UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE AUTOMAÇÃO E SISTEMAS

Alexandre Perin de Souza

UM MODELO DE DESCOBERTA DINÂMICA DE SERVIÇOS DE SOFTWARE BASEADO NO CONTEXTO DE PROCESSOS DE NEGÓCIOS E EM QUALIDADE DE SERVIÇO

Tese submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina para a obtenção do Grau de Doutor em Engenharia de Automação e Sistemas.
Orientador: Prof. Dr. Ricardo José Rabelo.

Florianópolis

2011

Catalogação na fonte elaborada pela biblioteca
da
Universidade Federal de Santa Catarina

S729m Souza, Alexandre Perin de
Um modelo de descoberta dinâmica de serviços de software baseado no contexto de processos de negócios e em qualidade de serviço [tese] / Alexandre Perin de Souza ; orientador, Ricardo José Rabelo. - Florianópolis, SC, 2011.
330 p.: il., grafs., tabs.

Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Inclui referências

1. Engenharia de sistemas. 2. Negócios – Processamento de dados. 3. Arquitetura orientada a serviços. 4. Controle de qualidade. 5. Software. I. Rabelo, Ricardo José. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. III. Título.

CDU 621.3-231.2(021)

Alexandre Perin de Souza

Um Modelo de Descoberta Dinâmica de Serviços de Software Baseado no Contexto de Processos de Negócios e em Qualidade de Serviço

Esta Tese foi julgada adequada para obtenção do Título de "Doutor em Engenharia de Automação e Sistemas" e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas

Florianópolis, 10 de outubro de 2011.

_____
Prof. José Eduardo Ribeiro Cury, Dr.
Coordenador do Curso

Banca Examinadora:

_____ Prof. Ricardo J. Rabelo, Dr. Orientador Universidade Federal de Santa Catarina	_____ Profª. Flávia M. Santoro, Dra. Relatora Universidade Federal do Estado do Rio de Janeiro
_____ Prof. Nelson S. Rosa, Dr. Membro Universidade Federal de Pernambuco	_____ Prof. Frank A. Siqueira, Dr. Membro Universidade Federal de Santa Catarina
_____ Prof. Carlos B. Montez, Dr. Membro Universidade Federal de Santa Catarina	_____ Prof. João Bosco M. Sobral, Dr. Membro Universidade Federal de Santa Catarina

Dedico esta tese a minha família e a Deus.

AGRADECIMENTOS

Muito obrigado...

aos colegas Valdeci J. Costa e Wilson C. Branco Neto, pelo envio das cartas de recomendação ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

aos colegas de turma Rafael J. Deitos, Breno C. Pinheiro e Mathias Erdtmann.

aos mestres do Departamento de Automação e Sistemas da UFSC, pela competência.

aos colegas do GSigma, principalmente ao Roque O. Bezerra, pela decisiva e competente ajuda, ao Rui Tramontin pelo auxílio durante a qualificação, à Maiara H. Cancian, pelas conversas e discussões sobre temas que diziam respeito aos nossos trabalhos, ao Saulo Zambiasi, pelo suporte no ambiente do GSigma, ao Marcus V. Drissen da Silva, pela boa conversa e ao Fabiano Baldo, pelo companheirismo.

aos colegas da UNIPLAC que sempre compreenderam os vários nãos que fui obrigado a dizer.

ao apoio que sempre recebi do Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

ao CNPq, pelo apoio financeiro.

aos membros da banca de qualificação, professores Joni da Silva Fraga, Frank Siqueira, Carlos Montez, Ricardo J. Rabelo e Flávia M. Santoro, pela capacidade de análise, idoneidade e críticas.

ao meu orientador, Prof^{o}. Ricardo José Rabelo, pela oportunidade, pelos conselhos, pelo comprometimento e pela sempre importante participação.

à Manuela e à Karen, meus amores, pela paciência nos momentos difíceis.

RESUMO

Esta tese aborda o problema da integração do nível de negócios (BPM) com nível dos sistemas (SOA) através da concepção de um modelo de descoberta dinâmica de serviços. Estudos mostram que há diversas formas de realizar essa integração. No entanto, o que se observa é a falta de uma teoria geral que unifique e resolva o problema da descoberta de forma alinhada ao nível dos processos de negócios. Para contornar isso, vários trabalhos foram propostos considerando-se contextos específicos, usando-se restrições e adotando pressupostos que os tornam pouco aplicáveis em cenários reais. Esta pesquisa mostrou que é possível que os níveis BPM e SOA trabalhem de forma mais integrada e que o processo de descoberta seja mais ágil e transparente. Foi desenvolvido um modelo de descoberta, integrado, aberto, flexível, que busca e seleciona o serviço de software mais adequado, considerando a funcionalidade do serviço desejado, o contexto dos processos de negócios, os aspectos de QoS associados ao serviço desejado, e o momento em que a aplicação é executada. O modelo considera um cenário onde diversas empresas provedoras de serviços de software, autônomas e heterogêneas, disponibilizam seus serviços (na forma de *web services*) em repositórios largamente distribuídos. Para mitigar os problemas de interoperabilidade nos vários níveis envolvidos, o modelo faz intenso uso de padrões abertos e consolidados, além de utilizar um catálogo de processos de negócios e duas ontologias: uma de processos de negócios (baseados no padrão UBL) e uma de QoS. A abordagem frente ao problema de descoberta divide-o em duas fases: de projeto e de execução de aplicações. O modelo faz uso de um *crawling* para encontrar e selecionar serviços candidatos (na fase de projeto) e utiliza um algoritmo de descoberta dinâmica para encontrar e selecionar os serviços (na fase de execução) para a execução de aplicações SOA. O modelo faz a descoberta considerando o estilo arquitetural e de disponibilização SaaS, incluindo a geração dinâmica de SLAs. Uma entidade lógica chamada *Federação* é usada para gerenciar provedores, ontologias e os repositórios de serviços. Os resultados obtidos permitem inferir que o modelo tem potencial de servir também como um moderno ambiente de competitividade e sustentabilidade para empresas desenvolvedoras de software como serviço SaaS.

Palavras-chave: BPM. SOA. Descoberta de serviços. QoS. UBL. SaaS.

ABSTRACT

This thesis addresses the problem of integrating the business level (BPM) systems level (SOA) through the conception of a model of dynamic discovery of services. Studies show that there are several ways to achieve this integration. However, what is observed is the lack of a general theory which unifies and solve the problem of service discovery aligned with the level of business process. To address this, several studies have been proposed considering specific contexts, using constraints and adopting assumptions that make it less applicable in real scenarios. This research showed that it is possible levels BPM and SOA work in a more integrated way and that the discovery process is more agile and transparent. In this sense, a discovery model has been conceived. It is integrated, open, flexible, finding and selecting the most appropriate software service, considering the required services' functionalities, business processes' contexts, the required QoS levels, and the moment the SOA application will run. The developed model considers a scenario where several autonomous and heterogeneous companies provide software services (as *web services*) and made them available in largely distributed repositories. Aiming at mitigating interoperability problems in the various involved levels, the model strongly relies on open and consolidated standards, besides using a business process catalog and two ontologies: of business processes (based on UBL standard) and of QoS. The approach proposed in the model splits the discovery problem into two phases: design and running applications. It applies a crawling algorithm to find out and to select service candidates (in the design phase) and a dynamic discovery algorithm for finding and selecting the service in the execution of SOA applications. The model performs the discovery taking the SaaS architectural and availability style into account, including the dynamic generation of SLAs. A logical entity called *Federation* is used to manage providers, ontologies and services repositories. Research´s results allow inferring that the developed model has the potential to also serve as a modern environment for levering the competitiveness and sustainability of companies that develop software under SaaS model.

Keywords: BPM. SOA. Web Service Discovery. QoS. UBL. SaaS.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE GRÁFICOS

LISTA DE ABREVIATURAS E SIGLAS

API	-	*Application Programming Interface*
APS	-	*Application Service Provider*
B2B	-	*Business to Business*
B2C	-	*Business to Consumer*
BPM	-	*Business Process Management*
BPMS	-	*Business Process Management Suite*
BPMN	-	*Business Process Modeling Notation*
CRM	-	*Customer Relationship Management*
CNPq	-	Centro Nacional de Desenvolvimento Científico e Tecnológico
CSP	-	*Constraint Programming*
DAML	-	*DARPA Agent Markup Language*
DAMLS	-	*DARPA Agent Markup Language for Web Services*
DAS	-	Departamento de Automação e Sistemas
DDS	-	Descoberta Dinâmica de Serviços
DTD	-	*Document Type Definition*
EDI	-	*Electronic Data Interchange*
ebXML	-	*Electronic Business using eXtensible Markup Language*
ECOLEAD	-	*European Collaborative Networked Organization Leaddership Iniciative*
ER	-	Entidade Relacionamento
ERP	-	*Enterprise Resource Planning*
GSigma	-	a alta, espaçamento simples (todo o trabalho)
HSQLDB	-	*HyperSQL Database*
HTTP	-	*Hypertext Transfer Protocol*
IBM	-	*International Business Machines*
IDE	-	*Integrated Development Environment*
ICWS	-	*International Conference on Web Services*
IFM	-	Instituto Fábrica do Milênio
IJWSR	-	*International Journal of Web Services Research*
KPI	-	*Key Performance Index*
LF	-	Linguagem Formal
LN	-	Linguagem Natural
OASIS	-	*Organization for the Advancement of Structured Information Standards*
ODE	-	*Orchestration Director Engine*
OMG	-	*Object Management Group*
OWL	-	*Web Ontology Language*

OWL-S	-	*Semantic Markup for Web Services*
P2P	-	*Peer to Peer*
PIP	-	*Partner Interface Processes*
PME	-	Pequenas e Médias Empresas
PPGEAS	-	Programa de Pós-Graduação em Engenharia de Automação e Sistemas
QoS	-	*Quality of Service*
RI	-	Recuperação da Informação
RMI	-	*Remote Method Invocation*
RPC	-	*Remote Procedure Call*
SaaS	-	*Software as a Service*
SAWDSL	-	*Semantic Annotating for WSDL and XML Schema*
SCA	-	*Service Component Architecture*
SCDL	-	*Service Composition Description Language*
SCM	-	*Supply Chain Management*
SGML	-	*Standard Generalized Markup Language*
SLA	-	*Service Level Agreement*
SOA	-	*Service Oriented Architecture*
SWT	-	*Standard Widget Toolkit*
TCP/IP	-	*Transmission Control Protocol e Internet Protocol*
TI	-	Tecnologias de Informação
TIC	-	Tecnologias de Informação e Comunicação
TLP	-	*Top Level Project*
UBL	-	*Universal Business Language*
UDDI	-	*Universal Description, Discovery and Integration*
UFSC	-	Universidade Federal de Santa Catarina
UP	-	*Unified Process*
UML	-	*Unified Modeling Language*
UNIPLAC	-	Universidade do Planalto Catarinense
URI	-	*Uniform Resource Identifier*
URL	-	*Uniform Resource Location*
URN	-	*Uniform Resource Name*
XML	-	*Extensible Markup Language*
XTRO	-	*Extended Registries Ontology*
W3C	-	*World Wide Web Consortium*
WBPM	-	*Websphere Business Modeler*
WS	-	*Web Service*
WSA	-	*Web Service Architecture*
WS-BPEL	-	*Web Services Business Process Execution Language*
WS-CDL	-	*Web Service Choreography Description Language*

WSDL	-	*Web Service Description Language*
WSFA	-	*Web Services Agent Framework*
WS-I	-	*The Web Services Interoperability Organization*
WSLA	-	*Web Service Level Agreement Language*
WSMF	-	*Web Service Modeling Framework*
WSMO	-	*Web Service Modeling Ontology*
WSMX	-	*Web Service Modelling eXecution Environment*

SUMÁRIO

Capítulo 1

Introdução

1.1 **Apresentação**

A convergência da base tecnológica, aliada à dinâmica da indústria e o crescimento da Internet, são aspectos inter-relacionados, responsáveis por várias transformações na vida das pessoas e das organizações. O primeiro aspecto decorre do fato de poder representar qualquer tipo de informação de uma única forma, a digital. O segundo se preocupa em criar máquinas mais poderosas, com mais recursos, além de torná-las mais acessíveis. E o terceiro diz respeito ao crescimento da Internet, a qual se tornou um fenômeno singular ao ponto de ser considerado como fator estratégico para desenvolvimento de uma nação (TAKAHASHI, 2000).

Estes fatores criam um fenômeno global, com elevado potencial transformador das atividades sociais, financeiras e de negócios, como por exemplo, a competição internacional acirrada, o aumento crescente dos custos operacionais das empresas, os ciclos de vida dos produtos cada vez menores, o constante desafio frente às mudanças dos requisitos de negócio, entre outros (TAKAHASHI, 2000).

Em virtude desta nova dinâmica imposta, um grande esforço vem sendo despendido pelas empresas para integrar negócios e Tecnologia da Informação (TI), uma vez que tanto negócios quanto TI tradicionalmente trabalham fracamente conectados (NEUBAUER, 2009).

Devido a isto, várias iniciativas surgiram nos últimos anos, dentre elas o *Business Process Management* (BPM) e *Service Oriented Architecture* (SOA). Assim, BPM vem se destacando em virtude de representar o conjunto das melhores experiências obtidas em gerenciamento de processos de negócios (GARIMELLA; LEES; WILLIAMS, 2008). Fundamentalmente, BPM proporciona uma visão organizacional, na qual o processo tem prioridade em relação à estruturação departamental clássica encontrada nas empresas. O objetivo desta mudança é privilegiar cada atividade que faz parte do processo. O que se espera é criar um maior valor agregado aos processos como um todo (GOLDKUHL; LIND, 2008).

Já SOA surge como um paradigma para desenvolvimento de sistemas distribuídos, onde os componentes essenciais são serviços (pequenos módulos de software), os quais podem ser executados em computadores diferentes a partir de provedores de serviços distintos (SOMMERVILLE, 2007) e (JOSUTTIS, 2008).

Recentemente, a combinação de BPM&SOA tem sido defendida como a melhor estratégia para que empresas obtenham um alinhamento mais próximo entre processos de negócios e recursos tecnológicos e, com isto, consigam projetar aplicações que melhor respondam a mudanças dos requisitos dos negócios (BPMInstitute.org, 2006). A combinação de BPM&SOA permite às empresas automatizar e otimizar processos de negócios através de componentes reusáveis (serviços *web*), integrando complexos e heterogêneos sistemas (KAMOUN, 2007).

Nesta junção, BPM é uma disciplina que possibilita modelar um negócio em termos de atividades que perpassam e envolvem vários sistemas dentro de uma empresa, podendo até mesmo abranger várias empresas. Estas atividades formam os processos de negócios, os quais são representados em um formato que possa ser entendido e processado por máquinas. No que diz respeito a SOA, serviços de software implementam funções presentes nos processos de negócios, de tal forma que diferentes aplicações podem compartilhar um mesmo serviço devido à alta interoperabilidade e ao baixo acoplamento advindos da fundação tecnológica padronizada dos serviços *web* (KAMOUN, 2007).

A prática das soluções de software que permitem a integração BPM&SOA mostra que analistas de negócios costumam especificar processos no nível de negócios, enquanto engenheiros de sistemas, no nível tecnológico, realizam a vinculação desta especificação com serviços próprios ou de terceiros, criando uma aplicação SOA. Após a

concepção da aplicação, a mesma é executada em alguma ferramenta computacional.

Além disso, analistas normalmente criam suas especificações sem considerar padrões de processos, tais como as UBL (*Universal Business Language*) (OASIS, 2006) ou RosettaNet (ROSETTANET, 2010), os quais têm um potencial para minimizar problemas de interoperabilidade em aplicações SOA. Nesse caso, ocorre que informações sobre o processo em si não são repassadas à camada SOA. Isto cria uma maior dificuldade quando o propósito está relacionado à descoberta de serviços, pois o contexto dos processos é perdido (INAGANTI; BEHARA, 2007). Neste trabalho, o contexto dos processos é usado para definir os processos de negócios (OMG, 2011) em termos do seu fluxo de execução e do conjunto de atividades que o compõem (OMG, 2011), (KHURANA; MANDKE, 2009) e (WANG; LU, 2010). Por outro lado, considerar o contexto dos processos de negócios permite que a descoberta seja automatizada e realizada com menores intervenções de quem projeta a aplicação, pois o contexto é capturado e refletido no processo de descoberta de serviços.

A descoberta pode admitir duas formas: estática ou dinâmica. Na forma estática ou tradicional, serviços escolhidos são vinculados às aplicações. Assim, independentemente de quando a aplicação irá ser executada, o vínculo permanece perene, ou seja, para o serviço em questão, deve ser garantido que seu endereço não muda e que ele esteja disponível e acessível para que a aplicação possa ser executada com sucesso. Na descoberta dinâmica, serviços são selecionados e imediatamente vinculados às aplicações. Neste tipo, um serviço pode ser substituído por outro, a qualquer tempo, de acordo com demandas da aplicação. Isto é uma vantagem significativa, uma vez que esta capacidade melhor responde aos requisitos de um mercado que sofre modificações em tempos menores, cada vez mais profundas e exige maior produtividade das empresas a cada instante.

Considerando a integração BPM&SOA tradicional, uma dificuldade enfrentada pelas empresas é o gerenciamento e a manutenção de aspectos de qualidade relacionados aos serviços (QoS). Esta dificuldade é mais intensa quando um processo de negócio possui restrições e estas devem ser traduzidas em limites respeitados pelos serviços presentes na camada tecnológica. Isto impacta sobremaneira na integração BPM&SOA, já que atributos de qualidade são dispositivos que potencializam a descoberta do serviço que melhor atenda às

necessidades da aplicação (PAPAZOGLOU *et al.*, 2007) e (CHAARI *et al.*, 2008).

Esta forma de integração e de atuação das empresas caracteriza uma menor flexibilidade e agilidade em relação a mudanças em seus processos de negócios, já que cada modificação exige nova programação e vinculação de serviços, consumindo recursos e tempo das empresas. Ademais, há que se considerar, neste cenário de integração, a forte tendência do crescimento de fornecedores de software ofertando serviços, bem como a necessidade de comercialização destes via algum modelo de negócios, tal como proposto no SaaS (GREER JR, 2009) e em (CANCIAN, 2009).

Estas barreiras formam um panorama que dificulta as organizações escolherem o serviço mais adequado, seja em função das funcionalidades requeridas, seja dos aspectos de qualidade associados aos serviços desejados ou do contexto no qual o processo de negócio esteja inserido.

Somado aos obstáculos presentes nas soluções BPM&SOA, há que se considerar a limitação das tecnologias existentes usadas na descoberta de serviços: i) a mais recente especificação do *Universal Description, Discovery and Integration* (UDDI) (OASIS, 2004) permite apenas formas simples de pesquisa, a partir de um conjunto de funções presentes e disponíveis na *Application Programming Interface* (API) dos repositórios de serviços; ii) a pesquisa em repositórios é sintática, ou seja, baseada em palavras-chave e em categorias dos serviços. Esta forma não permite que serviços *web*, contendo as mesmas funções, escritas de maneira diferente, sejam facilmente encontrados; iii) o processo não é sensível às informações de Qualidade de Serviço, fato que dificulta a seleção de serviços em função das necessidades de uma aplicação (ZANG; ZHOU, 2002).

Assim, nesta tese, o foco está em explorar o processo de descoberta de serviços *web* no âmbito da integração BPM&SOA, de forma dinâmica, através da concepção de um modelo[1] de descoberta de serviços de software baseado no contexto de processos de negócios e em qualidade de serviço. Além disso, considera-se a forte tendência dos serviços estarem sendo ofertados em diversos repositórios distribuídos e

[1] A noção de modelo adotada neste trabalho refere-se à representação de algo (processo ou entidade), seja para mostrar sua aparência, funcionamento ou construção (IEEE, 1990), (PRESSMAN, 2001) e (SOMMERVILLE, 2007).

considera-se que estes serviços podem ser disponibilizados por diversos provedores.

1.2 **Problema de Pesquisa**

A união do nível de processo ao nível tecnológico é um fator crítico para sobrevivência das organizações (SENTANIN; SANTOS; JABBOUR, 2008). Esta integração possibilita aproveitar sistemas legados[2], através do acesso às funcionalidades dos mesmos via serviços *web*. A consequência direta disso é tornar aplicações flexíveis ao ponto de estarem preparadas para enfrentar mudanças exigidas pelo mercado, uma vez que é possível, por exemplo, substituir um serviço por outro sem a necessidade de alterar a aplicação no nível de negócios.

No atual estado de desenvolvimento desta integração, BPM e SOA são consideradas as melhores soluções para que este processo ocorra. Estudos apontam diversas formas de integrar estas duas iniciativas (WOODLEY; GAGNON, 2005), (BEHARA, 2006) e (ADAM; DOERR, 2008). Entretanto, a dificuldade inerente a cada uma destas técnicas é a falta de uma teoria geral que unifique e resolva o problema da descoberta (INAGANTI; BEHARA, 2007), pois as abordagens de pesquisa são bastante fragmentadas (PAPAZOGLOU *et al.*, 2007).

Para suprimir esta necessidade, diversos mecanismos foram criados considerando contextos específicos e aplicando-se restrições. Embora muitos destes mecanismos tenham atingido sucesso na tarefa de descobrir serviços, percebeu-se na revisão do estado da arte que eles apresentam várias limitações quando o problema está em descobrir o serviço mais adequado, aquele que melhor atenda às necessidades de uma aplicação, pois deixam de considerar: as funcionalidades do serviço, o contexto dos processos de negócios, os aspectos de qualidade associados aos serviços (QoS), o momento em que aplicação é executada e o cenário altamente distribuído, no qual provedores disponibilizam serviços e estes são ofertados em repositórios espalhados em diversos locais, fato que dificulta uma mais ágil, transparente e eficiente integração BPM&SOA.

---

[2] Sistemas computacionais normalmente complexos, antigos, de difícil manutenção, que fornecem funções essenciais para uma organização.

O que se deseja responder neste trabalho de pesquisa se resume à pergunta: *é possível criar um modelo que integre processos a sistemas através da descoberta dinâmica de serviços?E que essa descoberta leve em conta os aspectos funcionais e não-funcionais dos serviços, o contexto dos processos de negócios e a oferta de serviços por diversos provedores largamente distribuídos?*

Na pergunta de pesquisa, a expressão "descobrir de forma dinâmica o serviço *web*" quer dizer:

- *Descobrir*, diz respeito ao processo de buscar e selecionar, onde:
  - A busca se refere a serviços de software, os quais estarão ofertados em repositórios de serviços (UDDIs) heterogêneos (independente das tecnologias e estruturas de dados usadas para criar serviços) e largamente distribuídos (em um cenário pervasivo[3] de provedores) e independentes entre si;
  - E, seleção, significa escolher o serviço de software mais adequado em função dos critérios de QoS associados e do contexto da aplicação em questão;
  - Serviços *web* diz respeito aos serviços de software que estarão implementados na tecnologia *web services*, considerado o padrão atual *de facto* (W3C, 2004);
- A palavra *dinâmica* se refere ao fato de que a localização física do serviço não é conhecida *a priori*, ou seja, pode mudar (PIRES, 2008).

Em termos do problema a ser resolvido, o mesmo pertence à linha de pesquisa designada por descoberta de serviços *web* (*Web Services Discovery*). Já em termos de abordagem geral adotada nesta proposta, ele se enquadra na linha de pesquisa designada por Recuperação de Informação (*Information Retrieval*).

---

[3] Embora a palavra *pervasivo* não exista no dicionário da língua portuguesa, ela será usada neste trabalho como tradução do termo em inglês "pervasive", indicando que o computador está embarcado em ambientes de forma invisível para o usuário. Nesta concepção, o computador tem a capacidade de obter informação do ambiente no qual ele está embarcado e utiliza estas informações para dinamicamente construir controlar, configurar e ajustar a aplicação para melhor atender as necessidades do dispositivo ou usuário (ARAUJO, 2003).

1.3 **Justificativa**

Business Process Management e *Service-Oriented Architecture* são duas iniciativas diferentes. BPM é uma camada tecnológica de alto nível que vai além de uma aplicação individual (BPMInstitute.org, 2006). Esta camada inclui métodos, técnicas e ferramentas para o projeto, execução, gerenciamento e análise de processos de negócios (AALST; HOFSTEDE; MATHIAS, 2003).

O uso de uma solução BPM permite aos analistas de negócio investigarem processos sob várias perspectivas, tais como: tarefas intra-organizacionais a serem executadas, interações existentes entre entidades participantes de um processo de negócio através da troca de informações, entre outras. O resultado desta análise permite a uma dada empresa alinhar melhor sistemas e tecnologia da informação ao seu negócio e, com isso, melhorar sua competitividade.

Já SOA é definido como uma forma arquitetural para sistemas distribuídos (W3C, 2004) e serviços *web* são uma das existentes implementações tecnológicas para SOA.

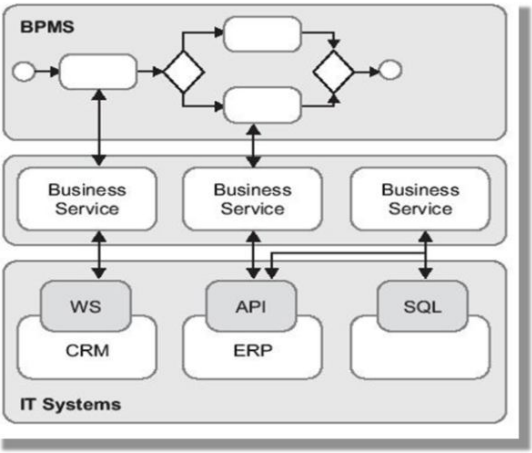


Figura 1 - Integração BPM&SOA, no qual SOA funciona como elo entre sistemas legados e processos de negócios.
Fonte: (ORACLE, 2008).

Quando as duas iniciativas são aplicadas juntas, BPM e SOA tornam-se sinérgicas. Ambas fornecem uma estratégia para tratar os vários desafios impostos pelas mudanças constantes dos requisitos

relacionados às aplicações que implementam processos de negócios. Esta cooperação permite a redução de custos, o aumento da eficiência e uma grande flexibilidade no desenvolvimento ou manutenção de aplicações (KAMOUN, 2007).

No cenário no qual BPM e SOA trabalham juntos é possível especificar processos de negócios (parte superior da Figura 1) e usar serviços para implementá-los (parte central da Figura 1), permitindo que sistemas heterogêneos e legados participem da composição de aplicações (parte inferior da Figura 1, denominada de *IT Systems*).

Assim, devido ao encapsulamento dos serviços, a comunicação entre o ambiente BPM e um serviço é feita a partir de uma interface que contém informações sobre ele. Em função da flexibilidade e do baixo acoplamento proporcionados por SOA, é possível substituir um serviço dinamicamente, sem prejuízo ao processo de negócio como um todo. Desta forma, é possível usar sistemas de informações já construídos para implementar processos de negócios, pois a fundação tecnológica padronizada dos serviços permite esta abstração.

Assim, além da necessidade de integrar BPM&SOA, surge também a necessidade da descoberta do serviço mais adequado. Somado a isto, deve-se considerar também o cenário no qual provedores de serviços (empresas de software) vem como oportunidade de negócio a disponibilização de seus produtos à base de serviços de software (CANCIAN, 2009).

