ServiceLevelAgreements*
• *nfr3 . Class = c1* $\vee$ *nfr3 . Class = c2*
$\wedge$ *Contract! . QoSAgreed = QoSAgreed!*
$\wedge$ *O1! = Requester? . Profile*
$\wedge$ *O2! = Provider? . Profile*
$\wedge$ *ServiceLevelAgreements' = ServiceLevelAgreements* $\cup$ {*Contract!*}

O esquema *ContractIntegration* representa a fase final do processo de estabelecimento de contrato, onde a ontologia de SLA (*SLAOntology*) é gerada pela integração dos perfis do cliente e do provedor. Finalmente, a variável *ServiceLevelAgreements* é atualizada pela criação de um novo contrato (*Contract!*). A concatenação de processos foi utilizada para simplificar a formação de contrato mediante a utilização de duas técnicas diferentes de reconciliação ontológica: integração de perfis e alinhamento de requisitos de QoS.

*ContractIntegration*

$\Delta PERSONÆ$
*Contract!: ServiceLevelAgreement*
*CoreOntology?: Ontology*

---

$Contract!.SLAOntology = Contract!.SLAOntology \cup \{CoreOntology?\}$
$ServiceLevelAgreements' = ServiceLevelAgreements \cup \{Contract!\}$

Por fim, o conjunto de atividades executadas através de uma aplicação PERSONÆ, é formalmente definido pela composição seqüencial das fases de descoberta (*ServiceDiscovery*), seleção (*ServiceSelection*) e estabelecimento de contrato de serviços (*ContractEstablishment*):

$PERSONÆApplication \mathrel{\widehat=}$
$ServiceDiscovery \fatsemi ServiceSelection \fatsemi ContractEstablishment$

## 4.5 DIRETRIZES PARA REFINAMENTO E PROVA

O uso da notação Z na descrição da funcionalidade de sistemas computacionais complexos compreende basicamente as fases de especificação, refinamento e prova (WOOD, 1993), (WOODCOCK; DAVIES, 1996). A primeira fase consiste na definição precisa de requisitos funcionais e aspectos comportamentais de um sistema. A segunda fase consiste na minimização do indeterminismo ou incerteza das operações especificadas. Neste sentido, algumas técnicas como cálculo de pré-condições, cálculo funcional de esquemas ou funções e/ou relações de recuperação (*retrieve relations/functions*, respectivamente) podem ser utilizadas para a tradução de uma especificação mais abstrata em código executável. A terceira fase consiste basicamente na construção de teoremas que permitam a verificação da aplicabilidade e corretude das operações mais concretas (em nível de implementação) que refinam as operações mais abstratas (nível de especificação e projeto).

A especificação de operações de um sistema em nível abstrato geralmente baseia-se em operadores matemáticos elementares. Isto confere maior clareza na definição dos conceitos envolvidos. Os principais elementos para definição de postulados na notação Z são os conjuntos (*given sets*, elementos básicos na definição de tipos utilizados), variáveis (definidas a partir dos conjuntos), expressões ou axiomas (para definição dos valores que as variáveis podem assumir) e predicados (que estabelecem restrições sobre as variáveis contidas nas expressões). Estas estruturas elementares podem ser combinadas para a definição de outras estruturas mais complexas, mais próximas de estruturas de dados reais, que compõem a especificação em nível de projeto.

As principais estruturas em nível de projeto são as tuplas (para definição de registros ou produtos cartesianos para cruzamento de tabelas), relações e funções (que podem agregar conjuntos de tuplas, úteis na modelagem de tabelas ou bancos de dados), seqüências (para definição de estruturas mais complexas como árvores, pilhas e filas) e esquemas (para formação de tipos complexos, com base na agregação de variáveis de tipos heterogêneos e definição de restrições sobre estas variáveis). A combinação destas estruturas com um conjunto de operadores matemáticos definidos pela notação agrega um nível de expressividade suficiente para a modelagem de estruturados em nível de projeto. O refinamento destas estruturas é direcionado à geração de código executável, em nível de implementação (vide Figura 4.4).

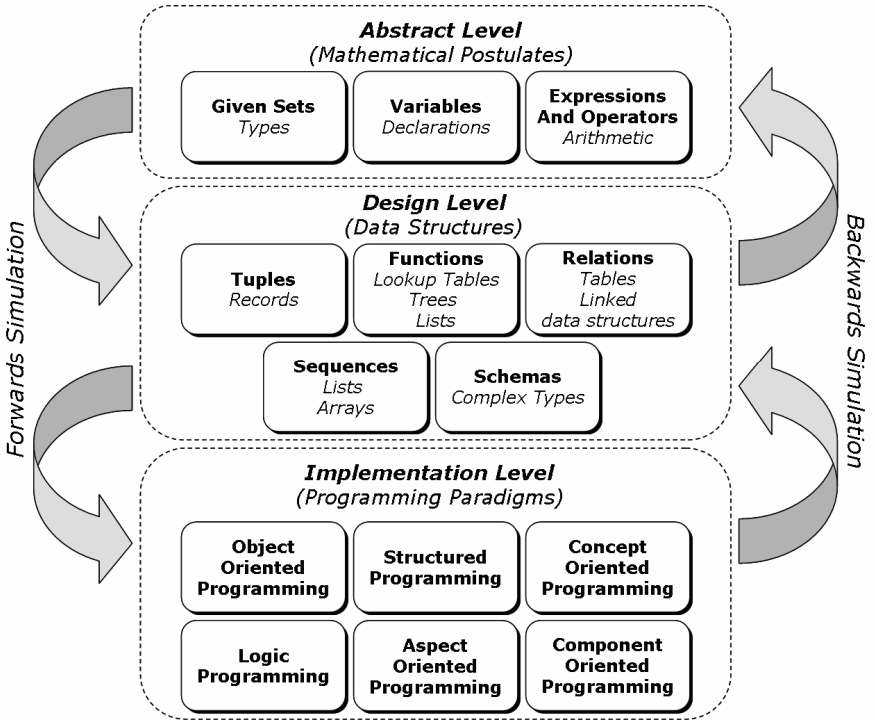



**Figura 4.4 – Mapeamentos de estruturas de especificação em Z**

A adequação da notação Z ao paradigma orientado a objetos é um aspecto discutido em alguns trabalhos (BOWEN, 1996), (JACKY, 1997), (POTTER; SINCLAIR; TILL, 1996). Por exemplo, os conjuntos assemelham-se às definições de tipos de variáveis, assim como esquemas de estado abstrato e esquemas de operação são semelhantes às definições de classes e métodos, respectivamente, Entretanto, combinações diferentes de técnicas de transformação podem conduzir à geração de código pertinente a outros paradigmas, tais como orientação a aspectos (pela modularização e separação de conceitos) (KICZALES, 1997), orientação a

conceitos (como a utilização de ontologias na especificação de modelos conceituais), orientação a componentes (pela combinação de teorias como orientação a objetos, *frameworks* e padrões de projeto) (SZYPERSKI, 2002) ou programação lógica (pela resolução de problemas com auxílio de provadores de teoremas) (LLOYD, 1987). A tradução de uma especificação abstrata em código executável é denominada simulação com encadeamento para frente (*forwards simulation*). O processo inverso, o qual pode implicar a redefinição conceitual do modelo, consiste na simulação com encadeamento para trás (*backwards simulation*).

A especificação funcional em nível abstrato corresponde à perspectiva do usuário quanto à utilização do sistema (vide Figura 4.5). No caso da tarefa de seleção de serviços, por exemplo, um usuário forneceria apenas os requisitos de qualidade de seu interesse, mas não estaria interessado em detalhes de comunicação relacionados com a execução do serviço. Desta forma, detalhes específicos de implementação são suprimidos neste nível. Suponha que a perspectiva do usuário seja definida por um estado abstrato *AS*, então cada operação abstrata será descrita em termos dos estados abstratos antecedentes (representados por *AS*) e pelos seus subseqüentes (representados pelo estado *AS'*). Além disso, um estado inicial válido para todas as operações deve ser definido, para garantir as condições necessárias à inicialização do sistema, (representado pelo esquema *AI*).

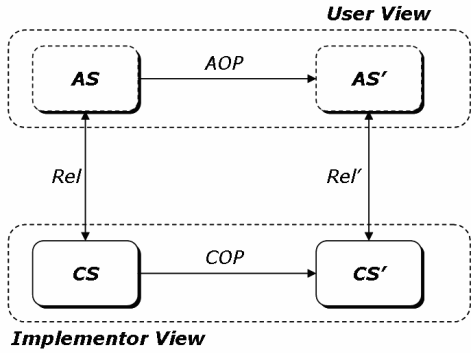



**Figura 4.5 – Relacionamento entre esquemas abstratos e concretos (BOWEN, 1996)**

A especificação funcional em nível de projeto de implementação corresponde à visão do programador sobre o sistema (vide Figura 4.5). Esta representação deve apresentar um maior nível de detalhe quando a aspectos operacionais (e.g. tratamento de exceções, estruturas de dados utilizadas ou tipo de paradigma de programação utilizado). Estes aspectos fornecem uma visão mais detalhada sobre o comportamento interno dos componentes do sistema. Suponha-se que a perspectiva do programador seja definida por um esquema concreto *CS*, então cada operação concreta será descrita em termos dos estados concretos antecedentes (representados por *CS*) e pelos seus subseqüentes (representados

pelo esquema *CS'*). A inicialização ocorre de forma análoga ao caso da perspectiva do usuário (onde o esquema de inicialização é representado com *CI*).

A verificação da compatibilidade entre as perspectivas do usuário e do programador consiste na verificação dos teoremas da inicialização, da aplicabilidade e da corretude. As expressões que postulam estes teoremas variam de acordo com o tipo de mapeamento utilizado. Considerando que PERSONÆ constitui um modelo definido em nível abstrato (i.e não orientado a um paradigma em específico, em conformidade com os princípios de SOA), a estratégia de simulação com encadeamento para frente (*forwards simulation*) seria a mais adequada. Os teoremas correspondentes são estabelecidos com base no relacionamento entre os esquemas de estado (*AS, AS', CS* e *CS*'), os esquemas de operações (*AOp* e *COp*) e na relação de recuperação (*Rel* e *Rel'*, que associam visões abstratas e concretas de estados e operações).

Para cada operação abstrata *AOp*, deve existir uma operação concreta correspondente *COp*, a qual é aplicável ao domínio correspondente da operação abstrata, produzindo um resultado que satisfaz a especificação em nível abstrato. Em outras palavras, se *AOp* modifica o estado do sistema de *AS* para *AS'*, então a operação concreta correspondente *COp* deve, dado um estado inicial *CS*, o qual está relacionado com *AS*, mediante a relação *Rel*, produz um novo estado *CS'*, o qual está relacionado com *AS'*, mediante a relação *Rel'*. Estas relações são formalizadas segundo os teoremas abaixo:

*Teorema da Inicialização*

$\forall\ CS' \bullet CI \Rightarrow \exists\ AS' \bullet AI \wedge Rel'$

*Teorema da Aplicabilidade*

$\forall\ AS; CS \bullet \text{pre}\ AOp \wedge Rel \Rightarrow \text{pre}\ COp$

*Teorema da Corretude*

$\forall\ AS; CS; CS' \bullet \text{pre}\ AOp \wedge Rel \wedge COp \Rightarrow \exists AS' \bullet AOp \wedge Rel'$

O operador "pre" fornece as pré-condições que definem o correto funcionamento da operação definida pelo esquema, no qual todas as variáveis de estado posterior, decoradas pelo símbolo apóstrofo (e.g. *ServiceLevelAgreements'*) e as variáveis de saída (terminadas com "**!**") são quantificadas existencialmente. As variáveis de entrada e saída foram suprimidas para fins de simplicidade da apresentação. Desta forma, o estado concreto é considerado como um refinamento do estado abstrato do sistema. De igual modo, cada

operação concreta descreve o mesmo aspecto comportamental descrito na operação correspondente abstrata. Um tratamento mais completo relacionado aos mecanismos de prova de acordo com os dois modelos de simulação (*forwards simulation* e *backwards simulation*) é proposto por Woodcock & Davies (WOODCOCK; DAVIES, 1996).

A primeira condição estabelece que uma inicialização válida nos estados concretos também é válida no estado abstrato correspondente (pelos esquemas de inicialização abstratos e concretos, *AI* e *CI*, respectivamente).

A segunda condição estabelece que uma operação concreta *COp* pode ser aplicada em qualquer estado no qual sejam satisfeitas as pré-condições definidas pela operação equivalente abstrata *AOp*. Esta condição é chamada de teorema da aplicabilidade. Isto quer dizer que a versão concreta (mais próxima da implementação) obedece às precondições definidas pela sua operação ancestral abstrata.

A terceira condição, que representa o teorema da corretude, estabelece que uma operação concreta produza resultados consistentes com aqueles produzidos pela correspondente abstrata.

Suponha-se que dois estados concretos *CS* e *CS'* estão relacionados pela operação concreta *COp*. Suponha-se ainda que *AS*, um esquema estado abstrato correspondente ao esquema de estado concreto *CS*, obedece às precondições de *AOp*. Então, para que *COp* seja considerado como um refinamento correto de *AOp*, deve existir uma operação abstrata *AOp*, correspondente a *COp*, que transforma *AS* em *AS'* que, por sua vez, está relacionado a *CS'*.

Em síntese, uma estratégia de refinamento e prova depende do tipo de estrutura de dados utilizada em nível de projeto (*design level*), do paradigma de programação alvo e do tipo de propriedade que se pretende demonstrar. Desta forma, para cada esquema de estado e operação em nível abstrato (perspectiva do usuário) devem ser construídos os esquemas correspondentes em nível de implementação (perspectiva do programador). O mapeamento entre estas visões é definido por um esquema de relações, também chamado de relação de recuperação (*retrieve relation*). Para cada relacionamento são aplicados os teoremas da aplicabilidade e corretude. O teorema da inicialização é único para o estado global do sistema.

É importante destacar que a geração automática de código não se enquadra no escopo deste trabalho. Em PERSONÆ, o foco é a especificação de aspectos conceituais relacionados com a aplicação de reconciliação ontológica como abordagem para mediação semântica em SOA, dada a heterogeneidade semântica nas descrições de serviços. Portanto, o modelo proposto tem como propósito a estruturação de um modelo que considere explicitamente a tarefa da mediação em SOA, definindo como as técnicas de reconciliação

podem ser utilizadas nas tarefas de descoberta, seleção e estabelecimento de contratos de serviços. Considerações posteriores à aplicação destas técnicas em outras tarefas de maior nível de complexidade, como a composição de serviços, monitoramento, negociação e reconfiguração de serviços compostos, partem do pressuposto da sedimentação do modelo conceitual nas tarefas iniciais.

## 4.6 CONSIDERAÇÕES FINAIS

Neste capítulo, foi descrito o processo de formalização do modelo PERSONÆ. Inicialmente foram formalmente especificados os componentes que fazem parte do estado abstrato do modelo PERSONÆ. Em seguida, a formalização dos aspectos dinâmicos do modelo, representados pelas atividades sobre serviços, foi detalhada. Por fim, foram apresentadas as diretrizes para refinamento e prova das operações do modelo, que podem ser utilizadas para fins de implementação e verificação de propriedades, durante uma fase posterior de geração automática de código.

# CAPÍTULO 5

# CENÁRIO DE USO DO MODELO PERSONÆ

*"Não se pode provar uma definição. O que se pode fazer é mostrar que ela faz sentido."*
[Albert Einstein]

*Neste capítulo, é apresentado um cenário de uso que exemplifica a aplicação das operações definidas pelo modelo PERSONAE. O Sistema de Informações Turísticas (SEI-Tur) é utilizado para demonstração da abordagem. Esta aplicação consiste em uma solução baseada em Web services semânticos para provisão de serviços de pacotes turísticos. O objetivo é demonstrar como a atividade de mediação pode auxiliar na aproximação de clientes e provedores mais especificamente nas fases de descoberta e seleção de serviços. Uma abordagem simplificada, baseada em integração e alinhamento de ontologias é proposta para o estabelecimento de contratos de serviços. O capítulo encerra com algumas considerações sobre a implementação do modelo proposto.*

## 5.1 SISTEMAS DE INFORMAÇÃO TURÍSTICA

Nos últimos anos, a indústria do turismo tem vivenciado mudanças significativas no cenário da prestação de serviços em geral. Com o advento da Internet, a publicação de informações sobre serviços neste campo ganha novo fôlego. A ampla disseminação de portais de turismo na Web aponta para o surgimento de um mercado cada vez mais competitivo, onde a personalização na provisão de serviços pode constituir um importante diferencial na conquista por espaço.

Os Sistemas de Informação Turística ou Agências Virtuais de Turismo representam um recente tipo de aplicação para comércio eletrônico, provendo informações como serviços de passagens aéreas, hospedagem, aluguel de veículos, seguros ou opções de entretenimento (ANDRADE, 2006), (DOGAC, 2004), (MAIA, 2004), (STOLLBERG, 2004). A operacionalização deste tipo de aplicação está relacionada com a automatização de tarefas como a descoberta e seleção de serviços, no oferecimento de produtos e pacotes

turísticos. A extração deste tipo de informação serve de base para a execução de várias tarefas, incluindo a composição de pacotes turísticos, planejamento de viagens e comparação de preços, com base em requisitos definidos pelo usuário do serviço.

Neste cenário, a Arquitetura Orientada a Serviços Semânticos pode representar uma alternativa como paradigma de desenvolvimento dessas aplicações. Isto inclui não apenas a disseminação exponencial de informações e serviços, sem limitações temporais ou geográficas, mas também na recomendação de informação pertinente aos interesses dos usuários, mediante a busca, descoberta, seleção e composição de vários serviços. Prover automatização na provisão de pacotes turísticos envolve a consideração de vários aspectos, desde a captura dos requisitos do usuário até a negociação sobre o nível de qualidade oferecida pelo serviço. Portanto, o turismo eletrônico constitui um rico cenário para o desenvolvimento e aplicação das tecnologias de *Web services* semânticos.

Recentemente, algumas iniciativas têm sido identificadas quanto ao desenvolvimento de *Web services* para turismo. Sabre[17] e Datalex[18] estão entre as primeiras organizações a desenvolver *Web services* neste campo. Os *Web services* providos pela Sabre fornecem várias funcionalidades relacionadas com a venda de serviços de viagem (e.g. venda de passagens aéreas, aluguel de automóveis e reservas de hotéis). Galileo[19] também oferece um conjunto de *Web services* para execução de processos básicos como a compra, reserva e modificação de serviços turísticos (incluindo reservas de passagens aéreas, informações de vôos em tempo real e planejamento de itinerários), constituindo uma alternativa de parceria com agências de turismo virtuais, na redução de tempo, custo e manutenção de aplicações orientadas a serviços. Outra empresa de viagens beneficiada com a tecnologia de *Web services* é a Continental Airlines[20], nos Estados Unidos. Em busca de garantia de espaço no mercado, a Continental provê serviços de informação sobre vôos com base em computação pervasiva ciente de contexto. Usuários podem acompanhar informações detalhadas sobre condições de vôo e percurso em dispositivos portáveis como PDAs e celulares.

Embora a Arquitetura Orientada a Serviços constitua um modelo adequado para a construção de agências de turismo virtuais, não existe um consenso sobre a terminologia empregada na classificação de serviços nesta área. Isto torna clara a necessidade de uma

[17] Vide http://www.sabre.com
[18] Vide http://www.datalex.com
[19] Vide http://www.galileo.com
[20] Vide http://www.continental.com

abordagem orientada à mediação semântica, que permita uma aproximação de clientes e provedores, com a preservação das diferenças entre os vocabulários utilizados.

### 5.1.1 ONTOLOGIAS DE REQUISITOS FUNCIONAIS

Conforme mencionado na seção 2.4 do capítulo 2, as abordagens atuais de ontologias para descrição de serviços diferem em sua estrutura. Em OWL-S, a sub-ontologia *Service Profile* descreve o que o serviço faz, em termos de entradas, saídas, precondições e efeitos (embora os dois últimos aspectos ainda não estejam bem definidos). Em WSMO, isto é definido pela especificação de *Web Service Descriptions* (que inclui *capabilities*, para descoberta e *interface,* para composição de serviços). Entretanto, ambas apresentam uma forma genérica para descrição de requisitos funcionais. A extensão destas abordagens mediante o uso de ontologias de domínio pode proporcionar maior especificidade na busca por serviços, visto que um mesmo requisito funcional pode ter significados diferentes, de acordo com o contexto ou domínio de aplicação (e.g. uma operação de reserva de hotel pode ser diferente de uma reserva de um livro).

Existem várias propostas para a construção de ontologias para o domínio do turismo. O projeto Harmonise (FODOR; WERTHNER, 2004) permite que organizações de turismo mantenham seus formatos de dados proprietários, desde que sejam utilizados mecanismos de mapeamentos de termos. Para este propósito, o projeto utiliza a ontologia IMHO (*Interoperability Minimum Harmonization Ontology*), que provê um vocabulário genérico para turismo. Desta forma, para que uma aplicação utilize esta plataforma, é necessária a aplicação de técnicas de mediação semântica, para a garantia de interoperabilidade com outras aplicações. O projeto Harmonise inclui parcerias com várias organizações internacionais de padronização no ramo do turismo, como WTO[21] (*World Tourism Organization*), TTI[22] (*Travel Technology Initiative*), IFITT[23] (*International Federation for IT and Travel & Tourism*) e OTA[24] (*Open Travel Alliance*).

O padrão OTA (*Open Travel Alliance*) define um amplo conjunto de especificações em XML Schema para a descrição funcional de serviços em nível de troca de mensagens. Estas especificações incluem termos utilizados em vários segmentos do turismo, como reserva e compra de passagens aéreas, hotelaria, aluguel de veículos, linhas terrestres e

[21] Vide www.world-tourism.org
[22] Vide www.tti.org
[23] Vide www.ifitt.org
[24] Vide www.opentravel.org

seguros. As especificações OTA incluem a descrição de parâmetros de operações de reserva, aluguel, compra, venda, cancelamento, modificação e consulta de informações de serviços turísticos (incluindo requisitos de QoS e dados estatísticos sobre os serviços). Desta forma, o padrão OTA constitui uma alternativa para enriquecimento das descrições de serviços em nível de troca de mensagens, para o domínio de turismo (vide Figura 5.1).