Desta forma, para que o processo de descoberta de serviços resulte em resultados mais próximos à situação que se deseja, percebe-se a necessidade de um modelo que integre os níveis de processos e de tecnologia, considerando aspectos funcionais (o que um serviço deve fazer) e não-funcionais (o conjunto desejado de restrições de QoS associadas aos serviços) e o cenário largamente distribuído composto por provedores que ofertam e disponibilizam serviços.

1.4 **Hipótese de Pesquisa**

Com base na pergunta de pesquisa, a hipótese de pesquisa é apresentada no Quadro 1. Considerando-se o método hipotético-dedutivo a ser aplicado nesta tese, a hipótese está formulada de forma afirmativa e é do tipo plausível. Isso significa que se buscará comprová-la ao longo do seu desenvolvimento e do seu resultado, observando-se o experimento e suas consequências com base na atual verdade e nos fundamentos teóricos e científicos da área de descoberta de serviços.

Problema	É possível criar um modelo que integre processos a sistemas através da descoberta dinâmica de serviços?E que essa descoberta leve em conta os aspectos funcionais e não-funcionais dos serviços, o contexto dos processos de negócios e a oferta de serviços por diversos provedores largamente distribuídos?
Hipótese	A concepção de um modelo que una o nível de processos de negócios da empresa e o nível no qual os serviços de software são globalmente disponibilizados, guiado pelas desejadas métricas de QoS e pelo contexto do processo, tem o potencial de garantir que o serviço *web* mais adequado para executar o processo de negócio em andamento seja descoberto.

Quadro 1 - Hipótese de Pesquisa.

1.5 **Objetivos**

1.5.1 Geral

Esta tese tem como objetivo desenvolver um modelo integrado, flexível e aberto de descoberta dinâmica de serviços *web* é capaz de buscar e selecionar o serviço mais adequado consoante o contexto do processo de negócio em questão e os requisitos de qualidade vigentes.

Esta descoberta deve seguir critérios funcionais e não-funcionais em um cenário pervasivo composto por provedores ofertando e disponibilizando serviços de software.

1.5.2 Específicos

Serão objetivos específicos desta Tese:

- Levantar e registrar o referencial teórico em função dos elementos de estudo e registrar o estado da arte relativo à descoberta de serviços *web*;
- Integrar o catálogo de processos de negócios UBL proposto em Bezerra (2011) ao ambiente BPM de concepção de aplicações SOA. Esta integração permite que o projetista de aplicações use processos prontos e consolidados para compor suas aplicações;
- Implementar um *crawling* para serviços. O *crawling* é usado na fase de concepção de aplicações, que munido da expressão

de descoberta de serviços monta uma lista de serviços candidatos a implementarem determinada atividade. Além do *crawling*, um algoritmo de descoberta dinâmica foi implementado, o qual usa informações colhidas pelo *crawling*, tendo como objetivo definir (1) um serviço a ser utilizado quando aplicações SOA forem executadas;

- Organizar um ente lógico denominado Federação (RABELO, 2008) a ser consultado em relação à descoberta de serviços. A Federação é fonte de publicação e pesquisa de serviços e conta com um conjunto de ontologias. Uma ontologia QoS para especificação de QoS e outra ontologia (chamada UBL) usada para definir o contexto dos processos de negócios a serem descobertos e publicados;
- Criar um mecanismo de exportação do processo para um formato padrão de execução, de forma a permitir que esses processos ou aplicações possam ser executados em um ambiente de execução de aplicações;
- Construir um ambiente de execução de aplicações, de forma a permitir que aplicações prontas possam ser executadas e acompanhadas; e
- Projetar um protótipo do modelo proposto que contemple todos os elementos conceituais a serem definidos e instanciados na forma de software.

1.6 **Originalidade Pretendida**

Tomando como base a motivação e relevância do problema de descoberta de serviços, o estado da arte e da prática na área, bem como as tendências gerais identificadas sobre as mudanças nas empresas em termos de processos e TICs (Tecnologias da Informação e Comunicação), observa-se que praticamente todos os elementos necessários para um ambiente de descoberta dinâmica já foram de alguma forma tratados na literatura. Todavia, sempre explorando tais elementos isoladamente. Por exemplo, há inúmeros trabalhos que tratam exclusivamente de semântica na descoberta, alguns usam QoS e outros atuam no processo de descoberta em tempo de execução. Na pesquisa efetuada não se encontrou um modelo de descoberta dinâmica que compreendesse todos esses elementos de forma integrada e simultânea. Adicionalmente, que usasse um catálogo de processos de negócios e utilizasse padrões reconhecidos e abertos em todas as etapas da

descoberta. Portanto, trata-se de um modelo arquiteturalmente inédito. Do ponto de vista do estado-da-arte em relação à descoberta em si, a principal contribuição científica está na consideração da camada de negócios (BPM) no ambiente de descoberta, fazendo com que o mecanismo de descoberta desenvolvido (um *crawling*) tenha a capacidade de buscar com mais precisão os serviços de software (*web services*) mais adequados para o contexto do processo de negócio em questão, de acordo com os requisitos de QoS desejados, num cenário distribuído de repositórios de serviços. Finalmente, embora não tenha sido o foco principal desta tese e assim não se tenha validado isso, deve-se ressaltar outra perspectiva de ineditismo, que é uma visão que prepara o ambiente de descoberta para uma composição mais ágil de aplicações SOA inserido em um modelo de negócios SaaS[4].

1.7 Adequação às Linhas de Pesquisa do Programa de Pós-Graduação

O Programa de Pós-Graduação em Engenharia de Automação e Sistemas desenvolve atividades de formação e pesquisa ligadas às linhas de pesquisa de Controle, Automação e Sistemas Mecatrônicos e Sistemas Computacionais. Em função do tema, da problemática de pesquisa e dos objetivos propostos neste trabalho, este possui forte aderência à linha de pesquisa denominada de Sistemas Computacionais.

1.8 Elementos de Pesquisa

Definido o problema e as principais características do que se deseja desenvolver, é necessário precisar o que estudar. Isso permite conhecer melhor cada aspecto do tema a ser abordado, além de definir um contorno mais nítido em relação ao objeto de estudo (GONSALVES, 2005).

Desta forma, os assuntos ou elementos de pesquisa usados para o desenvolvimento da revisão bibliográfica emergiram em função: i) de um levantamento bibliográfico inicial e ii) da análise sobre a literatura

[4] Esta maior agilidade pode ser verificada na dissertação de mestrado de Bezerra (2011), que desenvolveu um dos componentes da arquitetura geral do ambiente de descoberta construído nesta tese.

disponível sobre o tema. Eles permitiram realizar um recorte no campo de atuação da pesquisa e a definição do ineditismo a ser trabalhado.

Os assuntos que compõem o domínio de atuação desta pesquisa são:

- *Business Process Management* (BPM);
- Catálogos e Padrões de Processos de Negócios;
- Integração BPM&SOA;
- Arquitetura Orientada a Serviços e serviços *web*;
- Descoberta de serviços *web*;
- Modelo SaaS.

Como fonte de realização desta pesquisa, um conjunto de bases de dados virtuais espalhadas pela Internet foi consultado. O Quadro 2 sintetiza a lista das principais fontes de pesquisa e informação utilizadas na revisão bibliográfica e do estado da arte deste trabalho.

Nome	Endereço (URL)
Biblioteca de Teses e Dissertações do Instituto Brasileiro de Informação em Ciência e Tecnologia	http://www.ibict.br/
Biblioteca Digital de Domínio Público	http://www.dominiopublico.gov.br
Biblioteca Digital Brasileira de Computação	http://www.lbd.dcc.ufmg.br/bdbcomp/bdbcomp.jsp
Biblioteca Digital da PUC-RIO	http://www.maxwell.lambda.ele.puc-rio.br
IEEE *Digital Library*	http://ieeexplore.ieee.org/
ACM *Digital Library*	http://portal.acm.org/dl.cfm
Portal de Periódicos da CAPES	http://www.periodicos.capes.gov.br/portugues/index.jsp
CiteSeer	http://citeseerx.ist.psu.edu/

Quadro 2 – Principais Fontes de Pesquisa e Informação a serem usadas nesta pesquisa.

Além dessas, o Quadro 3 apresenta uma lista de entidades responsáveis pelo gerenciamento de padrões. Documentos disponíveis

nos *sites* destas entidades também foram consultados para o desenvolvimento da revisão bibliográfica.

Entidade	Sigla	Breve descrição
Organization for the Advancement of Structured Information Standards	OASIS	Orienta o desenvolvimento, convergência e a adoção de padrões abertos em âmbito global.
World Wide Web Consortium	W3C	Consórcio Internacional composto de representantes de organizações, cujo objetivo maior é desenvolver padrões com vistas a obter o máximo da *Web*.
Networked European Software and Services Initiative	NESSI	É um esforço europeu no sentido de criar uma plataforma tecnológica genérica que impulsione o aumento dos negócios europeus em âmbito mundial. Sua principal inspiração está no uso Tecnologias de Informação e de Comunicação e no desenvolvimento de aplicações SaaS (*software* como serviços).
Business Process Management Initiative	BPMI	Iniciativa para crescimento e melhoria da área de BPM. Em 2005, BPMI e OMG uniram esforços para incremento de especificações sobre BPM.
RosettaNet	RosettaNet	É uma organização global, cujo objetivo é promover padrões relacionados ao comércio colaborativo.

Quadro 3 - Lista de Entidades usadas como Fonte de Pesquisa e Informação.

1.9 **Enquadramento Metodológico Científico**

A realização de uma pesquisa científica pressupõe rigor no conjunto de atividades encampadas pelo pesquisador para atingir seu objetivo. Estas atividades devem apresentar reflexões conceituais sólidas, alicerçadas em conhecimentos já existentes, além de resultados relevantes e possíveis de serem verificados por outros pesquisadores (GONSALVES, 2005).

O propósito do enquadramento metodológico realizado nesta seção está em conhecer as características e métodos necessários para

condução da pesquisa. Desta forma, a pesquisa científica está enquadrada nos seguintes aspectos:

- Quanto à Natureza da Pesquisa
- Quanto aos Objetivos da Pesquisa
- Quanto à Abordagem de Pesquisa
- Quanto ao Método de Pesquisa
- Quanto ao Paradigma de Pesquisa
- Quanto aos Procedimentos de Pesquisa
- Quanto ao Tipo de Observação na Pesquisa
- Quanto ao Tempo da Pesquisa

Em relação à Natureza da Pesquisa, este trabalho toma a forma de uma pesquisa aplicada, pois são usadas tecnologias de base, tal como linguagens de programação, infra-estrutura de comunicação (Internet), aliada a um conjunto de especificações e recomendações fornecidas por várias iniciativas ligadas ao desenvolvimento de padrões, tais como: W3C, OMG, OASIS entre outras, e alguns modelos e paradigmas, tais como BPM, SOA e SaaS para o desenvolvimento do modelo.

Quanto aos seus Objetivos, trata-se de uma pesquisa exploratória, pois visa proporcionar maior familiaridade com o problema com vistas a torná-lo explícito. Nesse sentido, os objetivos específicos constituem instâncias que permitem explorar e tornar a pesquisa compreensível e precisa. Mais concretamente, ela visa contribuir ao conhecimento científico na forma da geração de conhecimentos adicionais sobre o problema da descoberta de serviços quando esta for guiada pelo contexto e dependente de QoS.

Em relação à Abordagem de Pesquisa, trata-se de uma pesquisa qualitativa, cujo objetivo é a geração de um modelo. Neste tipo de pesquisa, considera-se que há uma relação dinâmica entre o mundo real e o sujeito (SILVA; MENEZES, 2005). Isto é, um vínculo indissociável entre o mundo objeto e a subjetividade do sujeito que não pode ser traduzido em números. Embora seja um trabalho essencialmente qualitativo, alguns indicadores quantitativos deverão ser aplicados para fins de análise global dos resultados. Porém, aspectos quantitativos (por exemplo, tempo de execução de serviços) não serão foco deste trabalho. Também não serão foco de atenção aspectos relacionados à segurança e a tolerância a faltas.

Quanto ao Método de Pesquisa, este trabalho caracteriza-se como sendo uma pesquisa do tipo hipotético-dedutiva. Este método se aplica essencialmente quando não há um conjunto de conhecimentos

completos sobre algo ou fenômeno; assim cria-se a necessidade de conjecturar (na forma de hipóteses), com algum empirismo, visando-se gerar um novo conceito ou teoria. Nesta categoria, a diretriz básica se concentra em estabelecer uma hipótese de pesquisa e, interativamente, busca-se atingir a solução pretendida.

Em relação ao Paradigma de Pesquisa, este trabalho classifica-se como sendo positivista, pois a realidade – o contexto do problema – deve ser tangível. Nesse tipo de paradigma, busca-se o conjunto de causas que afetam o processo de descoberta de serviços, por exemplo: falta de QoS, não considerar semântica, incapacidade das tecnologias associadas aos serviços *web* em representar os vários aspectos presentes nos serviços, como expressar a busca por serviços, entre outras. Esse processo, por exemplo, deve identificar quais e quantos atributos de QoS devem ser considerados na descoberta, quais e quantas iniciativas existem para representar semântica nos serviços e quais estratégias podem ser usadas para resolver ou minimizar as consequências da falta de recursos das tecnologias ligadas aos serviços, para representar os vários aspectos considerados no processo de descoberta de serviços.

Do ponto de vista dos procedimentos técnicos, a pesquisa se caracteriza como pesquisa bibliográfica e experimental. É uma pesquisa que será elaborada em função de materiais bibliográficos já publicados. Serão realizados experimentos envolvendo as várias dimensões presentes no processo de descoberta de serviços *web*, inicialmente em repositórios de serviços pouco povoados e centralizados. Incrementalmente, serão adicionadas as demais dimensões e, por fim, os experimentos evoluirão para um ambiente distribuído, com um número maior de provedores.

Esta pesquisa ainda se caracteriza como sendo sistemática e factual na dimensão Tipo de Observação da Pesquisa. Isso decorre do fato de que a coleta de dados será realizada em um ambiente com condições controladas. Os experimentos serão desenvolvidos, conduzidos e analisados ao nível de protótipos computacionais, com dados (por exemplo: serviços e QoS) previamente introduzidos – e portanto conhecidos *a priori*. Ao longo dos experimentos, foram observadas as respostas do protótipo quanto ao problema da descoberta, com ênfase voltada principalmente à qualidade da descoberta, ou seja, se o mais adequado serviço desejado/especificado, considerando o contexto e aspectos de QoS, foi selecionado corretamente.

Por fim, em relação à dimensão Tempo de Pesquisa, este trabalho tem característica de estudo longitudinal. Sendo assim, especificações

desejadas para serviços, bem como serviços resultantes do processo de descoberta, serão registrados em vários momentos para fins de realização de ensaios e análises com vista à aproximação ou confirmação da hipótese de pesquisa.

Em termos de execução do trabalho, ela seu deu parcialmente nas instalações do Grupo de Sistemas Inteligentes de Manufatura (GSigma), no Departamento de Automação e Sistemas da Universidade Federal de Santa Catarina (DAS/UFSC), e nas dependências da Universidade do Planalto Catarinense (UNIPLAC), no Departamento de Ciências Exatas e Tecnológicas, local de trabalho deste doutorando. Em ambas as instalações existem condições adequadas em termos de software, hardware, rede e material bibliográfico de base.

1.10 **Metodologia de Execução do Trabalho**

A realização desta tese foi planejada para ser realizada em cinco macro etapas, sendo as seguintes:

1. Avaliação do Estado da Arte e Revisão Bibliográfica;
2. Concepção do Modelo Conceitual de Descoberta de Serviços;
3. Implementação do Modelo;
4. Análise do Modelo;
5. Documentação e Publicação.

A primeira etapa, chamada de Avaliação do Estado da Arte e Revisão Bibliográfica, compreendeu a leitura de assuntos e trabalhos correlatos ao tema proposto. Foi elaborado um relatório de pesquisa, contendo os aspectos mais fortemente e mais fracamente cobertos presentes nas abordagens usadas na resolução do problema-foco.

O relatório constituiu o principal elemento para a construção do modelo conceitual a ser proposto, assim como para a identificação do ineditismo do trabalho. Os meios usados nessa etapa se restringiram à captura, análise e fichamento bibliográfico de documentos presentes em anais, livros e repositórios virtuais espalhados pela Internet, cuja relevância fosse reconhecida pela comunidade científica.

A sequência do trabalho se deu com a etapa chamada de Concepção do Modelo. O passo inicial para execução dessa etapa foi a identificação de requisitos do modelo de busca e seleção de serviços. A base para execução dessa atividade se constituiu em dois trabalhos de mestrado do Programa de Pós-Graduação em Engenharia de Automação e Sistemas (PPGEAS), que foram usados como fontes iniciais de

informações. Um desses trabalhos foi defendido 2007, sobre interoperabilidade entre serviços *web* (PIAZZA, 2007), e, o outro, foi defendido em 2009, que explora o conceito de Software como Serviços (CANCIAN, 2009).

A terceira etapa correspondeu a Implementação e Testes do Modelo. A atividade de implementação do modelo foi realizada com base no Processo Unificado (*Unified Process*), originalmente proposto por Jacobson, Booch & Rumbaugh, descrito em Wazlawick (2004), e em ferramentas computacionais livres, escolhidas a partir das informações colhidas nas etapas de Avaliação do Estado da Arte, Revisão Bibliográfica e Concepção do Modelo.

A etapa seguinte consistiu da Análise do Modelo. Essa etapa teve como meta conduzir um conjunto de experimentos em um cenário cientificamente válido dentro do qual foram avaliados dois pontos: i) o cumprimento do objetivo da Tese e ii) a comprovação da hipótese de pesquisa. Ainda, o resultado do trabalho foi profundamente analisado, verificando-se as qualidades do modelo assim como suas limitações. Finalmente, o algoritmo de descoberta proposto foi formalmente verificado.

Por fim, a etapa de documentação e publicação foi composta das seguintes atividades: qualificação da tese, escrita de artigos e a redação e defesa da tese.

1.11 **Projeto de Ambientação**

Este trabalho está relacionado a dois projetos de pesquisa recém-finalizados. Um internacional, chamado ECOLEAD (*European Collaborative Networked Organization LEADdership Iniciative*), e outro nacional, chamado IFM (*Instituto Fábrica do Milênio*), apoiado pelo Ministério de Ciência e Tecnologia/CNPq.

1.11.1 Projeto ECOLEAD[5]

O *European Collaborative Networked Organization LEADdership Iniciative* teve seu início em abril de 2004 e o término se deu em junho de 2008. Contou com a participação de diversos parceiros, de 14 diferentes países. A finalidade do projeto ECOLEAD foi de

[5] http://www.ecolead.org

estabelecer os fundamentos e mecanismos necessários para uma futura sociedade industrial européia baseada em redes de colaboração. Dada à complexidade do projeto, o mesmo foi dividido em três áreas que constituem a base para organizações em rede. São elas: Ambientes de Criação de Organizações Virtuais, Organizações Virtuais e Comunidades Virtuais Profissionais.

No ECOLEAD, uma infra-estrutura de TIC para o suporte de redes colaborativas foi desenvolvida, e uma das funcionalidades planejadas por essa infra-estrutura é a descoberta de serviços *web* nos repositórios de serviços das empresas-membro das redes e de prestadoras de serviços de software. Assim sendo, conceitualmente, o resultado desta Tese será representado como sendo este serviço, já que aquela infra-estrutura foi desenhada para ser escalável e seus desenvolvimentos incrementais.

1.11.2 Projeto IFM[6]

O Instituto Fábrica do Milênio iniciou suas atividades de pesquisa em abril de 2002 e teve seu término em dezembro de 2008. O IFM é uma organização em rede apoiada pelo Ministério da Ciência e Tecnologia. O IFM congrega cerca de 800 pesquisadores espalhados em 39 grupos de pesquisa e alocados em 32 Instituições de Ensino Superior do Brasil. Seu objetivo focou pesquisas em manufatura voltadas para carências nacionais.

No IFM, o tema de redes colaborativas também foi abordado, dentro do qual o aspecto de infra-estrutura de TIC esteve também contemplado. O serviço de descoberta desenvolvido no projeto ECOLEAD foi conceitualmente previsto para ser desenvolvido. O objetivo no IFM foi coletar requisitos das indústrias que atuaram como piloto para fins de verificação das suas necessidades, características e restrições em termos de arquiteturas SOA e infra-estruturas de TIC de suporte.

[6] http://www.ifm.org.br

1.12 **Organização do Documento**

Esta seção mostra a estrutura da tese em capítulos:

- Capítulo 1: Introdução
  Além de uma breve apresentação sobre o tema de pesquisa, esse capítulo discorre sobre os elementos fundamentais para contextualizar a pesquisa.

- Capítulo 2: Revisão Bibliográfica
  Esse capítulo registra os assuntos essenciais para o entendimento da pesquisa como um todo. Sua organização e ênfase foram baseadas na literatura lida, apresentando assuntos relacionados ao tema desta pesquisa sob o ponto de vista conceitual.

- Capítulo 3: Descoberta de Serviços
  Esse capítulo enfatiza problemas e subproblemas relacionados à descoberta de serviços, bem como mostra estratégias e aponta lacunas ainda pouco exploradas na descoberta dinâmica de serviços *web*.

- Capítulo 4: Modelo de Descoberta
  Este capítulo apresenta o modelo de descoberta dinâmica proposto. As características que descrevem o modelo, sua arquitetura conceitual, a originalidade pretendida e o conjunto de pressupostos são registrados.

- Capítulo 5: Protótipo Computacional
  O capítulo detalha a implementação do modelo proposto. Além da apresentação do protótipo computacional implementado, o capítulo mostra o conjunto de tecnologias e ferramentas utilizadas e um exemplo do funcionamento do protótipo computacional.

- Capítulo 6: Avaliação do Modelo
  O capítulo registra a avaliação do modelo de descoberta com base em um conjunto de métricas que permitem avaliar o modelo. Além disso, diversos experimentos são

descritos e o registro formal da análise da complexidade do algoritmo, usado na descobrir serviços, é realizado.

- Capítulo 7: Conclusões e Trabalhos Futuros
  O capítulo encerra o documento, listando as principais contribuições da pesquisa, as limitações do modelo de descoberta e sugere atividades que podem ser exploradas em pesquisas futuras.

Capítulo 2

Revisão Bibliográfica

Este capítulo apresenta a revisão bibliográfica referente aos temas essenciais para o entendimento do trabalho como um todo. Os assuntos aqui tratados são: *Business Process Management* (BPM), Arquitetura Orientada a Serviços (SOA), Integração BPM&SOA, Catálogo de Processos de Negócios e Software como Serviço (SaaS).

2.1 **Business Process Management (BPM)**

Esta seção apresenta uma série de conceitos relacionados à BPM, sendo eles: definição, perspectivas, motivação e a sua importância frente ao desenvolvimento deste trabalho.

2.1.1 Definição

BPM é definido como uma técnica que suporta processos de negócios usando software para especificar, controlar, executar e analisar processos empresariais que envolvem pessoas, empresas, aplicações, documentos e outras fontes de informações (AALST; HOFSTEDE; MATHIAS, 2003).

O significado de BPM pode ser melhor compreendido a partir da análise da suas dimensões (GARIMELLA; LEES; WILLIAMS, 2008):

- Negócio: algo que interessa ao cliente e, portanto, tem um valor agregado alto para a empresa;
- Gerenciamento: se aplica ao uso de ferramentas que permitem produzir ou se chegar ao produto desejado;
- Processo: transforma entradas em saídas, sendo caracterizado pela necessidade de possuir:
    - Agilidade;
    - Transparência; e
    - Produtividade.

Esta última dimensão (Processo) refere-se a um termo que aparece em vários contextos para designar uma sequência ou encadeamento de atividades que envolvem recursos para se atingir um determinado fim. Por exemplo, no domínio jurídico, a palavra processo é usada para descrever atividades relacionadas aos processos jurídicos. Nos processos químicos, a palavra processo é usada para descrever tarefas relacionadas à aquele contexto, e assim por diante (BALDAM *et al.*, 2007).

O enfoque dado a este trabalho enfatiza processos de negócios empresariais, ou seja, processos destinados a desenvolver as atividades fim de uma empresa, tais como: comercialização de produtos, solicitação de compra de materiais, entre outros.

O gerenciamento de processos de negócios empresariais tornou-se um importante tema nos anos 90, uma vez que muitas empresas foram obrigadas a realizar um esforço para aumentar sua produtividade, melhorar seu relacionamento com o cliente e reduzir o tempo para lançar novos produtos (SENTANIN; SANTOS; JABBOUR, 2008).

2.1.2 Motivação

Recentemente, pesquisas apontam BPM como uma forte tendência. Neste sentido, quatro razões são apontadas para justificar o interesse por BPM (BALDAM *et al.*, 2007):

- Alta competitividade entre empresas: a queda das barreiras alfandegárias (início da globalização) permitiu que novas empresas participassem de mercados até então desconhecidos;
- Controle da complexidade: a pressão por melhores resultados é uma constante, principalmente, em grandes empresas; mesmo quando não há uma grande competição de mercado.

Nesse caso, a pressão se restringe à obtenção de maiores lucros, isso em função do mercado financeiro;

- Necessidade de maior rapidez no desenvolvimento de produtos: o foco passa a ser o cliente e o processo, não o produto. O produto deve ter módulos básicos e customizáveis que podem ser alterados e evoluídos. Nesse cenário, passa-se a usar tecnologias adaptáveis, nas quais regras e fluxos possam ser alterados sem grande envolvimento de recursos;
- Aumento das exigências de transparência: o mercado financeiro tende a valorizar ações destas empresas.

Neste cenário, existe uma enorme atenção dada ao BPM, uma vez que ele representa o conjunto das melhores idéias e experiências obtidas em gerenciamento de processos de negócios nos últimos anos (GARIMELLA; LEES; WILLIAMS, 2008).

2.1.3 Importância

Enquanto sistemas tradicionais de *workflow* têm seu foco principal na execução de processos de negócios, BPM amplia o tratamento dado aos processos de negócios, agindo, principalmente, na fase de diagnóstico.

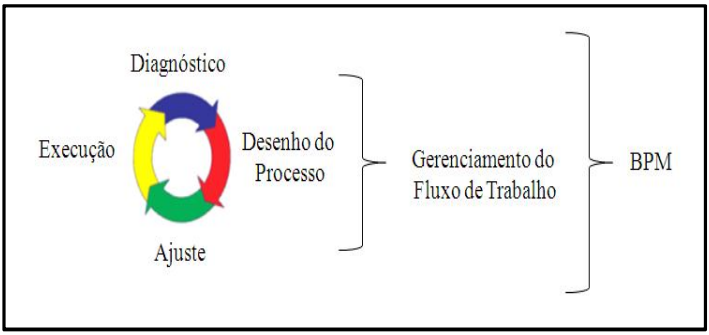


Figura 2 - Ciclo de vida BPM comparado Sistema tradicionais de *Workflow*. Fonte: (AALST; HOFSTEDE; MATHIAS, 2003).

A Figura 2 ilustra diferenças entre BPM e sistemas tradicionais de *workflow*. A fase de desenho permite a especificação de processos de negócios. Na fase de configuração ou ajuste, informações sobre um processo são inseridas e adequadas para que a fase de execução ocorra.

Na fase de diagnóstico, os processos de negócios são analisados em função de problemas e melhorias que podem ser realizadas.

Em outro viés, BPM proporciona uma visão organizacional, na qual a ênfase está na execução de atividades horizontais que envolvem: sistemas, pessoas, departamentos entre outros. A Figura 3 mostra um exemplo desta visão.

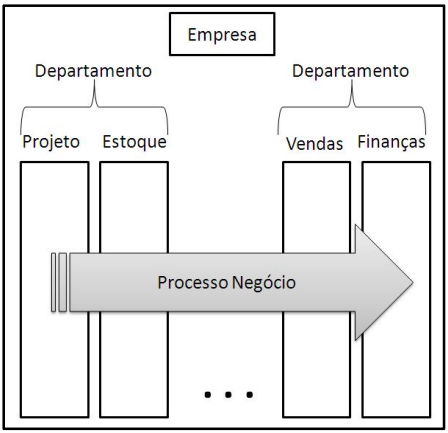


Figura 3 – Empresa organizada a base de Processos.

Esta visão contrasta com a visão clássica, a qual descreve a forma com que trabalha uma empresa tradicional. A visão horizontal envolve o gerenciamento de recursos e privilegia clientes, fornecedores e atividades que fazem parte do processo. Esta forma de organização resulta em um maior valor agregado aos processos como um todo (GOLDKUHL; LIND, 2008).

2.1.4 Considerações

É fato que o mercado de BPM tende a crescer muito nos próximos anos, principalmente pela necessidade de melhoria dos processos de negócios, sendo este um fator crítico de sucesso e sobrevivência para as empresas.

Esse crescimento abre novas oportunidades, seja profissional, empresarial e de integração com o nível de tecnologia das empresas.

Nesse alinhamento, BPM atua no nível dos processos de negócios. Enquanto, em outro extremo, a tecnologia permite que processos sejam executados por sistemas presentes nas empresas.

Portanto, a atenção dada à estratégia BPM, nesse trabalho, reside no fato de ela ser a camada que permite interação e intervenção nos processos de negócios presentes em uma empresa.

2.2 **Arquitetura Orientada a Serviço (SOA)**

A crescente demanda de aplicações voltadas à Internet, aliada à necessidade destas aplicações lidarem com variadas plataformas, protocolos, *devices*[7] entre outros; somadas às limitações do paradigma de desenvolvimento de sistemas baseado em componentes e o surgimento dos serviços *web* fez com que o paradigma de desenvolvimento orientado a serviços emergisse (PIAZZA, 2007).

Desta forma, esta seção é dedicada a apresentar os principais conceitos relacionados a SOA, sua arquitetura e tecnologias associadas.

2.2.1 Definição

Arquitetura Orientada a Serviço ou simplesmente SOA é um estilo arquitetural independente de tecnologia. É uma disciplina para construção de sistemas distribuídos baseados em serviços *web*, os quais são componentes de software descritos, que podem ser descobertos e usados para implementar tarefas previstas em processos de negócios (DECKER *et al.*, 2007).

Além de ser um paradigma de desenvolvimento de sistemas distribuídos, SOA tem como intenção dar suporte ao desenvolvimento rápido, com baixo custo, e facilitar a composição de aplicações distribuídas em ambientes heterogêneos (BREOVOLD; LARSSON, 2007).

2.2.2 Características

Em W3C (2004), Papazoglou *et al.* (2007) e Josuttis (2008) podem ser encontradas um conjunto de características chaves presente no paradigma SOA:

---

[7] Componentes e dispositivos de hardware ou software.

- Abstração: um serviço é visto como uma caixa-preta. Ou seja, ele é usado em função do que faz e não de como é implementado. Esse aspecto permite visualizar e compreender um sistema sob o ponto de vista lógico das funções que o implementam;
- Orientado a mensagem: a troca de informações entre um provedor e um consumidor de serviço é feito mediante mensagens, as quais obedecem a um formato conhecido tanto pelo provedor quanto pelo consumidor de serviços. Isto possibilita que sistemas legados sejam conectados e incorporados a outros sistemas, uma vez que não é necessário conhecer e trabalhar com comandos internos presentes nesses sistemas que ainda operam;
- Descrito para máquinas processarem: através da tecnologia *eXtensible Markup Language* (XML), máquinas e aplicações podem interpretar solicitações ou respostas de um serviço, o que garante a automação de tarefas;
- Orientado à rede: corresponde à natural distribuição dos serviços, que tendem a ser usados sobre uma rede de comunicação, normalmente a Internet;
- Plataforma neutra: mensagens são trocadas em um formato independente de plataforma.

Estas características motivam o uso de SOA e viabilizam a construção de aplicações flexíveis, heterogêneas, dinâmicas e distribuídas através do uso de componentes de software denominados serviços *web*.

2.2.3 Serviços *Web*

SOA utiliza serviços como elementos fundamentais para o desenvolvimento de aplicações (BREOVOLD; LARSSON, 2007). Assim, um Serviço *web* ou *Web Service* (WS) é definido pelo Grupo de Trabalho, ligado ao W3C, responsável pela especificação da *Web Services Architecture* (WSA) como:

> Um serviço *web* é um software projetado para suportar integração entre máquinas que fazem parte de uma rede. Todo serviço tem uma interface (WSDL) que o define, sendo ela descrita uma linguagem (XML) que máquinas podem processar. Outros sistemas interagem com um serviço a partir

> de um protocolo (SOAP) que descreve a estrutura das mensagens trocadas entre máquinas. Tipicamente este protocolo é utilizado usando HTTP em conjunto com outros padrões *web* (W3C, 2004).

Algumas características inerentes aos serviços *web* (PIAZZA, 2007):

- Definição do serviço com base em uma interface: aplicações são criadas como uma coleção de serviços interligados usando a descrição presente em suas interfaces. Interfaces descrevem serviços e especificam pontos de acesso à implementação. Em função disso, serviços possuem baixo acoplamento;
- Capacidade de composição: serviços podem compor outros serviços, promovendo a reusabilidade dos serviços em diversas aplicações, processos e lógicas distintas;
- Granularidade: indica que o serviço engloba funções de valor agregado já que um serviço é uma fachada para um conjunto de regras de negócio de granularidade fina;
- Não gerenciam estados: os serviços não tratam informações de estado (serviços são *stateless*), fato que privilegia o baixo acoplamento dos serviços;
- Reusabilidade: a lógica da aplicação encapsulada como um serviço pode ser reutilizada em diferentes aplicações;
- Interoperabilidade: devido à fundação tecnológica padronizada dos serviços, os mesmos possuem um alto poder de comunicação com sistemas construídos com tecnologias diferentes.

A Figura 4 apresenta uma síntese dos principais atributos relacionados aos serviços *web* comparado a outras formas de desenvolvimento de sistemas.

Desta forma, aplicações compostas por serviços podem sofrer ajustes mais rápidos, frequentes, com custo mais baixo e menor intervenção humana, permitindo às empresas ganharem em competitividade (DAN; JOHNSON; ARSANJANI, 2007), (CASTRO-LEON; HE; CHANG, 2007) e (BREOVOLD; LARSSON, 2007).

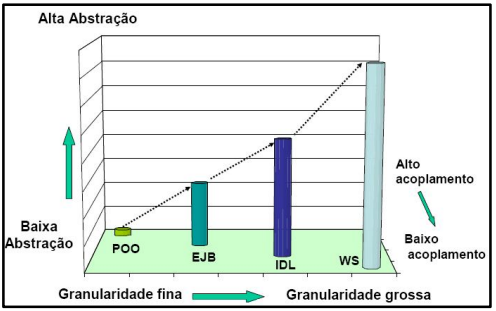


Figura 4 - Síntese dos principais atributos dos serviços comparados a outras tecnologias.
Fonte: (PIRES, 2008).

2.2.4 Arquitetura

O desenvolvimento de sistemas SOA, implementados à base de serviços *web*, vem tomando corpo à medida que o número de serviços *web* cresce e são disponibilizados na Internet (BREITMAN, 2005).

Diante disso, serviços *web* têm sido tema de várias pesquisas, pois se vislumbra que eles poderão dar um novo impulso à Internet. Neste sentido, uma série de padrões foram desenvolvidos, sendo que três deles compõem a fundação dos serviços *web* (BAILEY; ZHU, 2006).

- *Web Service Description Language* (WSDL),
- SOAP e
- *Universal Description, Discovery and Integration (UDDI*).

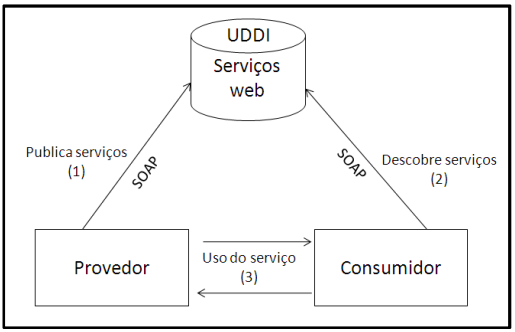


Figura 5 – Ciclo de vida dos serviços *web*.
Fonte: (DEITEL; DEITEL, 2003).

Na construção de aplicações orientadas a serviços, cabe aos provedores a oferta dos mesmos. A publicação de serviços é feita a partir de informações presentes em um arquivo WSDL, formado a base de XML.

No processo de publicação, o provedor usa funções presentes na API-UDDI para publicar características dos serviços em repositórios de serviços, os UDDIs (passo 1 na Figura 5).

Um consumidor acessa o repositório de serviços UDDI e faz buscas por serviços que lhe interessam. Encontrado o serviço desejado, lhe são entregues informações que permitem invocar o serviço (passo 2 na Figura 5).

Por fim, provedor e consumidor interagem através da troca de mensagens prevista na interface do serviço, sendo tais mensagens formatadas usando o protocolo SOAP (passo 3 na Figura 5).

Para dar suporte ao desenvolvimento de aplicações orientadas a serviços, foi projetada uma arquitetura organizada em camadas com uma série de tecnologias relacionadas, conforme pode ser constatado a partir da análise da Figura 6.

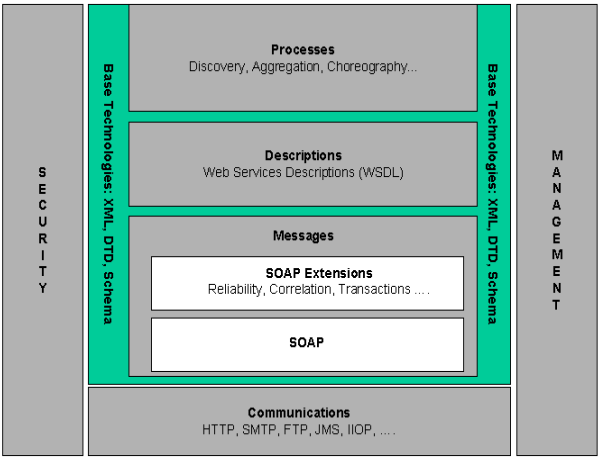


Figura 6 - Tecnologias e Camadas da *Web Service Architecture*.
Fonte: (W3C, 2004).

2.2.4.1 Universal Description Discovery and Integration (UDDI)

A versão 1.0 do UDDI foi lançada em setembro de 2000. Seu objetivo era criar uma estrutura para registro de informações sobre

serviços *web*. Em junho de 2001, a versão 2.0 do UDDI teve como principal característica uma maior flexibilidade em relação à classificação dos serviços. Já em 2004, a versão 3.0 teve como maior avanço o suporte a uma grande interação no processo de publicação de serviços (OASIS, 2004).

O *Universal Description Discovery and Integration* (UDDI) fornece serviço de diretório e serviço de nomes, onde as descrições dos serviços podem ser pesquisadas por: nome ou categoria ou, ainda, podem ser acessados diretamente, bastando informar o seu *Uniform Resource Location* (URL).

Três tipos de informações estão disponíveis em um UDDI (OASIS, 2004):

- *White pages* (semelhantes às páginas brancas de uma lista telefônica) incluem informações básicas de uma organização, tais como: nome, endereço, contato entre outras;
- *Yellow pages*: descrevem serviços usando diferentes categorias: bancos, oficinas, hospitais entre outras. É possível descobrir serviços *web* informando uma categoria;
- *Green Pages*: incluem informações técnicas (por exemplo: formato de dados, parâmetros de entrada/saída e outros) relativas às funções disponíveis em um determinado *serviço web*.

Os UDDIs são providos de estruturas de dados que organizam as informações dos serviços e, assim, permitem realizar publicação e pesquisa. Em um UDDI os dados são organizados em termos de quatro estruturas, conforme mostra o Quadro 4.

Nome	Descrição
businessEntity	Contém informações de quem fornece o serviço
businessService	Contém informações descritivas de um serviço em particular
bindingTemplate	Contém informações de como invocar um serviço
tModel	Descreve especificações técnicas implementadas pelo serviço

Quadro 4 - Estruturas de dados usadas em UDDIs.

A Figura 7 ilustra uma entrada hipotética em um repositório UDDI, mostrando o uso das estruturas de dados presentes no Quadro 4.

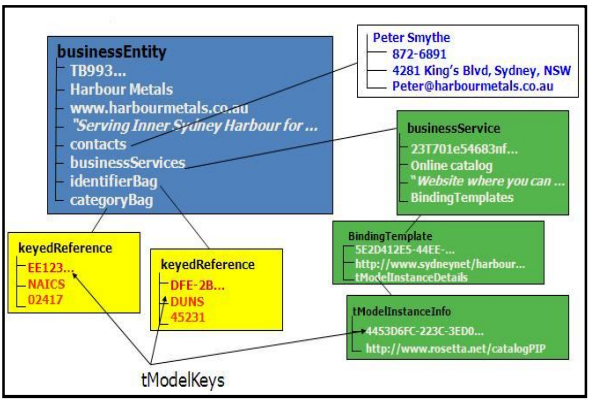


Figura 7 - Exemplos de entradas em um UDDI.
Fonte: (OASIS, 2000).

As entradas contidas na estrutura *businessEntity* contém dados da empresa responsável pelo serviço. O elemento *businessService*, presente no elemento *businessEntity*, é um link que leva à descrição de uma família de serviços. O link *identifierBag* (opcional) remete ao elemento *KeyedReference*. Nesse elemento são armazenados dados sobre o negócio da empresa através de um par (nome, valor). Informações de categorização do serviço são armazenados em elementos *keyedReference*, acessadas através do *link categoryBag*, também opcional na estrutura *businessEntity*.

A estrutura *businessService* armazena informações sobre serviços, tais como: descrição, nome, identificador do serviço, entre outros. Há um link para um elemento *bindingTemplate*, o qual possui links para elementos *tModelInstanceInfo* que armazenam informações técnicas sobre um serviço: endereço de acesso, protocolo de acesso, entre outros. Os identificadores *tModelKeys* são associados aos *KeyedReference* de forma a estabelecerem uma relação entre a entidade que fornece o serviço, o serviço em si e sua especificação técnica.

Em um UDDI, a pesquisa é realizada através de um conjunto de funções, presentes na API-UDDI e que permitem realizar buscas com base em dois conjuntos de operações (COULOURIS; DOLLIMORE; KINDBERG, 2007).

Um conjunto rotulado através do prefixo *get* (por exemplo: *get_businessDetail*, *get_serviceDetail*, *get_bindingDetail* e *get_tModelDetail*), tem a função de recuperar uma entidade em particular, um serviço *web*. Outro conjunto, com prefixo *find* (*find_business*, *find_service*, *find_binding* e *find_tModel*) permite recuperar um conjunto de entidades ou serviços *web*.

Já os métodos com prefixos *save* (*save_business*, *save_service*, *save_binding* e *save_tModel*) são usados para publicar informações sobre serviços. Os métodos com prefixo *delete* (*delete_business*, *delete_service*, *delete_binding* e *delete_tModel*) são utilizados para remover informações relacionadas a um serviço. Esta relação se restringe aos métodos clássicos. A lista completa de métodos que podem ser usados está disponível na API do padrão UDDI.

2.2.4.2 Web Service Description Language (WSDL)

Em 2000, iniciaram-se os trabalhos para desenvolvimento da especificação WSDL. Rapidamente, devido a um conjunto de melhorias, a versão 1.0 foi seguida pela WSDL 1.1 a qual tem sido usada para o desenvolvimento de várias ferramentas voltadas aos serviços *web* (MILLER *et al.*, 2004).

Muitos esforços foram realizados para melhorar a especificação WSDL 1.1 e o resultado desse trabalho se deu com a criação de uma nova versão da especificação, a WSDL 1.2, que, por conter mudanças bastante significativas foi renomeada para WSDL 2.0, sendo recomendado como padrão em 26 de junho de 2007, pelo W3C (W3C, 2007c).

A WSDL é um documento escrito em XML que especifica informações necessárias para se conectar a um serviço *web*.

A WSDL é usada para responder três perguntas sobre um serviço *web* (MENÉNDEZ, 2002):

- O que é?
- Onde encontrá-lo?
- Como chamar um serviço?

Os elementos que fazem parte um documento WSDL são mostrados na Figura 8.

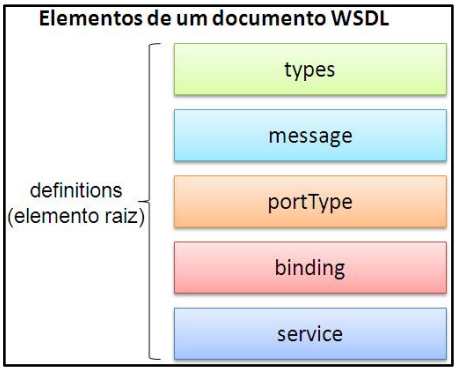


Figura 8 - Elementos presentes em um documento WSDL.

Os elementos WSDL são (W3C, 2007c):

- *definitions*: é o elemento raiz de um documento WSDL e contém todas as definições globais através do uso de *namespaces*;
- *types*: indica uma série de definições de tipos usadas em declarações ao longo do documento WSDL;
- *message*: este elemento descreve de forma abstrata a organização das mensagens utilizadas nas operações de envio ou recebimento de uma mensagem. Para essa finalidade cada elemento <*message*> contém um ou mais elementos <*part*> os quais usam definições presentes no elemento <*types*> para formar as partes de uma mensagem.
- *portType*: esse elemento agrupa um conjunto de operações disponíveis em um *Web Service*;
    - *operation*: descrição de uma ação ou função disponibilizada pelo serviço. Existem três mensagens permitidas:
        - Mensagem de Entrada;
        - Mensagem de Saída;
        - Mensagem de Falha (opcional).
- *binding*: descreve o mecanismo usado por um serviço para se comunicar com um cliente. O elemento *binding* é utilizado para definir o formato das mensagens e o mapeamento dos elementos *operations* para cada *portType*;
- *service*: define a localização real do serviço, tendo em vista que o mesmo pode conter várias portas e cada uma delas é

específica para um tipo de ligação, descrita no elemento *binding*.

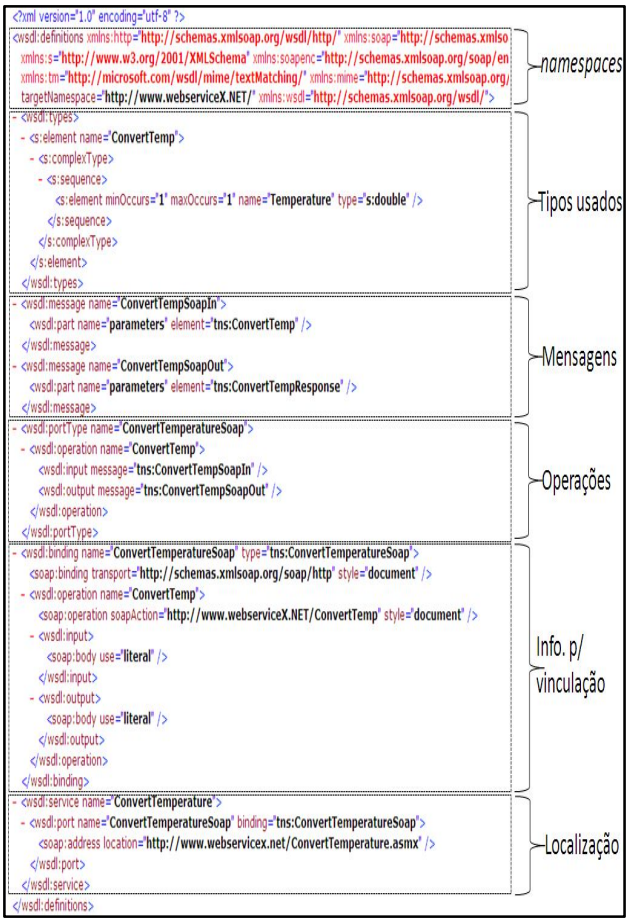


Figura 9 - Exemplo de um documento WSDL.

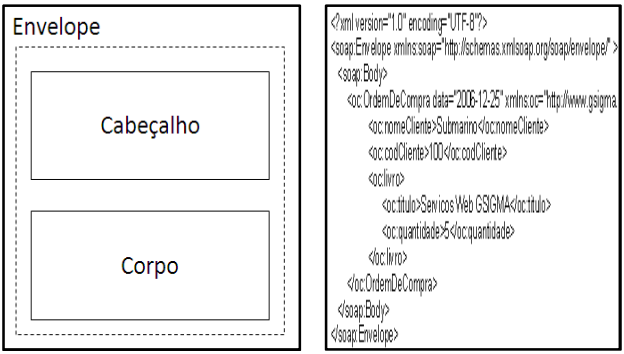
A Figura 9 apresenta um exemplo de documento WSDL. No exemplo, são realçados cada um dos elementos presentes em um documento WSDL, bem como a estrutura e organização dos elementos no trabalho de especificação de serviços.

2.2.4.3 SOAP

O SOAP faz uso da tecnologia XML, sendo independente de qualquer modelo de programação em particular, definindo assim uma estrutura de transferência de mensagens extensível (W3C, 2007b).

Embora XML seja feito para a troca de dados, ele sozinho não é suficiente para o intercâmbio de informações através da *web*, é necessário ter um protocolo para realizar chamadas a procedimentos remotos tal como o protocolo HTTP (*Hypertext Transfer Protocol*) (W3C, 2007b).

Uma mensagem SOAP é um documento XML que é formado por um elemento *Envelope* que abriga a definição de outros dois elementos: *Cabeçalho* (opcional) e o *corpo* (BASHA *et al.*, 2002), conforme ilustra a Figura 10. O *Envelope* é o elemento de nível mais elevado de uma mensagem SOAP. Ele contém um *cabeçalho* usado para adicionar recursos de autenticação, gerenciamento de transação entre outros. O elemento *corpo* é a parte principal da mensagem SOAP. Nesse elemento estão armazenadas as informações necessárias para que o processador de destino possa processar o pedido e retornar uma resposta.



a) Estrutura b) Exemplo

Figura 10 - Mensagem SOAP.
Fonte: (PIAZZA, 2007).

Um exemplo da estrutura de um envelope SOAP pode ser visualizado na Figura 10a e uma instância SOAP pode ser visualizada na Figura 10b.

2.2.5 Considerações

SOA representa uma forma de pensar quanto ao projeto de sistemas e sua posterior integração a outros sistemas. Em função das suas características arquiteturais, sistemas podem ser implementados usando-se de várias tecnologias.

O paradigma SOA é vantajoso em relação às abordagens tradicionais, pois permite projetar aplicações fortemente coesas e fracamente acopladas através de módulos especializados, distribuídos, acessados remotamente, interoperáveis e reutilizáveis, chamados de serviços. Nesse cenário, serviços determinam o comportamento de um sistema associado a um dado processo de negócio.

Em relação ao restante do trabalho, a contribuição desta seção está em permitir que a lógica de negócios seja implementada através de serviços, os quais serão especificados e descobertos por um mecanismo para serem vinculados às aplicações.

2.3 Integração BPM&SOA

O conteúdo desta seção mostra a necessidade e a importância da integração entre duas iniciativas diferentes: BPM e SOA. Por um lado BPM, preocupado em alinhar melhor processos de negócios frente às mudanças constantes e rápidas dos seus requisitos e, por outro, SOA como solução para ligar atividades presentes em processos de negócios a serviços de forma flexível, rápida, com baixo custo e independente de tecnologia.

2.3.1 Problema

Requisitos associados a negócios mudam rapidamente, não permitindo que sistemas, aplicações ou processos acompanhem o ritmo destas necessidades de mercado. As razões para isto são (PIRES, 2008):

- Sistemas atuais (grande parte deles) ainda são monolíticos ou estáticos, possuindo baixa flexibilidade em relação a mudanças;
- Existe um alto custo para alterar um sistema monolítico, pois é preciso intervenção humana, por meio de programadores, para executar esta tarefa;
- Alto custo associado à atualização de um sistema monolítico;

- Como garantir que ajustes nos processos de negócios não comprometerão funcionalidades dos demais sistemas de informação empresariais;
- Nem sempre uma empresa dispõe de recursos e tempo para realizar esta tarefa.

Em função disso, processos de negócios ficam congelados e impossibilitados de agregar novos requisitos, fazendo com que empresas percam em competitividade.

2.3.2 Solução

Em virtude da necessidade de tornar os processos de negócios alinhados às necessidades de mercado, um grande esforço vem sendo despendido para integrar negócios e Tecnologia da Informação (TI), uma vez que empresas são obrigadas a, rapidamente, reverem e adaptarem seus processos de negócios e sistemas para sobreviverem (NEUBAUER, 2009).

Recentemente, a combinação de BPM e SOA tem sido defendida como a melhor estratégia para que empresas obtenham um alinhamento mais próximo entre processos de negócios e recursos tecnológicos e, com isto, consigam projetar aplicações que melhor respondam às mudanças dos requisitos dos negócios.

A combinação de BPM e SOA permite às empresas automatizar e otimizar processos de negócios através de componentes reusáveis (serviços *web*), integrando complexos e heterogêneos sistemas (KAMOUN, 2007).

Nesta junção, BPM é uma estratégia que possibilita modelar um negócio em termos de atividades, as quais perpassam e envolvem vários sistemas dentro de uma empresa. Estas atividades formam os processos de negócios, os quais são representados em um formato que possa ser entendido e processado por máquinas. No que diz respeito a SOA, serviços *web* implementam funções presentes nos processos de negócios, de tal forma que diferentes aplicações podem compartilhar um mesmo serviço devido à alta interoperabilidade e ao baixo acoplamento advindos da fundação tecnológica padronizada dos serviços *web* (KAMOUN, 2007).

Para um melhor entendimento das iniciativas, a Figura 11 lista as principais diferenças entre elas.

BPM	SOA
Dirigido pelo Negócio	Dirigido por TI
Abordagem *Top-down*	Abordagem *Bottom-up*
Reusa modelos de processo	Reusa implementações de serviços
Orientado a projeto	Orientado a infra-estrutura da aplicação corporativa
Métricas de negócio e *Key Performace Indicators (KPIs)*	Métricas arquiteturais, de consistência lógica, de facilidade de integração e de custo de mudanças

Figura 11 - Diferenças entre BPM e SOA.
Fonte: (PIRES, 2008).

A integração BPM&SOA, conforme ilustra a Figura 12, tem sido defendida como a melhor estratégia para empresas obterem um alinhamento mais próximo entre processos e tecnologia (NEUBAUER, 2009).

Figura 12 - Parceria BPM&SOA.
Fonte: (PIRES, 2008).

Nesta junção, BPM trata da especificação, simulação, execução e diagnóstico de processos, SOA é definido como uma forma arquitetural para sistemas distribuídos (W3C, 2004) e serviços *web* são uma das existentes implementações tecnológicas para SOA.