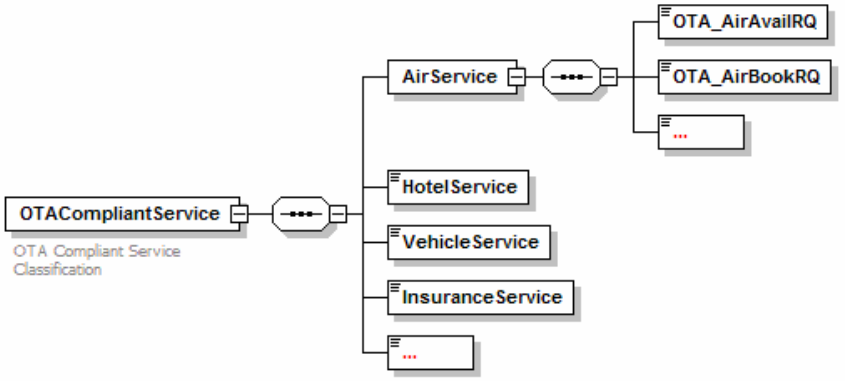



**Figura 5.1 – Versão parcial da ontologia OTA para descrição de requisitos funcionais de serviços**

A ontologia IMHO, do projeto Harmonise, tem sido referenciada em estudos de caso de OWL-S (DOGAC, 2004), (FODOR; WERTHNER, 2004), e WSMO (STOLLBERG, 2004). Ou seja, esta ontologia representa um ponto de interoperabilidade em potencial, entre outros padrões, no que concerne à especificação de termos pertinentes ao domínio do turismo. Neste caso, a aplicação de técnicas de reconciliação de ontologias pode permitir a comunicação de organizações neste domínio, em específico.

### 5.1.2 Ontologias de Requisitos Não-Funcionais

Existem várias abordagens para taxonomias de requisitos não-funcionais, existindo certo consenso sobre a classificação destes requisitos como atributos de qualidade de serviço. Isto significa que requisitos de QoS são considerados requisitos não-funcionais. A recíproca não é necessariamente verdadeira. Uma ontologia de QoS define os relacionamentos entre requisitos não-funcionais (e.g. convergência ou conflitos) e os intervalos de valores relacionados com os níveis de qualidade oferecida, os quais podem ser especificados como propriedades de tipos de dados (*datatype properties*).

Ontologias de QoS também podem ser utilizadas como extensão de ontologias de descrição de serviços. A abordagem OWL-S não considera este aspecto explicitamente, embora seja possível que uma ontologia de QoS possa estender ontologias de domínio na descrição de serviços (DOGAK, 2004). Em relação à abordagem WSMO, requisitos não-

funcionais são utilizados na descrição de seus componentes arquiteturais. Contudo, os requisitos utilizados correspondem às definições do padrão *Dublin Core* (WEIBEL, 1998), que não inclui informações relacionadas às condições de custo, tempo ou espaço, por exemplo. Uma abordagem sistematizada para descrição formal de requisitos não-funcionais é proposta por O'Sullivan et al. (O'SULLIVAN; EDMOND; HOFSTEDE, 2005), fornecendo subsídios para a definição de ontologias de QoS específicas para *Web services* (vide Figura 5.2)

Enquanto as ontologias de domínio são utilizadas para conferir maior especificidade na fase de descoberta de serviços, as ontologias de QoS conferem um maior nível de refinamento no processo de seleção. Entretanto, não existe consenso sobre a utilização de uma única ontologia de QoS, visto que as métricas dos requisitos não-funcionais variam de acordo com o domínio de aplicação. Em outras palavras, a instanciação de determinados tipos de serviços define qual subconjunto de requisitos não-funcionais será considerado. Ainda que um mesmo conjunto de dados seja considerado como base (e.g. padrões de tabelas de preço em serviços de hotelaria), o relacionamento desses tipos de dados com as classes de requisitos de QoS pode variar de uma ontologia para outra. Isto deixa claro que, para uma seleção mais refinada, é necessário um alinhamento de parâmetros de qualidade de serviço, do nível de classe até o nível de instâncias de conceitos.

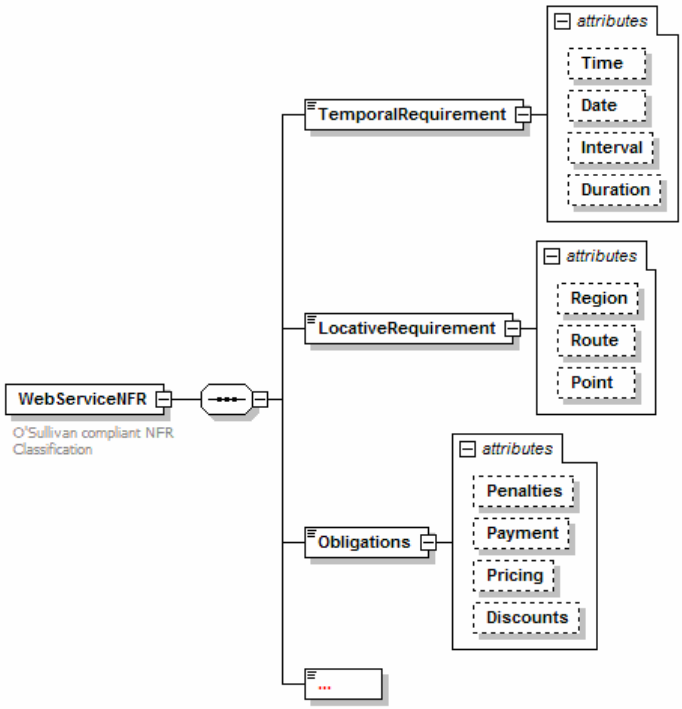



**Figura 5.2 – Visão parcial de uma ontologia de QoS (O'SULLIVAN, 2005)**

## 5.2 SEI-TUR – SERVIÇO DE INFORMAÇÕES TURÍSTICAS

O SEI-Tur – Serviço de Informações Turísticas (MAIA, 2004) – é um sistema de recomendação de construção de pacotes turísticos baseado na tecnologia de *Web services* semânticos. O sistema utiliza uma interface para captura de requisitos do usuário e realiza a descoberta, seleção e composição dos serviços, com base na ferramenta WebSComposer (ANDRADE, 2006). A montagem do roteiro turístico (que inclui serviços de transporte, hospedagem, alimentação e entretenimento) é realizada com base na execução de um algoritmo de escalonamento, considerando a compatibilidade de entradas e saídas dos serviços e condições espaço-temporais. Para cada descrição de serviço existe uma ontologia de domínio, que permite uma descoberta mais precisa. Os requisitos de qualidade de serviço são definidos nas ontologias de domínio, ou seja, não existe uma ontologia específica para requisitos não-funcionais. Além disso, o sistema fornece um estudo de caso suficientemente rico para a aplicação de mecanismos típicos de uma Arquitetura Orientada a Serviços Semânticos.

Atualmente o sistema é restrito a informações do turismo local, no estado da Paraíba – Brasil. Entretanto, futuras expansões do cadastro do sistema incluem o cadastro de empresas de outras regiões do país, ampliando as possibilidades oferecidas pelo serviço. A possibilidade do aumento do número de serviços cadastrados aponta para a inclusão de bases heterogêneas de dados, desde o nível estrutural (diferentes ontologias de domínio) até o nível de instâncias dos conceitos. Por exemplo, as empresas aéreas utilizam diferentes parâmetros de "classe" nos compartimentos das aeronaves, oferecem serviços diferentes para entretenimento e diferentes tipos de bonificação de clientes em processo de fidelização. O mesmo ocorre para hotéis, que utilizam diferentes padrões de classificação de quartos e de perfis de clientes.

Desta forma, pode-se prever que o aumento da oferta de novos serviços cadastrados no sistema indica o surgimento de terminologias diferentes para um mesmo domínio de serviço. As diferenças podem ocorrer em nível de descrição funcional ou de parâmetros de qualidade de serviço. Isto torna clara a necessidade de uma abordagem centrada na mediação, para resolução de possíveis conflitos.

Nas seções a seguir, será discutida a aplicabilidade do modelo PERSONÆ no cenário da provisão de serviços turísticos. Visto que o modelo não adentra em aspectos de níveis de composição de serviços, somente serão considerados na fase de seleção os serviços que atendem totalmente à funcionalidade desejada. Portanto, o objetivo é mostrar

como a mediação pode ser aplicada na descoberta e na seleção de serviços simples. O tratamento de aspectos relacionados com mediação e requisitos de QoS em nível de composição de serviços são discutidos na seção de trabalhos futuros, no capítulo 6. Para efeito de simplicidade na apresentação das propriedades do modelo, será considerado um cenário simples de serviços reserva de passagens aéreas.

### 5.2.1 DESCOBERTA DE SERVIÇOS E FUSÃO DE ONTOLOGIAS

Para a delimitação do cenário de uso, algumas premissas devem ser estabelecidas. Neste caso, os serviços disponíveis possuem descrições baseadas em duas representações diferentes (WSMO e OWL-S). Conforme dito anteriormente, a funcionalidade de um serviço específico pode ser definida mediante a utilização de ontologias de domínio. A definição dos requisitos não-funcionais de um serviço pode ser incluída na ontologia de domínio, fornecendo uma descrição mais específica de parâmetros de qualidade do domínio considerado. Os parâmetros de entradas, saídas, precondições e efeitos são de interesse da aplicação, para fins de execução e composição de serviços. As informações contidas na ontologia de domínio especificam conceitos mais próximos dos termos utilizados pelos usuários.

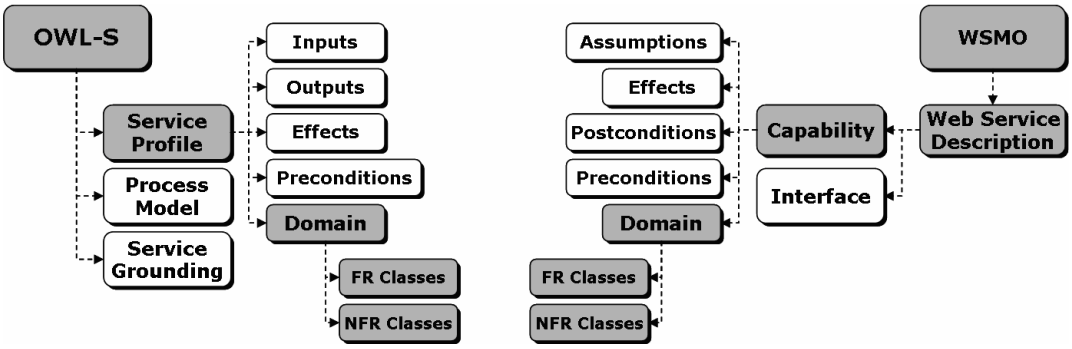



**Figura 5.3 – Parâmetros utilizados na descoberta de serviços (Perspectiva do usuário)**

Também foram utilizadas as especificações OTA para a definição dos conceitos da ontologia de domínio na modelagem do cenário. Entretanto, inúmeros detalhes destas especificações foram suprimidos para fins de simplificação na demonstração das propriedades do modelo PERSONÆ. De acordo com a pesquisa realizada neste trabalho, as especificações OTA representam um dos principais esforços no estabelecimento de termos gerais utilizados na indústria do turismo. Estas especificações representam, em potencial, um ponto em comum

na definição de ontologias de domínio para o turismo. Este fator comum permite a aplicação da técnica de fusão de conceitos na fase de descoberta de serviços.

Considere o exemplo de um usuário à procura por serviços de passagens aéreas, de acordo com os requisitos mostrados na Tabela 5.1. A primeira coluna corresponde aos requisitos funcionais desejados. Estes requisitos serão utilizados na descoberta de serviços. A segunda coluna corresponde aos requisitos não-funcionais, que serão utilizados no processo de seleção, que será detalhado adiante.

**Tabela 5.1 – Requisitos do usuário para requisição de serviços de reserva de passagens aéreas**

| REQUISITOS FUNCIONAIS | REQUISITOS NÃO-FUNCIONAIS |
|---|---|
| SERVIÇO DE RESERVA DE PASSAGENS *ONLINE*<br>ORIGEM: FLORIANÓPOLIS<br>DESTINO: SÃO PAULO<br>HORÁRIO/DATA DE IDA:<br>17/05/2007 – 12:00H | PREFERÊNCIA PELAS COMPANHIAS:<br>TAM E AIR FRANCE<br>CLASSE EXECUTIVA<br>PRIORIDADE:<br>MENOR CUSTO |

Considere ainda que os serviços de passagens aéreas disponíveis sejam classificados segundo a Tabela 5.2.

**Tabela 5.2 – Classificação de serviços de passagens aéreas disponíveis**

| WEB SERVICE | REQUISITOS FUNCIONAIS | REQUISITOS NÃO-FUNCIONAIS |
|---|---|---|
| *OTA_AvailRQ* | Fornece informações sobre a disponibilidade de vôos, considerando um par de cidades origem/destino, dado um número de passageiros. | Companhia aérea<br>Classe de acomodação<br>Conexão de cidades<br>Refeições<br>Tempo de vôo<br>Hora de partida/chegada |
| *OTA_AirBookRQ* | Fornece informações sobre itinerários para um ou mais passageiros | Preço<br>Classe de acomodação<br>Refeições<br>Serviços especiais<br>Serviços de Informação<br>Condições de pagamento<br>Tipo de ticket |
| *OTA_AirFareDisplayRQ* | Fornece informações sobre taxas adicionais no translado entre um par de cidades origem/destino | Taxas adicionais |
| *OTA_AirLowFareSearchRQ* | Fornece informações sobre itinerários entre vários pares de cidade origem/destino intermediários. | Pontos de conexão<br>Tipo do vôo (com paradas ou direto)<br>Preços e Taxas adicionais |
| *OTA_AirSchedulesRQ* | Fornece uma programação de vôos mediante sincronização de informações relacionadas com pares de cidades origem/destino, data de partida/chegada e hora de partida/chegada. | Refeições<br>Tempo de duração de vôo<br>Estatísticas de vôo<br>Classe de acomodação |

Na fase de descoberta, parte-se do pressuposto de que os requisitos funcionais do usuário estão definidos em uma ontologia que representa o padrão utilizado pela aplicação. Sendo assim, a fusão de conceitos nesta fase está baseada na propagação de similaridades entre os conceitos definidos nesta ontologia e os conceitos definidos em outras ontologias de domínio mais específicas (estas últimas importam a ontologia padrão definida pela aplicação). Conforme visto no capítulo 3, a fusão dos conceitos de ontologias diferentes depende da propagação dos conceitos da ontologia importada nestas ontologias. Quanto mais fortes forem os axiomas de ligação, maior será a fusão de classes e propriedades. De acordo com o exemplo considerado anteriormente, os serviços de reserva de vôos correspondem à classe definida por *OTA_AirAvailRQ*. Na Figura 5.4, ilustram-se os conceitos definidos por esta classe de serviço.

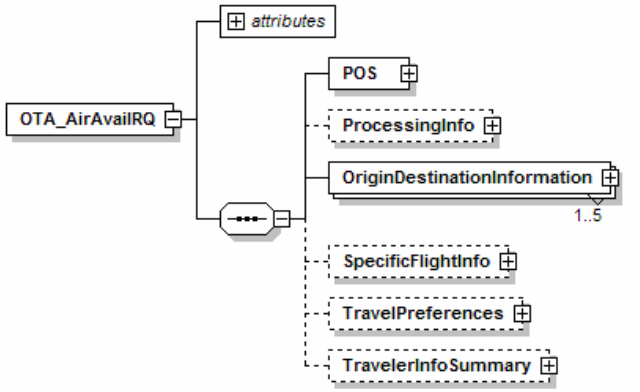



**Figura 5.4 – Classificação de parâmetros de serviços de reserva de passagem aérea (OTA, 2005)**

Além do tipo de serviço desejado, os outros requisitos funcionais mencionados correspondem às cidades de origem e destino e aos horários de partida e chegada. Estes requisitos são definidos pela classe *OriginDestinationInformation*, como é mostrado na Figura 5.5.

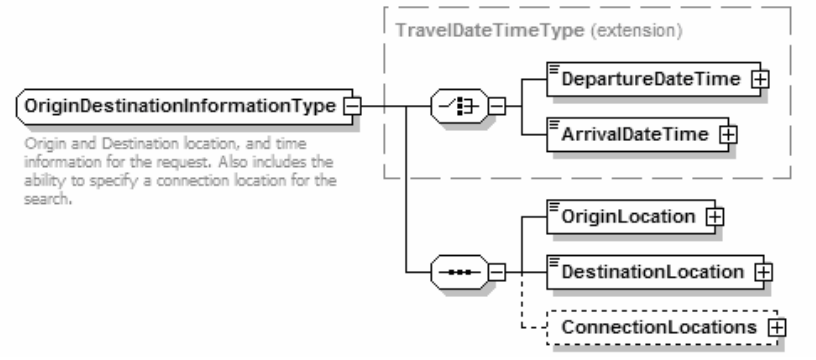



**Figura 5.5 – Detalhamento de requisitos funcionais para serviços de reserva de passagens aéreas (OTA, 2005)**

Considerem-se duas ontologias de domínio diferentes, $O_1$ e $O_2$, as quais importam os conceitos ilustrados na Figura 5.5. Embora um cenário realístico possa incluir um número bem maior de ontologias diferentes, a análise aqui proposta inclui mapeamentos duas a duas, para fins de simplicidade. Na ontologia $O_1$ a classe *DepartureArriveInformation* estende o conceito original da ontologia importada $O_3$ mediante uma relação de equivalência (*equivalentFrom*). Na ontologia $O_2$, a classe *Point2PointInformation* estende o mesmo conceito da ontologia $O_3$, mediante uma relação de generalização (*isSuperClassOf*). Segundo é mostrado na Figura 5.6, estas relações se propagam das classes originais importadas de $O_3$ para as ontologias de destino. A ontologia-núcleo gerada (*Core Ontology*) contém não somente os conceitos da ontologia $O_3$, fator comum entre as ontologias consideradas, mas também a intersecção das propagações destes conceitos. Desta forma, a ontologia gerada no processo de fusão de classes representa um esquema global que agrega conceitos comuns às duas ontologias consideradas.

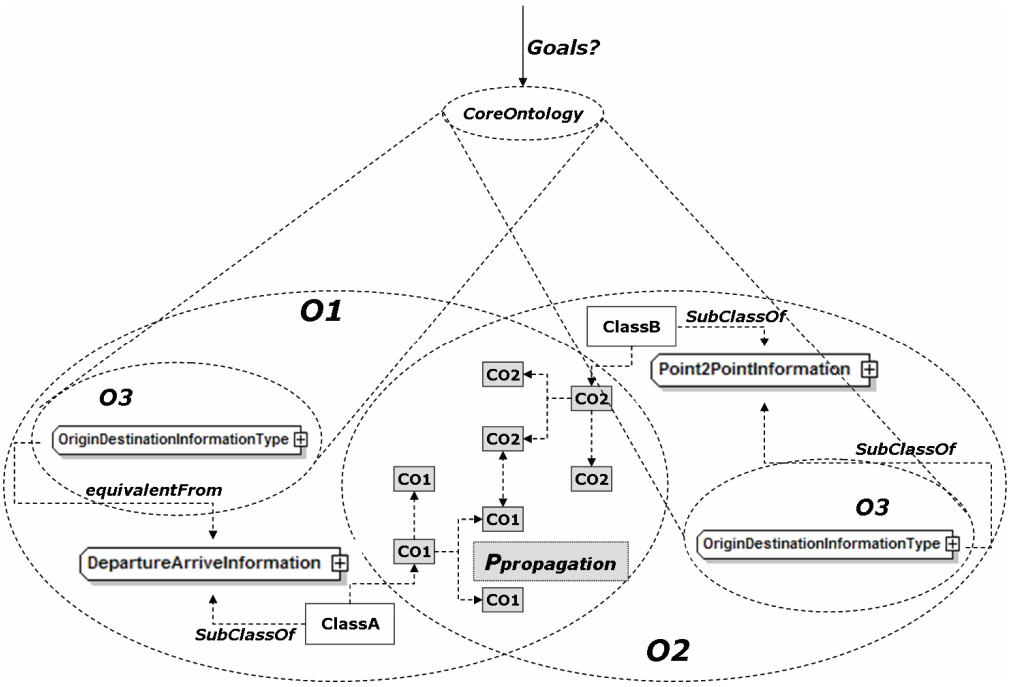



**Figura 5.6 – Propagação de conceitos na descoberta de serviços**

Visto que o propósito na descoberta de serviços é a recuperação do maior número possível de serviços que atendam à funcionalidade desejada, é suficiente a fusão em nível de classes. Uma fusão em nível de propriedades (ou mesmo de indivíduos) pode gerar uma ontologia-núcleo muito específica, estreitando as possibilidades na descoberta de serviços. Este nível de refinamento pode ocorrer em uma fase posterior de seleção. A fusão dos

conceitos permite que a descoberta possa ser realizada posteriormente, com base na recuperação das instâncias específicas (indivíduos).

*ClassMergingInIntersectionImport*

$\Delta Ontologies$
$\Xi ClassAxioms$
$O1?, O2?, CoreOntology!: Ontology$

---

$\exists ClassAxiomSet: ClassAxioms; ImportedOntology: Ontology$
$\mid ClassAxiomSet \in CoreOntology! . OntologicalClassAxioms$
$\wedge O1? \in OntologySet \wedge O2? \in OntologySet$
$\wedge ImportedOntology \in OntologySet$
$\wedge CoreOntology! \notin OntologySet$
$\wedge ImportedOntology . Classes \subseteq O1? . Classes$
$\wedge ImportedOntology . Classes \subseteq O2? . Classes$
$\bullet ClassAxiomSet . isSubClassOf$
$= \{ c1, c2, c3: CLASS$
$\mid c1 \in ImportedOntology . Classes$
$\wedge c2 \in O1? . Classes$
$\wedge c3 \in O2? . Classes$
$\wedge$ **(c1 EquivalentClass c2 $\wedge$ c1 isSubClassOf c3)** $\bullet (c2, c3) \}$
$\wedge CoreOntology! . OntologicalClassAxioms$
$= CoreOntology! . OntologicalClassAxioms \cup \{ClassAxiomSet\}$
$\wedge OntologySet' = OntologySet \cup \{CoreOntology!\}$

A intersecção entre as propagações dos conceitos da ontologia importada nas ontologias de domínio depende dos axiomas de ligação. Por exemplo, o axioma *equivalentFrom* representa um nível de similaridade maior que o axioma *isSubClassOf*. O esquema *ClassMergingInIntersectionImport* relaciona as pré-condições a serem consideradas no processo de inferência de novos relacionamentos entre as ontologias de origem. Esta operação se refere ao caso em que ambas as ontologias estendem a ontologia importada. O esquema foi reduzido para fins de demonstração das propriedades relacionadas com o exemplo aqui discutido. O Apêndice traz uma versão mais completa deste esquema.

As variáveis de entrada e saída do processo (*O1? O2?* e *CoreOntology!*) são fornecidas na parte declarativa do esquema. Na parte predicativa, as pré-condições para o processo de integração são definidas. A variável *ClassAxiomSet* representa os axiomas de classe que serão incorporados na ontologia-núcleo gerada. A classe *c1* pertence à ontologia *O3* (instanciada pela classe *OriginDestinationInformationType*, no exemplo); a classe *c2* pertence à ontologia *O2* (instanciada pela classe *DepartureArriveInformation*); a classe *c3* pertence à ontologia *O3* (instanciada pela classe *Point2PointInformation*). A condição

seguinte verifica a existência de relações de equivalência e generalização, conforme ilustrado na Figura 5.6. A compreensão de conjuntos é encerrada com a atribuição de uma nova relação de especificação (pelo conjunto *ClassAxiomSet.isSubClassOf*), entre as classes *c2* e *c3*. A Tabela 5.3 sumariza algumas das principais implicações no processo de fusão de ontologias, conforme os axiomas de ligação considerados.

**Tabela 5.3 – Implicações no processo de propagação de propriedades na integração de ontologias**

| PROPAGAÇÃO DE PROPRIEDADES | IMPLICAÇÕES |
|---|---|
| **$O_1 \leftrightarrow O_3$** *OriginDestinationInformationType* **equivalentFrom** *DepartureArriveInformation* | A classe *DepartureArriveInformation* contém todas as subclasses de *OriginDestinationInformationType*. Novas relações entre os conceitos da primeira podem torná-la mais específica do que a segunda. |
| **$O_2 \leftrightarrow O_3$** *OriginDestinationInformationType* **isSubClassOf** *Point2PointInformation* | As subclasses da classe *OriginDestinationInformationType* tornam-se subclasses de *Point2PointInformation*. A segunda classe torna-se mais genérica que a primeira. Isto é possível pelo fato de que, na construção de ontologias, generalização ou especialização dependem do uso que se faz das extensões realizadas com base nos conceitos importados. |
| **$O_1 \leftrightarrow O_2$** *DepartureArriveInformation* **isSubClassOf** *Point2PointInformation* | As classes de *DepartureArriveInformation* tornam-se subclasses de *Point2PointInformation*. A ontologia gerada a partir da integração contém a ontologia importada com acréscimo desta relação. |

Após o processo de integração, a descoberta pode ser realizada com base na recuperação das instâncias específicas (indivíduos), pelo mapeamento produzido pela ontologia-núcleo gerada. Os resultados da busca são mostrados na Tabela 5.4, de acordo com o exemplo mencionado.

**Tabela 5.4 – Resultado da descoberta de serviços com base em requisitos funcionais**

| WEB SERVICE | DEPARTURE DATETIME | ARRIVAL DATETIME | ORIGIN LOCATION | DESTINATION LOCATION |
|---|---|---|---|---|
| TAM OTA_AirAvailRQ | 17/05/2007 12:00h | 17/05/2007 13:00h | Florianópolis | São Paulo |
| TAM OTA_AirAvailRQ | 17/05/2007 11:39h | 17/05/2007 12:40h | Florianópolis | São Paulo |
| GOL OTA_AirAvailRQ | 17/05/2007 11:51h | 17/05/2007 12:55h | Florianópolis | São Paulo |
| AirFrance OTA_AirAvailRQ | 17/05/2007 12:00h | 17/05/2007 13:00h | Florianópolis | São Paulo |
| AirFrance OTA_AirAvailRQ | 17/05/2007 10:53h | 17/05/2007 11:55h | Florianópolis | São Paulo |

### 5.2.3 SELEÇÃO DE SERVIÇOS E ALINHAMENTO DE ONTOLOGIAS

Conforme mencionado anteriormente, no caso da multiplicidade da oferta, a seleção se respalda na especificidade da procura. Este processo é realizado com base na consideração de requisitos de qualidade que diferenciam serviços similares. Estes requisitos podem ser definidos segundo uma ontologia específica de requisitos de QoS ou podem ser estruturados em ontologias de domínio, fornecendo uma caracterização mais específica dos parâmetros de qualidade de um determinado tipo de serviço. A especificação *OTA_AirAvailRQ* também inclui estes requisitos, definidos pelas classes *SpecificFlightInfo*, *TravelPreference*s e *TravelerInfoSummary*. Estas preferências podem ser de uso mais geral (e.g. *PriceRequestInformation*) ou mais específico (e.g. *Airline*, no caso da existência de algum tipo de relação de fidelização entre cliente e provedor do serviço), conforme é ilustrado na Figura 5.7.

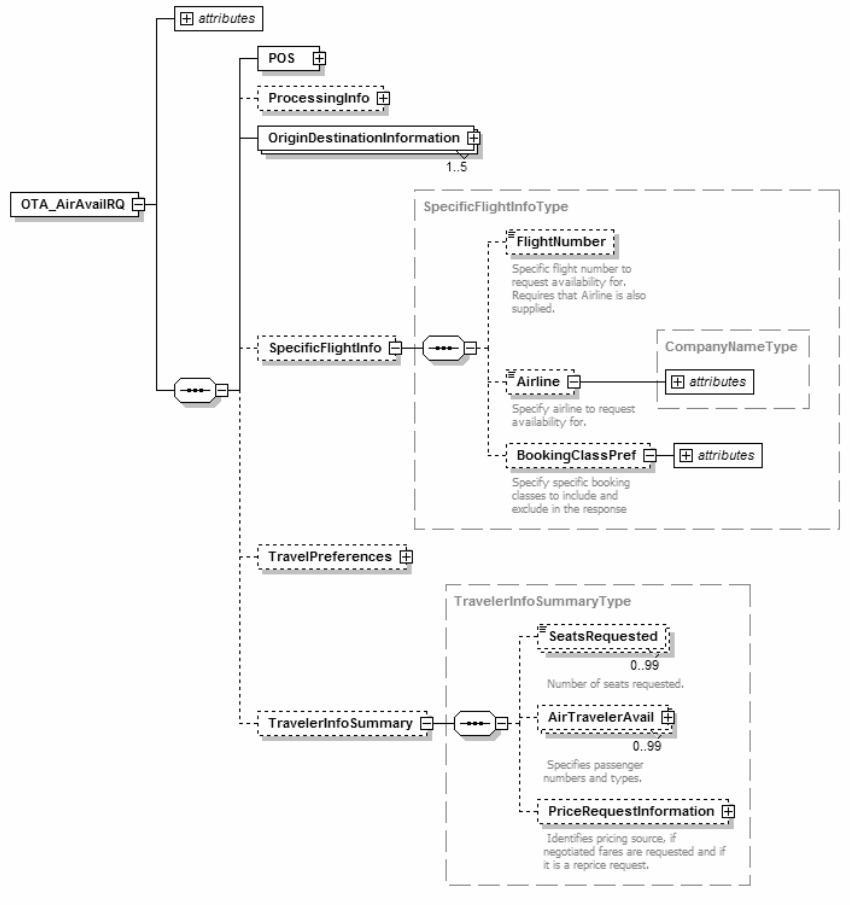



**Figura 5.7 – Detalhamento de requisitos não-funcionais para serviços de reserva de passagens aéreas (OTA, 2005)**

Considerando ainda o exemplo discutido neste capítulo, é possível verificar que, neste caso, o usuário não estabelece restrições (*QoSContraints*) em relação ao nível de qualidade desejada. Portanto, somente serão consideradas as suas preferências (*QoSPreferences*) e prioridades (*QoSPriorities*), conforme é mostrado na Tabela 5.1. Considerando ainda que a ontologia de domínio referenciada como padrão já inclui definições de parâmetros de QoS, além do fato de que na fase anterior de descoberta foi realizado um alinhamento de classes, é suficiente um alinhamento de propriedades de tipos de dados neste exemplo. Caso fossem utilizadas diferentes ontologias para requisitos de QoS, seria necessário um alinhamento prévio de classes. O alinhamento de classes neste contexto também permitiria a verificação de propriedades de correlação entre os requisitos não-funcionais (e.g. convergência ou conflitos). Este procedimento pode ser utilizado na fase de negociação de serviços, antes do estabelecimento de um contrato sobre nível de qualidade acordada (*Service Level Agreement*).

Conforme é especificado em PERSONÆ, os requisitos não-funcionais definem o nível de qualidade desejada (*QoSConstraints*, *QoSPreferences* e *QoSPriorities*, para o usuário) e o nível de qualidade provida (*QoSProvided*, segundo a especificação do serviço). Estas relações foram formalizadas da seguinte forma:

*NonFunctionalRequirement*

$Id: ID$
$Class: CLASS$
$Domain: Ontology$

---

$Class \in Domain\ .\ Classes$

$LEVEL: \mathbb{P}\ DATATYPE$

$QoSProvided: NonFunctionalRequirement \rightarrow\!\!\!\!\!\!\rightarrow LEVEL$
$QoSPreferences: NonFunctionalRequirement \rightarrow\!\!\!\!\!\!\rightarrow LEVEL$
$QoSPriorities: NonFunctionalRequirement \times NonFunctionalRequirement$
$\rightarrow\!\!\!\!\!\!\rightarrow NonFunctionalRequirement$

Neste caso, o domínio das relações *QoSProvided* e *QoSPreferences* é do tipo *NonFunctionalRequirement* (definidos por classes em uma ontologia) e a imagem é do tipo *LEVEL* (definido por um *power set* '$\mathbb{P}$' de *DATATYPE*). Isto justifica a utilização do alinhamento de propriedades de tipos de dados (*DataTypeProperties*) no processo de seleção de serviços. Os intervalos de tipos de dados específicos variam de acordo com a

implementação. Portanto, o conjunto definido por *LEVEL* abstrai todas as possibilidades de tipos de dados, na modelagem de propriedades e atributos.

Na Figura 5.8, são mostrados alguns dos requisitos não-funcionais considerados no exemplo. Considerando que o conjunto de requisitos não-funcionais são os mesmos (de acordo com a ontologia importada), com pequenas variações de nomenclatura, foi estabelecida uma relação de equivalência entre os requisitos *PriceRequestInformationType* e *Pricing*. Visto que o alinhamento produz um conjunto de mapeamentos individuais entre os conceitos, é possível fornecer ao usuário apenas as relações pertinentes às suas preferências. De acordo com o exemplo anterior, estas propriedades são representadas por *PricingSource/Source* e *CabinType/CabinClass*.

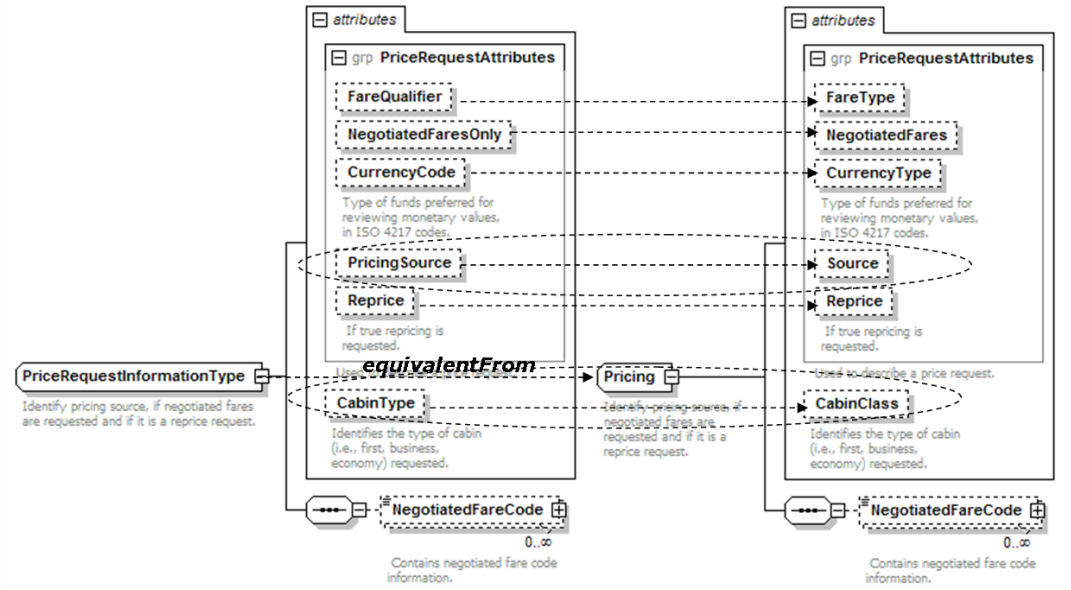



**Figura 5.8 – Alinhamento de *DataTypeProperties* na seleção de parâmetros de QoS**

O esquema *DataTypePropertyAlignment* relaciona as pré-condições a serem consideradas no processo de inferência de novos relacionamentos entre as ontologias de origem. O esquema foi reduzido para fins de demonstração das propriedades relacionadas com o exemplo aqui discutido. O Apêndice traz uma versão mais completa deste esquema. As variáveis de entrada e saída do processo (*O1? O2?* e *MapSet!*) são fornecidas na parte declarativa do esquema. Na parte predicativa, as pré-condições para o processo de integração são definidas. A variável *MapSet* é definida pelo esquema de tipo *MapSet*, que contém um conjunto de relações geradas entre classes, propriedades e indivíduos de duas ontologias diferentes. A propriedade *dtp1* pertence à ontologia *O3*, a propriedade *dtp2*

pertence à ontologia *O2* e a propriedade *dtp3* pertence à ontologia *O3*. A condição seguinte verifica a existência de relações de equivalência, conforme ilustrado na Figura 5.8. A compreensão de conjuntos é encerrada com a atribuição de um novo axioma de ligação entre as ontologias (pelo conjunto *MapSet!.DataTypePropertyMappings*), entre as propriedades *dtp2* e *dtp3*.

*DataTypePropertyAlignment*

$\Xi$ *Ontologies*
$\Xi$ *DataTypePropertyAxioms*
*O1?, O2?: Ontology*
*MapSet!: MappingSet*
*Report!: REPORT*

---

*MapSet! . DataTypePropertyMappings*
= { *dtp1, dtp2, dtp3: DataTypeProperty;*
*EquivalentDataTypeProperty: DataTypeProperty* $\leftrightarrow$ *DataTypeProperty*
| *O1?* $\in$ *OntologySet*
$\wedge$ *O2?* $\in$ *OntologySet*
$\wedge$ *dtp1* $\in$ dom *EquivalentDataTypePropertys*
$\wedge$ *dtp2* $\in$ ran *EquivalentDataTypeProperty*
$\wedge$ *dtp3* $\in$ ran *EquivalentDataTypeProperty*
$\wedge$ dom *EquivalentDataTypeProperty* $\subseteq$ *O1? . DataTypeProperties*
$\wedge$ ran *EquivalentDataTypeProperty* $\subseteq$ *O2? . DataTypeProperties*
$\wedge$ **(*dtp2*, *dtp3*) $\in$ *EquivalentDataTypeProperty*** $\bullet$ (*dtp1*, *dtp3*) }
*MapSet! . ClassMappings* = {} $\Rightarrow$ *Report! = Impasse*
*MapSet! . ClassMappings* $\neq$ {} $\Rightarrow$ *Report! = Match*

---

De acordo com o exemplo, as preferências do usuário incluem as companhias TAM e Air France e disponibilidade de vaga em classe executiva. No caso de similaridades nestas preferências, o critério de diferenciação é a prioridade do menor custo (vide Tabela 5.5).

**Tabela 5.5 - Resultado da seleção de serviços com base em requisitos não-funcionais**

| WEB SERVICE | EXECUTIVE CLASS | PRICING |
|---|---|---|
| TAM OTA_AirAvailRQ | 17/05/2007 12:00h | R$ 310,00 |
| TAM OTA_AirAvailRQ | 17/05/2007 11:39h | R$ 310,00 |
| AirFrance OTA_AirAvailRQ | 17/05/2007 12:00h | R$ 245,00 |
| AirFrance OTA_AirAvailRQ | 17/05/2007 10:53h | R$ 215,00 |

### 5.2.3 ESTABELECIMENTO DE CONTRATOS E INTEGRAÇÃO DE ONTOLOGIAS

Um contrato de serviço registra termos e condições relacionados com o nível de qualidade de serviço determinado por acordo entre cliente e provedor. Em PERSONÆ, a especificação de contrato é definida pelo esquema *ServiceLevelAgreement*. Este esquema contém os tipos *ServiceRequesterSection* e *ServiceProviderSection*, que representam os perfis do cliente e do provedor, respectivamente. O tipo *SLAOntology* especifica uma ontologia integrada, que reúne os perfis mencionados e os requisitos de QoS. O nível de qualidade acordada (*QoSAgreed*) depende de uma negociação entre os requisitos prioritários do usuário (*QoSPriorities*) e o nível de qualidade oferecida pelo provedor (*QoSOffered*). Conforme mencionado no capítulo 4, no contexto deste trabalho as correlações entre requisitos de QoS não são tratadas detalhadamente. Neste caso, é proposta uma abordagem simplificada para integração de perfis e alinhamento de requisitos de QoS, no estabelecimento do contrato de serviço.

*ServiceLevelAgreement*

$ServiceRequesterSection, ServiceProviderSection : Ontology$
$SLAOntology : \mathbb{P}\ Ontology$
$QoSAgreed : NonFunctionalRequirement \pfun LEVEL$

---

$\forall nfr : NonFunctionalRequirement \mid nfr \in \mathrm{dom}\ QoSAgreed$
$\bullet\ nfr\ .\ Class \in ServiceRequesterSection\ .\ Classes$
$\land\ nfr\ .\ Class \in ServiceProviderSection\ .\ Classes$
$ServiceRequesterSection \in SLAOntology$
$ServiceProviderSection \in SLAOntology$

Considerando que não existe consenso sobre padrões de representação de contratos de serviços (os quais variam de acordo com o domínio do serviço), a integração de ontologias pode ser utilizada para agregação dos perfis de clientes e provedores. No modelo ESCHER (RIBEIRO, 2004), por exemplo, é proposta uma estrutura de propriedades mais concretas, relacionadas ao serviço (*ServiceQoSspec*), e que também caracterizam o provedor. Esta especificação é definida em um tipo (*ServiceQoSspecType*) composto por cinco elementos, que são utilizados após o processo de negociação (vide Figura 5.9). Quanto ao perfil de usuários, o padrão OTA também disponibiliza uma especificação de parâmetros para este propósito (vide Figura 5.10).

O primeiro elemento refere-se ao nível de QoS acordada (*QoSagreed*), sendo definido por um conjunto de pares que relacionam parâmetros de QoS (*QoSparam*)

negociados e intervalos de valores (*ValueRange*). O segundo elemento refere-se à qualidade efetivamente fornecida (*QoSprovided*) pelo provedor, definido por um conjunto de estrutura similar à qualidade acordada, ou seja, pares de parâmetros de QoS (*QoSparam*) e intervalo de valores (*ValueRange*). O terceiro elemento está relacionado à qualidade medida (*QoSmeasured*). O quarto elemento caracteriza a necessidade de reter informações sobre os recursos alocados (*VirtualResourceAlloc*) para a prestação do serviço O quinto e último elemento representa a situação (*status*) do contrato. Valores possivelmente associados ao *status* são: em negociação (*Negotiating*), ativo (*Active*) e violado (*Violated*).

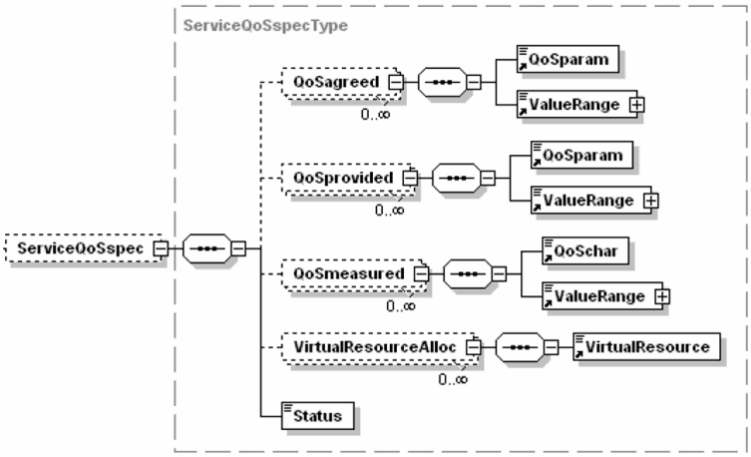



**Figura 5.9 – Componente de serviço no Contrato de QoS (RIBEIRO, 2004)**

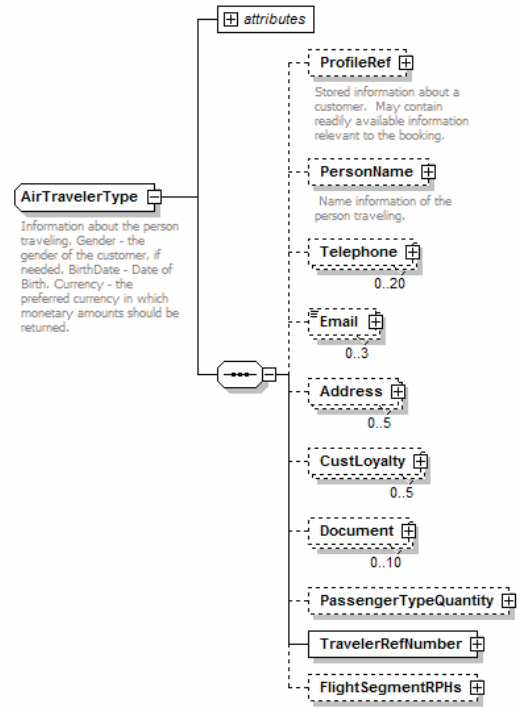



**Figura 5.10 – Componente de perfil de usuário de serviços de companhias aéreas (OTA, 2005)**

Desta forma, as duas representações mencionadas podem ser integradas na formação de uma ontologia-núcleo para representação de contrato. O esquema *ClassIntegration* especifica esta operação, considerando as variáveis *O1?* e *O2?* Como entradas do processo. A saída é definida pela variável *CoreOntology!*, que representa a ontologia integrada. Neste processo, ocorre apenas uma união generalizada de classes, permitindo o reuso das definições. Conforme é detalhado no capítulo 4, este processo representa uma das etapas para o estabelecimento do contrato. Para o estabelecimento do nível de QoS acordado (*QoSAgreed*) alinhamentos de propriedades de tipos de dados (*DataTypeProperties*) e de indivíduos (*individuals*), podem ser aplicados, conforme detalhado na seção 5.3.