2.3.3 Considerações

Esta seção tratou da integração entre as iniciativas BPM e SOA. Por um lado, BPM realizando o monitoramento, análise e simulação de processos de negócios e, por outro, SOA contribuindo com a execução de tarefas presentes nos processos de negócios, implementadas através de serviços.

A integração da camada de negócios BPM à camada tecnológica SOA permite às empresas tornarem seus processos de negócios mais alinhados às necessidades de mercado. Neste viés, mudanças nos processos reusam serviços, quando necessário, e mudanças de serviços não influenciam na especificação dos processos de negócios.

A estratégia BPM&SOA dá maior competitividade às empresas, uma vez que elas podem acompanhar as mudanças nos requisitos das suas aplicações de forma flexível, dinâmica e com custo baixo. Isto impulsiona às Pequenas e Médias Empresas (PMEs) para uma nova realidade, permitindo a elas participarem de novos mercados.

Em virtude do restante do trabalho estar constituído sobre a parceria BPM&SOA, essa associação é de fundamental importância para a compreensão dos demais tópicos abordados na revisão bibliográfica.

2.4 Catálogo de Processos de Negócios

Esta seção apresenta o catálogo de processos de negócios que está integrado ao ambiente BMP, sendo parte do modelo proposto nesta tese. Sua formação contém especificações de processos de negócios já consolidadas e aceitas como padrão. Um conjunto de especificações de processos de negócios empresariais armazenados no catálogo. Essas especificações definirão o domínio usado na atividade de descoberta dinâmica de serviços *web*.

Diante do exposto acima, esta seção aborda: o objetivo do catálogo de processos de negócios, como o catálogo será composto e apresenta a *Universal Business Language* (UBL), origem das especificações dos processos empresarias disponíveis no catálogo.

Este trabalho não teve por finalidade realizar uma exaustiva e profunda análise das iniciativas que exploram catálogos de processos. Mais informações e detalhes podem ser encontrados em Bezerra (2011).

2.4.1 Objetivo

O Catálogo de Processos de Negócios será usado como fonte de processos de negócios empresariais, sendo formado por um conjunto de especificações. Estas especificações servirão de modelo para ser usado na construção de aplicações.

Além disso, o conjunto de especificações presentes no catálogo será um elemento chave na definição do modelo proposto, pois a

especificação dos processos servirá de baliza para definir o domínio ou contexto a ser adotado na descoberta dinâmica de serviços *web*.

A Figura 13 apresenta o catálogo de processos de negócios integrado a um ambiente BPM. No ambiente BPM, o projetista usa um editor de processos que suporte a *Business Process Modeling Notation* (BPMN) (OMG, 2011) para montar uma aplicação ou criar uma aplicação customizada.

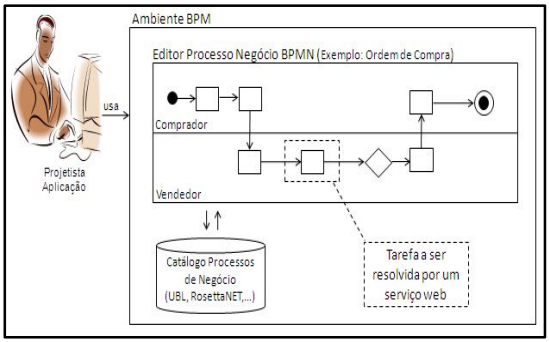


Figura 13 - Catálogo de Processos de Negócios integrado a um Ambiente BPM.

A presença do catálogo de processos de negócios integrado a um ambiente BPM&SOA cria um diferencial ainda mais poderoso para tratar das mudanças dos requisitos de mercado, pois suas especificações de processos já prontas e adotadas pela comunidade (padrão) facilitam a interoperabilidade entre processos de negócio de distintos.

2.4.2 Composição

As especificações de processos de negócio estudadas UBL (OASIS, 2006) e RosettaNET (ROSETTANET, 2010) - determinam o fluxo de informações necessárias para a execução de um processo de negócio. Esse fluxo é criado, geralmente, usando diagramas de atividades da UML.

Essas especificações são transportadas para o catálogo através da interpretação de arquivos XML atreladas a cada especificação. Assim, basta acrescentar ou retirar componentes BPMN de forma a particularizar um processo de negócio.

A tarefa de criação de modelos ou a importação de modelos prontos expressos em BPMN deve ser uma funcionalidade inerente ao ambiente BPM e totalmente transparente ao projetista de aplicações.

2.4.2.1 Universal Business Language (UBL)

UBL foi inspirado nas limitações do padrão *Electronic Data Interchange* (EDI) (OASIS, 2006). UBL é uma linguagem baseada no padrão XML para descrição de documentos trocados eletronicamente em transações eletrônicas. Um conjunto de padrões voltados a negócios eletrônicos, cuja finalidade é aumentar a interoperabilidade no *e-commerce* ao padronizar mensagens de negócio.

O Quadro 5 apresenta a evolução da UBL até a última versão 2.0, aprovada pelo OASIS em dezembro de 2006.

Geração	Nome do padrão
Geração 1 (1Q 1998)	CBL 1.0 (VEO/NIST)
Geração 2 (2Q 1999)	CBL 2.0 (*Commerce One*)
Geração 3 (4Q 2000)	xCBL 3.0 (*Commerce One og SAP*)
Geração 4 (1Q 2003)	UBL 0.7 (OASIS)
Geração 5 (4Q 2004)	UBL 1.0 (OASIS)
Geração 6 (1Q 2006)	UBL 2.0 (OASIS)

Quadro 5 - Evolução do Padrão UBL.

Todos os XML *Schemas* são especializados em um conjunto de documentos, os quais contêm regras de formação de documentos XML trocados em transações ou processos de negócios.

Os XML *Schemas* constituem o vocabulário de dados trocados entre parceiros de negócios, sendo caracterizados pelos seguintes aspectos chave:

- Modulares;
- Reusáveis;
- Extensíveis.

Esses aspectos permitem que o padrão possa evoluir para outros domínios ou necessidades específicas, caracterizando-se como um padrão consistente e genérico no que tange a especificação de informações envolvendo transações eletrônicas.

Uma explicação mais detalhada do funcionamento operacional do processo de negócio *Ordering* UBL é feito na Seção 4.2.2.4, onde um exemplo completo do funcionamento do modelo proposto é apresentado.

2.4.2.2 RosettaNet

RosettaNet é o nome dado a um consórcio global fundado em 1998 sem fins lucrativos, composto de mais de 500 empresas líderes de mercado no seguimento de componentes eletrônicos, fabricação de semicondutores e telecomunicações. Seu objetivo é definir processos de negócios pertencentes à classe B2B, assim como padrões para a troca de dados entre participantes de um negócio eletrônico através de *Partner Interface Processes* (PIPs). Os PIPs são responsáveis por especificar a estrutura e o formato de documentos, assim como atividades, ações e papéis de cada participante em um processo de negócio. Por exemplo, o PIP3A4 - Solicitação de Ordem de Compra (PIP 3A4: *Request Purchase Order*) permite a um comprador emitir uma ordem de compra e a um vendedor confirmar se a ordem será aceita, rejeitada ou se essa ficará pendente, conforme mostra a Figura 14.

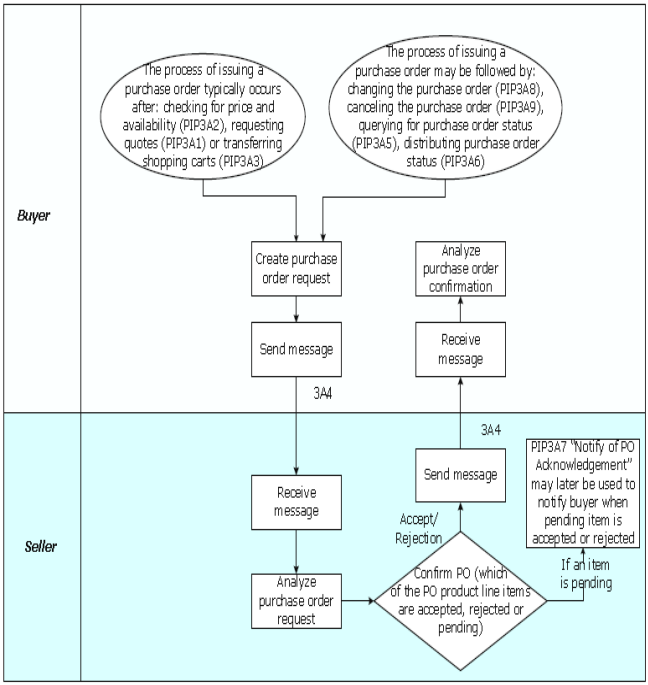


Figura 14 - PIP3A4 – Solicitação de Ordem de Compra.
Fonte: (ROSETTANET, 2010).

No PIP3A4 o fluxo de atividades é representado pelas setas direcionadas. O losango presente no parceiro de negócio *Seller* serve como representação de decisões, enquanto que os retângulos presentes em ambos os parceiros de negócios definem as atividades a serem executadas para que ocorram solicitações de ordens de compra. As duas elipses localizadas no parceiro *Buyer* descrevem fatos (expressos em outros PIPs) a serem considerados para que o PIP3A4 possa ser iniciado.

2.4.3 Considerações

Esta seção comentou sobre o catálogo de processos de negócios, fonte de especificação para o desenvolvimento de aplicações e aspecto chave para definir o domínio dos processos de negócios usado na descoberta de serviços.

Em relação aos processos de negócios, esta seção permitiu concluir que enquanto os RosettaNet PIPs estão preocupados em definir padrões verticais, com ênfase na indústria de fabricação de componentes eletrônicos, os padrões para processos UBL, horizontais, podem ser usados em vários tipos de atividades de indústria, comércio, finanças entre outras.

Assim, caberá a cada fornecedor de BPM implementar e ofertar um ambiente intuitivo o bastante para que usuários participantes de um processo possam construir aplicações, usando especificações presentes em um catálogo de processos. Isto reduz recursos alocados à tarefa de especificar processos, além de permitir que analistas de negócios usem modelos prontos na atividade de arquitetar aplicações.

Desta forma, o catálogo de processos de negócios será composto por processos de negócios UBL, em função dos seguintes motivos:

- Toda e qualquer transação acontece em um formato livre, comum e aberto;
- Seus processos de negócios objetivam contornar as limitações do padrão (*Electronic Data Interchange*) EDI, portanto seus modelos possuem base sólida, constituída há anos;
- Em virtude do custo reduzido da UBL, já que ela trafega sobre a Internet, os custos com configuração e manutenção de redes são mais baixos, permitindo que haja o aumento da participação de Pequenas e Médias Empresas (PMEs) no *e-commerce*.

Em função do exposto e considerando a proposta de construção do modelo de descoberta serviços, o conjunto de processos de negócios especificados pela UBL, na sua versão mais recente, deverá estar integrado ao ambiente BPM.

2.5 **Composição de Serviços**

Esta seção visa apresentar uma introdução sobre composição de serviços. Inicialmente, registram-se algumas definições para composição e justifica-se o seu uso no contexto da desejada integração BPM&SOA. Na sequência, dois modelos para composição são apresentados. A seção termina com algumas considerações sobre composição de serviços.

2.5.1 Definição

Uma das primeiras definições para composição de serviços registra que ela consiste de um conjunto de atividades necessárias para combinar e vincular serviços existentes com o intuito de criar novos processos (DUSTDAR, 2003). Uma definição um pouco mais recente aponta que o processo de composição envolve combinar diversos serviços (possivelmente oferecidos por diferentes provedores) para formar um único serviço (serviço composto ou resultante) que satisfaça as necessidades de um cliente (PAPAZOGLOU *et al.*, 2007). Nesta mesma direção, Zongyan *et al.* (2007) comentam que composição é o processo de combinar serviços para criar serviços de maior valor agregado. Assim, com base nas definições mencionadas, percebe-se que o processo de composição de serviços visa criar um novo serviço cuja composição é feita a partir de um conjunto de serviços existentes, previamente especificados, descobertos e vinculados que atendem a um conjunto de restrições associadas.

2.5.2 Motivação para Composição em ambientes BPM&SOA

A motivação para uso de composições em ambientes integrados BPM&SOA está concentrada em dois aspectos básicos (BEHARA, 2006). Um deles diz respeito ao reuso de serviços. Neste aspecto, quanto mais os serviços da camada tecnológica forem desacoplados, maior será o nível de resuo dos mesmos. A vantagem disso está na possibilidade de um serviço fazer parte da implementação de diversos processos de negócios. Isto rentabiliza serviços existentes e reduz custos de desenvolvimento de novos serviços. O segundo aspecto está associado à

possibilidade de combinar serviços visando prover implementações para funcionalidades presentes nos processos de negócios ou capaz de implementar um processo de negócio por completo. É neste aspecto que se justifica com maior ênfase o uso da composição de serviços, pois é através dela que processos de negócios podem ser modificados com maior agilidade, necessitando apenas que novas composições sejam ajustadas ou desenvolvidas.

2.5.3 Modelos para Composição

Bucchiarone e Gnesi (2006) definem dois modelos fundamentais de composição: a estática e dinâmica, conforme ilustra a Figura 15.

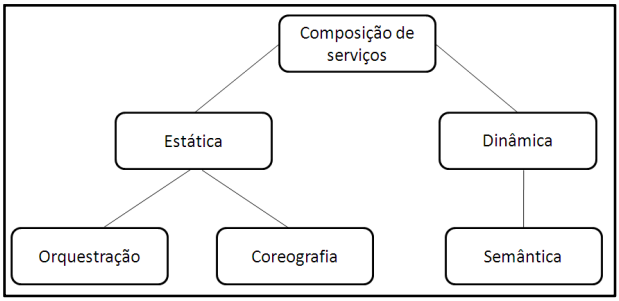


Figura 15 - Modelos de composição de serviços.
Fonte: (BUCCHIARONE; GNESI, 2006).

Na composição estática, serviços são compostos em tempo de projeto de aplicações, enquanto que a composição dinâmica ocorre em tempo de execução de aplicações (DUSTDAR, 2003). Em ambos os modelos, para que a composição ocorra com sucesso, três aspectos merecem destaque (DUSTDAR, 2003). O primeiro aspecto diz respeito à necessidade do uso de técnicas que permitam uma mais adequada representação ou especificação de serviços. O segundo aspecto envolve a dependência da composição do processo de pesquisa e seleção (descoberta) de serviços, aspecto também ratificado por Tsalgatidou e Pilioura (2002). O terceiro aspecto diz respeito à quão eficiente, em termos de desempenho, é a composição de serviços resultante.

Na composição por orquestração de serviços, a ideia central reside em definir o fluxo que determina a ordem de execução dos serviços e as condições sob as quais os serviços participantes da composição são invocados. Este processo é coordenado por um *engine* (motor de execução de aplicações SOA) que faz a invocação e a

combinação dos serviços presentes da composição, conforme ilustra a Figura 16. Nela, ilustra-se que a combinação do serviço A, do serviço B e mais do serviço C geram um novo serviço, o *serviço ABC*. É este *serviço ABC* que controla a ordem de execução e invocação dos demais serviços presentes na orquestração.

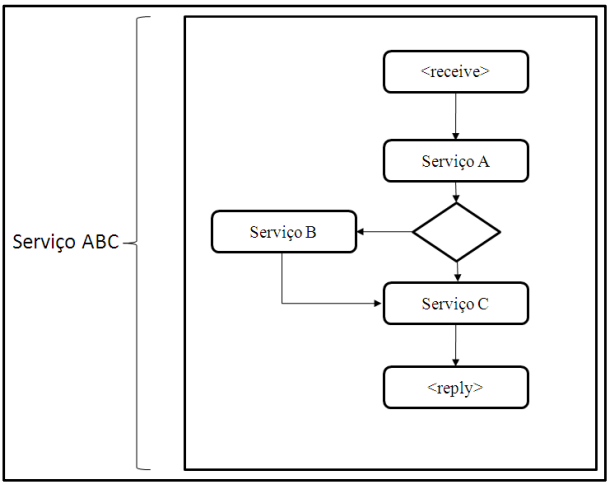


Figura 16 – Exemplo de orquestração de serviços.

A principal e mais expressiva linguagem utilizada para se fazer orquestração é a WS-BPEL (OASIS, 2010). Em essência, WS-BPEL é uma especificação XML composta de uma série de elementos, conforme mostra o Quadro 6.

Código WS-BPEL	Descrição
<process>	Define o início da orquestração de um processo ou aplicação SOA.
<partnerLink>	Define o serviço participante da composição.
<invoke> e <invoke>...<receive>	Define uma chamada síncrona ou assíncrona, respectivamente.
<variable>, <assign> e <copy>	Define um conjunto de manipuladores de variáveis resultantes de invoções de serviços.
<switch> e <flow>	Define um fluxo seqüencial e fluxo paralelo, respectivamente.

Quadro 6 - Resumo das principais primitivas WS-BPEL usadas na composição de aplicações

Além da linguagem WS-BPEL para orquestrar serviços, Milanovic e Malek (2004) comentam e comparam outras soluções para orquestração como: Redes de Petri, Π-*Calculus*, OWL-S, *Web* componentes e *Model Checking*/FSM.

Na composição baseada na coreografia de serviços, o processo de composição ocorre a partir da definição do comportamento de cada serviço e da interação (troca de mensagens) entre eles na composição. Em resumo, na coreografia os serviços conhecem e executam suas funções, cujo objetivo final está em auxiliar o fluxo das operações da composição.

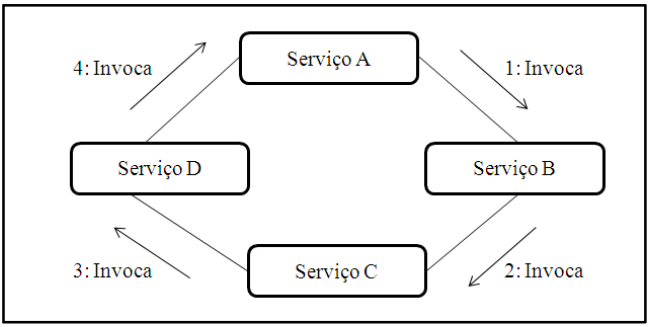


Figura 17 - Exemplo de coreografia de serviços.

Na Figura 17, tem-se um exemplo que ilustra a troca de mensagens entre os serviços A, B, C e D. Na coreografia, o Serviço A sabe que precisa fazer uma chamada ao Serviço B. O Serviço B sabe também que necessita invocar o Serviço C, que por sua vez sabe também que precisa invocar o Serviço D. Portanto, este conjunto de chamadas aos serviços objetiva garantir que um dado processo de negócio seja executado. Em relação à linguagem para permitir construção de aplicações por coreografia destaca-se uma primeira proposta desenvolvida pelo (W3C, 2005), a WS-CDL (*Web Services Choreography Description Language*), ainda em nível de recomendação. Além disso, há diversos ambientes computacionais comerciais que suportam orquestrações de serviços, como: *NetBeans*, *WebShpere* da IBM, *Microsoft BizTalk* e outras.

Tradicionalmente, tanto na composição por orquestração como na coreografia, o processo de composição é manual, necessitando que um usuário especifique, busque, selecione e associe serviços. Este processo é muito dispendioso e trabalhoso, principlamente quando o número de

serviços com características similares é grande. Por outro lado, a composição pode ocorrer de forma híbrida, ou seja, semi-automática. Nela, basicamente há uma divisão de responsabilidades entre o ambiente de composição e o usuário. Em outro extremo, a composição pode ser totalmente automática (sem a intervenção humana). Neste caso, a responsabilidade pelo processo de composição passa a ser de agentes de softwares, que providos de técnicas semânticas, especificam serviços, realizam escolhas e decidem pela composição mais adequada para o momento.

Ainda no lado direito da Figura 15, tem-se a composição de serviços dinâmica cuja base está firmada no uso da semântica para automatizar a tarefa de composição (BUCCHIARONE; GNESI, 2006). Neste sentido, a literatura apresenta diversas opções para especificar e associar semântica aos serviços *web*, como por exemplo: OWL-S e *Web Service Choreography Interface* (WSCI) (W3C, 2002).

2.5.4 Considerações

Esta seção tratou de apresentar uma introdução ao processo de composição de serviços, definindo e enfatizando os dois principais e mais usados modelos para composição de serviços: a orquestração e a coreografia.

Embora o foco maior deste trabalho não esteja direcionado para a geração da melhor composição de serviços possível, o conhecimento e o entendimento dos modelos de composição são importantes. Primeiro, porque o resultado da descoberta proposta neste trabalho gera uma composição de serviços que é executada em um ambiente de execução (ver Capítulo 4 mais maiores detalhes). Segundo, porque a composição da aplicação em si demonstra a integração BPM&SOA pretendida neste trabalho.

2.6 Software como um Serviço[8] (SaaS)

Esta seção apresenta o modelo SaaS, uma tendência que em muitos casos já é realidade como modelo de comercialização de software. A seção inicia com a definição de SaaS, mostra os principais

[8] Do inglês *Software as a Service*.

benefícios deste modelo e termina com algumas considerações em relação ao modelo SaaS.

2.6.1 Definição

SaaS é uma solução recente que consiste em provedores que distribuem software, na forma de serviços *web*, pela Internet. Interessados nesses serviços, os acessam sob demanda e pagam pelo uso, apesar de existir provedores que não cobram taxa alguma (GREER JR, 2009).

Semelhante ao modelo SaaS, tem-se o modelo *Application Service Provider* (ASP), no qual provedores fornecem aplicações completas, baseadas no modelo cliente-servidor, fornecidas sob a luz de um contrato de utilização. A grande limitação desse modelo está na falta de flexibilidade advinda das aplicações monolíticas e altamente acopladas, pois o processamento lógico e físico é centralizado, o que dificulta a evolução da aplicação (CANCIAN, 2009).

O modelo SaaS baseia-se na estratégia cliente-servidor, contando também com o uso de padrões e a distribuição fracamente desacoplada de serviços para fornecer software.

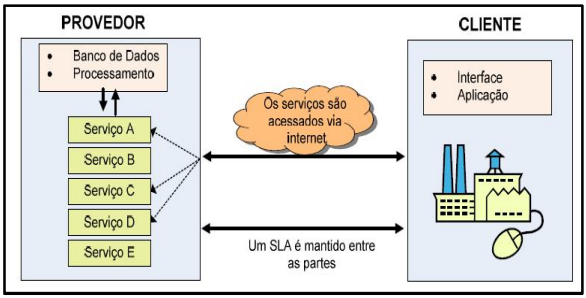


Figura 18 - Visão SaaS.
Fonte: (CANCIAN, 2009).

A Figura 18 ilustra o funcionamento do modelo SaaS. Nela, provedores criam e disponibilizam serviços através da Internet. Clientes descobrem serviços e negociam com o respectivo provedor o uso dos mesmos. Finalizada a negociação, um acordo de nível de serviço SLA (*Service Level Agreement*) é gerado entre as partes envolvidas (provedor e cliente), detalhando responsabilidades e direitos, bem como formas de acesso, disponibilidade e outros aspectos relacionados ao uso do serviço.

2.6.2 Benefícios

A maneira de comercializar software como um serviço traz uma série de benefícios, principalmente para o lado consumidor do serviço, conforme vários estudos apontam:

- Usufrui da alta disponibilidade do serviço;
- Redução de custos com aquisição de licenças, além de não exigir implantação de grande infra-estrutura no local do consumidor;
- Não há custo de manutenção do serviço;
- Pode haver o uso compartilhado dos serviços por membros de uma empresa;
- Há um contrato (SLA) que registra direitos e deveres das entidades envolvidas no negócio;
- Redução da dependência em relação ao departamento de TI;
- Independência de fornecedor, que pode ser trocado a qualquer momento;
- As tecnologias padronizadas, presentes na arquitetura dos serviços *web*, permitem a integração de serviços com sistemas legados.

Por outro lado, alguns cuidados devem ser observados quando da adoção desse modelo:

- Idoneidade do provedor do serviço;
- O estabelecimento de um contrato que irá reger o funcionamento, acesso e disponibilidade entre outros itens presentes no aplicativo a ser comercializado via modelo SaaS;
- Customização é limitada em função da proposta SaaS e, portanto, encarece a aplicação ou serviço.

O Quadro 7 apresenta alguns exemplos de empresas que comercializam serviços através do modelo SaaS.

Empresa	Descrição do Serviço	Site
Google	Disponibiliza aplicativos para edição de documentos, planilhas e apresentações de forma gratuita.	docs.google.com/
Zoho Office	Oferece uma gama de serviços voltados a aplicações colaborativas e de negócios.	www.zoho.com/
Salesforce	Fornece Sistemas para Gerenciamento de Relacionamento com Clientes (CRM)	www.salesforce.com

Quadro 7 - Exemplos de SaaS.

O Quadro 8 mostra o que muda com a adoção do modelo SaaS nas empresas.

Forma	Tradicional	Intermediário	SaaS
Pagamento	Pagamento de licenças	Usualmente com assinatura com taxa fixa	Pago por uso
Tipo de acesso	Independente de demanda	Independente de demanda	Sob-demanda
Gestão	TI da empresa	ASP (*Application Service Provider)* & SLA	*Services Providers* & SLA
Deployment	*Data center* da empresa ou terceirizado	*Data center* externo centralizado	*Data center* externo Distribuído

Quadro 8 - Mudanças em função da adoção do modelo SaaS.

2.6.3 Considerações

Esta seção apresentou a definição de SaaS, uma visão geral do seu funcionamento, bem como benefícios e aspectos importantes no momento de adotar uma solução SaaS.

O uso do modelo SaaS neste trabalho será subjacente à proposta deste trabalho. Embora não se pretenda investigá-lo, o seu uso é um importante elemento de agregação de valor à proposta.

De um lado, sustenta modelos de negócios associados aos modernos ambientes BPM&SOA. De outro lado, incrementa os modelos tradicionais de descoberta de serviços, que não tem se preocupado com a viabilidade econômica dos serviços encontrados. Isto é relevante à medida que há uma tendência de existirem inúmeros provedores de serviços distribuídos pela Internet, muitos dos quais oferecendo serviços funcionalmente equivalentes com preços diferenciados. Neste sentido, o custo acaba se tornando um item a mais de QoS a ser observado quando da seleção de um serviço.

É importante observar que os critérios de seleção poderão variar de caso a caso, ou seja, não necessariamente o serviço de menor preço será sempre o escolhido. Consoante aos objetivos da aplicação-cliente sendo composta e os demais critérios de QoS desejados (pelo cliente) e oferecidos (pelo provedor), um outro serviço mais "caro" pode ser escolhido como sendo o mais adequado.

2.7 **Recuperação da Informação**

Esta seção apresenta o conceito de Sistemas de Recuperação da Informação, descreve seus elementos fundamentais e aborda os indicadores de avaliação tradicionais usados na área de Recuperação da Informação. A seção é importante pelo fato do modelo proposto (descrito no Capítulo 4) definir e usar um mecanismo de recuperação de serviços *web* que necessita ser avaliado.

2.7.1 Sistemas de Recuperação da Informação

Um sistema de Recuperação de Informação (RI) é capaz de armazenar, recuperar e manter informações. Informações neste contexto podem ser de vários tipos, tais como datas, números, imagens, áudio, vídeo ou multimídia (KOWALSKI, 1997).

Os sistemas de RI consistem de um software que facilita um usuário a encontrar informações que ele precisa (KOWALSKI, 1997). Esses sistemas são tradicionalmente classificados em dois grandes grupos: aqueles que atuam como *web search* (sistemas que fazem pesquisa em inúmeros documentos armazenados em computadores servidores espalhados pela Internet), e os sistemas de recuperação específicos, projetados para atender uma determinada necessidade

(sistemas de busca e pesquisa presentes nos sistemas operacionais) (MANNING; RAGHAVAN; SCHÜTZE, 2008).

Independente do tipo, sistemas de recuperação de informação são organizados adotando uma arquitetura conceitual clássica, composta por três elementos fundamentais, conforme ilustra a Figura 19.

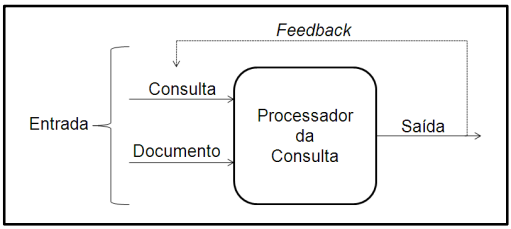


Figura 19 - Componentes essenciais de um Sistema de RI.
Fonte: (RIJSBERGEN, 1979).

Os componentes conceituais são as entradas, o processo e saídas. O primeiro elemento conceitual é a entrada. Nela, destaca-se a necessidade de usar uma representação da informação para que a mesma possa ser recuperada, bem como uma descrição dos desejos do usuário.

O segundo elemento, o processador, representa o motor que executa a consulta. Seu objetivo é interpretar uma consulta e gerar saídas que atendam os desejos informados através de entradas pelo usuário. O terceiro elemento, a saída, contém um conjunto de respostas a serem apresentadas para que o usuário possa avaliá-las frente aos seus objetivos. Em não concordando com as respostas, um usuário usa como base as saídas obtidas (como *feedback*) para realizar novas consultas (RIJSBERGEN, 1979).

Na arquitetura dos sistemas de recuperação de informação, comparando-se os seus elementos conceituais com os elementos conceituais que caracterizam o modelo proposto (ver Capítulo 4, Seção 4.2.2 para detalhes) tem-se: entradas (o conjunto de desejos do usuário projetista da aplicação), o processador de consultas que corresponde ao mecanismo de descoberta proposto e saídas que equivalem aos serviços *web* descobertos. Desta forma, o mecanismo de descoberta proposto (ver Capítulo 4) é caracterizado como um sistema de RI, sendo sua qualidade determinada através da análise dos valores dos indicadores da área de recuperação da informação. Esses indicadores são descritos brevemente na próxima seção.

2.7.2 Indicadores de Avaliação para Sistemas de RI

A literatura que trata da avaliação dos sistemas de RI apresenta diversos indicadores. Segundo Rijsbergen (1979) *apud* Cleverdon e Keen (1966) e Storb (1997) *apud* Salton (1975), Salton e McGill (1983) e Lancaster (1987) seis elementos são cruciais para avaliar um sistema de recuperação de informação:

- Precisão (*precision*) é capacidade de recuperar documentos relevantes sobre o total de recuperados;
- Cobertura (*recall*) é a capacidade de extrair documentos relevantes sobre o total de relevantes;
- Esforço necessário para formular consultas, conduzir a pesquisa e examinar o resultado;
- Tempo de resposta é o tempo entre a entrega da consulta para o mecanismo e a apresentação do resultado;
- Forma de apresentação do resultado é o que influencia a habilidade do usuário em utilizar informações recuperadas;
- Completude da coleção é uma medida que significa o quanto uma coleção possui assuntos relevantes.

Considerando o modelo proposto, apresentado no Capítulo 4, o *esforço* necessário para formular consultas, conduzir pesquisas e examinar resultados só ocorre na fase de projeto de aplicações SOA, já que na fase de execução, estas atividades são executadas automaticamente pelo ambiente de execução. Na fase de projeto de aplicações SOA o esforço para formular consultas, conduzir pesquisas e examinar resultados pode ser medido de várias maneiras. Um deles é o registro do tempo para realização destas tarefas. Outra é aplicar questionários a um conjunto de projetistas de aplicação.

Apesar da reconhecida importância de se conhecer o esforço, a avaliação desta métrica não é considerada neste trabalho. O principal motivo tem a ver com a dificuldade de implantar o modelo proposto em diversas organizações, condição necessária para avaliar o esforço na tarefa de formular consultas, conduzir pesquisas e examinar resultados.

O critério denominado *tempo de resposta* pode ser medido diretamente através do tempo de processamento da descoberta, ou seja, a partir do instante em que o mecanismo de descoberta é acionado até o momento que mostra resultados (STORB, 1997).

Quanto ao critério denominado *forma de apresentar resultados*, cabe salientar que os resultados da descoberta são mostrados apenas na

fase de projeto de aplicações (ver Capítulo 4). Nesta fase, apresenta-se uma lista de serviços candidatos (lista composta somente de serviços que satisfazem os desejos da aplicação).

O critério designado por *completude* mede o total de documentos relevantes em uma coleção. Em se tratando de se obter a efetividade de sistemas de RI, duas medidas são tradicionalmente utilizadas: precisão (*precision*) e cobertura (*recall*) (MANNING; RAGHAVAN; SCHÜTZE, 2008). A precisão evidencia a capacidade de um sistema de RI recuperar informações corretas que o usuário deseja. A cobertura mostra a abrangência do mecanismo de busca frente ao conjunto de informações disponíveis para consulta. Quanto maior for a cobertura atingida pelo mecanismo, menor é a chance de deixar algum melhor serviço fora do resultado da descoberta.

Assim, diante do exposto, a análise do resultado obtido pela combinação da precisão e cobertura diz o quão efetivo um sistema de RI é e apóia a decisão de usar ou não um modelo de recuperação da informação (STORB, 1997).

2.7.2.1 Cálculo da Precisão e Cobertura

Para entender como funcionam as métricas de precisão e cobertura é necessário compreender o significado da palavra relevância no contexto da avaliação dos sistemas de recuperação de informação. Relevância é um conceito subjetivo podendo ter significado diferente em contextos (RIJSBERGEN, 1979). Já Storb (1997) *apud* Salton e McGill (1983) comenta que ao menos duas definições relacionadas à relevância são possíveis. Uma objetiva, que considera a relevância uma característica lógica entre consulta e documento, podendo ser mensurada considerando se o documento trata ou não do assunto da consulta. Outra mais subjetiva, que não considera apenas o conteúdo do documento, mas o conhecimento do usuário no momento da pesquisa. É mais fácil determinar a relevância através da definição objetiva, uma definição lógica que corresponde a determinar se um documento é ou não relevante (STORB, 1997).

Com base na definição de relevância, define-se cobertura (*recall*) como a proporção entre os documentos relevantes e recuperados sobre os documentos relevantes na base de dados (RIJSBERGEN, 1979) e (MANNING; RAGHAVAN; SCHÜTZE, 2008).

$$Cobertura = \frac{Número\ de\ itens\ recuperados\ e\ relevantes}{Número\ de\ itens\ relevantes\ na\ base\ de\ documentos}$$

E precisão como a proporção entre os documentos relevantes e recuperados e os documentos recuperados.

$$Precisão = \frac{Número\ de\ itens\ recuperados\ e\ relevantes}{Número\ total\ de\ itens\ recuperados}$$

2.7.2.2 Relação entre Precisão e Cobertura

A determinação da precisão e da cobertura para uma dada consulta é feita percorrendo-se a lista de documentos resultantes da consulta, considerando os documentos recuperados na sub-lista percorrida (STORB, 1997). O Quadro 9 apresenta valores para cobertura e precisão considerando 3 documentos relevantes, sendo os valores calculados para precisão e cobertura gerados a partir dos valores presentes na lista resposta. Por exemplo, para determinar o valor da precisão e da cobertura após o documento 67 recuperado (ver linha 7 do Quadro 9), consideram-se 3 documentos relevantes em 7 recuperados, ou seja: precisão = 3/7 = 0,43 e cobertura = 3/3 = 1,0.

N	nº do documento	Relevante	Precisão	Cobertura
1	588	✓	1,00	0,33
2	34		0,50	0,33
3	2345		0,33	0,33
4	23	✓	0,50	0,67
5	1		0,40	0,67
6	45		0,33	0,67
7	67	✓	0,43	1,00
8	89		0,38	1,00
9	97		0,33	1,00
10	87		0,30	1,00

Quadro 9 - Cobertura e precisão após de n documentos recuperados.
Fonte: Storb (1979) *apud* Salton (1983).

Para determinar a relação entre cobertura e precisão, devem-se executar diversas consultas. Os resultados obtidos são classificados de acordo com valores de cobertura no intervalo [0,1] e usados para calcular a precisão destas consultas em cada nível de cobertura.

2.7.3 Considerações

Esta seção apresentou uma introdução aos sistemas de recuperação de informação. O conceito e os componentes essenciais que formam um sistema de RI foram abordados, bem como os principais indicadores usados no processo de avaliação de sistemas de RI foram apresentadas e descritas.

Tendo em vista que o mecanismo de descoberta é caracterizado como um sistema de RI, indicadores são aplicados para determinar a qualidade do mesmo. Esta qualidade pode ser vista de diferentes ângulos, exigindo a combinação ou aplicação de um conjunto de indicadores.

Em razão das características previstas para o modelo de descoberta proposto (ver Capítulo 4, Seção 4.1), dois indicadores (precisão e cobertura) foram definidos e serão combinados para avaliar o mecanismo de descoberta proposto.

Capítulo 3

Descoberta de Serviços

Este capítulo apresenta as principais e mais importantes iniciativas que abordam a descoberta de serviços *web*. Inicialmente, a Seção 3.1 apresenta a definição de descoberta adotada neste trabalho. Na Seção 3.2 são descritas as formas de descobertas de serviços. A Seção 3.3 descreve os trabalhos que envolvem a descoberta de serviços. A Seção 3.4 registra considerações sobre o estado da arte em descoberta de serviços.

3.1 **Definição**

Uma das primeiras definições para descoberta de serviços foi proposta por Tsalgatidou e Pilioura (2002). Segundo os autores, descoberta é o processo de encontrar um serviço apropriado para um cliente através de um mecanismo de descoberta.

O'Sullivan, Edmond e Ter Hofstede (2002) destacam a necessidade de encontrar e selecionar serviços e atribuem como definição para descoberta a ação de encontrar serviços candidatos. Já para Zhou *et al.* (2003) descoberta é habilidade que se tem para descobrir serviços disponíveis.

Além disso, descoberta é uma das principais tarefas que fazem parte da composição de serviços cujo objetivo é selecionar serviços

adequados em função das necessidades e preferências definidas por um usuário (WANG *et al.*, 2006a).

Stollberg, Hepp e Hoffmann (2007) apontam que a descoberta de serviços é um fator crítico dentro da arquitetura orientada a serviços e definem descoberta como o processo de encontrar serviços adequados para uma determinada tarefa.

Li *et al.* (2009) comentam que descoberta de serviços é o processo de identificar serviços que atendem completamente os requisitos dos usuários.

Para o grupo de trabalho responsável pela especificação da *Web Services Architecture* (WSA), ligado ao W3C, descoberta de serviços se caracteriza como sendo o processo de descobrir serviços através da ação de localizar a descrição de um serviço *web* que atende a um conjunto de critérios (W3C, 2004).

Ao longo deste trabalho, a definição para descoberta de serviços proposta pelo grupo de trabalho da WSA será adotada, porque é ampla e contempla, em essência, as diversas definições propostas para descoberta de serviços.

3.2 **Formas**

Fundamentalmente a descoberta ocorre de forma manual, semi-automática (ou interativa) ou automática. Na manual, um usuário (uma entidade que deseja encontrar serviços) usa um mecanismo de descoberta para localizar e selecionar descrições de serviços que atendem suas necessidades (W3C, 2004). Nesta forma, fica a cargo do usuário informar ao sistema os seus desejos, os critérios a serem usados na escolha do serviço mais adequado e instruir como o mecanismo de descoberta deve proceder para realizar a descoberta. O inconveniente desta abordagem está no tempo e no esforço despendido pelo usuário no processo e na necessidade de conhecimento e controle do usuário sobre o processo de descoberta como um todo. Considerando que o número de serviços similares sendo ofertados tende a crescer, esta abordagem tende também a exigir mais do usuário (CARENINI *et al.*, 2008a) e (ELGAZZAR; HASSAN; MARTIN, 2010).

Na descoberta semi-automática (cuja base é a participação humana no processo de descoberta), o mecanismo auxilia usuários (guia-os) a encontrarem serviços mais adequados em função das suas

necessidades. Esta assistência é representada através de um conjunto de interfaces que permitem ao usuário interagir e avaliar resultados de descobertas efetuadas, podendo também envolver o uso de ontologias para minimizar problemas de interoperabilidade e para possibilitar a automatização de algumas tarefas e o uso de filtros que permitem obter respostas mais condizentes com os desejos de um usuário (ALBRESHNE; PASQUIER, 2010) (PÉREZ *et al.*, 2010).

Na descoberta automática (caracterizada pela não intervenção humana), um software (tipicamente um agente) executa esta tarefa. Neste caso, o uso de semântica associada aos serviços é uma técnica usada para permitir que máquinas possam realizar escolhas por elas mesmas, minimizando tempo, esforço e intervenção humana no processo (W3C, 2004). Nesta abordagem, o controle fica sob a responsabilidade do mecanismo de descoberta que encontra, filtra e vincula serviços de forma automática.

Além da descoberta manual, semi-automática e automática, a descoberta pode ocorrer de forma estática ou dinâmica.

Na forma estática ou tradicional, serviços são escolhidos e vinculados (*binding*) às aplicações. Nessa forma o vínculo permanece perene, mesmo quando a aplicação é executada. Portanto, nessa forma, deve-se garantir que o endereço do serviço não mude, ficando disponível e acessível quando a aplicação SOA for executada.

Na descoberta dinâmica (IBM, 2001) e (W3C, 2007a) serviços *web* são selecionados e imediatamente vinculados às aplicações. Neste tipo de descoberta, um serviço pode ser substituído por outro, a qualquer tempo, de acordo com demandas da aplicação ou outra. A possibilidade de substituir serviços a qualquer tempo é uma vantagem significativa desta abordagem, principalmente quando se considera um cenário cuja possibilidade de mudança nos requisitos das aplicações é frequente.

Um aspecto importante relacionado tanto à descoberta estática como à dinâmica é que ambas não garantem a execução do serviço, pois o serviço escolhido pode estar inacessível no exato momento do seu uso. Entretanto, a dinâmica se destaca porque serviços são selecionados e vinculados às aplicações em tempo de execução de aplicações.

Além disso, a descoberta pode ocorrer em tempo de projeto ou em tempo de execução de aplicações (KOVÁCS; MICSIK; PALLINGER, 2006) e (LI *et al.*, 2009). Tempo de projeto corresponde a fase de especificação de aplicações ou processos e tempo de execução contempla o conjunto de atividades para seleção, invocação e execução de serviços (KOVÁCS; MICSIK; PALLINGER, 2006).

3.3 **Estado da Arte**

3.3.1 Introdução

O ponto de partida para a apresentação do estado da arte se deu com a leitura e análise de sete trabalhos.

O trabalho de Bernstein e Klein (2002), a proposta de O'Sullivan, Edmond e Ter Hofstede (2002), a pesquisa feita por Tsalgatidou e Pilioura (2002), o trabalho apresentado por Pilioura, Tsalgatidou e Batsakis (2003), a análise do relatório técnico que detalha a Arquitetura dos Serviços *web* (WSA) do W3C (W3C, 2004), a análise do trabalho de proposto por Garofalakis *et al.* (2006) e, como forma de complementar e dar maior solidez ao estado da arte, o trabalho *Service-Oriented Computing: State of the art Research Challenges*, proposto por Papazoglou *et al.* (2007) foi igualmente analisado.

A opção pela análise inicial destes trabalhos foi baseada em dois pontos. O primeiro relacionado aos principais e diferentes desdobramentos que ocorreram na descoberta de serviços ao longo dos últimos anos. O segundo, pelos trabalhos formarem uma base para o entendimento global e gradual necessário para a construção do estado da arte em descoberta de serviços.

A principal contribuição advinda da análise dos trabalhos mencionados acima é a percepção da complexidade da descoberta, seu enorme escopo e a influência que a mesma exerce nas demais atividades ligadas ao ciclo de desenvolvimento de serviços, como por exemplo: a publicação e a invocação de serviços.

3.3.2 Trabalhos Relevantes

Em razão da grande quantidade de iniciativas que focam a descoberta de serviços e para facilitar o entendimento das mesmas, três grupos foram criados. A inspiração para a definição dos grupos nasceu da análise de dois trabalhos. O primeiro descreve e comenta sobre os desafios associados à computação orientada a serviços (PAPAZOGLOU *et al.*, 2007). O segundo apresenta um *survey* relacionando e categorizando as mais importantes estratégias usadas no processo de descoberta de serviços (GAROFALAKIS *et al.*, 2006).

Um primeiro grupo abriga iniciativas que visam melhorar a descrição dos serviços e, consequentemente, aumentar a precisão proporcionada pelo mecanismo de descoberta.

O segundo grupo foi organizado com trabalhos preocupados em aumentar a cobertura, a escalabilidade dos repositórios de serviços e melhorar o desempenho da descoberta.

O terceiro grupo foi formado por trabalhos que visam melhorar o processo de seleção de serviços. A seleção de serviços é considerada uma etapa posterior à descoberta por muitos autores (KRITIKOS; PLEXOUSAKIS, 2008), (CARENINI *et al.*, 2008a) e (LI *et al.*, 2009), cujo objetivo é realizar uma classificação (um *ranking*) dos serviços descobertos.

3.3.2.1 Iniciativas que visam melhorar a descrição dos Serviços

Uma das primeiras iniciativas a perceber a dificuldade de representar informação sobre serviços foi proposta por ShaikhAli *et al.* (2003). Os autores decidiram melhorar a descrição Dos serviços criando uma extensão do padrão UDDI para suportar atributos associados aos serviços como custo, desempenho e outros. Além disso, os autores estendem uma série de métodos padrões UDDI para suportar diferentes formas e opções de pesquisas. Embora o trabalho tenha preocupação em melhorar a descrição dos serviços (para alcançar uma maior precisão), o fato de alterar a estrutura padrão das UDDIs restringe o uso da estratégia, porque limita a solução para uso em casos particulares, e a torna mais difícil de ser integrada a outros sistemas.

Zhou *et al*. (2003) buscaram uma solução para dois problemas relacionados à descoberta de serviços. O primeiro reflete a necessidade de representar e medir atributos de QoS. O segundo afeta a escalabilidade das UDDIs, pois o número de serviços a serem publicados e a quantidade de descobertas tende a crescer nos próximos anos. O trabalho usa QoS para melhorar a descrição dos serviços, mas seu maior foco está na cobertura e escalabilidade de UDDIs, sendo descrito e analisado na Seção 3.3.2.2.

Na proposta de Ran (2003), os aspectos funcionais e não-funcionais dos serviços são contemplados. Para suportá-los, uma arquitetura diferenciada foi definida através da incorporação de um elemento chamado *Web Service QoS Certifier* à arquitetura padrão dos serviços *web*. O processo de descoberta ocorre da seguinte forma: o provedor, no momento da publicação, entra em contato com *Web Service QoS Certifier*. O *Web Service QoS Certifier* verifica e certifica

informações de QoS que o provedor quer associar ao serviço a ser publicado. Essas informações de QoS são registradas em um repositório de informações de serviços padrão UDDI. A análise do trabalho aponta que a descrição dos aspectos não-funcionais associados aos serviços é bastante rica, podendo servir de base para novas definições de QoS, bem como para definir os elementos de QoS usados no desenvolvimento de contratos de uso de serviços (SLAs). O problema é que a definição proposta não é considerada um padrão ou recomendação, dificultando seu uso no âmbito deste trabalho de pesquisa, pois o uso intenso de padrões é um dos principais e mais importantes elementos para obter uma mais fácil e ágil integração entre o nível de negócios e o nível dos sistemas.

Sycara *et al.* (2003), inspirados em descobertas mais precisas e na necessidade de tornar automática a descoberta, a composição e a invocação de serviços, propuseram a combinação de serviços *web* com *web* semântica. Os autores usam DAML-S. Ela especifica uma ontologia de nível superior[9] para descrever serviços, sendo composta por três elementos: *Service Profile* (usado para descrever as capacidades de um serviço), *Process Model* (elemento que permite detalhar o fluxo de execução de processos, ou seja, como um serviço trabalha) e o *Grounding* (que tem como papel informar como um serviço pode ser acessado). Um elemento chave no trabalho de Sycara *et al.* (2003) é DAML-S/UDDI que realiza mapeamentos entre informações presentes no *service profile* de um serviço para UDDI. Estas informações são guardadas através de *tModels elements* UDDI. Outro elemento importante é DAML-S *Matching Engine*, é responsável pela tarefa de *matching* entre o desejo e o que se tem, em termos de descrição de serviços. Uma contribuição importante do trabalho refere-se ao desenvolvimento de uma máquina virtual para execução de aplicações, a DAML-S *Virtual Machine*. A máquina usa informações presentes no elemento DAML-S *Process*, invoca e executa serviços.

A proposta é bastante relevante, principalmente em termos de abrangência. Ela envolve desde a concepção, descoberta, composição até a execução de serviços. Embora a proposta esteja alinhada aos

[9] Ontologia de alto nível ou de domínio é concebida para ser compartilhada por grandes comunidades de usuários.

objetivos do modelo proposto neste trabalho, um fato que diferencia o modelo proposto é o uso de um padrão (UBL) para especificar e descrever processos de negócios (através de uma ontologia), tanto na publicação dos serviços quanto na descoberta deles. O ganho desta estratégia está em criar um vocabulário comum entre clientes e provedores de serviços de forma a minimizar problemas de interoperabilidade, possibilitando uma mais ágil integração de processos e sistemas.

Sivashanmugam, Verma e Sheth (2004) apresentam uma proposta que envolve a organização de registros em uma federação. A justificativa mais expressiva dos autores para uso de federações é baseada no aumento da cobertura da descoberta, o que proporciona ao mecanismo de busca usar outros registros para encontrar serviços com características similares. Os autores usam ontologias associadas aos serviços para melhorar a descrição dos mesmos. Em razão do foco do trabalho estar em sua maior parte na cobertura e escalabilidade de UDDIs, o trabalho é descrito em mais detalhes na Seção 3.3.2.2.

Maximilien e Singh (2004) comentam a necessidade de considerar aspectos de qualidade na seleção de serviços e ratificam a dificuldade que os protocolos padrão para desenvolvimento de serviços possuem em representar estes aspectos. Com este objetivo, os autores apresentam um *framework* WSFA (*Web Services Agent Framework*), que estende o padrão de desenvolvimento de serviços, adicionando agentes de software. Além da proposta do *framework*, um destaque do trabalho é a ontologia para qualidade de serviço proposta pelos autores. A ontologia desenvolvida é bastante completa, possuindo vários níveis de detalhamento, representando tanto aspectos qualitativos, por exemplo: maior disponibilidade, maior desempenho entre outros, como quantitativos associados aos serviços.

Embora a ontologia desenvolvida para QoS seja bastante completa, a mesma não é um padrão ou recomendação. Isto, novamente, não vai ao encontro de um dos principais e mais importantes diferenciais do modelo proposto neste trabalho que é o uso intenso de padrões e recomendações para propiciar uma maior agilidade na integração entre os ambientes BPM&SOA.

O trabalho proposto por Kokash (2005) prevê a criação de um *framework* para descoberta de serviços. Nesse *framework*, cada cliente tem um agente dedicado (chamado de *personal agent*), cujo objetivo maior é realizar pedidos, enviar respostas e realizar vinculação de serviços. O *personal agent* aceita requisições e encaminha estas para o *matching agent*. O *matching agent*, outro elemento presente na

arquitetura do *framework*, gerencia o repositório de informações de serviços e também a descoberta de serviços. Assim, cabe ao *matching agent* realizar a descoberta de serviços de acordo com o pedido feito pelo *personal agent* de um cliente. Além disso, o *matching agent* mantêm registros sobre requisições efetuadas, fornecendo aos *personals agent* informações sobre serviços ou descobertas realizadas que envolveram serviços ou requisições similares. Isso permite aos *personals agents* solicitarem sugestões em relação aos serviços requisitados anteriormente.

Além de usar ontologia e QoS para descrever serviços, o destaque do trabalho está na utilização de descoberta a partir de agentes e no uso de descoberta antigas para subsidiar ou facilitar novas descobertas. Em relação aos agentes, eles podem ser trabalhados em uma perspectiva que os torne "inteligentes", para que possam auxiliar clientes no processo de descoberta, seja no momento de construir a expressão da descoberta seja para classificar ou escolher serviços. Em razão disso, abre-se uma oportunidade para criar um mecanismo de descoberta assistido, ou seja, com tarefas automatizadas (classificação dos serviços descobertos), conforme o que se prevê no modelo de descoberta proposto neste trabalho.

Pathak *et al.* (2005) apresentam um *framework* para descoberta de serviços centrado no *matching* semântico funcional e na seleção de serviços candidatos. Provedores, quando desejam publicar serviços, descrevem capacidades dos mesmos através da OWL-S (*Semantic Markup for Web Services*). Da mesma forma, clientes descrevem seus desejos usando API disponível no *framework* utilizando OWL-S para informar capacidades desejadas para um serviço. No momento da descoberta, um *matching engine* realiza o mapeamento de domínio entre as ontologias informadas pelo provedor e pelo cliente. O resultado deste processo é um conjunto composto de serviços candidatos, classificados em função do grau de *matching* (perfeito ou total, parcial, *plug-in*, ou seja, o serviço descoberto atende o especificado e tem mais capacidades, sem correspondência) obtido em relação ao mapeamento das ontologias. Os autores ainda prevêem uma seleção em função de atributos de QoS associados aos serviços.

Em essência, o trabalho proposto ratifica a importância do uso de semântica e QoS na descoberta e seleção de serviços. Cabe destacar a ontologia de QoS proposta, a qual apresenta aspectos de QoS

dependentes (preço de comercialização de um serviço) e independentes (disponibilidade, desempenho) de um domínio. Isto mostra a complexidade de definição de aspectos de QoS dentro do processo de descoberta e cria uma nova perspectiva associada aos aspectos de QoS, a de uso de aspectos dependentes do domínio. Outro ponto de destaque é que fica a critério do cliente informar pesos ou preferências para atributos de QoS na seleção. Isto viabiliza a seleção de serviços com características similares, permitindo que o serviço mais adequado possível seja encontrado. Em relação ao modelo proposto, usa-se um conceito que é associado ao provedor do serviço, para selecionar serviços similares, sendo possível que outros critérios sejam incorporados à seleção de serviços, tais como os apresentados no trabalho descrito anterioremente.

Este trabalho é muito semelhante ao proposto em termos de abordagem geral. Ele usa uma ontologia explícita e na proposta há uma ontologia implícita, baseada no padrão UBL. Além disso, ele não se preocupa com a integração com os níveis de processos de negócios (nível BPM).