*ClassIntegration*

$\Delta Ontologies$
$O1?, O2?, CoreOntology!: Ontology$

---

$O1? \in OntologySet$
$O2? \in OntologySet$
$CoreOntology! \notin OntologySet$
$CoreOntology!.Classes = O1?.Classes \cup O2?.Classes$
$OntologySet' = OntologySet \cup \{CoreOntology!\}$

## 5.3 ORIENTAÇÕES PARA IMPLEMENTAÇÃO

O modelo PERSONÆ consiste basicamente em uma abordagem de mediação semântica para Arquiteturas Orientadas a Serviços. A mediação tratada neste trabalho refere-se a um mecanismo que visa à aproximação de clientes e provedores em um cenário de provisão de serviços personalizados. Esta necessidade de aproximação refere-se ao fato de que nem sempre provedores e clientes compartilham uma mesma visão de um serviço. Estas visões podem diferir em termos de interesses envolvidos (e.g. a busca por vantagens na relação custo/benefício) ou em termos de terminologias utilizadas. PERSONÆ representa uma abordagem centrada no segundo aspecto. Em relação à operacionalização desta abordagem, esta seção visa somente indicar "o que" é necessário implementar, sem aprofundar "como" se dará esta implementação.

Conforme mencionado anteriormente, a principal diferença entre o modelo SOA tradicional e o modelo orientado a serviços semânticos consiste na ampliação das possibilidades de descoberta de serviços que melhor atendam às necessidades dos usuários.

O enriquecimento das descrições destes serviços é o principal fator envolvido na melhoria da recuperação da informação.

Junto com os benefícios da automatização de processos, surgem problemas envolvidos com a multiplicidade de padrões e formas de descrição de serviços. Isto dificulta a aproximação entre clientes e provedores envolvidos, tornando clara a necessidade de uma abordagem orientada à mediação, de forma a garantir comunicação entre as entidades envolvidas, com preservação das diferenças de vocabulários.

Conforme mencionado no capítulo 2, existem diversas opções de implementação para este cenário, desde as descrições dos serviços, até os registros enriquecidos com estas descrições (e.g. UDDI enriquecido com OWL-S (AKKIRAJU, 2003), (PAOLUCCI, 2002a). Considerando que uma mesma aplicação cliente pode acessar diferentes representações de serviços semelhantes, o uso de um serviço de mediação torna-se imprescindível. As especificações de mediadores WSMO (PAOLUCCI; SRINIVASAN; SYCARA, 2004) constituem um esforço neste sentido. Entretanto, alguns aspectos relacionados com a funcionalidade, técnicas e contexto de atuação de serviços de mediação ainda carecem de melhores considerações. O modelo PERSONÆ representa uma alternativa neste sentido, como forma de orientar a implementação de mecanismos de mediação com base em reconciliação de ontologias.

De forma análoga a outros tipos de serviços, um mediador também deve ter uma descrição que especifique precisamente a funcionalidade oferecida (e.g. técnicas de mediação utilizadas) e aspectos de qualidade na mediação (e.g. confiabilidade e imparcialidade da mediação oferecida). Tratar um mediador como um tipo de serviço específico, o qual possa ser invocado automaticamente, pode trazer benefícios às partes interessadas na negociação de serviços. No lado do cliente, a invocação de mediadores pode simplificar o projeto de aplicações.

Vale destacar que, com relação à aplicação cliente (*ServiceRequester*), é de grande importância o projeto de interfaces adequadas para receber as especificações de qualidade do usuário. Para qualquer tipo de aplicação, a interface do usuário é um elemento fundamental. No contexto da personalização de serviços esta interface do usuário é o meio, a partir do qual, a qualidade de serviço desejada é especificada, negociada, percebida e adaptada. Estas múltiplas funcionalidades sem dúvida tornam a construção da interface um grande desafio. Embora a discussão de técnicas voltadas à construção de interfaces amigáveis e funcionais não faça parte do escopo deste trabalho, é fundamental entender a importância deste elemento em um cenário real, que envolva a personalização de serviços.

No lado do usuário que utiliza estas aplicações, as buscas podem retornar um maior número de resultados, provenientes de vários tipos de provedores (i.e. não apenas provedores que utilizem um determinado padrão de descrição de serviços). No lado do provedor, ampliam-se as possibilidades de uso de diversos padrões e de melhor atendimento aos usuários. Em outras palavras, a desoneração da arquitetura, mediante o uso de serviços específicos de mediação, permite que provedores e clientes desloquem seus esforços para tarefas como a negociação de serviços. Esta flexibilidade se traduz na garantia de comunicação entre as partes com a preservação das diferenças.

## 5.4 CONSIDERAÇÕES FINAIS

Neste capítulo, foi mostrado um cenário de utilização do modelo PERSONÆ, com base no domínio de serviços de reservas de passagens aéreas. Foram discutidos aspectos relacionados com a aplicação de técnicas específicas de reconciliação de ontologias de acordo com o contexto da tarefa. O objetivo foi a demonstração da aplicabilidade do modelo como abordagem orientada à mediação semântica, tomando como base o Sistema de Informações Turísticas (SEI-Tur) em exemplos hipotéticos. Finalmente foram discutidos alguns aspectos a serem considerados na implementação do modelo.

# CAPÍTULO 6

# CONCLUSÕES E TRABALHOS FUTUROS

*"A mente que se abre a uma nova idéia jamais voltará ao seu tamanho original."*
[Albert Einstein]

*Um modelo de mediação que permita a aproximação de clientes e provedores no oferecimento de serviços diferenciados inclui vários aspectos. A abordagem utilizada para a concepção do modelo PERSONÆ permitiu visualizar estes aspectos sob o ponto de vista da reconciliação de conceitos que representam as diferentes visões que clientes e provedores detêm sobre um mesmo serviço.*

*Neste capítulo são discutidas as contribuições do modelo PERSONÆ no aspecto da provisão de serviços personalizados. Contribuições secundárias também são descritas. Finalmente, são apresentadas as perspectivas que orientam trabalhos futuros.*

## 6.1 RESUMO DAS CONTRIBUIÇÕES

Uma das principais contribuições deste trabalho refere-se à mediação como alternativa de aproximação entre clientes e provedores em um cenário orientado ao oferecimento de serviços personalizados. No contexto deste trabalho, a mediação consiste na reconciliação de termos utilizados para a descrição de serviços de um mesmo domínio. Este procedimento, denominado *reconciliação ontológica*, representa a essência da mediação semântica do modelo PERSONÆ.

Conforme detalhada nos capítulos 3 e 4, a estratégia utilizada na concepção deste modelo consistiu primeiramente na formalização dos mecanismos de fusão, integração e alinhamento de ontologias. O formalismo utilizado foi a notação Z, baseado em teoria de conjuntos e lógica de primeira ordem. A escolha deste formalismo foi orientada pela existência de relações de morfismo com linguagens da Web semântica (e.g. *Web Ontology Language* – OWL), objeto de estudo de trabalhos correlatos (DONG, 2004), (LUCANU; LI; DONG, 2005). O nível de expressividade da notação Z torna este formalismo adequado

para a verificação de propriedades de ontologias em nível de prova, conforme os níveis de linguagens definidas para a Web semântica. Isto permitiu não somente um entendimento mais aprofundado de cada técnica, como também a verificação de propriedades específicas que justificam a aplicabilidade destas técnicas nas tarefas de descoberta, seleção e estabelecimento de contratos de serviços.

A caracterização da fusão de ontologias permitiu a associação desta técnica com a fase de descoberta de serviços. Visto que o objetivo nesta fase é agregar o maior número de serviços que possam atender, em potencial, a um determinado cliente, a fusão permite a formação de uma visão comum na consulta por serviços que estão descritos segundo representações diferentes. Isto amplia as possibilidades na descoberta, permitindo uma busca independente de definições de padrões específicos para definição dos conceitos que caracterizam um serviço.

A análise das propriedades da técnica de alinhamento permitiu a associação desta com a fase de seleção de serviços. Considerando que esta técnica representa um mecanismo semi-automático, onde o usuário é quem define os mapeamentos de conceitos que lhe interessam, existe uma forte relação deste mecanismo com a fase de seleção de serviços. Neste caso, o objetivo é colocar à mão do usuário as opções relacionadas com os níveis de qualidade de serviço oferecidas. Como estes requisitos de qualidade são de natureza subjetiva, variando de acordo com o nível de exigência de cada usuário, o alinhamento permite a verificação das intersecções que possam existir entre o que o usuário quer e o que o provedor está disposto a oferecer, em termos de qualidade de serviço.

Contribuições secundárias também podem ser relacionadas, dentre elas a proposta de uma abordagem que combina integração e alinhamento de ontologias no estabelecimento de contratos de serviços. Neste caso, a integração de ontologias foi proposta como abordagem para reunir perfis de usuários e provedores em diferentes formas de representação. Estes perfis variam de acordo com o domínio de aplicação do serviço. Por exemplo, um perfil de provedor de serviços de informações sobre companhias aéreas pode ser diferente do perfil de um provedor de serviços de hotelaria. De igual modo, existem inúmeras propostas de ontologias para descrever perfis de usuários. O alinhamento de ontologias foi proposto para a verificação de intersecções entre as prioridades sobre requisitos de qualidade do usuário e os requisitos oferecidos de fato pelo provedor. Possíveis intersecções podem ser utilizadas para a definição do nível de qualidade acordada. Vale salientar que a proposta considera apenas o alinhamento de propriedades de valores. Idealmente, um processo de negociação de serviços precede o estabelecimento de contratos.

Isto está relacionado com a análise de correlações entre requisitos não-funcionais, através da qual pode ser verificada a existência de conflitos ou convergências entre estes requisitos.

Outras contribuições incluem a verificação do uso de métodos formais no tratamento de problemas da Web semântica. A formalização fornece um nível de abstração suficiente para o entendimento do problema, além de permitir que ênfase especial seja dada no tratamento de propriedades específicas que caracterizam o problema em foco. Isto confere um caráter mais genérico ao modelo gerado, tornando possível a sua utilização na construção de diversas aplicações que operam sobre o paradigma orientado a serviços.

Finalmente, o modelo PERSONÆ representa um esforço no sentido de reafirmar a importância da tarefa de mediação para aproximar clientes e provedores de serviços. Neste sentido, o modelo pode servir de base para a implementação de serviços específicos de reconciliação de ontologias, que permitam uma comunicação entre as partes envolvidas na preservação da liberdade pela escolha dos padrões mais apropriados para a descrição de seus interesses.

## 6.2 TRABALHOS FUTUROS

Todas as áreas citadas durante a descrição das contribuições do modelo PERSONÆ abrem inúmeras possibilidades para trabalhos futuros. Contudo, existe um especial interesse na implementação de serviços de mediação específicos para reconciliação de ontologias. Isto pode envolver o uso de uma abordagem de refinamento progressivo da especificação, desde o nível de definições matemáticas, até a geração de código executável.

Outro aspecto a ser investigado é a mediação em nível de composição de serviços. De acordo com Medjahed et al. (MEDJAHED; BOUGUETTAYA; ELMAGARMID, 2003), a composição de serviços pode ocorrer em vários níveis, desde o nível de troca de mensagens até o nível de coesão de serviços (*composition soundness*). As informações que definem os modelos de composição de serviços (*workflows*) também podem ser definidas de diferentes formas, conforme detalhado no capítulo 2. Isto torna necessária uma abordagem centrada na mediação para a fase de composição de serviços. Além disso, uma análise sobre a coesão de serviços (*composition soundness*) pode permitir avaliar até que ponto um serviço simples é mais vantajoso para o cliente do que uma composição de serviços. Por exemplo, um único serviço de reserva de transportes pode ter um custo mais alto do que uma composição de serviços mais simples que forneçam a mesma funcionalidade. Isto implica na investigação de quais técnicas de reconciliação podem ser

utilizadas por um mediador de forma a maximizar a coesão de serviços em termos de requisitos de QoS.

A exploração de ontologias na inferência de correlações entre requisitos não-funcionais também representam outro aspecto de investigação posterior. Estas relações representam o aspecto principal na definição de políticas de negociação sobre serviços. O acordo sobre estas políticas de negociação pode minimizar as possibilidades de rupturas de contratos, no caso da definição precisa das penalidades resultantes neste ato. Desta forma, torna-se necessária uma abordagem para monitoramento da qualidade de serviço oferecida. Desta fase, dependem os processos de renegociação e reconfiguração dinâmica de serviços.

De forma geral, o modelo proposto nesta dissertação de mestrado, bem como as perspectivas de trabalhos futuros, estão direcionados para a validação da hipótese de que uma abordagem centrada na mediação pode aproximar clientes e provedores em um cenário orientado a serviços personalizados. A transparência neste processo pode ser traduzida em termos de vantagens para ambos os lados. No lado do cliente, ampliam-se as possibilidades na descoberta e escolha dos serviços que melhor atendam às suas reais necessidades. No lado do provedor, o termo "padrão" na descrição de seus serviços pode passar a significar uma livre "opção". Isto significa basicamente a possibilidade de comunicação com a preservação das diferenças.

# REFERÊNCIAS BIBLIOGRÁFICAS


ABELS, S.; HAAK, L.; HAHN, A. Identification of Common Methods Used for Ontology Integration Tasks. In: *Proc. ACM IHIS,* 2005. p. 75-78.

AGGARWAL, R.; VERMA, K.; MILLER, J.; MILNOR, W. Constraint Driven Web Service Composition in METEOR-S. In: *Proc. IEEE SCC*, 2004. p. 23-30.

AKKIRAJU, R.; FARRELL, J.; MILLER, J.; NAGARAJAN, M.; SCHMIDT, M.; SHETH, A.; VERMA, K. Web Service Semantics – WSDL-S. W3C Member Submission, version 1.0, November 2005. Disponível em: <http://www.w3.org/Submission/WSDL-S/>. Acesso em: 01 maio 2007.

AKKIRAJU, R.; GOODWIN, R.; DOSHI, P.; ROEDER, S. A Method for Semantically Enhancing the Service Discovery Capabilities of UDDI. In: *Proceedings of the Workshop on Information Integration on the Web*, p. 87–92, August 2003.

ALONSO, G.; CASATI, F.; KUNO, H.; MACHIRAJU, V. Web Services: Concepts, Architecture and Applications. Springer-Verlag, 2004.

ANDRADE, F. G. WebS Composer: Uma Ferramenta baseada em Ontologias para Descoberta e Composição de Serviços na Web. Dissertação de Mestrado, UFCG, Brasil, 2006.

BALKE, W.; WAGNER, M. Through Different Eyes: Assessing Multiple Conceptual Views for Querying Web Services. In: *Proc. WWW Conference* 2004. p. 196-205.

BALKE, W.; WAGNER, M. Towards Personalized Selection of Web Services. In: *Proc. WWW Conference* 2003.

BARDEN, R.; STEPNEY, S.; COOPER, D. *Z in Practice*. BCS Practitioners Series. Prentice Hall, 1994.

BATTLE, S. ET AL. Semantic Web Services Ontology (SWSO) W3C Member Submission 9 September 2005. Disponível em: <http://www.w3.org/Submission/SWSF-SWSO/>. Acesso em: 01 maio 2007.

BAUMEISTER, H. Relating Abstract Datatypes and Z-schemata. In: BERT, D.; CHOPPY, C. (eds.). In: *Recent Trends in Algebraic Development Techniques - Selected Papers*, v. 1827 of LNCS, p. 366-382, Springer-Verlag, 2000.

BECK, K. *Smalltalk Best Practice Patterns*. Prentice Hall, 1997.

BECKETT, D. RDF/XML Syntax Specification (Revised) W3C Recommendation 10 February 2004. Disponível em: <http://www.w3.org/TR/rdf-syntax-grammar/>. Acesso em: 01 maio 2007.

BELLUCCI, E.; ZELEZNIKOW, J. Representations for Decision Making Support in Negotiation. *Journal of Decision Support*, v. 10, n.3-4, p. 449-479, 2001.

BENATALLAH, B.; HACID, M. S.; REY, C.; TOUMANI, F. Towards Semantic Reasoning for Web Services Discovery. In: *Proc. ISWC*, 2003. Springer Verlag.

BENBERNOU, S.; HACID, M. Resolution and Constraint Propagation for Semantic Web Services Discovery. *Distributed and Parallel Databases,* v. 18, n. 1, p. 65-81, July 2005.

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

BERNSTEIN, A.; KLEIN, M. Towards High-Precision Service Retrieval. *IEEE Internet Computing*, v. 8, n. 1, p. 30-36, 2004.

BIANCHINI, D.; ANTONELLIS, V. D.; MELCHIORI, M. QoS in Ontology-Based Service Classification and Discovery. In: *Proc. IEEE DEXA*, 2004. p. 145-150.

BONNET, V.; BOUDAOUD, K.; GAGNEBIN, M.; HARMS, J.; SHULTZ, T. Online Dispute Resolution Systems as Web Services. In: *Proc. Hewlett-Packard Open View University Association Workshop*, June 2002.

BOOTH, D.; HAAS, H.; MCCABE, F.; NEWCOMER, E.; CHAMPION, M.; FERRIS, C.; ORCHARD, D. (eds.). Web Service Architecture, W3C Working Group Note, 11 February 2004. Disponível em: <http://www.w3.org/TR/ws-arch/>. Acesso em: 01 maio 2007.

BOWEN, J. P. *Formal Specification and Documentation using Z: A Case Study Approach*. International Thomson Computer Press, 1996.

BRAY, T. ET AL. Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation. Disponível em: <www.w3.org/TR/REC-xml>. Acesso em: 01 maio 2007.

BRUIJN, J. D. ET AL. State-of-the-art Survey on Ontology Merging and Aligning. SEKT-project report D4.2.1 (WP4), IST-2003-506826, 2003.

BURSTEIN, M.; BUSSLER, C.; ZAREMBA, M.; FININ, T.; HUHNS, M. N.; PAOLUCCI, M.; SHETH, A. P.; WILLIAMS, S. A Semantic Web Services Architecture. *IEEE Internet Computing*, v. 9, n. 5, p. 72-81, September 2005.

BUSSLER, C.; FENSEL, D.; SADEH, N. The Role of Semantic Web Services in Enterprise Application Integration and E-commerce. *International Journal of Electronic Commerce (IJEC)*, v. 9, n. 2, 2005.

BUSSLER, C.; MAEDCHE, A.; FENSEL, D. Web Services: Quo Vadis? *IEEE Intelligent Systems, Trends & Controversies*, Jan./Feb. 2003.

CASATI, F.; ILNICKI, S.; JIN, L.; KRISHNAMOORTHY, V.; SHAN, M. Adaptive and Dynamic Service Composition in e-Flow. In: *Proc. CAiSE 2000*. LNCS, v. 1789, p. 13-31.

CHEN, W.; KIFER, M.; WARREN, D. S. HiLog: A Foundation for Higher-Order Logic Programming. *Journal of Logic Programming*, v. 15, n. 3, p. 187–230, 2003.

CHRISTENSEN, E.; CURBERA, F.; MEREDITH, G.; WEERAWARANA, S. Web Services Description Language (WSDL). W3C Note 15 March 2001. Disponível em: <www.w3.org/TR/wsdl>. Acesso em: 01 maio 2007.

CIMPIAN, E.; MOCAN, A.; STOLLBERG, M. Mediation Enabled SemanticWeb Services Usage. In: *Proc. ASWC 2006.* LNCS, Springer-Verlag.

COHEN, B. Justification of Formal Methods for System Specification. *Software Engineering Journal*, p. 26-35, 1989.

COHEN, W.; RAVIKUMAR, P.; FIENBERG, S. A Comparison of String Distance Metrics for Name-matching Tasks. In: *Proc. IJCAI* 2003.

COLUCCI, S.; NOIA, T. D.; SCIASCIO, E. D.; DONINI, F.; MONGIELLO, M. Concept Abduction and Contraction for Semantic-Based Discovery of Matches and Negotiation Spaces in an e-Marketplace. In: *Proc. ICEC 2004*. ACM Press, 2004.

CONSTANTINESCU, I.; FALTINGS, B. Efficient Matchmaking and Directory Services. In: Proc. of *IEEE/WIC International Conference on Web Intelligence*, 2003.

COPLIEN, J. O. A Generative Development-Process Pattern Language. In: COPLIEN, J. O.; SCHMIDT, D. C. (eds.), *Pattern Languages of Program Design*. Addison Wesley, 1995.

DAVIS, N. J.; FENSEL, D.; RICHARSON, M. The Future of Web Services. *BT Technology Journal*, v. 22, n. 1, p.118-130, 2004.