Wang *et al.* (2006a) comentam da dificuldade de descobrir serviços e apresentam uma solução para descoberta que envolve quatro características. A primeira trata da descrição de serviços em função de múltiplas dimensões. A segunda relacionada ao aprendizado em relação a descobertas antigas. A terceira envolve o tratamento de requisitos vagos ou imprecisos e a quarta aborda a composição de serviços em função de duas características: estática e dinâmica. Sobre múltiplas dimensões, os autores se referem a descrições de serviços sob três aspectos: a) funcionais; b) não-funcionais e c) propriedades econômicas (por exemplo: preço, tipo de pagamento pelo uso do serviço, confiança). Cada aspecto de um serviço é descrito formalmente em função dessas dimensões e a ontologia OWL-S é usada para representar significados relacionados aos aspectos funcionais de um serviço. Em relação ao aprendizado, em essência, cada dimensão (funcional, não-funcional e econômica) tem uma contribuição no resultado final da seleção. Para cada elemento, de cada dimensão, é atribuído um peso que varia entre 0 (para menos relevante) e 1 (para mais relevante). Aplica-se um algoritmo de aprendizagem de máquina para determinar serviços através de exemplos de *matching* anteriores ou passados. Já em relação aos aspectos relacionados à QoS qualitativos, lógica *fuzzy* é a estratégia escolhida para modelar e resolver requisições vagas ou imprecisas tais como "o mais barato serviço, com o menor tempo de resposta".

É um trabalho que se preocupa com vários problemas relacionados ao ciclo de serviços *web*: descrição, descoberta de serviços e ainda propõe uma solução para a composição de aplicações. Ele envolve diversas estratégias para contornar estes problemas tais como: representação de serviços através de diversas dimensões, além de haver a preocupação com o tratamento de aspectos de QoS vagos ou imprecisos.

O trabalho é uma inspiração importante para o modelo de descoberta proposto, pois contempla diversos elementos que permitem definir um ambiente tanto para projeto como para execução de aplicações SOA, além de trazer outra perspectiva associada aos atributos de QoS que é o uso de atributos vagos ou imprecisos. Isto pode servir caso a estratégia preliminar do modelo de descoberta proposto para selecionar serviços não se mostre adequada, principalmente em razão do crescimento do número de serviços sendo ofertados.

O trabalho proposto por Stollberg, Hepp e Hoffmann (2007) usa um elemento formal chamado de *goal templates* para descrever serviços. Os serviços são descritos através de objetivos genéricos em função do domínio ou problema que deseja resolver. A ideia da descrição dos objetivos gerais é baseada em lógica clássica de 1ª ordem. Uma descrição mais detalhada do trabalho de Stollberg, Hepp e Hoffmann (2007) está na Seção 3.3.2.2, pelo fato do mesmo enfatizar também o problema da necessidade de melhores desempenhos associadas à descoberta.

Vitvar, Zaremba e Moran (2007) mencionam a necessidade de realizar descoberta de serviços usando informações associadas aos provedores de serviços. Os autores justificam esta necessidade em razão da impossibilidade dos atuais protocolos em representar e armazenar informações que permitam criar um nível de detalhe maior em relação às capacidades de um serviço. Em termos de descoberta, é apresentado um algoritmo semântico que seleciona o serviço mais adequado em relação às capacidades descritas por um provedor e necessidades exigidas por um cliente. O algoritmo de descoberta prevê três níveis de análise. Um associado a capacidades dos serviços, as quais são contrapostas aos desejos dos clientes. Em outro nível, faz-se uma análise das entradas e saídas dos serviços para verificar se elas são compatíveis com as necessidades expressas pelo cliente. Por fim, em um terceiro nível, a análise se dá em função de informações extras obtidas junto ao

provedor de serviços. Através delas o mecanismo de descoberta chega à seleção do mais adequado serviço em tempo de execução.

O fato interessante do trabalho é que, de forma similar ao trabalho proposto, informações associadas aos provedores podem também ser usadas como critérios de seleção de serviços, quando há diversos serviços com características similares que precisam ser classificados ou organizados em razão de algum critério.

Kourtesis *et al.* (2007) motivam para a necessidade da composição de aplicações através da descoberta de serviços. O trabalho reforça o conjunto de limitações técnicas associadas aos protocolos tradicionais voltados a suportar o ciclo de vida dos serviços e, principalmente, destaca carências nas etapas de publicação e descoberta (por serem baseadas em palavras-chave, implicando em: falta de precisão, impossibilidade de selecionar serviços de acordo com um contexto). Os autores inserem semântica na descrição dos serviços para melhorar a precisão dos resultados. Falam de várias abordagens que adotam semântica e optam por uma recente, a SAWSDL (*Semantic Annotations for WSDL and XML Schema*) (W3C, 2006). Eles usam uma ontologia do próprio projeto, denominado FUSION, para estabelecer significado aos serviços. A FUSION *ontology* é extensível, serve como vocabulário comum, sendo considerado um veículo de interoperabilidade dentro do projeto. Sendo assim, a publicação requer que classes OWL sejam ligadas à ontologia FUSION e associadas aos vários elementos da WSDL de um serviço via SAWSDL. Na descoberta, o mecanismo resolve *matchings* em função dos níveis estabelecidos anteriormente para encontrar serviços.

Novamente, percebe-se o interesse de melhorar ainda mais a descrição de um serviço. O trabalho usa inserir anotações semânticas junto aos elementos da WSDL de um serviço, para posterior realização de *matchings* para descoberta de serviços. Esta abordagem é semelhante, em termos de associação de ontologias para melhorar a descrição dos serviços, ao modelo proposto neste trabalho. A principal diferença é que o modelo proposto usa e associa uma ontologia baseada em uma especificação de processos de negócios (no caso a UBL), cujo objetivo principal é definir funcionalmente serviços.

Carenini *et al.* (2008a) apontam para mecanismos de descoberta de serviços necessidade melhores e mais sofisticados. Isso em razão do crescimento do número de serviços similares e em virtude da heterogeneidade dos mesmos, principalmente entre provedores e clientes. O trabalho tem como objetivo propor um cenário usado para contextualizar e assim resolver semanticamente problemas de

heterogeneidade no que tange a descrição de atributos funcionais e não-funcionais. Uma contribuição interessante do artigo é a afirmação de que seleção de serviços é uma atividade de avaliação de serviços descobertos com objetivo de identificar aquele que melhor atende aos requisitos exigidos por um cliente em um processo de descoberta. Na visão dos autores, a seleção é um processo que pode ser considerada um enriquecimento da descoberta, implicando utilização de critérios para se chegar ao melhor serviço, ou seja, aquele que atende completamente as necessidades do cliente. O trabalho mostra e ratifica a importância da separação entre as fases de busca e da fase seleção, sendo está última uma fase posterior que usa um conjunto de critérios para selecionar o serviço mais adequado dentre um conjunto de serviços com características similares. Esta separação é importante, pois como comentado anteriormente, há um aumento do número de serviços. A separação das fases de busca e de seleção permite que uma atividade seja menos dependente da outra, o que facilita a alteração e a incorporação de novos critérios de seleção, por exemplo.

Chaari *et al.* (2008) comentam que o processo de seleção de serviços é altamente dependente de decisões humanas e que serviços são encontrados basicamente em função dos requisitos funcionais associados aos mesmos. Os autores comentam da necessidade de melhorar ou tornar mais rica a descrição de serviços através de atributos de qualidade, já que aspectos de qualidade podem estabelecer diferenças entre serviços com funcionalidades similares. Com a intenção de gerenciar atributos de qualidade, os autores adotaram o *WS-Policy Framework*, embora este *framework* possua deficiências para representar aspectos não-funcionais em nível semântico. Diante dessa necessidade, os autores propuseram uma extensão para o *Framework* através do uso de ontologias. Desta forma, foi possível melhorar a riqueza na descrição de aspectos de QoS e a precisão da descoberta.

O trabalho se constitui em mais uma opção para melhorar a descrição dos serviços através do uso de semântica associadas aos atributos de qualidade. No tocante ao modelo de descoberta proposto, tanto o provedor de serviços como o cliente usam uma mesma ontologia de QoS. Isto evita que problemas de interoperabilidade semântica presentes em definições diferentes de QoS possam afetar a descoberta e, consequentemente, gera este vocabulário comum em termos de QoS, facilitando e tornando mais ágil a integração entre BPM&SOA.

A motivação maior para o trabalho de Kritikos e Plexousakis (2008) está na carência técnica do padrão UDDI. A proposta dos autores recai sobre os atributos de QoS. Os autores propõem criar uma extensão da OWL-S (a OWL-Q) para permitir que atributos de QoS possam ser representados em um nível semântico e, com isso, se possa ganhar em precisão na descoberta de serviços. Os autores listam um conjunto de requisitos que devem ser seguidos para descrição dos aspectos de QoS e usam esse conjunto de requisitos como base para criar a extensão OWL-Q. Uma das justificativas para usar OWL-S como alicerce para representar QoS está na possibilidade de especificar regras através do elemento *ProcessModel* da OWL. Isso permite uma melhor representação dos serviços, principalmente em termos de qualidade de serviço. A descoberta é realizada mediante uma requisição de um cliente que expressa, via CSP (*Constraint Programming*)[10], um conjunto atributos de QoS desejados para um serviço. O algoritmo de *matching* é preparado para realizar relaxamentos, no caso de retornar serviços que não atendem ou atendem parcialmente os requisitos de um cliente. Ao cliente é apresentado um conjunto de serviços e é indicado que a descoberta foi relaxada.

Um dos pontos que chama a atenção no trabalho está relacionado à linguagem formal usada para descrever desejos e o relaxamento feito para selecionar o serviço mais adequado. Em relação ao uso de uma linguagem formal, ela permite ganhos em relação à precisão, mas por outro lado necessita de um *template* para que um cliente, que não a conhece, informe seus desejos. Outro ponto de destaque é a técnica de relaxamentos automáticos. Em relação ao modelo descoberta proposto, o mesmo indica quais critérios podem ser relaxados em uma próxima descoberta, cabendo ao usuário a tarefa de analisá-los e usá-los.

Carenini *et al.* (2008b) apresentam uma solução para descoberta de serviços organizada em uma arquitetura denominada *Glue2*. *Glue2* proporciona descoberta de serviços envolvendo critérios funcionais e não funcionais, considerando aspectos semânticos associados aos serviços. Seu funcionamento está alicerçado no *framework* WSMO. O processo de descoberta é descrito em duas grandes fases. A primeira relacionada à identificação de serviços candidatos. Nesta fase um cliente descreve ou expressa capacidades funcionais e não-funcionais de um serviço, definindo seus significados através de ontologias. Isso é

[10] Um problema expresso com uma série de restrições é chamado *CSP problem*.

feito a partir de uma linguagem formal presente no *framework* WSMO, chamada de WSML. Com base na expressão da descoberta, um conjunto de serviços que expressam cenários é recuperado de um registro presente no *Glue2*. Estes cenários correspondem a descrições de serviços e ontologias associadas a eles. Este primeiro filtro forma um conjunto de serviços candidatos que, em seguida, em uma segunda etapa, são classificados usando critérios informados pelo cliente na expressão da descoberta.

Um ponto importante no trabalho é a representação de aspectos de qualidade tanto quantitativo como qualitativo (menor preço, maior disponibilidade, maior escalabilidade). Isto torna o processo de seleção mais realístico, uma vez que ele se aproxima do mundo real do desenvolvimento de aplicações baseadas em SOA. O tratamento dado a avaliação dos atributos de QoS é outro destaque. Os autores usam avaliar QoS em duas fases. Na primeira verifica-se o grau, o qual mostra se um atributo de QoS atende ou não os requisitos do cliente. Na segunda fase, verifica-se e define-se um indicador que representa a distância entre o valor do atributo de QoS em relação ao máximo valor de QoS. Embora o trabalho proponha uma interessante representação de QoS, seus atributos não são considerados um padrão, dificultando a integração de ambientes (como BPM e SOA), servindo para situações particulares.

A proliferação da *web* e a quantidade de tarefas executadas por serviços motivam para uma descoberta de serviços mais efetiva (ELGAZZAR; HASSAN; MARTIN, 2010). Neste contexto, os autores propõem melhorar a precisão da descoberta, através da extração de informações disponíveis nas WSDLs dos serviços. O objetivo é obter informações que permitam melhor descrever funcionalmente os serviços. São extraídos da WSDL aspectos que descrevem a semântica e o comportamento do serviço para formar grupos com funcionalidades similares. Os grupos são formados por um *web crawler* que, em um passo anterior à descoberta, pesquisa e analisa WSDLs para montar os grupos. Quando um usuário deseja buscar por um serviço, ele aciona um motor de busca que realiza a consulta usando como fonte os grupos já montados. Finalizada a consulta, o motor devolve e apresenta os serviços que satisfazem os objetivos do usuário. Em relação ao modelo de descoberta proposto, percebe-se uma estratégia similar, que é de facilitar a descoberta em tempo de execução de aplicações SOA, através

do uso de um *crawling* que retorna serviços candidatos em tempo de projeto, evitando que descobertas exaustivas em tempo de execução de aplicação SOA sejam realizadas. Além disso, esse trabalho não trata de QoS.

O Quadro 10 apresenta a lista de iniciativas estudadas que procuram melhorar a descrição dos serviços. A primeira coluna mostra as iniciativas. A segunda coluna está dividida de forma a contemplar as formas mais relevantes para melhorar a descrição dos serviços. A última linha apresenta e compara o modelo de descoberta proposto em relação às estratégias que visam melhorar a descrição dos serviços.

Grupo com foco na melhoria na descrição dos serviços			
Iniciativas	Estratégia adotada		
	Semântica (ontologia)	QoS	Outra
ShaikhAli *et al*. (2003)		✓	
Zhou *et al*. (2003)		✓	
Ran (2003)		✓	
Sycara *et al*. (2003)	✓		
Sivashanmugam, Verma e Sheth (2004)	✓		
Maximilien e Singh (2004)	✓	✓	
Kokash (2005)	✓	✓	
Pathak *et al*. (2005)	✓	✓	
Wang *et al*. (2006a)	✓	✓	
Stollberg, Hepp e Hoffmann (2007)			✓ (usa lógica de 1ª ordem)
Vitvar, Zaremba e Moran (2007)	✓		✓ (usa informações associadas ao provedor)
Kourtesis *et al*. (2007)	✓ (usa SAWSDL)		
Carenini *et al*. (2008a)	✓		
Chaari *et al*. (2008)	✓	✓	

Kritikos e Plexousakis (2008)	✓	✓	
Carenini *et al*. (2008b)	✓	✓	
Elgazzar, Hassan e Martin (2010)			✓ (analisa WSDLs via *web crawler* para formar grupos de serviços com funcionalidades similares)
Modelo de Descoberta Proposto	✓ (usa uma ontologia para identificar e definir a funcionalidade dos serviços desejados)	✓	

Quadro 10 - Resumo das iniciativas que desejam melhorar a descrição dos serviços para aumentar a precisão.

3.3.2.2 Iniciativas que visam aumentar a Cobertura, a Escalabilidade dos repositórios de serviços e o Desempenho da descoberta

Zhou *et al*. (2003) buscaram uma solução para dois problemas relacionados à descoberta de serviços. O primeiro reflete a necessidade de representar e medir atributos de QoS. O segundo afeta a escalabilidade dos repositórios UDDI, uma vez que o número de serviços a serem publicados e a quantidade de descobertas tende a crescer.

A base da solução proposta pelos autores está em um conjunto de registros federados, organizados em domínios como: empresas e universidades. Em cada domínio, há diversos registros padrão UDDI com suporte a descoberta de serviços. Cada domínio registra informações relacionadas à qualidade do serviço através do retorno do uso do serviço pelo cliente (*feedback requestor's*). Estas informações

são armazenadas em um banco de dados local (*caching*[11]), usados para descobertas futuras. Caso não exista retorno do uso do serviço por um cliente, um elemento chamado *Test Host* elemento é utilizado para gerar informações de qualidade sobre um serviço. A descoberta ocorre com o cliente expressando seus desejos através de uma interface *web*. Em um segundo passo, a descoberta ocorre localmente, consultando a *cache*, e se necessário, o registro local (UDDI local). Se o resultado da descoberta atender as necessidades do cliente, a mesma é concluída. Caso contrário, a descoberta passa a envolver vários domínios, os quais são organizados através do *Cooperating Server Graph* (CSG) *Model*[12]. A descoberta segue verificando capacidades dos serviços e aspectos de qualidade associados, consultando cada banco de dados local (*caching*). Por fim, o resultado da descoberta é ordenado em função dos aspectos de qualidade e apresentado ao cliente.

Um ponto de destaque do trabalho é o uso de *caching*, usadas para melhorar o tempo de resposta (eficiência) da descoberta. Um segundo ponto é a descoberta federada. Isto permite que mais serviços possam ser cobertos e verificados em relação às necessidade de um cliente, podendo ocasionar melhor qualidade em relação à resposta do processo de descoberta. Outro ponto que chama a atenção é o uso do retorno de cliente (*feedback requestor's*). Isto cria informações sobre a qualidade dos serviços bastante próximas da realidade. No entanto, percebe-se que o procedimento de medir e coletar é complexo, pois intrinsecamente deve atender a uma série de demandas, como por exemplo: quando e com qual freqüência se deve medir e coletar, quem realiza este processo, quais indicadores (aspectos de QoS) usar, como coletar estas informações e como medir aspectos qualitativos?

A estratégia de usar uma *caching* para QoS evita descobertas exaustivas, implicando melhora no desempenho geral do processo de descoberta. O modelo proposto de descoberta também adota uma estratégia semelhante. Um *crawler* é ativado para encontrar e selecionar serviços candidatos e associar esses serviços à aplicação SOA. Semelhante à abordagem proposta por Zhou *et al*. (2003), o principal objetivo é obter ganhos em relação ao tempo total da descoberta.

---

[11] Uma memória ou um banco de dados contendo informações sobre serviços usado para otimizar descobertas futuras. (ZHOU *et al.*, 2003) e (STOLLBERG; HEPP; HOFFMANN, 2007).

[12] Modelo que otimiza e gerencia ligações entre os diversos domínios.

Rompothong e Senivongse (2003) apresentam uma proposta de descoberta de serviços que usa consultar registros organizados em uma federação de repositórios. Para os autores, a cooperação entre os diversos serviços e a variedade de serviços em um ambiente organizado como uma federação pode trazer resultados mais expressivos em relação à cobertura da descoberta. Em essência, a descoberta usa, inicialmente, o registro local e segue ou é propagada para os demais registros presentes e ativos na federação. A descoberta nesse caso é uma descoberta sintática envolvendo critérios funcionais associados aos serviços. Esta iniciativa mostra como descobertas federadas podem ser projetadas. Assim é possível que clientes realizem consultas ou descobertas em diversas UDDIs de forma transparente, ou seja, como se a descoberta envolvesse apenas um registro local.

O conjunto de características apresentadas em termos de consulta federada é muito similar à proposta de Federação de Provedores adotada neste trabalho. Entretanto, a ideia de federação proposta no modelo de descoberta vai além da organização de repositórios e da consulta federada como se houvesse apenas uma única fonte e, ainda, adota um conjunto de normas usadas pelos provedores no processo de publicação de serviços, para garantir a qualidade dos serviços publicados. Além disso, há um módulo de avaliação de provedores, cujo conceito do provedor é usado para classificar serviços descobertos com características similares. Ainda, esse trabalho não considera QoS.

ShaikhAli *et al*. (2003) propõem habilitar o registro de serviços para garantir consultas persistentes. Desta maneira, é possível publicar serviços por um tempo limite, ficando assim garantido que dentro deste tempo informações associadas aos serviços não mudam. Este aspecto proporciona ao ambiente de descoberta uma maior robustez comparada a outros trabalhos, pois neste caso, além de aumentar a eficiência da descoberta, ele não necessita verificar se o serviço está ativo antes de ser invocado.

A iniciativa de dotar mecanismos de descoberta com suporte a consultas persistentes é um aspecto interessante. O modelo proposto de descoberta neste trabalho não suporta consultas persistentes, muito embora seja possível em um trabalho futuro incorporar esta funcionalidade. Entretanto, antes de implantá-la deve-se responder duas questões. A primeira delas diz respeito à quantidade de tempo que se garante a consulta persistente. A segunda, caso alguma característica do

serviço se altere no intervalo de tempo garantido pela consulta persistente, como fazer para que a aplicação ou o cliente do serviço seja avisado e tome as providências necessárias para não prejudicar a execução de aplicações SOA e a integração BPM&SOA.

Sivashanmugam, Verma e Sheth (2004) apresentam uma proposta que envolve a organização de registros em uma federação. A justificativa dos autores para uso de federações é baseada no aumento da cobertura, o que proporciona a busca em outros registros serviços com características similares. A federação é organizada em uma arquitetura de rede *Peer to Peer* (P2P), sendo os registros classificados ou estruturados em função de uma ontologia. Esta ontologia, chamada de XTRO (*Extended Registries Ontology*), permite classificar registros em função de domínios e subdomínios. Os autores definem federação como sendo uma coleção de registros autônomos, conectados e, possivelmente, diferentes entre si (por exemplo: UDDI, ebXML). Na descoberta de serviços, clientes expressam suas necessidades através de um *template* (um modelo) disponível através de uma interface. A consulta é enviada à rede P2P, que através de um elemento chamado de *operator peer*, seleciona registros e realiza a descoberta. É possível, também, realizar consultas federadas em diversas federações ou em uma, em particular. O resultado da descoberta é enviado ao cliente para avaliação.

O trabalho considera a heterogeneidade dos registros (estruturas de dados e API de acesso), a distribuição de dados (ao invés de replicação de informações em registros como ocorre na versão 3 do UDDI) e outras características que reafirmam a importância de registros estarem conectados de forma a possibilitarem maior cobertura em relação aos serviços publicados.

No que tange ao modelo de descoberta proposto, a preocupação em considerar a heterogeneidade dos registros é fato, porque abre a possibilidade de incluir outros padrões de repositórios de serviços (UDDI, ebXML), garantindo uma maior cobertura na descoberta e consequentemente que serviços melhores possam fazer parte do processo de integração BPM&SOA.

Kovács, Micsik e Pallinger (2006) propõem um mecanismo de descoberta que combina recuperação de informação sintática com

matching lógico implementado em Prolog[13]. Os autores comentam da divisão do ciclo de vida dos serviços em dois momentos importantes: tempo de projeto (que compreende as atividades necessárias para descrição de serviços) e tempo de execução (que compreende as atividades de: descoberta, seleção e execução de serviços). O mecanismo de descoberta proposto é baseado em três fases. A primeira corresponde à obtenção de serviços candidatos através do *matching* sintático usando palavras-chave. A segunda etapa realiza o *matching* lógico. Nesta etapa as descrições dos serviços e dos desejos de um cliente são mapeadas para primitivas em Prolog e a partir delas faz-se o *matching* sobre os serviços candidatos da etapa anterior. O mecanismo de descoberta proposto pelos autores possui complexidade computacional linear. Considerando a tendência do aumento do número de serviços a serem publicados e ofertados, a complexidade de ordem linear representa um bom desempenho em termos de tempo de descoberta de serviço.

Um dos destaques da proposta é a informação da complexidade linear do algoritmo proposto para descoberta. Essa informação é importante para o modelo proposto, porque são propostos dois algoritmos bastante similares que podem ser comparados através do critério complexidade computacional com esta proposta, por exemplo.

No trabalho proposto por Stollberg, Hepp e Hoffmann (2007), o desempenho computacional (tempo para encontrar o serviço mais adequado, ou seja, aquele que atenda as necessidades de um cliente) das soluções baseadas em semântica ainda não receberam atenção suficiente, uma vez que o número de serviços vem aumentando e as consequências práticas disso ainda não foram tratadas. Os autores definiram alguns parâmetros para avaliar os mecanismos de descoberta: eficiência (tempo computacional requerido para encontrar o serviço desejado); escalabilidade (habilidade de lidar com grandes quantidades de serviços em um intervalo de tempo) e estabilidade (quão baixa é a variabilidade de tempo para realizar invocações de serviços). Desta forma, os autores adaptaram o conceito *caching* no cenário de descoberta de serviços *web*.

---

[13] Prolog é uma linguagem de programação que se enquadra no paradigma de Programação em Lógica Matemática.

A proposta se resume a capturar conhecimento acerca dos serviços requeridos através de descrições genéricas de objetivos, e traduzir esse conhecimento em informação que possa popular um *caching* de serviços. Isso, segundo os autores, otimiza (melhora) a descoberta quando se deseja descobrir serviços em tempo de execução. Um ponto de destaque nessa proposta é a percepção dos tempos de descoberta: em tempo de projeto e em tempo de execução. Estes elementos requerem projetos de descoberta diferenciados, já que os requisitos e atividades inerentes a cada tempo são diferentes. Em tempo de projeto, escolhas interativas podem ser realizadas. Já em tempo de execução, escolhas automáticas, podem permitir uma melhor potencialização de serviços disponíveis. Assim, desconsiderar estes tempos pode levar a projetos que na prática sejam difíceis de serem usados e se distanciem das reais necessidades do mundo que usa serviços como paradigma de desenvolvimento de aplicações. Cabe ressaltar que os autores apontam para necessidade de classificação de serviços resultantes do processo de descoberta. Para isso, os autores sugerem algumas formas ou critérios, como por exemplo, o mais recente fornecido ou descoberto. Além da política de guardar serviços em uma memória mais rápida para otimizar descobertas futuras, destaca-se a importância para identificação das fases de projeto e de execução de aplicações SOA. O problema é que não há uma descrição das operações inerentes a cada uma das fases. Isto abriu espaço para que no modelo proposto fosse possível definir um conjunto de operações para cada um delas, o que permitiu um alinhamento mais claro e transparente entre o nível de processos e o nível de TI através da descoberta de serviços.

O Quadro 11 resume a lista de iniciativas estudadas com objetivo de melhorar a cobertura, a escalabilidade dos repositórios de serviços e o desempenho da descoberta. Na primeira coluna são listadas as iniciativas analisadas. A segunda coluna está dividida de forma a contemplar as formas mais relevantes usadas para melhorar a cobertura, a escalabilidade dos repositórios de serviços e o desempenho da descoberta. A coluna *Adota fase de projeto e fase de execução* representa a separação da descoberta em dois momentos distintos. Na última linha, apresenta-se o conjunto de estratégias usadas pelo modelo de descoberta proposto para melhorar a cobertura, a escalabilidade dos repositórios de serviços e o desempenho da descoberta.

Analisando o Quadro 11, há poucas iniciativas preocupadas em garantir uma melhor cobertura ou ainda que almejar atender a critérios como desempenho, escalabilidade entre outros. Uma das razões para isso está na necessidade de aumentar o escopo da descoberta para

também considerar esses aspectos. Isso eleva a complexidade da solução, pois requer o envolvimento e aplicação de estratégias de outras áreas como: Sistemas distribuídos e Segurança.

Grupo com foco na melhoria da cobertura, escalabilidade das UDDIs e desempenho da descoberta					
Iniciativas	Estratégia usada				
	Arquitetura da federação (P2P)	Consulta persistente	Uso de *caching*	Adota fase de projeto e fase de execução	Consulta federada
Zhou *et al.* (2003)			✓ (armazena valores de QoS)		✓
Rompothong e Senivongse (2003)					✓
ShaikhAli *et al.* (2003)		✓			✓
Sivashanmugam, Verma e Sheth (2004)	✓				✓
Kovács, Micsik e Pallinger (2006)				✓ (fase de descrição dos serviços e fase de descoberta, invocação e execução)	
tollberg, Hepp e Hoffmann (2007)	✓		✓ (para armazenar serviços)	✓ (ressalta as fases de projeto e de execução de aplicações SOA)	
Modelo de Descoberta Proposto			✓ (*crawling* usado na fase de projeto para descobrir serviços candidatos)	✓ (usa especificar serviços na fase de projeto, enquanto que na fase de execução serviços são descobertos dinamicamente, inovocados e executados)	✓ (usa como fonte de consulta diversos repositórios de serviços, distribuídos, gerenciados por uma federação)

Quadro 11 - Estratégias que visam melhorar a cobertura, escalabilidade dos repositórios e desempenho.