DOAN, A.; MADHAVAN, J.; DHAMANKAR, R.; DOMINGOS, P.; HALEVY, A. Learning to Match Ontologies on the Semantic Web. *VLDB Journal*, v.12, n. 4, p. 303-319, November 2003.

DOGAC, A.; KABAK, Y.; LALECI, G.; SINIR, S.; YILDIZ, A.; KIRBAS, S.; GURCAN, Y. Semantically Enriched Web Services for the Travel Industry. *SIGMOD Record,* v.33, n. 3, p. 21-27, 2004.

DOMINGUE, J.; CABRAL, L.; HAKIMPOUR, F.; SELL, D.; MOTTA, E. IRS-III: A Platform and Infrastructure for Creating WSMO-based Semantic Web Services. In: *Proc. WIW* 2004.

DONG, J. S.; LEE, C. H.; LI, Y. F.; WANG, H. Verifying DAML+OIL and Beyond in Z/EVES. In: Proc. ACM/IEEE ICSE 2004. p. 201-210.

DONG, J. S.; SUN, J.; WANG, H. Z Approach to Semantic Web. In: *Proc. ICFEM* 2002. LNCS, Springer-Verlag, p. 156-167.

DOU, D.; MCDERMOTT, D.; QI, P. Ontology Translation by Ontology Merging and Automated Reasoning. In: *Proc. EKAW* 2002. p. 3–18.

EHRIG, M. *Ontology Alignment: Bridging the Semantic Gap (Semantic Web and Beyond)*. Springer-Verlag, 2006.

EHRIG, M.; STAAB, S. QOM – Quick Ontology Mapping. In: *Proc. ISWC* 2004. LNCS, Springer-Verlag, p. 683-696.

FODOR, O.; WERTHNER, H. Harmonise: A Step Toward an Interoperable E-Tourism Marketplace. *International Journal of Electronic Commerce*, v. 9, n. 2, p. 11, 2004.

FOWLER, M. *Analysis Patterns*. Addison-Wesley, 1997.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. *Design Patterns*. Addison-Wesley, 1995.

GANGEMI, A.; PISANELLI, D.; STEVE, G. Ontology Integration: Experiences with Medical Terminologies. In: GUARINO, N. (ed.), *Formal Ontology in Information Systems*, IOS Press, p. 163-178, 1998.

GARCIA-MOLINA, H. ET AL. The TSIMMIS Approach to Mediation: Data Models and Languages. *Journal of Intelligent Information Systems*, p. 117-132, 1997.

GIUNCHIGLIA, F.; SHVAIKO, P. Semantic Matching. *The Knowledge Engineering Review*, v. 18, n. 3, p. 265–280, 2004.

GÓMEZ-PEREZ, A.; FERNÁNDES-LÓPEZ, M.; CORCHO, O. Ontological Engineering: With Examples from the Areas of Knowledge Management, e-Commerce and the Semantic Web. Springer-Verlag, 2004.

GROSOF B. N. A Courteous Compiler from Generalized Courteous Logic Programs to Ordinary Logic Programs. In: *IBM Report included as part of documentation in the IBM CommonRules 1.0 software toolkit and documentation*, July 1999.

GRUNINGER, M. A Guide to the Ontology of the Process Specification Language. In: STUDER, R.; STAAB, S. (eds.) *Handbook on Ontologies in Information Systems*, Springer-Verlag, 2003.

HAARSLEV, V.; MÖLLER, R. Racer: A Core Inference Engine for the Semantic Web. In: *Proc. EON* 2003, p. 27-36.

HAMEED, A.; PREECE, A.; SLEEMAN, D. Ontology Reconciliation. In: STAAB, S., STUDER, R. (eds.), *Handbook on Ontologies in Information Systems*, Springer-Verlag, p. 231-250, 2003.

HORROCKS, I.; PARSIA, P.; PATEL-SCHNEIDER, P.; HENDLER, J. Semantic Web architecture: Stack or Two Towers? In: *Proc. PPSWR* 2005.

HUANG, J.; DANG, J.; HUHNS, M. N.; SHAO, Y. Ontology Alignment as a Basis for Mobile Service Integration and Invocation. *Journal of Pervasive Computing and Communications*, 2006.

HUHNS, M. N.; STEPHENS, L. M. Personal Ontologies. *IEEE Internet Computing*, v. 3, n. 5, p. 85-87, Sep./Oct. 1999.

ISO/IEC 13568:2002 – Z Formal Specification Notation – Syntax, Type System and Semantics. Disponível em: <http://www.iso.org>. Acesso em 01 maio 2007.

JACKY, J. *The Way of Z: Practical Programming with Formal Methods*. Cambridge University Press, 1997.

JAEGER, M.; ROJEC-GOLDMANN, G.; MUHL, C.; GEIHS, K. Ranked Matching for Service Descriptions Using OWL-S. Technical report, TU Berlin, Institute of Telekommunication Systems.

JANNINK, J.; PICHAI, S.; VERHEIJEN, D.; WIEDERHOLD, G. Encapsulation and Composition of Ontologies. In: *Proc. AAAI* 1998, July 1998.

KELLER, U.; LARA, R.; LAUSEN, H.; POLLERES, A.; FENSEL, D. Automatic Location of Services. In: *Proc. ESWC* 2005. LNCS v. 3532.

KICZALES, G. ET AL. Aspect-Oriented Programming. In: *Proc. ECOOP* 1997. v. 1241, p. 220-242.

KIFER, M.; LARA, R.; POLLERES, A.; ZHAO, C.; KELLER, U.; LAUSEN, H.; FENSEL, D. A Logical Framework for Web Service Discovery. In: *Proc. ISWC* 2004.

KLEIN M. Combining and Relating Ontologies: An Analysis of Problems and Solutions. In: *Proc. IJCAI* 2001.

KOTONYA, G.; SOMMERVILLE, I. *Requirements Engineering: Processes and Techniques*. John Wiley & Sons, 1998.

LAUSEN, H. ; POLLERES, A. ; ROMAN, D. (eds). Web Service Modeling Ontology - W3C Member Submission 3 June 2005. Disponível em: <http://www.w3.org/Submission/WSMO/>. Acesso em: 01 maio 2007.

LIN, M.; XIE, J.; GUO, H.; WANG, H. Solving QoS-Driven Web Service Dynamic Composition as Fuzzy Constraint Satisfaction. In: *Proc. IEEE EEE* 2005.

LLOYD, J. W. *Foundations of Logic Programming 2nd edition.* Springer-Verlag 1987.

LUCANU, D.; LI, Y. F.; DONG, J. S. Soundness Proof of Z Semantics of OWL Using Institutions. In: *Proc. WWW* Conference 2005. ACM Press, p. 1048-1049.

MAAMAR, Z.; MOSTEFAOUI, S. K.; MAHMOUD, Q. H. Context for Personalized Web Services. In: *Proc. HICSS* 2005, v. 7. IEEE Computer Society, Washington, DC, 166.2.

MAIA, A. C. J. SEI-Tur: Um Sistema de Criação de Roteiros Turísticos. Dissertação de Mestrado, UFCG, Brasil, 2004.

MANDELL, D.; MCILLRAITH, S. A Bottom-up Approach to Automating Web Service Discovery, Customization, and Semantic Translation. In: *Proc. WWW* 2003. ACM Press, 2003.

MARTIN, D.; BURSTEIN, M.; HOBBS, J.; LASSILA, O.; MCDERMOTT, D.; MCILRAITH, S.; NARAYANAN, S.; PAOLUCCI, M. OWL-S: Semantic Markup for Web Services - W3C Member Submission 22 November 2004. Disponível em: <http://www.w3.org/Submission/OWL-S>. Acesso em : 01 maio 2007.

MCGUINESS, D. L.; HARMELEN, F.V. (eds.). Ontology Language Overview. W3C Recomendation 10 February 2004. Disponível em: <http://www.w3.org/TR/owl-features/>. Acesso em: 01 maio 2007.

MEDJAHED, B.; BOUGUETTAYA, A.; ELMAGARMID, A. Composing Web services on the Semantic Web. *The VLDB Journal*, v.12, n. 4, p. 333-351, November 2003.

MENA, E.; KASHYAP, V.; ILLARRAMENDI, A.; SHETH, A. Domain Specific Ontologies for Semantic Information Brokering on the Global Information Infrastructure. In: GUARINO, N. (ed.), *Proc. FOIS* 1998, IOS Press, p. 269–283.

MILANOVIC, N.; MALEK, M. Current Solutions for Web Service Composition. *IEEE Internet Computing*, v. 8, n. 1, p. 51-59, Jan./Fev. 2004.

MITRA, P.; WIEDERHOLD, G. An Ontology-Composition Algebra. In: STAAB, S., AND STUDER, R. (eds.) In: *Handbook on Ontologies*, Springer Series: International Handbooks on Information Systems, p. 93-113, 2004.

MOCAN, A. ET AL.(eds.). WSMO Mediators, Working Draft 16 September 2005. Disponível em: <http://www.wsmo.org/TR/d29/>. Acesso em 01 maio 2007.

MOTTA, E. An Overview of the OCML Modelling Language. In: *Proc. KEML* 1998.

NADLER, J. Electronically-Mediated Dispute Resolution and E-Commerce. *Negotiation Journal*, v.17, p. 333–347, 2001.

NILES, I.; PEASE, A. Towards a Standard Upper Ontology. In: *Proc. FOIS* 2001. ACM Press, p. 2-9.

NOY, N. F. Tools for Mapping and Merging Ontologies. In: STAAB, ST.; STUDER, R. (eds.). In: *Handbook on Ontologies*. International Handbooks on Information Systems, Springer 2004.

NOY, N.; MUSEN, M. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In: *Proc. AAAI* 2000.

Object Management Group Inc. (OMG). Meta Object Facility (MOF) Specification v1.4, 2002.

O'SULLIVAN, J.; EDMOND, D.; HOFSTEDE, A.H.T. Formal Description of Non-Functional Service Properties. Technical FIT-TR-2005-01, Queensland University of Technology, Brisbane, 2005.

PAOLUCCI, M.; KAWAMURA, T.; PAYNE, T. R.; SYCARA, K. P. Importing the Semantic Web in UDDI. In: *Revised Papers From the International Workshop on Web Services, E-Business, and the Semantic Web,* LNCS, Springer-Verlag, v. 2512, p. 225-236.

PAOLUCCI, M.; KAWAMURA, T.; PAYNE, T. R.; SYCARA, K. P. Semantic Matching of Web Services Capabilities. In: *Proc. ISWC* 2002. LNCS, Springer-Verlag, v. 2342, p. 333-347.

PAOLUCCI, M.; SRINIVASAN, N.; SYCARA, K. Expressing WSMO Mediators in OWL-S. In: *Proc. ISWC* 2004, p.120–134.

PAPAIOANNOU, I. V.; TSESMETZIS, D. T.; ROUSSAKI, I. G.; ANAGNOSTOU, M. E. A QoS Ontology Language for Web-Services. In: *Proc. AINA* 2006. v. 1, p. 101-106.

PATIL, A. A.; OUNDHAKAR, S. A.; SHETH, A. P.; VERMA, K. METEOR-S Web Service Annotation Framework. In: *Proc. WWW* 2004. ACM Press, p. 553-562.

PINTO, H. S.; GÓMEZ-PEREZ, A.; MARTINS, J. P. Some Issues on Ontology Integration. In: *Proc. IJCAI* 1999.

PINTO, H. S.; MARTINS, J. P. A Methodology for Ontology Integration. In: *Proc. K-CAP* 2001. ACM Press, p. 131-138.

POTTER, B.; SINCLAIR, J.; TILL, D. *An Introduction to Formal Specification and Z.* Prentice-Hall, 1996.

PREECE, A.; DECKER, S. Intelligent Web Services. *IEEE Intelligent Systems*, v. 17, n. 1, p. 15-17, Jan./Feb. 2002.

RAHM, E.; BERNSTEIN, P. A. A Survey of Approaches to Automatic Schema Matching. *VLDB Journal*, v. 10, n. 4, p.334–350, 2001.

RIBEIRO, C. M. F. A. ESCHER: Uma Arquitetura de Qualidade de Serviço para Tratar a Percepção do Usuário. Tese de doutorado, UFPE, Brasil, 2004.

RIBEIRO, C. M. F. A.; ROSA, N. S.; CUNHA, P. R. F. An Ontological Approach for Personalized Services. In: *Proc. IEEE AINA* 2006. v. 2, p. 729-733.

RIBEIRO, C. M. F. A.; ROSA, N. S.; CUNHA, P. R. F. Meaningful SLA: Towards a QoS Contract Based on User Perception. In: *Proc. IEEE EDOC* 2004.

RIBEIRO, C. M. F. A.; ROSA, N. S.; CUNHA, P. R. F. Towards a Model for Personalized Communication Services based on User Perception. In: *Proc. ICEIS* 2004. v. 5, p. 211-215.

RIBEIRO, C.M.F.A.; ROSA, N. S.; CUNHA, P. R.F. Towards a Model for Personalized Communication Services. In: *Proc. IEEE AINA* 2004, vol.2, p. 99- 102.

SAALTINK, M. The Z/EVES System. In: BOWEN, J. P., HINCHEY, M. G., TILL, D. (eds.), ZUM'97: Z Formal Specification Notation, LNCS, Springer-Verlag, p. 72-85, 1997.

SCHORLEMMER, M.; KALFOGLOU, Y. Progressive Ontology Alignment for Meaning Coordination: An Information-theoretic Foundation. In: *Proc. ACM AAMAS* 2005. ACM Press, p. 737-744.

SHETH, A.; THACKER, S.; PATEL, S. Complex Relationships and Knowledge Discovery Support in the InfoQuilt System. *VLDB Journal* v. 12, n. 1, p. 2-27, May 2003.

SHNEIDERMAN, B. Designing Trust into Online Experiences. *ACM Communications,* v. 43, n. 12, p. 7-59, December 2000.

SILVA, P. A.; RIBEIRO, C. M. F. A.; SCHIEL, U.; QUEIROZ, J. E. R. A Formal Approach to Semantic Mediation in SOA: Specification and Proof. In: *Proc. IRMA* 2007.

SILVA, P. A.; SCHIEL, U.; RIBEIRO, C. M. F. A.; QUEIROZ, J. E. R. Extending SOA with Semantic Mediators. In: *ACM/IEEE Proc. SITIS* 2006. p. 208-217.

SPIVEY, J. M. *The Z Notation: A Reference Manual. 2ª Edição*. Prentice Hall, 1992.

SRINIVASAN, N.; PAOLUCCI, M.; SYCARA, K. Adding OWL-S to UDDI, Implementation and Throughput. In: *Proc. SWSWPC* 2004.

SRINIVASAN, N.; PAOLUCCI, M.; SYCARA, K. Semantic Web Service Discovery in the OWL-S IDE. In: *Proc. IEEE HICSS* 2006. v. 6.

STEPHENS, L. M.; GANGAM, A. K.; HUHNS, M. N. Constructing Consensus Ontologies for the Semantic Web: A Conceptual Approach. In: *Proc. WWW Conference* 2004. v. 7, n. 4, p. 421-442.

STEPNEY, S.; POLACK, F.; TOYN, I. A Z Patterns Catalogue I: Specification and Refactorings, v0.1. University of York Technical Report YCS-2003-349, January 2003.

STOLLBERG, M.; CIMPIAN, E.; FENSEL, D. Mediating Capabilities with Delta-Relations. In: *Proc. International Workshop on Mediation in Semantic Web Services*, 2005.

STOLLBERG, M.; KELLER, U.; FENSEL, D. Partner and Service Discovery for Collaboration Establishment with Semantic Web Services. In: *Proc. IEEE ICWS* 2005, v. 0, p. 473-480.

STOLLBERG, M.; LAUSEN, H.; POLLERES, A.; LARA, R. (eds.). WSMO Use case "Virtual Travel Agency", WSMO Working Draft 04 October 2004, v. 0.1. Disponível em: < http://www.wsmo.org/2004/d3/d3.2/b2c/20041004/>. Acesso em : 01 maio 2007.

STUMME, G.; MAEDCHE, A. Ontology Merging for Ontologies on the Semantic Web. In: *Proc. FMII* 2001.

SYCARA, K.; PAOLUCCI, M.; ANOLEKAR, A.; SRINIVASAN, N. Automated Discovery, Interaction and Composition of Semantic Web Services. *Web Semantics*, v. 1, n. 1, Elsevier, 2003.

SZYPERSKI, C. *Component Software: Beyond Object-Oriented Programming*. 2nd ed. Addison-Wesley Professional, Boston 2002

TUNG, H.; LIN, R. J. Automated Contract Negotiation Using a Mediation Service. In: *Proc. IEEE Cec* 2005. v. 0, p. 374-377.

TZITZIKAS, Y.; SPYRATOS, N.; CONSTANTOPOULOS, P. Mediators Over Ontology-based Information Sources. In: *Proc. WISE* 2001. v. 1, p. 31.

USCHOLD, M. Where is the Semantics in the Semantic Web?. *Artificial Inteligence Magazine*, 2003.

VALENTINE, S. H.; STEPNEY, S.; TOYN, I. A Z Patterns Catalogue II: Definitions and Laws, v. 01. University of York Technical Report YCS-2004-383, October 2004.

HARMELEN, F. van; FENSEL, D. Formal Methods in Knowledge Engineering. *The Knowledge Engineering Review*, v. 10, n. 4, p. 345-360, 1995.

VISSER, U. *Intelligent Information Integration for the Semantic Web*. Springer-Verlag, 2004.

WAGNER, M.; KELLERER, W. Web Services Selection for Distributed Composition of Multimedia Content. In: *Proc. ACM MULTIMEDIA* 2004. ACM Press, p. 104-107.

WANG, H.; HUANG, J.; QU, Y.; XIE, J. Web Services: Problems and Future Directions. *Journal of Web Semantics,* v. 1, p. 309-320, 2004.

WANG, J.; GASSER, L. Mutual Online Ontology Alignment. In: *Proc. AAMAS* 2002.

WEIBEL, S.; KUNZE, J.; LAGOZE, C.; WOLF M. Dublin Core Metadata for Resource Discovery. RFC 2413, IETF, September 1998.

WIEDERHOLD, G. Intelligent Integration of Information. *International Journal of Intelligent Information Systems,* v. 6, n. 2–3, p. 93–97, June 1996.

WIEDERHOLD, G. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, v. 25, n. 3, p. 38-49, March 1992.

WIEDERHOLD, G. Obtaining Precision when Integrating Information. In: FILIPE, J., SHARP, B., MIRANDA, P. (eds.), *Enterprise Information Systems III*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

WIEDERHOLD, G.; GENESERETH, M. The Conceptual Basis for Mediation Services. *IEEE Expert, Intelligent Systems and their Applications*, p. 38-47, Sep./Oct. 1997.

WOOD, K. R. A Practical Approach to Software Engineering Using Z and the Refinement Calculus. In: *SIGSOFT Software Engineering Notes,* v. 18, n. 5, p. 79-88, December 1993.

WOODCOCK, J.; DAVIE, J. *Using Z – Specification, Refinement, and Proof*. Prentice Hall, 1996.

YAN, L.; ÖZSU, M. T.; LIU, L. Accessing Heterogeneous Data through Homogenization and Integration Mediators. In: *Proc. CoopIS* 1997. p. 130-139.

YERNENI, R.; LI, C.; GARCIA-MOLINA, H.; ULLMAN, J. Computing Capabilities of Mediators. In: *Proc. ACM SIGMOD 1999*. ACM Press, p. 443-454.

ZHOU, C.; CHIA, L. T.; LEE, B. S. DAML-QoS Ontology for Web Services. In: *Proc. IEEE ICWS* 2004. p. 472.

# APÊNDICE. ESPECIFICAÇÃO FORMAL EM NOTAÇÃO Z DA ARQUITETURA PERSONÆ

[*CLASS*, *DATATYPE*, *ID*, *INSTANCE*, *MESSAGE*, *OPERATION*, *PROTOCOL*, *RESOURCE*, *URI*]

*REPORT* ::= *Match*
| *Impasse*
| *ServiceDiscovered*
| *NoServiceDiscovered*
| *NoServiceSelected*
| *ServiceSelected*

*ObjectProperty*

$$Id: ID$$
$$Relationship: CLASS \leftrightarrow CLASS$$

*DataTypeProperty*

$$Id: ID$$
$$Relationship: CLASS \nrightarrow DATATYPE$$

$$Individual: CLASS \rightarrow \mathbb{P}\ INSTANCE$$

$$LEVEL: \mathbb{P}\ DATATYPE$$

$$ObjPropInstantiation: ObjectProperty \rightarrow Individual \leftrightarrow Individual$$
$$DtPropInstantiation: DataTypeProperty \rightarrow Individual \nrightarrow DATATYPE$$

*ClassConstructors*

$$IntersectionOf: \text{seq}\ CLASS \rightarrow CLASS$$
$$UnionOf: \text{seq}\ CLASS \rightarrow CLASS$$
$$ComplementOf: CLASS \leftrightarrow CLASS$$

---

$$\forall c1, c2: CLASS;\ cn: \text{seq}\ CLASS \bullet IntersectionOf\ cn = c1 \Leftrightarrow Individual\ c1 = \cap \{ x: \text{ran}\ cn \bullet (Individual\ x) \} \wedge (UnionOf\ cn = c1 \Leftrightarrow Individual\ c1 = \cup \{ x: \text{ran}\ cn \bullet (Individual\ x) \}) \wedge (c1\ ComplementOf\ c2 \Leftrightarrow INSTANCE \setminus Individual\ c1 = Individual\ c2)$$

*ClassAxioms*

$$
\begin{array}{l}
isSuperClassOf: CLASS \leftrightarrow CLASS \\
isSubClassOf: CLASS \leftrightarrow CLASS \\
isSiblingClassOf: CLASS \leftrightarrow CLASS \\
EquivalentClass: CLASS \leftrightarrow CLASS \\
DisjointWith: CLASS \leftrightarrow CLASS \\
\hline
\forall c1, c2, c3: CLASS \\
\quad \bullet\ c1\ isSuperClassOf\ c2 \\
\qquad \Leftrightarrow Individual\ c2 \subseteq Individual\ c1 \\
\qquad\quad \land\ (c1\ isSubClassOf\ c2 \Leftrightarrow Individual\ c1 \subseteq Individual\ c2) \\
\qquad\quad \land\ (c1\ isSiblingClassOf\ c2 \\
\qquad\qquad \Leftrightarrow c3\ isSuperClassOf\ c1 \land c3\ isSuperClassOf\ c2 \\
\qquad\qquad\quad \lor\ c1\ isSubClassOf\ c3 \land c2\ isSubClassOf\ c3) \\
\qquad\quad \land\ (c1\ EquivalentClass\ c2 \Leftrightarrow Individual\ c1 = Individual\ c2) \\
\qquad\quad \land\ (c1\ DisjointWith\ c2 \Leftrightarrow Individual\ c1 \cap Individual\ c2 = \varnothing)
\end{array}
$$

*ObjectPropertyAxioms*

$$
\begin{array}{l}
SubObjectPropertyOf: ObjectProperty \leftrightarrow ObjectProperty \\
SuperObjectPropertyOf: ObjectProperty \leftrightarrow ObjectProperty \\
EquivalentObjectProperty: ObjectProperty \leftrightarrow ObjectProperty \\
TransitiveProperty: \mathbb{P}\ ObjectProperty \\
SymmetricProperty: \mathbb{P}\ ObjectProperty \\
InversePropertyOf: ObjectProperty \leftrightarrow ObjectProperty \\
\hline
\forall objP1, objP2, objPn: ObjectProperty; i1, i2, i3: Individual \\
\quad \bullet\ objP1\ SubObjectPropertyOf\ objP2 \\
\qquad \Leftrightarrow objP1 \in ObjectProperty \land objP2 \in ObjectProperty \\
\qquad\quad \Rightarrow ObjPropInstantiation\ objP1 \subseteq ObjPropInstantiation\ objP2 \\
\qquad\qquad \land\ (objP1\ SuperObjectPropertyOf\ objP2 \\
\qquad\qquad\quad \Leftrightarrow objP1 \in ObjectProperty \land objP2 \in ObjectProperty \\
\qquad\qquad\qquad \Rightarrow ObjPropInstantiation\ objP2 \subseteq ObjPropInstantiation\ objP1) \\
\qquad\qquad \land\ (objP1\ EquivalentObjectProperty\ objP2 \\
\qquad\qquad\quad \Leftrightarrow objP1 \in ObjectProperty \land objP2 \in ObjectProperty \\
\qquad\qquad\qquad \Rightarrow ObjPropInstantiation\ objP1 = ObjPropInstantiation\ objP2) \\
\qquad\qquad \land\ (objPn \in TransitiveProperty \\
\qquad\qquad\quad \Leftrightarrow (i1, i2) \in ObjPropInstantiation\ objPn) \\
\qquad\qquad \land\ ((i2, i3) \in ObjPropInstantiation\ objPn \\
\qquad\qquad\quad \Rightarrow (i1, i3) \in ObjPropInstantiation\ objPn) \\
\qquad\qquad \land\ (objPn \in SymmetricProperty \\
\qquad\qquad\quad \Leftrightarrow (i1, i2) \in ObjPropInstantiation\ objPn \\
\qquad\qquad\qquad \Rightarrow (i2, i1) \in ObjPropInstantiation\ objPn) \\
\qquad\qquad \land\ (objP1\ InversePropertyOf\ objP2 \\
\qquad\qquad\quad \Leftrightarrow ObjPropInstantiation\ objP1 = (ObjPropInstantiation\ objP2)^{\sim})
\end{array}
$$

*DataTypePropertyAxioms*

$SuperDataTypePropertyOf: DataTypeProperty \leftrightarrow DataTypeProperty$
$SubDataTypePropertyOf: DataTypeProperty \leftrightarrow DataTypeProperty$
$EquivalentDataTypeProperty: DataTypeProperty \leftrightarrow DataTypeProperty$

---

$\forall dtP1, dtP2: DataTypeProperty$
$\bullet\ dtP1\ SuperDataTypePropertyOf\ dtP2$
$\Leftrightarrow dtP1 \in DataTypeProperty \land dtP2 \in DataTypeProperty$
$\Rightarrow DtPropInstantiation\ dtP2 \subseteq DtPropInstantiation\ dtP1$
$\land\ dtP1\ SubDataTypePropertyOf\ dtP2$
$\Leftrightarrow dtP1 \in DataTypeProperty \land dtP2 \in DataTypeProperty$
$\Rightarrow DtPropInstantiation\ dtP1 \subseteq DtPropInstantiation\ dtP2$
$\land\ (dtP1\ EquivalentDataTypeProperty\ dtP2$
$\Leftrightarrow dtP1 \in DataTypeProperty \land dtP2 \in DataTypeProperty$
$\Rightarrow DtPropInstantiation\ dtP1 = DtPropInstantiation\ dtP2)$

*IndividualAxioms*

$SameAs: Individual \leftrightarrow Individual$
$DifferentFrom: Individual \leftrightarrow Individual$

---

$\forall i1, i2: Individual \bullet\ i1\ SameAs\ i2 \Leftrightarrow i1 = i2 \land i1\ DifferentFrom\ i2 \Leftrightarrow i1 \neq i2$

*Ontology*

$Classes: \mathbb{P}\ CLASS$
$ObjectProperties: \mathbb{P}\ ObjectProperty$
$DataTypeProperties: \mathbb{P}\ DataTypeProperty$
$Individuals: \mathbb{P}\ Individual$
$ClassConstructAxioms: \mathbb{P}\ ClassConstructors$
$OntologicalClassAxioms: \mathbb{P}\ ClassAxioms$
$OntologicalObjectPropertyAxioms: \mathbb{P}\ ObjectPropertyAxioms$
$OntologicalDataTypePropertyAxioms: \mathbb{P}\ DataTypePropertyAxioms$
$OntologicalIndividualAxioms: \mathbb{P}\ IndividualAxioms$

---

$\forall c1, c2: CLASS; objp1, objp2: ObjectProperty; dtp1, dtp2: DataTypeProperty;$
$i1, i2: Individual$
$\mid\ c1 \in Classes$
$\land\ c2 \in Classes$
$\land\ objp1 \in ObjectProperties$
$\land\ objp2 \in ObjectProperties$
$\land\ dtp1 \in DataTypeProperties$
$\land\ dtp2 \in DataTypeProperties$
$\land\ i1 \in Individuals$
$\land\ i2 \in Individuals \bullet\ c1 \neq c2 \land objp1 \neq objp2 \land dtp1 \neq dtp2 \land i1 \neq i2$

*FunctionalRequirement*

*Id: ID*
*Class: CLASS*
*Domain: Ontology*

---

*Class* ∈ *Domain . Classes*

*NonFunctionalRequirement*

*Id: ID*
*Class: CLASS*
*Domain: Ontology*

---

*Class* ∈ *Domain . Classes*

*ServiceDescription*

*ServiceName: DATATYPE*
*Domain: Ontology*
*Input, Output:* ℙ *DataTypeProperty*
*Capability: FunctionalRequirement*
*QoSOffered: NonFunctionalRequirement* ⇸ *LEVEL*

---

∃*nfr: NonFunctionalRequirement*
- *nfr* ∈ dom *QoSOffered*
  - ∧ *Capability . Class* ∈ *Domain . Classes*
  - ∧ *nfr . Class* ∈ *Domain . Classes*
  - ∧ *Input* ⊆ *Domain . DataTypeProperties*
  - ∧ *Output* ⊆ *Domain . DataTypeProperties*

*ServiceInterface*

*ServiceName: DATATYPE*
*Types:* ℙ *DATATYPE*
*Bindings:* ℙ *PROTOCOL*
*PortType:* ℙ *OPERATION*
*Messages:* ℙ *MESSAGE*
*Port: DATATYPE* ⇸ *URI*

---

dom *Port* ⊆ *Types*

*Service*

*ServiceName: DATATYPE*
*Description: ServiceDescription*
*ServiceImplementation: ServiceInterface*

*ServiceRegistry*

$RegistryID : ID$

$Services : ID \pfun ServiceDescription$

*ServiceRequester*

$Profile : Ontology$

$Goal : FunctionalRequirement$

$QoSConstraints : NonFunctionalRequirement \pfun LEVEL$

$QoSPreferences : NonFunctionalRequirement \pfun LEVEL$

$QoSPriorities : NonFunctionalRequirement \times NonFunctionalRequirement \pfun NonFunctionalRequirement$

$QoSDesired : NonFunctionalRequirement \pfun LEVEL$

---

$\forall nfr1, nfr2 : NonFunctionalRequirement \mid (nfr1, nfr2) \in \mathrm{dom}\ QoSPriorities \land (nfr2, nfr1) \in \mathrm{dom}\ QoSPriorities \bullet nfr1.Domain = nfr2.Domain$

$\land (QoSPriorities\ (nfr1, nfr2) = nfr1 \land QoSPriorities\ (nfr2, nfr1) = nfr1 \lor QoSPriorities\ (nfr1, nfr2) = nfr2 \land QoSPriorities\ (nfr2, nfr1) = nfr2)$

$\land\ \mathrm{ran}\ QoSPriorities \subseteq \mathrm{dom}\ QoSPreferences$

$\land\ \mathrm{dom}\ QoSConstraints \cap \mathrm{dom}\ QoSPreferences = \emptyset$

$\land\ QoSDesired = QoSConstraints \cup QoSPreferences$

*ServiceProvider*

$Profile : Ontology$

$Services : \mathbb{P}\ Service$

$Resource : \mathbb{P}\ RESOURCE$

$QoSProvided : NonFunctionalRequirement \pfun LEVEL$

---

$\forall S : Service \mid S \in Services \bullet QoSProvided \subseteq S.Description.QoSOffered$

*ServiceLevelAgreement*

$ServiceRequesterSection, ServiceProviderSection : Ontology$

$SLAOntology : \mathbb{P}\ Ontology$

$QoSAgreed : NonFunctionalRequirement \pfun LEVEL$

---

$\forall nfr : NonFunctionalRequirement \mid nfr \in \mathrm{dom}\ QoSAgreed \bullet nfr.Class \in ServiceRequesterSection.Classes \land nfr.Class \in ServiceProviderSection.Classes$

$ServiceRequesterSection \in SLAOntology$

$ServiceProviderSection \in SLAOntology$

*Ontologies*

$OntologySet: \mathbb{P}\ Ontology$

*Mediator*

$Type: Service$

$Description: Ontology$

$Context: Ontology \leftrightarrow Ontology$

*Entities*

$ServiceRequesters: \mathbb{P}\ ServiceRequester$

$ServiceProviders: \mathbb{P}\ ServiceProvider$

$Registries: \mathbb{P}\ ServiceRegistry$

$Mediators: \mathbb{P}\ Mediator$

*SLA*

$ServiceLevelAgreements: \mathbb{P}\ ServiceLevelAgreement$

*Relationships*

$Find: ServiceRequester \nrightarrow ServiceDescription \leftrightarrow ServiceRegistry$

$Publish: ServiceProvider \nrightarrow ServiceDescription \leftrightarrow ServiceRegistry$

$Bind: ServiceRequester \rightarrow ServiceProvider$

$GeneratedBy: SLA \nrightarrow Mediator$

$ProvidedBy: Service \leftrightarrow ServiceProvider$

$DescribedBy: Entities \rightarrow Ontology$

$Mediates: Mediator \nrightarrow ServiceRequester \leftrightarrow ServiceProvider$

---

$\text{ran } Find \subseteq \text{ran } Publish$

$\text{ran } GeneratedBy \subseteq \text{dom } Mediates$

$\text{dom } Bind \subseteq \text{dom } Find$

$\text{ran } Bind \subseteq \text{ran } ProvidedBy$

*PERSONÆ*

*Ontologies*

*Entities*

*SLA*

*Relationships*

*initPERSONÆ*

$\Delta PERSONÆ$

---

$OntologySet' = \emptyset$
$Registries' = \emptyset$
$ServiceRequesters' = \emptyset$
$ServiceProviders' = \emptyset$
$Mediators' = \emptyset$
$ServiceLevelAgreements' = \emptyset$

*AddOntology*

$\Delta Ontologies$
$O?: Ontology$

---

$O? \notin OntologySet$
$OntologySet = OntologySet' \cup \{O?\}$

*AddClass2Ontology*

$\Delta Ontologies$
$O?: Ontology$
$Class?: CLASS$

---

$O? \in OntologySet$
$Class? \notin O?.Classes$
$O?.Classes = O?.Classes \cup \{Class?\}$

*AddObjectProperty2Ontology*

$\Delta Ontologies$
$O?: Ontology$
$ObjID?: ID$

---

$\exists ObjProp: ObjectProperty \mid O? \in OntologySet \land ObjProp \notin O?.ObjectProperties$
• $ObjProp.Id = ObjID?$
$\land\ ObjProp.Relationship = \emptyset$
$\land\ O?.ObjectProperties = O?.ObjectProperties \cup \{ObjProp\}$

*AddDataTypeProperty2Ontology*

$\Delta Ontologies$
$O?: Ontology$
$DtpID?: ID$

---

$\exists DtProp: DataTypeProperty$
$\mid O? \in OntologySet \land DtProp \notin O?.DataTypeProperties$
• $DtProp.Id = DtpID?$
$\land\ DtProp.Relationship = \emptyset$
$\land\ O?.DataTypeProperties = O?.DataTypeProperties \cup \{DtProp\}$

*AddIndividual2Ontology*

$\Delta Ontologies$
$O?: Ontology$
$Individual?: Individual$

---

$O? \in OntologySet$
$Individual? \notin O? . Individuals$
$O? . Individuals = O? . Individuals \cup \{Individual?\}$

*SetObjectProperties*

$\Delta Ontologies$
$O?: Ontology$
$ObjID?: ID$
$ClassSet?: CLASS \leftrightarrow CLASS$

---

$\exists ObjProp: ObjectProperty$
$\mid O? \in OntologySet$
$\land ObjProp \in O? . ObjectProperties$
$\land \text{dom } ClassSet? \subseteq O? . Classes$
$\land \text{ran } ClassSet? \subseteq O? . Classes$
$\bullet ObjProp . Relationship = ClassSet?$
$\land ObjProp . Id = ObjID?$
$\land O? . ObjectProperties = O? . ObjectProperties \cup \{ObjProp\}$

*SetDataTypeProperties*

$\Delta Ontologies$
$O?: Ontology$
$DtPropID?: ID$
$DtPropSet?: CLASS \pfun DATATYPE$

---

$\exists DtProp: DataTypeProperty$
$\mid O? \in OntologySet$
$\land DtProp \in O? . DataTypeProperties$
$\land \text{dom } DtPropSet? \subseteq O? . Classes$
$\bullet DtProp . Relationship = DtPropSet?$
$\land DtProp . Id = DtPropID?$
$\land O? . DataTypeProperties = O? . DataTypeProperties \cup \{DtProp\}$

*SetClassAxioms*

$\Delta Ontologies$

$O?: Ontology$

$SuperClass?, SubClass?, SiblingClass?, EquivalentClass?,$
$DisjointClass?: CLASS \leftrightarrow CLASS$

---

$\exists ClassAxiomSet: ClassAxioms$
$\mid O? \in OntologySet$
$\land \text{dom } SuperClass? \subseteq O? . Classes$
$\land \text{ran } SuperClass? \subseteq O? . Classes$
$\land \text{dom } SubClass? \subseteq O? . Classes$
$\land \text{ran } SubClass? \subseteq O? . Classes$
$\land \text{dom } SiblingClass? \subseteq O? . Classes$
$\land \text{ran } SiblingClass? \subseteq O? . Classes$
$\land \text{dom } DisjointClass? \subseteq O? . Classes$
$\land \text{ran } DisjointClass? \subseteq O? . Classes$
$\land \text{dom } EquivalentClass? \subseteq O? . Classes$
$\land \text{ran } EquivalentClass? \subseteq O? . Classes$
$\land ClassAxiomSet \in O? . OntologicalClassAxioms$
$\bullet\ ClassAxiomSet . isSuperClassOf = SuperClass?$
$\land ClassAxiomSet . isSubClassOf = SubClass?$
$\land ClassAxiomSet . isSiblingClassOf = SiblingClass?$
$\land ClassAxiomSet . EquivalentClass = EquivalentClass?$
$\land ClassAxiomSet . DisjointWith = DisjointClass?$
$\land O? . OntologicalClassAxioms$
$= O? . OntologicalClassAxioms \cup \{ClassAxiomSet\}$

*SetObjectPropertyAxioms*

$\Delta Ontologies$
$O?: Ontology$
$SubObjectPropertyOf?, SuperObjectPropertyOf?, EquivalentObjectProperty?,$
$InversePropertyOf?: ObjectProperty \leftrightarrow ObjectProperty$
$TransitiveProperty?, SymmetricProperty?: ObjectProperty$

---

$\exists ObjectPropertyAxiomSet: ObjectPropertyAxioms;$
$ObjectPropertySet: \mathbb{P}\ ObjectProperty$
$\mid O? \in OntologySet$
$\land \text{dom}\ SubObjectPropertyOf? \subseteq O?\ .\ ObjectProperties$
$\land \text{ran}\ SubObjectPropertyOf? \subseteq O?\ .\ ObjectProperties$
$\land \text{dom}\ SuperObjectPropertyOf? \subseteq O?\ .\ ObjectProperties$
$\land \text{ran}\ SuperObjectPropertyOf? \subseteq O?\ .\ ObjectProperties$
$\land \text{dom}\ EquivalentObjectProperty? \subseteq O?\ .\ ObjectProperties$
$\land \text{ran}\ EquivalentObjectProperty? \subseteq O?\ .\ ObjectProperties$
$\land \text{dom}\ InversePropertyOf? \subseteq O?\ .\ ObjectProperties$
$\land ObjectPropertyAxiomSet \in O?\ .\ OntologicalObjectPropertyAxioms$
$\land ObjectPropertySet \subseteq O?\ .\ ObjectProperties$
$\bullet\ ObjectPropertyAxiomSet\ .\ SubObjectPropertyOf = SubObjectPropertyOf?$
$\land ObjectPropertyAxiomSet\ .\ SuperObjectPropertyOf = SuperObjectPropertyOf?$
$\land ObjectPropertyAxiomSet\ .\ EquivalentObjectProperty$
$= EquivalentObjectProperty?$
$\land ObjectPropertyAxiomSet\ .\ InversePropertyOf = InversePropertyOf?$
$\land ObjectPropertyAxiomSet\ .\ TransitiveProperty$
$= ObjectPropertySet \cup \{TransitiveProperty?\}$
$\land ObjectPropertyAxiomSet\ .\ SymmetricProperty$
$= ObjectPropertySet \cup \{SymmetricProperty?\}$
$\land O?\ .\ OntologicalObjectPropertyAxioms$
$= O?\ .\ OntologicalObjectPropertyAxioms \cup \{ObjectPropertyAxiomSet\}$

*SetDataTypePropertyAxioms*

$\Delta Ontologies$
$O?: Ontology$
$SubDataTypePropertyOf?, SuperDataTypePropertyOf?,$
$EquivalentDataTypeProperty?: DataTypeProperty \leftrightarrow DataTypeProperty$

---

$\exists DataTypePropertyAxiomSet: DataTypePropertyAxioms$
$\mid O? \in OntologySet$
$\land \operatorname{dom} SubDataTypePropertyOf? \subseteq O? . DataTypeProperties$
$\land \operatorname{ran} SubDataTypePropertyOf? \subseteq O? . DataTypeProperties$
$\land \operatorname{dom} SuperDataTypePropertyOf? \subseteq O? . DataTypeProperties$
$\land \operatorname{ran} SuperDataTypePropertyOf? \subseteq O? . DataTypeProperties$
$\land \operatorname{dom} EquivalentDataTypeProperty? \subseteq O? . DataTypeProperties$
$\land \operatorname{ran} EquivalentDataTypeProperty? \subseteq O? . DataTypeProperties$
$\land DataTypePropertyAxiomSet \in O? . OntologicalDataTypePropertyAxioms$
$\bullet DataTypePropertyAxiomSet . SubDataTypePropertyOf = SubDataTypePropertyOf?$
$\land DataTypePropertyAxiomSet . SuperDataTypePropertyOf$
$= SuperDataTypePropertyOf?$
$\land DataTypePropertyAxiomSet . EquivalentDataTypeProperty$
$= EquivalentDataTypeProperty?$
$\land O? . OntologicalDataTypePropertyAxioms$
$= O? . OntologicalDataTypePropertyAxioms \cup \{DataTypePropertyAxiomSet\}$

---

*SetIndividualAxioms*

$\Delta Ontologies$
$O?: Ontology$
$SameAs?, DifferentFrom?: Individual \leftrightarrow Individual$

---

$\exists IndividualAxiomSet: IndividualAxioms$
$\mid O? \in OntologySet$
$\land \operatorname{dom} SameAs? \subseteq O? . Individuals$
$\land \operatorname{ran} SameAs? \subseteq O? . Individuals$
$\land \operatorname{dom} DifferentFrom? \subseteq O? . Individuals$
$\land \operatorname{ran} DifferentFrom? \subseteq O? . Individuals$
$\land IndividualAxiomSet \in O? . OntologicalIndividualAxioms$
$\bullet IndividualAxiomSet . SameAs = SameAs?$
$\land IndividualAxiomSet . DifferentFrom = DifferentFrom?$
$\land O? . OntologicalIndividualAxioms$
$= O? . OntologicalIndividualAxioms \cup \{IndividualAxiomSet\}$

---

*MappingSet*

$O1?, O2?: Ontology$
$ClassMappings: CLASS \leftrightarrow CLASS$
$ObjectPropertyMappings: ObjectProperty \leftrightarrow ObjectProperty$
$DataTypePropertyMappings: DataTypeProperty \leftrightarrow DataTypeProperty$
$IndividualMappings: Individual \leftrightarrow Individual$