3.3.2.3 Iniciativas que visam melhorar o Processo de Seleção de Serviços

Relacionados a esse grupo foram encontrados onze trabalhos. Em razão do foco de alguns deles pertencerem há mais de um grupo, sete deles já foram descritos nas seções 3.3.2.1 e 3.3.2.2, sendo eles: Zhou *et al*. (2003), Maximilien e Singh (2004), Kokash (2005), Pathak *et al*. (2005), Wang *et al*. (2006a), Carenini *et al*. (2008b) e Kritikos e Plexousakis (2008). Os quatro restantes estão descritos nos parágrafos que seguem.

Wang *et al*. (2006b) são motivados pela dificuldade de selecionar serviços e, em um segundo plano, pela influência da seleção em outras atividades presentes no ciclo de vida dos serviços *web*. Desta forma, os autores propõem um mecanismo de seleção dinâmica de serviços baseado em QoS. Os autores definem uma ontologia e apresentam um mecanismo de seleção dinâmica de serviços e utilizam o *framework* WSMO (*Web Service Modeling Ontology*)[14]. O diferencial do trabalho de Wang *et al*. (2006b) é a normalização dos valores dos atributos de QoS. Segundo os autores, em geral, diversos trabalhos destacam o uso de QoS, uns na especificação de atributos e outros na forma de medição. Muitos falham quando comparam grandezas diferentes, pois nem sempre as métricas possuem as mesmas faixas de valores. A idéia principal do trabalho reside na proposta de um algoritmo que normalize cada métrica de QoS em valores entre 0 e 1, para tornar a comparação e seleção de valores *justa*.

Shein *et al.* (2008) apresentam um trabalho voltado à seleção de serviços através de atributos de QoS. A idéia está em propor uma sistemática de classificação (*ranking*) a partir de serviços semanticamente descobertos. Tais serviços são chamados de candidatos, já que podem possuir funcionalidades iguais ou semelhantes. Os autores adaptaram um algoritmo CLIQUE[15] que permite lidar com diversas dimensões de QoS. A proposta usa uma medida chamada de *Key*

---

[14] Modelo conceitual para descrição de várias características relacionadas aos serviços, gerenciado pela ESSI (*European Semantic Systems Initiative*) http://www.essi-cluster.org/working-groups/wsmo/ e http://www.wsmo.org/.

[15] Algoritmo de *clustering* aplicado como ferramenta para resolução de problemas em *data mining*.

Performance Index (KPI) para denotar valores de QoS para um dado serviço. O algoritmo produz uma saída contendo um *ranking* dos serviços que melhor atendem aos critérios funcionais de um cliente, organizados em função de valores associados à qualidade de serviço.

A estratégia de classificar serviços candidatos é semelhante à adotada no modelo de descoberta proposto. Entretanto, o modelo de descoberta opta por usar o conceito do provedor (valor obtido a partir do histórico do provedor do serviço na federação) para diferenciar e classificar serviços candidatos com características similares (que atendem critérios de funcionais e de qualidade). A vantagem desta estratégia é a de criar uma alternativa a seleção de serviços, não ficando restrita apenas aos atributos de QoS.

Badr *et al.* (2008) propõem uma solução para seleção e classificação de serviços. A proposta é baseada em atributos de qualidade para selecionar serviços e em preferência do usuário para classificá-los, ou seja, os autores investem na solução de dar pesos diferenciados para atributos de qualidade, sendo que os demais atributos permanecem com pesos iguais. Estes pesos permitem ao mecanismo de seleção e classificação estabelecer ordem de apresentação dos serviços resultantes do processo de descoberta. Um ponto recorrente em relação a outros trabalhos está na classificação de atributos de qualidade, embora tenha destaque a possibilidade de pesos para aqueles atributos de QoS mais importantes.

O modelo proposto usa atributos de QoS para selecionar serviços, mas também usa preferências do usuários, através do conceito de provedor para classficar serviços. Isto flexibiliza a forma de seleção de serviços, facilitando que o serviço mais adequado possa ser determinado, permitindo aproximar mais os ambientes BPM&SOA.

O trabalho proposto por Li *et al.* (2009) define descoberta como o processo de identificar serviços que satisfazem por completo os requisitos dos usuários. Na proposta, os autores separam o processo de descoberta em duas fases. A primeira se destina a encontrar serviços candidatos e a segunda refere-se à avaliação e classificação dos serviços candidatos encontrados na primeira etapa. A contribuição do trabalho reside na concepção de um modelo para descrever aspectos não funcionais, incorporando também atributos associados a negócios (preço) e outros não diretamente ligados às funcionalidades dos

serviços. Um algoritmo de decisão baseado em múltiplos critérios é usado tanto para penalizar como para premiar serviços.

A separação da descoberta em dois momentos distintos se assemelha à estratégia do modelo proposto. No modelo proposto, inicialmente buscam-se serviços que atendam a critérios funcionais e de qualidade, para então serem classificados por algum critério, tal como conceito do provedor. O objetivo é realizar um filtro preliminar para então classificar uma quantidade de serviços menor, evitando buscas exaustivas a todo o momento e prejuducando que a integração BPM&SOA seja mais ágil.

O Quadro 12 resume a lista de iniciativas estudadas com objetivo de melhorar a seleção de serviços. Na primeira coluna são listadas as iniciativas analisadas. A segunda coluna está dividida de forma a contemplar as formas mais relevantes usadas para melhorar o processo de seleção de serviços. Na última linha, apresenta-se o conjunto de estratégias usadas pelo modelo de descoberta proposto para melhorar a seleção de serviços.

Analisando-se Quadro 12 nota-se que grande parte das iniciativas estudadas usam QoS para selecionar serviços. Esta estratégia de seleção pode ser insuficiente, principalmente quando o número de serviços similares sendo ofertados é grande, porque pode resultar em um conjunto ainda bastante grande de serviços. Como forma de resolver isto, algumas estratégias adotam outros critérios, que somados com QoS, ganham força em um cenário de inúmeros serviços com características parecidas. O modelo proposto usa seleção via QoS e também adota um conceito associado ao provedor do serviço (um valor) para chegar ao serviço mais adequado possível. O objetivo é dotar o modelo de proposto de critérios que permitam selecionar serviços mais adequados possíveis, mesmo em um cenário composto de inúmeros serviços similares sendo ofertados.

Grupo com foco na melhoria da seleção de serviços			
Iniciativas	Estratégia usada		
	QoS	Usa descobertas antigas como critério de seleção	Adota preferências do usuário e outros critérios
Zhou *et al.* (2003)	✓		
Maximilien e Singh (2004)	✓		
Kokash (2005)	✓	✓	
Pathak *et al.* (2005)	✓		✓ (filtra serviços usando aspectos tradicionais de QoS, mas é possível usar outros atributos com pesos)
Wang *et al.* (2006b)	✓		
Wang *et al.* (2006a)		✓	
Carenini *et al.* (2008b)			✓ (usa critérios informados pelo usuário)
Kritikos e Plexousakis (2008)	✓		
Shein *et al.* (2008)	✓		
Badr *et al.* (2008)	✓		✓ (usa atribuir peso diferenciado para atributos de QoS)
Li *et al.* (2009)	✓		✓ (usa atributos associados a negócios, como preço)
Modelo de Descoberta Proposto	✓		✓ (usa conceito do provedor)

Quadro 12 - Estratégias que visam melhorar a cobertura, escalabilidade dos repositórios e desempenho da descoberta.

3.3.2.4 Análise Geral dos Trabalhos

O Quadro 13 apresenta uma síntese das iniciativas estudadas que envolvem a descoberta de serviços. As primeiras três colunas trazem os grupos descritos nas seções anteriores. As demais colunas associam cada iniciativa estudada com um conjunto de critérios identificados ao longo do levantamento do estado-da-arte, sendo eles: Sendo eles: Modelo de comercialização de serviços e SLA, Suporte a catálogo de processos de negócios, Suporte a execução de serviços e Atividades potencializadas pela descoberta (composição de aplicações, execução de aplicações) (TSALGATIDOU; PILIOURA, 2002). A última linha apresenta e compara as características do modelo de descoberta proposto frente às iniciativas estudadas.

A análise do Quadro 13, somada à leitura e análise dos trabalhos correlatos, permitiram registrar algumas constatações sobre o estado-da-arte em descoberta de serviços.

A primeira delas está associada à importância da descoberta de serviços como atividade básica para o desenvolvimento de outras atividades presentes no ciclo de vida dos serviços, tais como a composição de aplicações e a invocação de serviços (SYCARA *et al.*, 2003) e (KOURTESIS *et al.*, 2007).

A segunda constatação reside no aumento de serviços com características similares sendo ofertados, passando de uma tendência para uma realidade, apontada por (ZHOU *et al.*, 2003), (CARENINI *et al.*, 2008a) e (ELGAZZAR; HASSAN; MARTIN, 2010).

Na terceira, nota-se a necessidade de criar mecanismos de descoberta mais precisos (SHAIKHALI *et al.*, 2003), (PATHAK *et al.*, 2005), (WANG *et al.*, 2006a) e (CHAARI *et al.*, 2008), porque os atuais protocolos usados no ciclo de vida dos serviços possuem limitações técnicas que inviabilizam descobertas precisas e automáticas (MAXIMILIEN; SINGH, 2004) e (KRITIKOS; PLEXOUSAKIS, 2008).

Principais Iniciativas	Grupo: Melhoria da Descrição			Grupo: Melhoria da Cobertura, Escalabilidade e Desempenho					Grupo: Melhoria na Seleção			Modelo de disponibilização de serviços e SLA	Suporte a Catálogo de Processos	Suporte à Execução de Serviços	Atividades Potencializadas pela Descoberta
	Semântica	QoS	Outra	Arquitetura federação	Consulta persistente	Uso de *caching*	Fases: projeto e execução	Consulta federada	QoS	Descobertas antigas	Preferência do usuário				
Bernstein e Klein (2002)															Composição
Shaikh-Ali et al. (2003)		✓			✓			✓							-
Ran (2003)		✓													-
Zhou et al. (2003)		✓				✓		✓	✓					✓	Execução
Sycara et al. (2003)	✓														Composição e Execução
Rompothong e Senivongse (2003)						✓		✓							-
Sivashanmugam; Verma e Sheth (2004)	✓			✓				✓							-
Maximilien e Singh (2004)	✓	✓							✓					✓	Execução
Kokash (2005)		✓							✓	✓				✓	Execução
Paolucci et al. (2005)	✓	✓							✓		✓				-
Wang et al. (2006a)	✓	✓								✓					Composição
Kovács, Micsik e Pallinger (2006)							✓								-
Wang et al. (2006b)									✓						Composição
Stollberg, Hepp e Hoffmann (2007)			✓	✓		✓	✓							✓	Composição e Execução
Vitvar, Zaremba e Moran (2007)	✓		✓											✓	Composição e Execução
Kourtesis et al. (2007)	✓														Composição
Carenini et al. (2008a)	✓														-
Chaari et al. (2008)	✓	✓													-
Kritikos e Plexousakis (2008)	✓	✓							✓						-
Carenini et al. (2008b)	✓	✓									✓				-
Stein et al. (2008)									✓						-
Badr et al. (2008)									✓		✓				-
Li, Comerio, et al., (2009)									✓		✓				-
Elgazzar, Hassan e Martin, (2010)			✓												-
Modelo Proposto	✓	✓				✓	✓	✓	✓		✓	✓	✓	✓	Composição e Execução

Quadro 13 - Comparação das características do modelo de descoberta proposto em relação às iniciativas estudadas.

A quarta diz respeito à semântica, que através de ontologias, é considerada um aspecto chave no processo da descoberta. Ela permite obter precisão e possibilita que máquinas possam interagir e executar descobertas e seleções de serviços com menos intervenção humana (KOKASH, 2005) e (PAPAZOGLOU *et al.*, 2007). Embora a semântica prevaleça sobre métodos sintáticos, há iniciativas que usam técnicas não semânticas, pois são mais populares e mais suportadas pela indústria e pelas ferramentas computacionais (ELGAZZAR; HASSAN; MARTIN, 2010).

A quinta trata da necessidade de seleções de serviços automáticas para poupar tempo e esforço humano na seleção de serviços (SYCARA *et al.*, 2003) e (PAPAZOGLOU *et al.*, 2007). Neste sentido, notam-se diversas formas de selecionar serviços. O uso de QoS é a principal técnica, mas outras podem ser incorporadas pelos mecanismos de descoberta para tratar de serviços com características similares (VITVAR; ZAREMBA; MORAN, 2007) e (CHAARI *et al.*, 2008).

A sexta constatação diz respeito à baixa preocupação com a qualidade dos serviços publicados e ofertados nos repositórios de serviços. Ou seja, muitos provedores publicam serviços em repositórios (principalmente públicos) e não se preocupam em mantê-lo em funcionamento e mesmo atualizado. Este procedimento fez com que muitos dos repositórios fossem desativados, forçando usuários a descobrir serviços via algum tipo de mecanismo de descoberta similar aos mecanismos de busca disponíveis na *web*, tais como: Google e Yahoo (EYHAB; MAHMOUD, 2008).

A sétima mostra que as estratégias usadas para descobrir serviços são bastante diversificadas e fragmentadas. Isto revela diferentes visões em relação ao melhor método para descobrir serviços e com base na literatura estudada, as iniciativas não se preocupam em integrar suas propostas ao nível de processos de negócios (nível BPM), apenas assumem o fato de descobrir serviços para algum processo de negócio.

A oitava e última constatação está relacionada quantidade de abordagens que adotam como ponto de partida para o projeto de descoberta os processos de negócio (STOLLBERG; HEPP; HOFFMANN, 2007) e (VITVAR; ZAREMBA; MORAN, 2007). Ou seja, parte-se da análise das necessidades dos processos de negócios para então identificar necessidades e posteriormente descobrir serviços. A dificuldade desta abordagem está em obter uma granularidade fina a partir dos processos de negócios para definir serviços, tarefa que nem sempre é fácil de ser realizada (AZEVEDO *et al.*, 2009).

A nona constatação diz respeito ao aumento de serviços ofertados. Neste sentido, há necessidade de aumentar a cobertura, desempenho e escalabilidade das UDDIs (SIVASHANMUGAM; VERMA; SHETH, 2004), (STOLLBERG; HEPP; HOFFMANN, 2007). Isto se justifica em razão do crescimento da oferta e do aparecimento de serviços cada vez mais similares.

A décima tem relação com a seleção de serviços. Observou-se que os trabalhos usam critérios tradicionais para seleção de serviços (QoS). Isso abre oportunidades para que novos critérios sejam desenvolvidos e incorporados a outros já existentes, tal como se verifica no trabalho proposto por (LI *et al.*, 2009). Entretanto, determinar quais serão esses critérios, a forma de medi-los, quando, quem e com qual frequência os mesmos devem ser coletados, são perguntas que necessitam ser trabalhadas.

3.4 **Aspectos associados à descoberta**

Esta seção apresenta cinco aspectos associados à descoberta de serviços, elicitados a partir do levantamento do estado da arte e em razão da desejada integração BPM&SOA buscada neste trabalho.

3.4.1 O que expressar para um serviço?

O primeiro deles refere-se ao conjunto de informações que devem ser expressas e constar na representação do serviço desejado. Isto inclui informações como o nome do serviço, entradas, saída e aspectos de qualidade.

3.4.2 Como expressar um serviço desejado?

O segundo diz respeito a como expressar uma necessidade, ou seja, qual estrutura deve ser usada para representar um serviço, por exemplo: linguagem natural, linguagem formal e assim por diante.

3.4.3 Quem expressa um serviço desejo?

O terceiro aspecto está associado a quem expressa o desejo por um serviço. Esse problema envolve quatro diferentes abordagens, conforme mostra o Quadro 14.

Abordagem	Características
Fortemente baseada no Projetista de Aplicações	• Controle do que será informado fica a cargo do projetista; • Alta interatividade do projetista com o ambiente BPM; • Ambiente BPM recolhe informações e as repassa; • A análise dos resultados (serviços) é feita pelo projetista, determinando se os mesmos são adequados ou não.
Automática	• Controle fica a cargo do ambiente BPM; • O ambiente BPM identifica necessidades de forma automática; • Modificações nos requisitos são feitas automaticamente pelo ambiente. Estas modificações acontecem quando o que se deseja não está em conformidade com o que se encontra; • Não há intervenção do projetista no processo de descoberta de serviços *web*.
Semi-automática	• O ambiente, automaticamente, identifica necessidades; • Modificações nos requisitos são realizadas quando o que se deseja não está em conformidade com o que se encontra. As mudanças podem ser feitas: o Automáticas pelo Ambiente BPM até encontrar o que se deseja (conformidade de 100% entre desejo *versus* resultado); o Quando o ambiente encontra dificuldade (conformidade não é total), entra em cena o projetista que interage com ambiente.
Assistida	• O projetista, além de informar o que deseja, é auxiliado pelo ambiente BPM que fornece informações (relativas ao processo de negócio) como forma de ajudá-lo a construir a expressão da necessidade. • Alguns pontos em aberto nesta abordagem: o Projetista: informa quais dados? o Ambiente: controla e fornece quais dados?

Quadro 14 - Abordagens e características de quem expressa necessidade.

3.4.4 Quem avalia e de que forma os resultados são avaliados?

O quarto problema aborda a avaliação dos resultados dos serviços descobertos. Nesse aspecto há que se considerar algumas situações em função dos resultados da descoberta:

- Vários (n) serviços perfeitos (um serviço é perfeito quando satisfaz completamente o pedido do serviço desejado);
- Um (1) serviço perfeito;
- Nenhum serviço;
- Um (1) serviço parcial (quando não se atinge 100% de conformidade entre o que se deseja e o que se encontra);
- Diversos (n) serviços parciais.

Na abordagem *Fortemente baseada no Projetista* (ver Quadro 14), o projetista escolhe e determina o serviço mais adequado para uma dada tarefa. Nessa abordagem há uma forte interação do usuário projetista com o mecanismo de descoberta, exigindo paciência do projetista para avaliar serviços perfeitos.

Na abordagem *Automática* (ver Quadro 14), o controle e, portanto, a dinâmica do processo de descoberta fica a cargo do ambiente BPM. Um aspecto que merece destaque refere-se à seleção realizada pelo ambiente BPM. Esta tarefa exige do ambiente conhecimento necessário para definir com precisão quais serviços são perfeitos e quais não são. Para os serviços parciais, o ambiente deverá ter inteligência suficiente para realizar automaticamente suavizações e restrições sucessivas nos critérios de seleção até encontrar o serviço mais adequado.

Na abordagem *Semi-automática* (ver Quadro 14), o controle e a dinâmica do processo de seleção ficam a cargo do ambiente. O grande problema desta abordagem está na identificação dos requisitos que devem ser atendidos por um serviço. Nesse sentido, o projetista de aplicação será envolvido no processo, auxiliando o ambiente BPM a encontrar soluções, caso o ambiente não consiga encontrar serviços perfeitos.

Na abordagem *Assistida* (ver Quadro 14), a dificuldade está em responder a um conjunto de questões: (1) quão híbrida ou assistida será esta abordagem? (2) quais tarefas serão automáticas? (3) quais tarefas exigirão intervenção do projetista? (4) nos resultados gerados por descobertas, até que ponto solicitar a intervenção do projetista? (5) suavizar ou restringir critérios, isso ficaria a cargo do ambiente ou do projetista?

3.4.5 Quando um serviço desejado é expresso e descoberto?

O quinto e último aspecto está associada a quando um serviço é expresso e qual o momento que a descoberta é acionada (tempo de projeto ou execução de aplicação). O modelo proposto de descoberta trata desse problema criando duas fases, uma chamada de projeto de aplicações e a outra de execução de aplicações, explicadas em detalhes na Seção 4.1 e 4.2.2.

O Quadro 15 resume as características associadas às abordagens Automática, Semi-Automática, Assistida ou Fortemente baseada no Projetista, levando em conta os problemas associados à descoberta mencionados.

Abordagem / Aspecto	Automática	Semi-automática	Assistida	Fortemente baseada no Projetista
Interatividade	Nenhuma	Pouca	Alta	Alta
Quem identifica e informa necessidade	Ambiente BPM	Ambiente BPM	BPM e Projetista	Projetista
Quem avalia (desejo x resultado)	Ambiente BPM	Ambiente BPM e Projetista (eventualmente)	Ambiente BPM e Projetista	Projetista
Comportamento/ Atitude	Ambiente BPM (ativo)	Ambiente BPM (ativo)	Ambiente BPM (passivo) e Projetista (ativo)	Ambiente BPM (passivo) e Projetista (ativo)

Quadro 15 – Resumo das características das abordagens: Automática, Semi-Automática, Assistida ou Fortemente baseada no Projetista.

3.5 **Considerações**

O objetivo do capítulo foi identificar e registrar o estado-da-arte sobre descoberta de serviços, descrevendo as principais lacunas e oportunidades. Inicialmente registrou-se um conjunto de definições sobre o termo descoberta e suas formas. Em um segundo momento, em razão do escopo e da inerente complexidade da descoberta de serviços, os trabalhos analisados e descritos foram organizados em grupos de afinidades, designados por: Melhoria da descrição dos serviços,

Aumento da cobertura, escalabilidade dos repositórios de serviços e desempenho da descoberta e Melhoria na seleção de serviços. Diversos quadros foram criados, cuja finalidade principal foi apontar semelhanças e diferenças entre as iniciativas estudadas e o modelo de descoberta proposto. Em essência, analisando-se o conjunto de lacunas percebe-se que há uma oportunidade para o desenvolvimento de um modelo que incorpore diversos elementos requeridos num único artefato, abordagem não contemplada por nenhum dos trabalhos verificados, dificultando que os níveis de negócios (BPM) e de sistemas (SOA) sejam integrados de forma ágil e transparente.

Capítulo 4

Modelo de Descoberta Proposto

O capítulo apresenta o modelo de descoberta dinâmica de serviços *web* proposto. Ele inicia descrevendo o conjunto de características identificadas para o modelo de descoberta proposto. A seção 4.2 mostra os diversos componentes conceituais que formam a arquitetura do modelo proposto. A seção 4.3 lista e explica os pressupostos adotados na concepção do modelo de descoberta proposto. A seção 4.4 apresenta o diferencial da proposta frente aos demais trabalhos.

4.1 Características do Modelo Proposto

A identificação e o registro das características do modelo proposto tiveram como ponto de partida três elementos fundamentais. O primeiro deles diz respeito à construção do referencial teórico e do estado da arte em descoberta de serviços, apresentados nos Capítulos 1, 2 e 3. O segundo partiu da desejada integração BPM&SOA. O terceiro nasceu da necessidade de obter respostas para os aspectos enunciados na Seção 3.4. Desta maneira, foram identificadas onze características que representam os requisitos gerais que o modelo proposto deve respeitar tendo em vista o cenário de integração e de descoberta desejado.

A primeira delas, *Integrado*, está associada à necessidade de integrar logicamente diversos provedores de serviços, ou seja, empresas que constroem softwares, ofertando-os na forma de serviços *web*,

independente da localização física destes provedores. Esta integração ocorre na forma de uma federação de provedores de serviços, usando como maior inspiração a proposta apresentada por Rabelo (2008). Com isso, pretende-se atingir um maior grau de confiabilidade em relação à qualidade dos serviços publicados e ofertados, pois para publicar serviços, provedores deverão seguir um guia (CANCIAN; RABELO; VON WANGENHEIM, 2009), cuja finalidade maior está em conhecer e certificar serviços e provedores para uso futuro. Além da confiabilidade mencionada, pretende-se obter uma maior cobertura em termos de quantidade de serviços, através de um conjunto de repositórios de serviços gerenciados pela federação. Desta forma, não sendo restrita somente a isto, a federação age como um ente lógico que tanto congrega provedores de serviços como fonte de informações para o processo de descoberta (permitindo que a descoberta possa atingir diversos repositórios remotos de forma transparente para o mecanismo de descoberta).

A segunda característica, *Aberto*, refere-se ao fato do modelo dever ser interoperável. Assim, a sua concepção usa, de forma intensa, padrões e recomendações abertos, especificados por iniciativas internacionais para permitir um menor esforço e menor tempo quando algum novo componente for incorporado ao modelo proposto. Esta é uma característica chave que diferencia o modelo proposto de outras abordagens estudadas, porque permite que problemas de interoperabilidade sejam resolvidos ou minimizados ao ponto de impactarem uma mais transparente e ágil integração entre o nível dos processos de negócios (BPM) e o nível dos sistemas (SOA).

A terceira característica, *Flexível*, advém da necessidade de usar diversos critérios para buscar e selecionar serviços, conforme apontam os trabalhos (VITVAR; ZAREMBA; MORAN, 2007), (PATHAK *et al.*, 2005), (WANG *et al.*, 2006a) e (KOKASH, 2005). Como critério funcional, o modelo proposto incorpora uma ontologia de processos UBL, usada para definir de forma única atividades presentes em um processo de negócio[16]. Além disso, características de QoS são usadas

[16] Cabe salientar que no contexto da especificação UBL, os processos de negócios são formados por um conjunto de atividades. Cada atividade representa uma ação a ser executada para que um dado processo UBL ocorra.

para descrever serviços e também para selecionar serviços perfeitos (aqueles que atendem completamente um desejo). Outro critério de seleção passa pela preferência do usuário em usar ou não um conceito associado ao provedor de serviços para determinar o serviço mais adequado. Este último, inspirado nos trabalhos de (LI *et al.*, 2009) e (PATHAK *et al.*, 2005).

A quarta característica, *Abordagem*, diz respeito à forma com que a descoberta deve ocorrer durante a concepção e a execução de aplicações SOA. Na concepção de aplicações, a descoberta assume a forma de uma *descoberta assistida*. O objetivo é auxiliar o projetista na execução de tarefas como: informar aspectos de QoS e avaliar o resultado da descoberta. Na abordagem assistida o modelo proposto conta com um *crawler* que faz a busca, a seleção e apresenta ao projetista um conjunto de serviços candidatos. O projetista pode ou não concordar com a resposta do *crawler*, descartando resultados ou aprimorando informações sobre o serviço desejado, restringindo aspectos de qualidade. Na fase de execução de aplicação SOA, o mecanismo de descoberta adota um comportamento automático, pois nesta fase um usuário executor de aplicação SOA está interessado na execução da aplicação SOA em si, já que, nesta fase, seu papel não é de informar ou avaliar resultados de descobertas ou ainda fazer mudanças na aplicação SOA.