---

$\forall c1, c2: CLASS; objp1, objp2: ObjectProperty; dtp1, dtp2: DataTypeProperty;$
$i1, i2: Individual$
$\bullet\ c1\ ClassMappings\ c2$
$\Leftrightarrow (c1 \in O1?\ .\ Classes \land c2 \in O2?\ .\ Classes$
$\lor c2 \in O1?\ .\ Classes \land c1 \in O2?\ .\ Classes)$
$\land\ objp1\ ObjectPropertyMappings\ objp2$
$\Leftrightarrow (objp1 \in O1?\ .\ ObjectProperties \land objp2 \in O2?\ .\ ObjectProperties$
$\lor objp2 \in O1?\ .\ ObjectProperties \land objp1 \in O2?\ .\ ObjectProperties)$
$\land\ dtp1\ DataTypePropertyMappings\ dtp2$
$\Leftrightarrow (dtp1 \in O1?\ .\ DataTypeProperties \land dtp2 \in O2?\ .\ DataTypeProperties$
$\lor dtp2 \in O1?\ .\ DataTypeProperties \land dtp1 \in O2?\ .\ DataTypeProperties)$
$\land\ i1\ IndividualMappings\ i2$
$\Leftrightarrow i1 \in O1?\ .\ Individuals \land i2 \in O2?\ .\ Individuals$
$\lor i2 \in O1?\ .\ Individuals \land i1 \in O2?\ .\ Individuals$

---

*ClassAlignment*

$\Xi Ontologies$
$\Xi ClassAxioms$
$O1?, O2?: Ontology$
$MapSet!: MappingSet$
$Report!: REPORT$

---

$MapSet!\ .\ ClassMappings$
$= \{\ c1, c2, c3: CLASS; EquivalentClass: CLASS \leftrightarrow CLASS$
$\mid O1? \in OntologySet \land O2? \in OntologySet$
$\land\ c1 \in \text{dom}\ EquivalentClass$
$\land\ c2 \in \text{ran}\ EquivalentClass$
$\land\ c3 \in \text{ran}\ EquivalentClass$
$\land\ \text{dom}\ EquivalentClass \subseteq O1?\ .\ Classes$
$\land\ \text{ran}\ EquivalentClass \subseteq O2?\ .\ Classes$
$\land\ (c2, c3) \in isSuperClassOf$
$\lor\ (c3, c2) \in isSuperClassOf$
$\lor\ (c2, c3) \in isSubClassOf$
$\lor\ (c3, c2) \in isSubClassOf$
$\lor\ (c2, c3) \in EquivalentClass$
$\lor\ (c2, c3) \in isSiblingClassOf \bullet\ (c1, c3)\ \}$
$MapSet!\ .\ ClassMappings = \{\} \Rightarrow Report! = Impasse$
$MapSet!\ .\ ClassMappings \neq \{\} \Rightarrow Report! = Match$

---

*ObjectPropertyAlignment*

$\Xi Ontologies$
$\Xi ObjectPropertyAxioms$
$O1?, O2?: Ontology$
$MapSet!: MappingSet$
$Report!: REPORT$

---

$$
\begin{aligned}
&MapSet!\ .\ ObjectPropertyMappings \\
&\quad = \{\ objp1, objp2, objp3: ObjectProperty; \\
&\qquad EquivalentObjProperty: ObjectProperty \leftrightarrow ObjectProperty \\
&\qquad\quad \mid\ O1? \in OntologySet \land O2? \in OntologySet \\
&\qquad\qquad \land\ objp1 \in \mathrm{dom}\ EquivalentObjProperty \land objp2 \in \mathrm{ran}\ EquivalentObjProperty \\
&\qquad\qquad \land\ objp3 \in \mathrm{ran}\ EquivalentObjProperty \\
&\qquad\qquad \land\ \mathrm{dom}\ EquivalentObjProperty \subseteq O1?\ .\ ObjectProperties \\
&\qquad\qquad \land\ \mathrm{ran}\ EquivalentObjProperty \subseteq O2?\ .\ ObjectProperties \\
&\qquad\qquad \land\ (objp2, objp3) \in EquivalentObjectProperty \\
&\qquad\qquad \lor\ (objp2, objp3) \in InversePropertyOf \lor (objp3, objp2) \in InversePropertyOf \\
&\qquad\qquad \lor\ (objp2, objp3) \in SubObjectPropertyOf\ \lor (objp3, objp2) \in SubObjectPropertyOf \\
&\qquad\qquad \lor\ (objp2, objp3) \in SuperObjectPropertyOf \\
&\qquad\qquad \lor\ (objp3, objp2) \in SuperObjectPropertyOf \bullet\ (objp1, objp3)\ \} \\
&MapSet!\ .\ ClassMappings = \{\} \Rightarrow Report! = Impasse \\
&MapSet!\ .\ ClassMappings \neq \{\} \Rightarrow Report! = Match
\end{aligned}
$$

---

*DataTypePropertyAlignment*

$\Xi Ontologies$
$\Xi DataTypePropertyAxioms$
$O1?, O2?: Ontology$
$MapSet!: MappingSet$
$Report!: REPORT$

---

$$
\begin{aligned}
&MapSet!\ .\ DataTypePropertyMappings \\
&\quad = \{\ dtp1, dtp2, dtp3: DataTypeProperty; \\
&\qquad EquivalentDtProperty: DataTypeProperty \leftrightarrow DataTypeProperty \\
&\qquad\quad \mid\ O1? \in OntologySet \land O2? \in OntologySet \\
&\qquad\qquad \land\ dtp1 \in \mathrm{dom}\ EquivalentDtProperty \land dtp2 \in \mathrm{ran}\ EquivalentDtProperty \\
&\qquad\qquad \land\ dtp3 \in \mathrm{ran}\ EquivalentDtProperty \\
&\qquad\qquad \land\ \mathrm{dom}\ EquivalentDtProperty \subseteq O1?\ .\ DataTypeProperties \\
&\qquad\qquad \land\ \mathrm{ran}\ EquivalentDtProperty \subseteq O2?\ .\ DataTypeProperties \\
&\qquad\qquad \land\ (dtp2, dtp3) \in EquivalentDataTypeProperty \\
&\qquad\qquad \lor\ (dtp2, dtp3) \in SubDataTypePropertyOf \lor (dtp3, dtp2) \in SubDataTypePropertyOf \\
&\qquad\qquad \lor\ (dtp2, dtp3) \in SuperDataTypePropertyOf \\
&\qquad\qquad \lor\ (dtp3, dtp2) \in SuperDataTypePropertyOf \bullet\ (dtp1, dtp3)\ \} \\
&MapSet!\ .\ ClassMappings = \{\} \Rightarrow Report! = Impasse \\
&MapSet!\ .\ ClassMappings \neq \{\} \Rightarrow Report! = Match
\end{aligned}
$$

---

*IndividualAlignment*

$\Xi Ontologies$
$\Xi IndividualAxioms$
$O1?, O2?: Ontology$
$MapSet!: MappingSet$
$Report!: REPORT$

---

$MapSet!\ .\ IndividualMappings$
$= \{\ i1, i2, i3: Individual;\ SameInstance: Individual \leftrightarrow Individual$
$\mid O1? \in OntologySet \land O2? \in OntologySet$
$\land\ i1 \in \text{dom}\ SameInstance$
$\land\ i2 \in \text{ran}\ SameInstance$
$\land\ i3 \in \text{ran}\ SameInstance$
$\land\ \text{dom}\ SameInstance \subseteq O1?\ .\ Individuals$
$\land\ \text{ran}\ SameInstance \subseteq O2?\ .\ Individuals$
$\land\ (i2, i3) \in SameAs \lor (i3, i2) \in SameAs \bullet\ (i1, i3)\ \}$
$MapSet!\ .\ ClassMappings = \{\} \Rightarrow Report! = Impasse$
$MapSet!\ .\ ClassMappings \neq \{\} \Rightarrow Report! = Match$

*ClassMergingInTotalImport*

$\Delta Ontologies$
$O1?, O2?, CoreOntology!: Ontology$

---

$\exists ImportedOntology: Ontology$
$\mid O1? \in OntologySet \land O2? \in OntologySet$
$\land\ ImportedOntology \in OntologySet$
$\land\ CoreOntology! \notin OntologySet$
$\land\ O1?\ .\ Classes = ImportedOntology\ .\ Classes$
$\land\ O2?\ .\ Classes = ImportedOntology\ .\ Classes$
$\bullet\ CoreOntology!\ .\ Classes = ImportedOntology\ .\ Classes$
$\land\ OntologySet' = OntologySet \cup \{CoreOntology!\}$

*ObjectPropertyMergingInTotalImport*

$\Delta Ontologies$
$O1?, O2?, CoreOntology!: Ontology$

---

$\exists ImportedOntology: Ontology$
$\mid O1? \in OntologySet \land O2? \in OntologySet$
$\land\ ImportedOntology \in OntologySet$
$\land\ CoreOntology! \notin OntologySet$
$\land\ O1?\ .\ ObjectProperties = ImportedOntology\ .\ ObjectProperties$
$\land\ O2?\ .\ ObjectProperties = ImportedOntology\ .\ ObjectProperties$
$\bullet\ CoreOntology!\ .\ ObjectProperties = ImportedOntology\ .\ ObjectProperties$
$\land\ OntologySet' = OntologySet \cup \{CoreOntology!\}$

*DataTypePropertyMergingInTotalImport*

$\Delta Ontologies$

$O1?, O2?, CoreOntology!: Ontology$

---

$\exists ImportedOntology: Ontology$

$\mid O1? \in OntologySet \land O2? \in OntologySet$

$\land ImportedOntology \in OntologySet$

$\land CoreOntology! \notin OntologySet$

$\land O1?\ .\ DataTypeProperties = ImportedOntology\ .\ DataTypeProperties$

$\land O2?\ .\ DataTypeProperties = ImportedOntology\ .\ DataTypeProperties$

$\bullet\ CoreOntology!\ .\ DataTypeProperties$

$= ImportedOntology\ .\ DataTypeProperties$

$\land OntologySet' = OntologySet \cup \{CoreOntology!\}$

*IndividualMergingInTotalImport*

$\Delta Ontologies$

$O1?, O2?, CoreOntology!: Ontology$

---

$\exists ImportedOntology, MergedOntology: Ontology$

$\mid O1? \in OntologySet$

$\land O2? \in OntologySet$

$\land ImportedOntology \in OntologySet$

$\land CoreOntology! \notin OntologySet$

$\land O1?\ .\ Individuals = ImportedOntology\ .\ Individuals$

$\land O2?\ .\ Individuals = ImportedOntology\ .\ Individuals$

$\bullet\ CoreOntology!\ .\ Individuals = ImportedOntology\ .\ Individuals$

$\land OntologySet' = OntologySet \cup \{CoreOntology!\}$

*ClassMergingInPluginAndSubsumptionImport*

$\Delta Ontologies$

$O1?, O2?, CoreOntology!: Ontology$

---

$\exists ImportedOntology, MergedOntology: Ontology$

$\mid O1? \in OntologySet$

$\land O2? \in OntologySet$

$\land ImportedOntology \in OntologySet$

$\land CoreOntology! \notin OntologySet$

$\land O1?\ .\ Classes = ImportedOntology\ .\ Classes$

$\land ImportedOntology\ .\ Classes \subseteq O2?\ .\ Classes$

$\Rightarrow CoreOntology!\ .\ Classes = CoreOntology!\ .\ Classes \cup O2?\ .\ Classes$

$\land ImportedOntology\ .\ Classes \subseteq O1?\ .\ Classes$

$\land O2?\ .\ Classes = ImportedOntology\ .\ Classes$

$\Rightarrow CoreOntology!\ .\ Classes = CoreOntology!\ .\ Classes \cup O1?\ .\ Classes$

$\bullet\ OntologySet' = OntologySet \cup \{CoreOntology!\}$

*ObjectPropertyMergingInPluginAndSubsumptionImport*

$\Delta Ontologies$
$O1?, O2?, CoreOntology!: Ontology$

---

$\exists ImportedOntology: Ontology$
$\mid O1? \in OntologySet$
$\land O2? \in OntologySet$
$\land ImportedOntology \in OntologySet$
$\land CoreOntology! \notin OntologySet$
$\land O1?.ObjectProperties = ImportedOntology.ObjectProperties$
$\land ImportedOntology.ObjectProperties \subseteq O2?.ObjectProperties$
$\Rightarrow CoreOntology!.ObjectProperties$
$= CoreOntology!.ObjectProperties \cup O2?.ObjectProperties$
$\land ImportedOntology.ObjectProperties \subseteq O1?.ObjectProperties$
$\land O2?.ObjectProperties = ImportedOntology.ObjectProperties$
$\Rightarrow CoreOntology!.ObjectProperties$
$= CoreOntology!.ObjectProperties \cup O1?.ObjectProperties$
$\bullet OntologySet' = OntologySet \cup \{CoreOntology!\}$

*DataTypePropertyMergingInPluginAndSubsumptionImport*

$\Delta Ontologies$
$O1?, O2?, CoreOntology!: Ontology$

---

$\exists ImportedOntology: Ontology$
$\mid O1? \in OntologySet$
$\land O2? \in OntologySet$
$\land ImportedOntology \in OntologySet$
$\land CoreOntology! \notin OntologySet$
$\land O1?.DataTypeProperties = ImportedOntology.DataTypeProperties$
$\land ImportedOntology.DataTypeProperties \subseteq O2?.DataTypeProperties$
$\Rightarrow CoreOntology!.DataTypeProperties$
$= CoreOntology!.DataTypeProperties \cup O2?.DataTypeProperties$
$\land ImportedOntology.DataTypeProperties \subseteq O1?.DataTypeProperties$
$\land O2?.DataTypeProperties = ImportedOntology.DataTypeProperties$
$\Rightarrow CoreOntology!.DataTypeProperties$
$= CoreOntology!.DataTypeProperties \cup O1?.DataTypeProperties$
$\bullet OntologySet' = OntologySet \cup \{CoreOntology!\}$

*IndividualMergingInPluginAndSubsumptionImport*

$\Delta Ontologies$
$O1?, O2?, CoreOntology!: Ontology$

---

$\exists ImportedOntology: Ontology$
$\mid O1? \in OntologySet \land O2? \in OntologySet$
$\land ImportedOntology \in OntologySet$
$\land CoreOntology! \notin OntologySet$
$\land O1? . Individuals = ImportedOntology . Individuals$
$\land ImportedOntology . Individuals \subseteq O2? . Individuals$
$\Rightarrow CoreOntology! . Individuals$
$= CoreOntology! . Individuals \cup O2? . Individuals$
$\land ImportedOntology . Individuals \subseteq O1? . Individuals$
$\land O2? . Individuals = ImportedOntology . Individuals$
$\Rightarrow CoreOntology! . Individuals$
$= CoreOntology! . Individuals \cup O1? . Individuals$
$\bullet OntologySet' = OntologySet \cup \{CoreOntology!\}$

*ClassMergingInIntersectionImport*

$\Delta Ontologies$
$\Xi ClassAxioms$
$O1?, O2?, CoreOntology!: Ontology$

---

$\exists ClassAxiomSet: ClassAxioms; ImportedOntology: Ontology$
$\mid ClassAxiomSet \in CoreOntology! . OntologicalClassAxioms$
$\land O1? \in OntologySet$
$\land O2? \in OntologySet$
$\land ImportedOntology \in OntologySet$
$\land CoreOntology! \notin OntologySet$
$\land ImportedOntology . Classes \subseteq O1? . Classes$
$\land ImportedOntology . Classes \subseteq O2? . Classes$
$\bullet ClassAxiomSet . isSuperClassOf$
$= \{ c1, c2, c3: CLASS$
$\mid c1 \in ImportedOntology . Classes$
$\land c2 \in O1? . Classes$
$\land c3 \in O2? . Classes$
$\land (c1 EquivalentClass c2 \land c1 isSuperClassOf c3)$
$\lor c1 isSuperClassOf c2 \land c1 isSubClassOf c3$
$\lor c1 isSubClassOf c2 \land c1 EquivalentClass c3$
$\lor c1 isSubClassOf c2 \land c1 isSiblingClassOf c3 \bullet (c2, c3) \}$
$\land ClassAxiomSet . isSubClassOf$
$= \{ c1, c2, c3: CLASS$
$\mid c1 \in ImportedOntology . Classes$
$\land c2 \in O1? . Classes$
$\land c3 \in O2? . Classes$
$\land (c1 EquivalentClass c2 \land c1 isSubClassOf c3)$
$\lor c1 isSuperClassOf c2 \land c1 EquivalentClass c3$

$$
\begin{array}{l}
\quad \vee\ c1\ isSubClassOf\ c2 \wedge c1\ isSuperClassOf\ c3 \\
\quad \vee\ c1\ isSiblingClassOf\ c2 \wedge c1\ isSubClassOf\ c3 \bullet\ (c2, c3)\ \} \\
\wedge\ ClassAxiomSet\ .\ isSiblingClassOf \\
\quad = \{\ c1, c2, c3: CLASS \\
\qquad |\ c1 \in ImportedOntology\ .\ Classes \\
\qquad \wedge\ c2 \in O1?\ .\ Classes \\
\qquad \wedge\ c3 \in O2?\ .\ Classes \\
\qquad \wedge\ (c1\ EquivalentClass\ c2 \wedge c1\ isSiblingClassOf\ c3) \\
\qquad \vee\ c1\ isSuperClassOf\ c2 \wedge c1\ isSuperClassOf\ c3 \\
\qquad \vee\ c1\ isSuperClassOf\ c2 \wedge c1\ isSiblingClassOf\ c3 \\
\qquad \vee\ c1\ isSiblingClassOf\ c2 \wedge c1\ EquivalentClass\ c3 \\
\qquad \vee\ c1\ isSiblingClassOf\ c2 \wedge c1\ isSuperClassOf\ c3 \\
\qquad \vee\ c1\ isSiblingClassOf\ c2 \wedge c1\ isSiblingClassOf\ c3 \\
\qquad \bullet\ (c2, c3)\ \} \\
\wedge\ ClassAxiomSet\ .\ EquivalentClass \\
\quad = \{\ c1, c2, c3: CLASS \\
\qquad |\ c1 \in ImportedOntology\ .\ Classes \\
\qquad \wedge\ c2 \in O1?\ .\ Classes \\
\qquad \wedge\ c3 \in O2?\ .\ Classes \\
\qquad \wedge\ (c1\ EquivalentClass\ c2 \wedge c1\ EquivalentClass\ c3) \\
\qquad \bullet\ (c2, c3)\ \} \\
\wedge\ ClassAxiomSet\ .\ DisjointWith \\
\quad = \{\ c1, c2, c3: CLASS \\
\qquad |\ c1 \in ImportedOntology\ .\ Classes \\
\qquad \wedge\ c2 \in O1?\ .\ Classes \\
\qquad \wedge\ c3 \in O2?\ .\ Classes \\
\qquad \wedge\ (c1\ DisjointWith\ c2 \vee c1\ DisjointWith\ c3) \\
\qquad \vee\ c1\ isSubClassOf\ c2 \wedge c1\ isSubClassOf\ c3 \bullet\ (c2, c3)\ \} \\
\wedge\ CoreOntology!\ .\ OntologicalClassAxioms \\
\quad = CoreOntology!\ .\ OntologicalClassAxioms \cup \{ClassAxiomSet\} \\
\wedge\ OntologySet' = OntologySet \cup \{CoreOntology!\}
\end{array}
$$

*ObjectPropertyMergingInIntersectionImport*

$\Delta$*Ontologies*

$\Xi$*ObjectPropertyAxioms*

*O1?, O2?, CoreOntology!: Ontology*

---

$\exists$*ObjectPropertyAxiomSet: ObjectPropertyAxioms; ImportedOntology: Ontology*

| *ObjectPropertyAxiomSet* $\in$ *CoreOntology! . OntologicalObjectPropertyAxioms*

$\wedge$ *O1?* $\in$ *OntologySet*

$\wedge$ *O2?* $\in$ *OntologySet*

$\wedge$ *ImportedOntology* $\in$ *OntologySet*

$\wedge$ *CoreOntology!* $\notin$ *OntologySet*

$\wedge$ *ImportedOntology . ObjectProperties* $\subseteq$ *O1? . ObjectProperties*

$\wedge$ *ImportedOntology . ObjectProperties* $\subseteq$ *O2? . ObjectProperties*

• *ObjectPropertyAxiomSet . SuperObjectPropertyOf*

= { *p1, p2, p3: ObjectProperty*

| *p1* $\in$ *ImportedOntology . ObjectProperties*

$\wedge$ *p2* $\in$ *O1? . ObjectProperties*

$\wedge$ *p3* $\in$ *O2? . ObjectProperties*

$\wedge$ (*p1 SubObjectPropertyOf p2* $\wedge$ *p1 SubObjectPropertyOf p3*)

$\vee$ *p1 SubObjectPropertyOf p2* $\wedge$ *p1 InversePropertyOf p3*

$\vee$ *p1 SubObjectPropertyOf p2* $\wedge$ *p1 EquivalentObjectProperty p3*

$\vee$ *p1 SuperObjectPropertyOf p2* $\wedge$ *p1 InversePropertyOf p3*

$\vee$ *p1 SuperObjectPropertyOf p2* $\wedge$ *p1 EquivalentObjectProperty p3*

$\vee$ *p1 InversePropertyOf p2* $\wedge$ *p1 SuperObjectPropertyOf p3*

$\vee$ *p1 EquivalentObjectProperty p2* $\wedge$ *p1 SuperObjectPropertyOf p3*

• (*p2, p3*) }

$\wedge$ *ObjectPropertyAxiomSet . SubObjectPropertyOf*

= { *p1, p2, p3: ObjectProperty*

| *p1* $\in$ *ImportedOntology . ObjectProperties*

$\wedge$ *p2* $\in$ *O1? . ObjectProperties*

$\wedge$ *p3* $\in$ *O2? . ObjectProperties*

$\wedge$ (*p1 SuperObjectPropertyOf p2* $\wedge$ *p1 SubObjectPropertyOf p3*)

$\vee$ *p1 InversePropertyOf p2* $\wedge$ *p1 SubObjectPropertyOf p3*

$\vee$ *p1 EquivalentObjectProperty p2* $\wedge$ *p1 SubObjectPropertyOf p3*

• (*p2, p3*) }

$\wedge$ *ObjectPropertyAxiomSet . EquivalentObjectProperty*

= { *p1, p2, p3: ObjectProperty*

| *p1* $\in$ *ImportedOntology . ObjectProperties*

$\wedge$ *p2* $\in$ *O1? . ObjectProperties*

$\wedge$ *p3* $\in$ *O2? . ObjectProperties*

$\wedge$ (*p1 EquivalentObjectProperty p2*

$\wedge$ *p1 EquivalentObjectProperty p3*) • (*p2, p3*) }

$\wedge$ *ObjectPropertyAxiomSet . InversePropertyOf*

= { *p1, p2, p3: ObjectProperty*

| *p1* $\in$ *ImportedOntology . ObjectProperties*

$\wedge$ *p2* $\in$ *O1? . ObjectProperties*

$\wedge$ *p3* $\in$ *O2? . ObjectProperties*

$\wedge$ (*p1 InversePropertyOf p2* $\wedge$ *p1 InversePropertyOf p3*)
$\vee$ *p1 InversePropertyOf p2* $\wedge$ *p1 EquivalentObjectProperty p3*
$\vee$ *p1 EquivalentObjectProperty p2* $\wedge$ *p1 InversePropertyOf p3*
• (*p2, p3*) }
$\wedge$ *CoreOntology! . OntologicalObjectPropertyAxioms*
= *CoreOntology! . OntologicalObjectPropertyAxioms*
$\cup$ {*ObjectPropertyAxiomSet*}
$\wedge$ *OntologySet'* = *OntologySet* $\cup$ {*CoreOntology!*}

---

*DataTypePropertyMergingInIntersectionImport*

$\Delta$*Ontologies*
$\Xi$*DataTypePropertyAxioms*
*O1?, O2?, CoreOntology!: Ontology*

---

$\exists$*DataTypePropertyAxiomSet: DataTypePropertyAxioms; ImportedOntology: Ontology*
| *DataTypePropertyAxiomSet*
$\in$ *CoreOntology! . OntologicalDataTypePropertyAxioms*
$\wedge$ *O1?* $\in$ *OntologySet*
$\wedge$ *O2?* $\in$ *OntologySet*
$\wedge$ *ImportedOntology* $\in$ *OntologySet*
$\wedge$ *CoreOntology!* $\notin$ *OntologySet*
$\wedge$ *ImportedOntology . DataTypeProperties* $\subseteq$ *O1? . DataTypeProperties*
$\wedge$ *ImportedOntology . DataTypeProperties* $\subseteq$ *O2? . DataTypeProperties*
• *DataTypePropertyAxiomSet . SuperDataTypePropertyOf*
= { *p1, p2, p3: DataTypeProperty*
| *p1* $\in$ *ImportedOntology . DataTypeProperties*
$\wedge$ *p2* $\in$ *O1? . DataTypeProperties*
$\wedge$ *p3* $\in$ *O2? . DataTypeProperties*
$\wedge$ (*p1 SubDataTypePropertyOf p2* $\wedge$ *p1 SuperDataTypePropertyOf p3*)
$\vee$ *p1 SubDataTypePropertyOf p2*
$\wedge$ *p1 EquivalentDataTypeProperty p3*
$\vee$ *p1 SuperDataTypePropertyOf p2*
$\wedge$ *p1 EquivalentDataTypeProperty p3*
$\vee$ *p1 EquivalentDataTypeProperty p2*
$\wedge$ *p1 SuperDataTypePropertyOf p3* • (*p2, p3*) }
$\wedge$ *DataTypePropertyAxiomSet . SubDataTypePropertyOf*
= { *p1, p2, p3: DataTypeProperty*
| *p1* $\in$ *ImportedOntology . DataTypeProperties*
$\wedge$ *p2* $\in$ *O1? . DataTypeProperties*
$\wedge$ *p3* $\in$ *O2? . DataTypeProperties*
$\wedge$ (*p1 SuperDataTypePropertyOf p2*
$\wedge$ *p1 SuperDataTypePropertyOf p3*)
$\vee$ *p1 EquivalentDataTypeProperty p2*
$\wedge$ *p1 SubDataTypePropertyOf p3* • (*p2, p3*) }
$\wedge$ *DataTypePropertyAxiomSet . EquivalentDataTypeProperty*
= { *p1, p2, p3: DataTypeProperty*
| *p1* $\in$ *ImportedOntology . DataTypeProperties*

$\wedge\ p2 \in O1?\ .\ DataTypeProperties$
$\wedge\ p3 \in O2?\ .\ DataTypeProperties$
$\wedge\ (p1\ EquivalentDataTypeProperty\ p2$
$\wedge\ p1\ EquivalentDataTypeProperty\ p3) \bullet (p2, p3)\ \}$
$\wedge\ CoreOntology!\ .\ OntologicalDataTypePropertyAxioms$
$= CoreOntology!\ .\ OntologicalDataTypePropertyAxioms$
$\cup \{DataTypePropertyAxiomSet\}$
$\wedge\ OntologySet' = OntologySet \cup \{CoreOntology!\}$

---

*IndividualMergingInIntersectionImport*

$\Delta Ontologies$
$\Xi IndividualAxioms$
$O1?, O2?, CoreOntology!: Ontology$

---

$\exists IndividualAxiomSet: IndividualAxioms; ImportedOntology: Ontology$
$\mid\ IndividualAxiomSet \in CoreOntology!\ .\ OntologicalIndividualAxioms$
$\wedge\ O1? \in OntologySet$
$\wedge\ O2? \in OntologySet$
$\wedge\ ImportedOntology \in OntologySet$
$\wedge\ CoreOntology! \notin OntologySet$
$\wedge\ ImportedOntology\ .\ Individuals \subseteq O1?\ .\ Individuals$
$\wedge\ ImportedOntology\ .\ Individuals \subseteq O2?\ .\ Individuals$
$\bullet\ IndividualAxiomSet\ .\ SameAs$
$= \{\ i1, i2, i3: Individual$
$\mid\ i1 \in ImportedOntology\ .\ Individuals$
$\wedge\ i2 \in O1?\ .\ Individuals$
$\wedge\ i3 \in O2?\ .\ Individuals$
$\wedge\ (i1\ SameAs\ i2 \wedge i1\ SameAs\ i3) \bullet (i2, i3)\ \}$
$\wedge\ IndividualAxiomSet\ .\ DifferentFrom$
$= \{\ i1, i2, i3: Individual$
$\mid\ i1 \in ImportedOntology\ .\ Individuals$
$\wedge\ i2 \in O1?\ .\ Individuals$
$\wedge\ i3 \in O2?\ .\ Individuals$
$\wedge\ (i1\ SameAs\ i2 \wedge i1\ DifferentFrom\ i3)$
$\vee\ i1\ DifferentFrom\ i2 \wedge i1\ DifferentFrom\ i3 \bullet (i2, i3)\ \}$
$\wedge\ CoreOntology!\ .\ OntologicalIndividualAxioms$
$= CoreOntology!\ .\ OntologicalIndividualAxioms \cup \{IndividualAxiomSet\}$
$\wedge\ OntologySet' = OntologySet \cup \{CoreOntology!\}$

---

*ClassIntegration*

$\Delta Ontologies$

$O1?, O2?, CoreOntology!: Ontology$

---

$O1? \in OntologySet$

$O2? \in OntologySet$

$CoreOntology! \notin OntologySet$

$CoreOntology!.Classes = O1?.Classes \cup O2?.Classes$

$OntologySet' = OntologySet \cup \{CoreOntology!\}$

*ObjectPropertyIntegration*

$\Delta Ontologies$

$O1?, O2?, CoreOntology!: Ontology$

---

$O1? \in OntologySet$

$O2? \in OntologySet$

$CoreOntology! \notin OntologySet$

$CoreOntology!.ObjectProperties = O1?.ObjectProperties \cup O2?.ObjectProperties$

$OntologySet' = OntologySet \cup \{CoreOntology!\}$

*DataTypePropertyIntegration*

$\Delta Ontologies$

$O1?, O2?, CoreOntology!: Ontology$

---

$O1? \in OntologySet$

$O2? \in OntologySet$

$CoreOntology! \notin OntologySet$

$CoreOntology!.DataTypeProperties = O1?.DataTypeProperties \cup O2?.DataTypeProperties$

$OntologySet' = OntologySet \cup \{CoreOntology!\}$

*IndividualIntegration*

$\Delta Ontologies$

$O1?, O2?, CoreOntology!: Ontology$

---

$O1? \in OntologySet$

$O2? \in OntologySet$

$CoreOntology! \notin OntologySet$

$CoreOntology!.Individuals = O1?.Individuals \cup O2?.Individuals$

$OntologySet' = OntologySet \cup \{CoreOntology!\}$

*SyntacticalServiceDiscovery*

$\Xi PERSONÆ$
*Goals?: FunctionalRequirement*
*ServiceName?: DATATYPE*
*ServiceDiscoveredSet!: ID* $\pfun$ *ServiceDescription*
*Response!: REPORT*

---

*ServiceDiscoveredSet!*
$= \{$ *Registry: ServiceRegistry; id: ID; SD: ServiceDescription*
$\mid$ *Registry* $\in$ *Registries*
$\land$ $(id, SD) \in$ *Registry . Services*
$\land$ *Goals? = SD . Capability*
$\lor$ *ServiceName? = SD . ServiceName* $\bullet$ $(id, SD)$ $\}$
*ServiceDiscoveredSet!* $= \{\} \Rightarrow$ *Response! = NoServiceDiscovered*
*ServiceDiscoveredSet!* $\neq \{\} \Rightarrow$ *Response! = ServiceDiscovered*

*GoalOrientedServiceDiscovery*

$\Xi PERSONÆ$
*Goals?: FunctionalRequirement*
*ServiceSet!: ID* $\pfun$ *ServiceDescription*
*O1!, O2!: Ontology*

---

$\exists$ *GoalClasses, CapabilityClasses:* $\mathbb{P}$ *CLASS*
$\mid$ *GoalClasses* $= \{$ *fr: FunctionalRequirement* $\mid$ *fr = Goals?* $\bullet$ *fr . Class* $\}$
$\land$ *CapabilityClasses*
$= \{$ *Registry: ServiceRegistry; id: ID; SD: ServiceDescription;*
*fr: FunctionalRequirement*
$\mid$ *Registry* $\in$ *Registries*
$\land$ $(id, SD) \in$ *Registry . Services*
$\land$ *fr = SD . Capability* $\bullet$ *fr . Class* $\}$
$\land$ *O1!* $\in$ *OntologySet*
$\land$ *O2!* $\in$ *OntologySet*
$\bullet$ *O1! . Classes = GoalClasses* $\land$ *O2! . Classes = CapabilityClasses*

*InputOrientedServiceDiscovery*

$\Xi PERSONÆ$
$Input?: DataTypeProperty$
$ServicesSet!: ID \pfun ServiceDescription$
$O1!, O2!: Ontology$

---

$\exists DtPropertySet: \mathbb{P}\ DataTypeProperty$
$\mid DtPropertySet$
$= \{\ Registry: ServiceRegistry; id: ID; SD: ServiceDescription;$
$input: DataTypeProperty$
$\mid Registry \in Registries$
$\land (id, SD) \in Registry\ .\ Services$
$\land input \in SD\ .\ Input \bullet input\ \}$
$\land O1! \in OntologySet$
$\land O2! \in OntologySet$
$\bullet O1!\ .\ DataTypeProperties = O1!\ .\ DataTypeProperties \cup \{Input?\}$
$\land O2!\ .\ DataTypeProperties = DtPropertySet$

*OutputOrientedServiceDiscovery*

$\Xi PERSONÆ$
$Output?: DataTypeProperty$
$ServicesSet!: ID \pfun ServiceDescription$
$O1!, O2!: Ontology$

---

$\exists DtPropertySet: \mathbb{P}\ DataTypeProperty$
$\mid DtPropertySet$
$= \{\ Registry: ServiceRegistry; id: ID; SD: ServiceDescription;$
$output: DataTypeProperty$
$\mid Registry \in Registries$
$\land (id, SD) \in Registry\ .\ Services$
$\land output \in SD\ .\ Output \bullet output\ \}$
$\land O1! \in OntologySet$
$\land O2! \in OntologySet$
$\bullet O1!\ .\ DataTypeProperties = O1!\ .\ DataTypeProperties \cup \{Output?\}$
$\land O2!\ .\ DataTypeProperties = DtPropertySet$

*SyntacticalServiceSelection*

$\Xi PERSONÆ$
$Goals?: FunctionalRequirement$
$QoSDesired?: NonFunctionalRequirement \pfun LEVEL$
$ServiceSelectedSet!: \mathbb{P}\ ID$
$Response!: REPORT$

---

$\exists ServiceSet: ID \pfun ServiceDescription$
$\mid ServiceSet$
$= \{\ Registry: ServiceRegistry; id: ID; SD: ServiceDescription$
$\mid Registry \in Registries$
$\land (id, SD) \in Registry\ .\ Services$
$\land Goals? = SD\ .\ Capability \bullet (id, SD)\ \}$
$\land ServiceSelectedSet!$
$= \{\ id: ID; SD: ServiceDescription$
$\mid (id, SD) \in ServiceSet \land QoSDesired? = SD\ .\ QoSOffered \bullet id\ \}$
$\bullet\ ServiceSelectedSet! = \{\}$
$\Rightarrow Response! = NoServiceSelected \land ServiceSelectedSet! \neq \{\}$
$\Rightarrow Response! = ServiceSelected$

*QoSConstraintsBasedServiceSelection*

$\Xi PERSONÆ$
$QoSConstraints?: NonFunctionalRequirement \pfun LEVEL$
$ServiceSet!: ID \pfun ServiceDescription$
$O1!: Ontology$

---

$\exists NFRConstrainedClasses, NFROfferedClasses: \mathbb{P}\ CLASS$
$\mid NFRConstrainedClasses$
$= \{\ nfr: NonFunctionalRequirement \mid nfr \in \mathrm{dom}\ QoSConstraints?$
$\bullet\ nfr\ .\ Class\ \}$
$\land NFROfferedClasses$
$= \{\ Registry: ServiceRegistry; id: ID; SD: ServiceDescription;$
$nfr: NonFunctionalRequirement$
$\mid Registry \in Registries$
$\land (id, SD) \in Registry\ .\ Services$
$\land nfr \in \mathrm{dom}\ SD\ .\ QoSOffered \bullet nfr\ .\ Class\ \}$
$\land O1! \in OntologySet$
$\land NFRConstrainedClasses \subseteq NFROfferedClasses$
$\bullet\ O1!\ .\ Classes = NFROfferedClasses \setminus NFRConstrainedClasses$

*QoSPreferencesBasedServiceSelection*

$\Xi PERSONÆ$
$QoSPreferences?: NonFunctionalRequirement \pfun LEVEL$
$ServiceSet!: ID \pfun ServiceDescription$
$O1!: Ontology$

---

$\exists NFRPreferredClasses, NFROfferedClasses: \mathbb{P}\ CLASS$
$\mid NFRPreferredClasses$
$= \{\ nfr: NonFunctionalRequirement \mid nfr \in \mathrm{dom}\ QoSPreferences?$
$\bullet\ nfr\ .\ Class\ \}$
$\land\ NFROfferedClasses$
$= \{\ Registry: ServiceRegistry;\ id: ID;\ SD: ServiceDescription;$
$nfr: NonFunctionalRequirement$
$\mid Registry \in Registries$
$\land\ (id, SD) \in Registry\ .\ Services$
$\land\ nfr \in \mathrm{dom}\ SD\ .\ QoSOffered \bullet nfr\ .\ Class\ \}$
$\land\ O1! \in OntologySet$
$\land\ NFRPreferredClasses \subseteq NFROfferedClasses$
$\bullet\ O1!\ .\ Classes = NFRPreferredClasses$

*QoSPrioritiesBasedServiceSelection*

$\Xi PERSONÆ$
$QoSPriorities?: NonFunctionalRequirement \times NonFunctionalRequirement$
$\pfun NonFunctionalRequirement$
$ServiceSet!: ID \pfun ServiceDescription$
$O1!, O2!: Ontology$

---

$\exists NFRPriorityClasses, NFROfferedClasses: \mathbb{P}\ CLASS$
$\mid NFRPriorityClasses$
$= \{\ nfr: NonFunctionalRequirement \mid nfr \in \mathrm{ran}\ QoSPriorities?$
$\bullet\ nfr\ .\ Class\ \}$
$\land\ NFROfferedClasses$
$= \{\ Registry: ServiceRegistry;\ id: ID;\ SD: ServiceDescription;$
$nfr: NonFunctionalRequirement$
$\mid Registry \in Registries$
$\land\ (id, SD) \in Registry\ .\ Services$
$\land\ nfr \in \mathrm{dom}\ SD\ .\ QoSOffered \bullet nfr\ .\ Class\ \}$
$\land\ O1! \in OntologySet$
$\land\ O2! \in OntologySet$
$\land\ NFRPriorityClasses \subseteq NFROfferedClasses$
$\bullet\ O1!\ .\ Classes = NFRPriorityClasses \land O2!\ .\ Classes = NFROfferedClasses$

*QoSAlignment*

$\Delta PERSONÆ$
*Requester?: ServiceRequester*
*Provider?: ServiceProvider*
*O1!, O2!: Ontology*

---

$\exists s: Service; nfr1, nfr2: NonFunctionalRequirement$
| $Requester? \in ServiceRequesters$
$\wedge\ Provider? \in ServiceProviders$
$\wedge\ s \in Provider?\ .\ Services$
$\wedge\ nfr1 \in \text{ran}\ Requester?\ .\ QoSPriorities$
$\wedge\ nfr2 \in \text{dom}\ s\ .\ Description\ .\ QoSOffered$
$\wedge\ O1! \in OntologySet$
$\wedge\ O2! \in OntologySet$
• $O1!\ .\ Classes = O1!\ .\ Classes \cup \{nfr1\ .\ Class\}$
$\wedge\ O2!\ .\ Classes = O2!\ .\ Classes \cup \{nfr2\ .\ Class\}$

*ProfileIntegration*

$\Delta PERSONÆ$
*Requester?: ServiceRequester*
*Provider?: ServiceProvider*
*O1!, O2!: Ontology*
$QoSAgreed!: NonFunctionalRequirement \rightarrow\!\!\!\!\!\!\rightarrow LEVEL$
*Contract!: ServiceLevelAgreement*
*MapSet?: MappingSet*

---

$\exists nfr1, nfr2, nfr3: NonFunctionalRequirement; c1, c2: CLASS$
| $MapSet?\ .\ ClassMappings \neq \emptyset$
$\wedge\ nfr3 \in \text{dom}\ QoSAgreed!$
$\wedge\ (c1, c2) \in MapSet?\ .\ ClassMappings$
$\wedge\ Contract! \notin ServiceLevelAgreements$
• $nfr3\ .\ Class = c1 \vee nfr3\ .\ Class = c2$
$\wedge\ Contract!\ .\ QoSAgreed = QoSAgreed!$
$\wedge\ O1! = Requester?\ .\ Profile$
$\wedge\ O2! = Provider?\ .\ Profile$
$\wedge\ ServiceLevelAgreements' = ServiceLevelAgreements \cup \{Contract!\}$

*ContractIntegration*

$\Delta PERSONÆ$
*Contract!: ServiceLevelAgreement*
*CoreOntology?: Ontology*

---

$Contract!\ .\ SLAOntology = Contract!\ .\ SLAOntology \cup \{CoreOntology?\}$
$ServiceLevelAgreements' = ServiceLevelAgreements \cup \{Contract!\}$

$CreateNewOntology \cong AddOntology \vee (AddClass2Ontology \vee AddObjectProperty2Ontology \vee AddDataTypeProperty2Ontology \vee AddIndividual2Ontology \vee SetClassAxioms \vee SetObjectPropertyAxioms \vee SetDataTypePropertyAxioms \vee SetIndividualAxioms)$

$OntologyAlignment \cong ClassAlignment \fatsemi ObjectPropertyAlignment \fatsemi DataTypePropertyAlignment \fatsemi IndividualAlignment$

$MergingInTotalImport \cong ClassMergingInTotalImport \fatsemi ObjectPropertyMergingInTotalImport \fatsemi DataTypePropertyMergingInTotalImport \fatsemi IndividualMergingInTotalImport$

$MergingInPluginAndSubsumptionImport \cong ClassMergingInPluginAndSubsumptionImport \fatsemi ObjectPropertyMergingInPluginAndSubsumptionImport \fatsemi DataTypePropertyMergingInPluginAndSubsumptionImport \fatsemi IndividualMergingInPluginAndSubsumptionImport$

$MergingInIntersectionImport \cong ClassMergingInIntersectionImport \fatsemi ObjectPropertyMergingInIntersectionImport \fatsemi DataTypePropertyMergingInIntersectionImport \fatsemi IndividualMergingInIntersectionImport$

$OntologyMerging \cong MergingInTotalImport \fatsemi MergingInPluginAndSubsumptionImport \fatsemi MergingInIntersectionImport$

$OntologyIntegration \cong ClassIntegration \fatsemi ObjectPropertyIntegration \fatsemi DataTypePropertyIntegration \fatsemi IndividualIntegration$

$ServiceDiscovery \cong SyntacticalServiceDiscovery \vee (GoalOrientedServiceDiscovery \vee InputOrientedServiceDiscovery \vee OutputOrientedServiceDiscovery) \gg OntologyMerging$

$SemanticServiceSelection \cong QoSPreferencesBasedServiceSelection \fatsemi QoSConstraintsBasedServiceSelection \fatsemi QoSPrioritiesBasedServiceSelection$

$ServiceSelection \cong SyntacticalServiceSelection \vee (SemanticServiceSelection \gg OntologyAlignment)$

$ContractEstablishment \cong (QoSAlignment \gg OntologyAlignment) \fatsemi (ProfileIntegration \gg OntologyIntegration) \fatsemi ContractIntegration$

$PERSONÆApplication \cong ServiceDiscovery \fatsemi ServiceSelection \fatsemi ContractEstablishment$