No modelo proposto de descoberta, a quinta característica corresponde à ontologia de processos, *no caso a ontologia UBL* (detalhes da concepção da ontologia estão descritos na Seção 4.2.3.3.1). Sua construção baseia-se na especificação de processos de negócios UBL e visa definir serviços em termos funcionais. Ela dá maior riqueza semântica à descoberta, resolve problemas semânticos e reduz problemas de interoperabilidade tanto para clientes e provedores, pois ambos usam um vocabulário comum quando interagem com o mecanismo de descoberta, seja para publicar ou descobrir serviços. Dessa maneira, a ontologia cria uma efetiva ponte (por ser baseada em um padrão) entre requisitos dos clientes e provedores, fornecendo maior qualidade na descoberta de serviços, garantindo somente o retorno de serviços semanticamente alinhados aos processos UBL (ao nível de negócios), possibilitando um alinhamento mais próximo entre BPM&SOA.

A sexta característica, *Catálogo de Processos*, está relacionada ao ambiente de concepção de aplicações que está integrado a um catálogo de processos de negócios (conforme proposta de Bezerra (2011)). Desta maneira, o projetista pode usar processos prontos baseados em um

padrão (no caso a UBL) para construir suas aplicações sem ter que, por exemplo, iniciar ou construir a aplicação SOA por completo a partir do zero. O repasse de informações da aplicação em construção para a camada tecnológica, via mecanismo de descoberta, também é outro aspecto importante. Isto faz com que a perda do contexto do processo não ocorra, possibilitando ao mecanismo de descoberta achar e selecionar o serviço mais adequado em função do contexto do processo atual (das características do processo definidas via ontologia UBL). Desta forma, com esta integração (do ambiente de edição com o catálogo) objetiva-se tornar o nível de processos (BPM) e o nível de sistemas (SOA) mais alinhados, proporcionando uma mais transparente e ágil integração entre os ambientes BPM&SOA.

A sétima característica refere-se ao uso de um *crawler de serviços*, que é usado no momento da concepção de aplicações pelo usuário projetista. O objetivo do *crawling* é retornar um conjunto de serviços candidatos, sendo seu uso justificado através de dois motivos-chave. O primeiro é evitar que descobertas exaustivas sejam realizadas em tempo de execução de aplicações SOA. O segundo é possibilitar que o projetista da aplicação possa usá-lo para verificar se há serviços com determinadas características desejadas. Caso isto não ocorra, o projetista pode testar ou usar outro conjunto de características de QoS e analisar novamente o resultados apresentado pelo *crawling*. Cabe salientar que nesta etapa o *crawler* assiste o projetista informando quais critérios de QoS não foram atendidos em uma descoberta, permitindo a ele relaxar valores ou substituir características de QoS por outras, em uma nova descoberta. O uso de um *crawling* no modelo proposto foi baseado em estratégias semelhantes adotadas nos trabalhos de (STOLLBERG; HEPP; HOFFMANN, 2007), (ZHOU *et al.*, 2003) e (ELGAZZAR; HASSAN; MARTIN, 2010).

A oitava característica, *Algortimo de descoberta dinâmica*, está relacionada à construção de um algoritmo de descoberta de serviços que em tempo de execução de aplicações SOA, automaticamente, faz a descoberta e a vinculação dos serviços mais adequados para aquele momento. Em termos conceituais, o algoritmo de descoberta dinâmica é acionado via mecanismo de execução de aplicações SOA que usa a lista de serviços candidatos associada a cada atividade da aplicações SOA com objetivo de verificar a disponilidade do serviço. Nesta situação, o primeiro serviço disponível e acessível é usado, sendo imediatamente

vinculado à aplicação e invocado. Caso nenhum serviço candidato possa ser usado na execução da aplicação SOA, o algoritmo de descoberta dinâmica faz uma nova descoberta usando os repositórios de serviços disponíveis na federação.

A nona característica, *Qualidade de Serviço (QoS)*, trata de dotar o modelo proposto de um conjunto de características de QoS que unifique seu uso tanto na descoberta como na publicação de serviços, minimizando problemas de interoperabilidade. A justificativa para uso dos critérios de qualidade reside em dois fatores principais. O primeiro é possibilitar uma melhor e mais rica descrição dos serviços tal como proposto em (PATHAK *et al.*, 2005) e (SHEN *et al.*, 2008). O segundo aspecto é usar características de qualidade para poder selecionar serviços de acordo com preferências do usuário, principalmente devido a crescente quantidade de serviços similares sendo ofertados e disponibilizados e à necessidade de tornar os mecanismos de descoberta mais automatizados (diminuindo o esforço e intervenção humana no processo de seleção). A ontologia de QoS tem como inspiração o trabalho Tondello (2008), sendo que a sua concepção está descrita na Seção 4.2.3.3.1.

A décima característica, *Modelo para Disponibilização de Serviços e SLA*, (*Service Level Agreement*/Contrato a nível de serviço) tem como aspecto principal dotar o modelo proposto de um moderno modelo de acesso a serviços que especifique o uso, o pagamento, as qualidades oferecidas, os direitos e deveres das partes envolvidas, através de um contrato (SLA) entre as partes que usam um serviço. Isto é extremamente relevante considerando que o número de serviços similares e ofertados vem crescendo (CARENINI *et al.*, 2008a) e (ELGAZZAR; HASSAN; MARTIN, 2010).

A décima primeira característica, *Ubiquidade dos provedores*, dos provedores de serviços de software distribuídos pela Internet, cuja característica principal é tornar os serviços *web* sempre disponíveis, a qualquer lugar e a qualquer tempo (NESSI, 2006). O Quadro 16 resume as características gerais presentes no modelo de descoberta proposto.

Característica	Breve descrição
Integrado	Deve integrar provedores de serviços, independentemente da localização dos mesmos.
Aberto	Deve ser interoperável. Deve usar um conjunto de padrões e recomendações abertos, especificados por algumas iniciativas como: W3C, OMG, OASIS e outras.

Flexível	Deve suportar vários critérios de descoberta de serviços.
Abordagem	Em tempo de projeto de aplicações (*off-line*) deve usar a abordagem assistida: • Projetista identifica e informa necessidade; • *Crawling* e Projetista avaliam (desejo x resultado); • Comportamento do Projetista é ativo e do ambiente BPM reativo. Em tempo de execução de aplicações (*on-line*) deve utilizar a abordagem automática: • Ambiente BPM identifica e informa necessidade; • Ambiente BPM faz *ranking* dos serviços, portanto avalia (desejo x resultado); • Não há interatividade; • Comportamento do ambiente BPM ativo.
Ontologia de Processos de Negócios	Deve usar uma ontologia de processos de negócios baseado no padrão UBL para representar e definir serviços em termos funcionais.
Catálogo de Processos	Deve estar integrado a um catálogo de processos de negócios UBL, nos moldes proposto por Bezerra (2011), usando-o como fonte de processos de negócios prontos.
Crawling para serviços *web*	Deve ser definido e construído um *crawler* para pesquisar serviços em repositórios locais e remotos, de maneira a localizar e selecionar serviços que satisfazem à expressão da descoberta. Além disso, o *crawling* pode identificar a necessidade de relaxar alguns requisitos, quando não existir serviços candidatos que satisfaçam a expressão da descoberta.
Algoritmo de descoberta dinâmica de serviços *web*	Deve ser definido e construído um mecanismo de descoberta dinâmica para identificar, escolher e vincular o serviço mais adequado, conforme requisitos previstos para a aplicação naquele momento. Nessa situação, o mecanismo de descoberta deve se valer de informações colhidas pelo *crawling* para descobrir serviços de forma mais eficiente, visando evitar exaustivas descobertas.
Qualidade de Serviço (QoS)	O modelo proposto deve possuir uma relação de aspectos de QoS aplicáveis para serviços *web*.
Modelo para	O modelo proposto adota o modelo SaaS (*Software-*

Disponibilização de Serviços e SLA	*as-a-service*/software-como-um-serviço) e incorpora a geração de um contrato em nível de serviço (SLA) que visa registrar direitos e deveres das partes envolvidas no uso dos serviços.
Ubiquidade dos provedores	O modelo proposto adota o *utility* paradigma que prevê um cenário no qual serviços de software estão distribuídos sobre a Internet, são acessados sob demanda, de qualquer lugar a qualquer momento e sempre estão disponíveis (NESSI, 2006).

Quadro 16 – Resumo das características presentes no modelo proposto.

4.2 **Arquitetura Conceitual**

A construção de modelos conceituais permite a criação de uma abstração que possa ser usada para o desenvolvimento de arquiteturas concretas. Neste sentido, a arquitetura conceitual do modelo proposto é composta de visões, que permitem observar e analisar o modelo proposto em diferentes perspectivas, independentes de tecnologias, implementação ou outros detalhes mais concretos (MERSON, 2005). A Figura 20 ilustra e sintetiza os principais elementos da proposta. Esses elementos se diluem nos cinco problemas chave, associados à abordagem concebida, que tem relação com: (1) o que expressar para um serviço, (2) como expressar um serviço desejado, (3) quem expressa um serviço desejado, (4) quem avalia resultados e de que forma e (5) quando um serviço desejado é expresso e buscado.

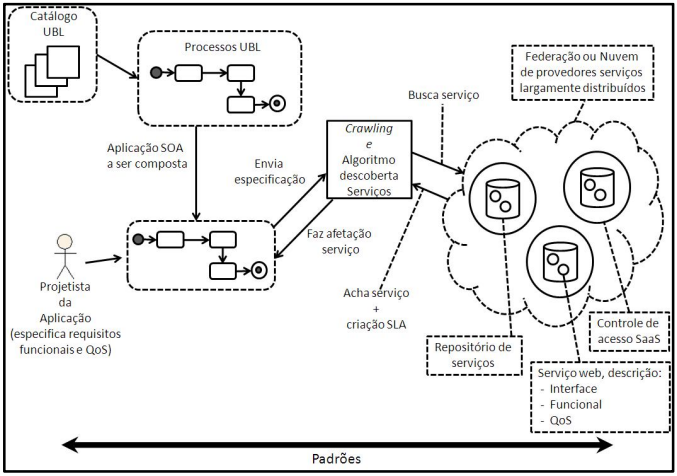


Figura 20 - Ilustração dos aspectos diferenciais da proposta.

4.2.1 Visão dos Casos de Uso

Com base nas características do modelo proposto, dois diagramas de casos de uso da UML foram desenvolvidos. O primeiro diagrama, mostrado na Figura 21, apresenta a relação de casos de uso utilizados pelo *Usuário Projetista de Aplicações*. O caso de uso, *Constrói Aplicações SOA*, utiliza informações provenientes do catálogo eletrônico de processos (BEZERRA, 2011) para definir e representar o contexto da aplicação e usa um *crawling* para obter serviços candidatos para implementar atividades presentes em uma aplicação.

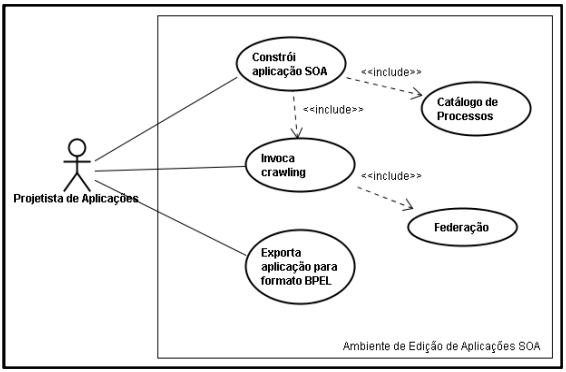


Figura 21 - Diagrama de caso de uso para Projetista de Aplicações.

O segundo caso de uso, *Invoca o crawling*, usa como fonte de pesquisa a Federação (o conjunto de repositórios associados a ela). Nela, o *crawling* faz a busca para selecionar e apresentar o serviço mais adequado, considerando o desejo do Projetista da aplicação.

O terceiro caso de uso, *Exporta aplicação para formato WS-BPEL*, permite ao projetista gravar uma aplicação SOA em um formato executável.

O segundo diagrama, mostrado na Figura 22, apresenta a lista de casos de uso relacionados com o *Usuário Executor de Aplicações*.

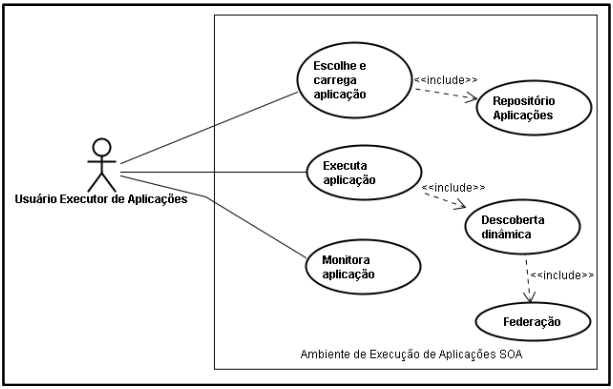


Figura 22 – Diagrama de caso de uso para Usuário Executor de Aplicações.

O caso de uso *Escolhe e Carrega Aplicação* permite ao *Usuário Executor* navegar (em um repositório de aplicações executáveis) para buscar e selecionar uma aplicação pronta para ser executada. O caso de uso *Executa aplicação,* permite ao *Usuário Executor de Aplicações* escolher e executar uma dada aplicação SOA. O terceiro caso de uso possibilita ao *Usuário Executor de Aplicações* monitorar o andamento da aplicação SOA. Para isso, ele usa uma interface gráfica que informa o estado atual da aplicação em execução.

Na seção seguinte, esses casos de uso são desdobrados e detalhados em termos do seu funcionamento, descrevendo a operação do modelo proposto.

4.2.2 Visão Operacional

O modelo integra o nível de processos ao nível tecnológico e está dividido em duas fases: projeto (Figura 23, lado esquerdo) e execução de aplicações SOA (Figura 23, lado direito).

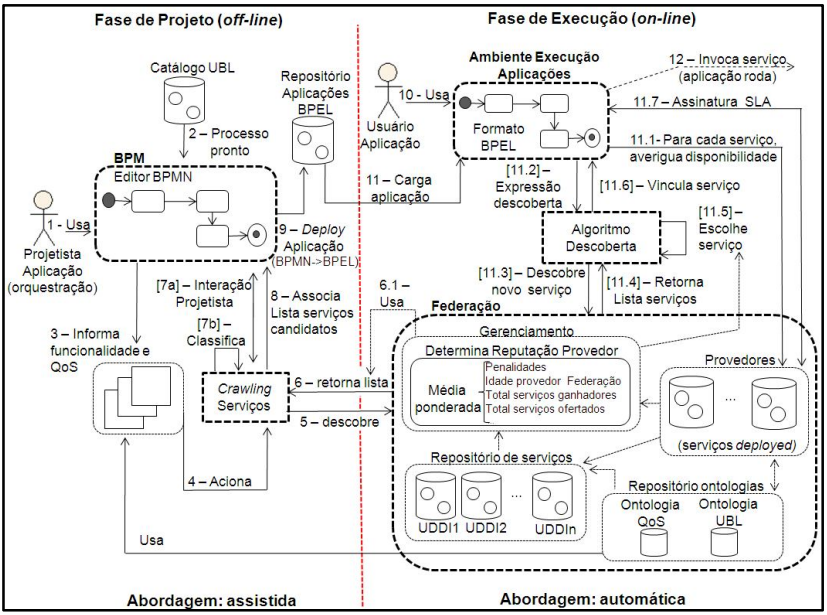


Figura 23 - Visão operacional do modelo proposto para descoberta.

A concepção do modelo proposto é baseada em cinco pressupostos (descritos em detalhes na Seção 4.3). O primeiro diz respeito ao conteúdo da pesquisa, que é por serviços *web*. O segundo refere-se ao uso do padrão UBL para modelar aplicações SOA. O terceiro pressuposto é sobre o uso de uma Federação de Provedores como fonte de pesquisa de serviços. O quarto pressuposto é chamado de relação 1:1. Isto significa que toda atividade presente na aplicação SOA será composta e estará associada a um único serviço descoberto. O quinto e último pressuposto considera o projetista um experiente profissional, que conhece processos e a ontologia UBL, tendo também conhecimento sobre aspectos relacionados à QoS.

A estratégia de separar o modelo proposto em fases distintas (o projeto da execução de aplicações) teve como inspiração os trabalhos de (STOLLBERG; HEPP; HOFFMANN, 2007) e (LI *et al.*, 2009) e é justificada a partir de dois aspectos importantes. O primeiro é caracterizado pela abstração dos detalhes de operacionalização de cada uma das fases. Com isso, um usuário (seja projetista ou executor de

aplicações) consegue focar mais e melhor o seu trabalho. Além disso, o fraco acoplamento entre as fases permite que o modelo proposto evolua de forma mais natural, seja incorporando ou substituindo elementos com menos tempo e esforço. O segundo aspecto enfatiza a desejada integração BPM&SOA, pois o conhecimento dos passos operacionais subsidia o projeto de novos ambientes computacionais, permitindo uma mais transparente e clara integração BPM&SOA.

Na fase de projeto de aplicações SOA (*off-line*), o usuário projetista usa um ambiente BPM com suporte a um editor BPMN para compor sua aplicação (passo 1). No passo 2, o ambiente BPM oferece acesso a um catálogo de processos de negócios padrão UBL. O projetista escolhe e usa processos de negócios UBL prontos para desenvolver sua aplicação. Isso agiliza a concepção da aplicação e dá maior riqueza semântica à descoberta, pois o uso de uma ontologia de processos UBL (ver Seção 4.2.3.3.1 para detalhes da concepção da ontologia UBL) serve para definir serviços em termos funcionais.

No passo 3, o projetista escolhe uma atividade, informa a funcionalidade requerida (usando a ontologia UBL), registra características de QoS (usando uma ontologia QoS - ver Seção 4.2.3.3.1 para detalhes da concepção da ontologia de QoS) e informa o uso ou não da reputação do provedor. Neste ponto, uma instância da expressão da descoberta é criada, conforme estrutura: $W_{desejado} = (F,Q)$, onde $W_{desejado}$ denota o serviço *web* requerido, F a funcionalidade exigida e Q descreve a lista das características de QoS desejadas.

De posse da expressão da descoberta (explicada em detalhes na Seção 4.2.2.1), um *crawler* é acionado (passo 4) para descobrir (pesquisar por) serviços. O *crawler* inicia a busca sistemática e sintática nos repositórios de serviços visando encontrar serviços candidatos (passo 5) que atendem a funcionalidade desejada e, ao mesmo tempo, selecionando os serviços que respeitam as desejadas restrições de QoS. O resultado (passo 6) é uma lista de serviços candidatos, que pode estar ou não classificada com base na reputação do provedor de serviço (passo 6.1). A reputação de cada provedor é expressa através de um valor obtido a partir da análise dos aspectos associados a cada provedor como por exemplo: penalidades sofridas, idade do provedor na federação, nível da sua certificação.

No passo 7, caso não haja serviço retornado pelo *crawling* (lista de serviços descobertos esteja vazia), o projetista tem a opção de relaxar os critérios de QoS a fim de encontrar algum serviço perfeito (aquele que satisfaz os critérios da expressão da descoberta) ou correr o risco de tentar fazer essa descoberta em tempo de execução (quando o processo

já estiver em execução no ambiente de execução de aplicações). Se a lista de serviços candidatos contiver apenas 1 (um) serviço, este é vinculado à atividade. Havendo diversos serviços candidatos perfeitos retornados pelo *crawling*, o projetista pode escolher um deles para implementar a atividade. Caso o projetista tenha escolhido usar a reputação do provedor, o primeiro da lista apresentada pelo *crawling* é vinculado à atividade. Cabe salientar que junto com o serviço candidato escolhido, vinculam-se também à atividade os demais serviços candidatos perfeitos retornados pelo *crawling* (passo 8). Assim, caso o serviço previamente alocado para implementar a atividade não esteja disponível e acessível, em tempo de execução de aplicação, busca-se o próximo serviço da lista de candidatos.

No passo 9, a aplicação pronta é convertida no formato executável e disponibilizada para execução futura, com os serviços mais adequados descobertos e respectivos contratos de uso de serviços (SLAs), gerados conforme proposta de Cancian, Rabelo e Von Wangenheim (2009).

Na fase de execução de aplicações SOA (Figura 23, lado direito), um usuário executor de aplicações SOA navega no repositório de aplicações WS-BPEL e identifica a aplicação que deseja executar (passos 10 e 11). Na execução da aplicação SOA, o ambiente de execução de aplicações interpreta e executa o código WS-BPEL. Nesta etapa, para cada atividade da aplicação implementada por um serviço, o ambiente usa a lista de serviços vinculados a procura do primeiro serviço disponível. Caso encontre, a execução da aplicação passa para outra atividade WS-BPEL. Caso não encontre, o ambiente de execução invoca o algoritmo de descoberta dinâmica para selecionar um novo serviço em tempo de execução da aplicação SOA. Este novo serviço é vinculado à aplicação, um novo contrato SLA é gerado automaticamente, e a aplicação segue a sua execução. O passo 12 representa um motor WS-BPEL executando a aplicação SOA com os mais adequados serviços para o momento.

4.2.2.1 Expressão da Descoberta

Um elemento fundamental e de difícil percepção no modelo proposto é a expressão da descoberta. Ela é usada pelo *crawling* e, se necessário, usada pelo algoritmo de descoberta dinâmica para selecionar um (1) novo serviço. Na fase de projeto (Figura 23, lado esquerdo, antes

da invocação do *crawling*), as informações de QoS somadas ao contexto da aplicação são organizadas na forma de uma expressão de pesquisa composta de uma *2-tupla*, conforme mostra a Figura 24.

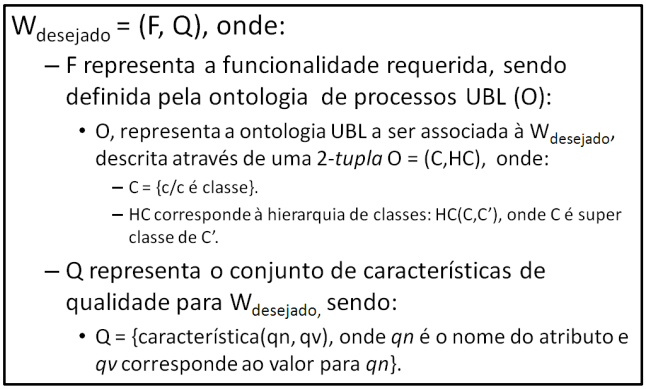


Figura 24 - Expressão da descoberta.

A Figura 25 ilustra um exemplo contendo valores fictícios presentes em uma expressão da descoberta. No canto superior esquerdo, ilustra-se uma determinada funcionalidade requerida (*AcceptOrderBuyerParty*). Esta funcionalidade é determinada a partir da ontologia UBL, representada na parte inferior da Figura 25. A ontologia UBL identifica univocamente a atividade *AcceptOrderBuyerParty* dentro da especificação UBL. Ou seja, a atividade *AcceptOrderBuyerParty* é executada pelo participante *BuyerParty* e faz parte do processo *OrderingProcess* que pertencente a categoria *Ordering* da especificação UBL. No lado superior direito da Figura 25, representa-se o conjunto de características de QoS desejadas para um serviço. No exemplo, tem-se um conjunto de atributos (*ResponseTime* e *ExecutionTime*) acompanhados de seus respectivos valores que formam uma dada característica de QoS desejada (no exemplo: *Performance*).

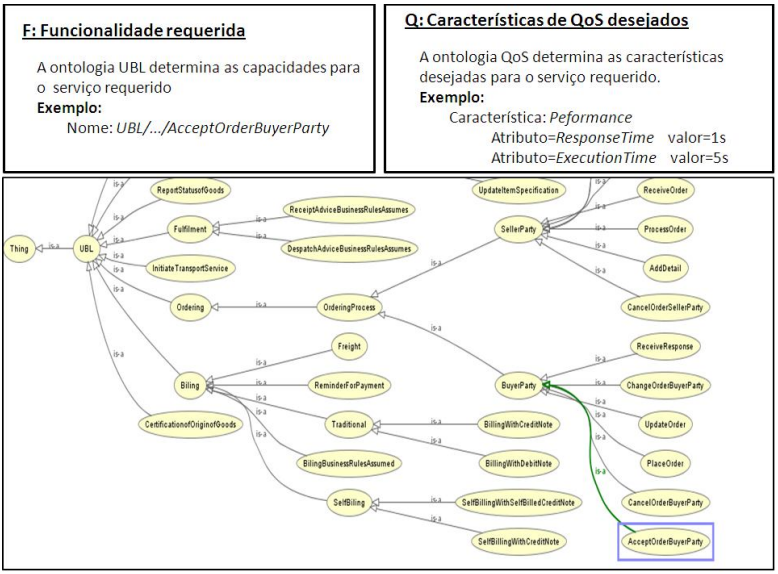


Figura 25 - Exemplo da expressão da descoberta, juntamente com a representação gráfica da ontologia UBL gerada a partir do *Protégé*.

Identificados os elementos presentes no modelo e conhecida a expressão da descoberta (Figura 24), as seções que seguem descrevem o funcionamento conceitual do modelo proposto levando em consideração as operações que compõem as fases de projeto e execução de aplicações.

4.2.2.2 Fase de Projeto de Aplicações (*off-line*)

A lista de operações efetuadas na fase de projeto de aplicações é apresentada no Quadro 17.

Resumo descritivo dos passos pertencentes à fase de projeto de aplicações	
1.	O projetista usa um ambiente de edição para compor aplicações SOA. O ambiente possui um editor BPMN que permite editar e customizar aplicações.
2.	O ambiente BPM oferece acesso a um catálogo de processos de negócios UBL (BEZERRA, 2011). O projetista pode usar processos de negócios UBL prontos na composição da sua aplicação.
3.	O projetista determina a funcionalidade requerida (usando a ontologia UBL), informa restrições de QoS (usando a ontologia de QoS) para às

	atividades a serem executadas por serviços *web* e comunica o uso ou não da reputação do provedor.
4.	O *crawler*, munido de uma instância da expressão de descoberta (contendo requisitos funcionais associado ao processo UBL em questão e aspectos de QoS informados), é acionado para descobrir serviços candidatos.
5.	Com base na expressão da descoberta, o *crawler* vai à procura de serviços nos repositórios de serviços (disponíveis na federação) visando montar uma lista de serviços candidatos, ou seja, uma lista composta de serviços que respeitam os requisitos expressos na expressão da descoberta.
6.	Uma lista de serviços é devolvida como resultado da descoberta, podendo ou não estar classificada usando como critério a reputação do provedor.
7.	Se a lista de serviços candidatos retornada pelo *crawling* for um (1), gera-se o SLA para o serviço e passa-se para o passo 8. Senão duas situações devem ser consideradas: a) Nenhum serviço: o *crawling* sugere ao projetista que relaxe requisitos para que a lista de candidatos seja formada; b) N serviços *perfeitos* (um serviço é perfeito quando é atingido o grau máximo (100%) de conformidade entre o que se deseja e o que se encontra): o provedor pode escolher um serviço a seu critério, ou pode pedir ao *crawling* que faça uma classificação dos serviços em função da reputação do provedor. Neste caso, o primeiro é selecionado para implementar a atividade.
8.	A lista de serviços candidatos é passada ao ambiente de edição BPM para ser associada à atividade da aplicação e enquanto houver atividades a serem implementadas por serviços *web*, os passos acima se repetem.
9.	A aplicação é convertida em um formato executável (WS-BPEL), disponibilizada para execução e armazenada em um repositório de aplicações.

Quadro 17 – Resumo dos passos operacionais executados durante a fase de projeto de aplicações.

O diagrama de sequência da UML, mostrado na Figura 26 e construído com base nas operações apresentadas no Quadro 17, ilustra a sequência de mensagens trocadas entre os diversos elementos conceituais envolvidos no projeto de aplicações SOA.

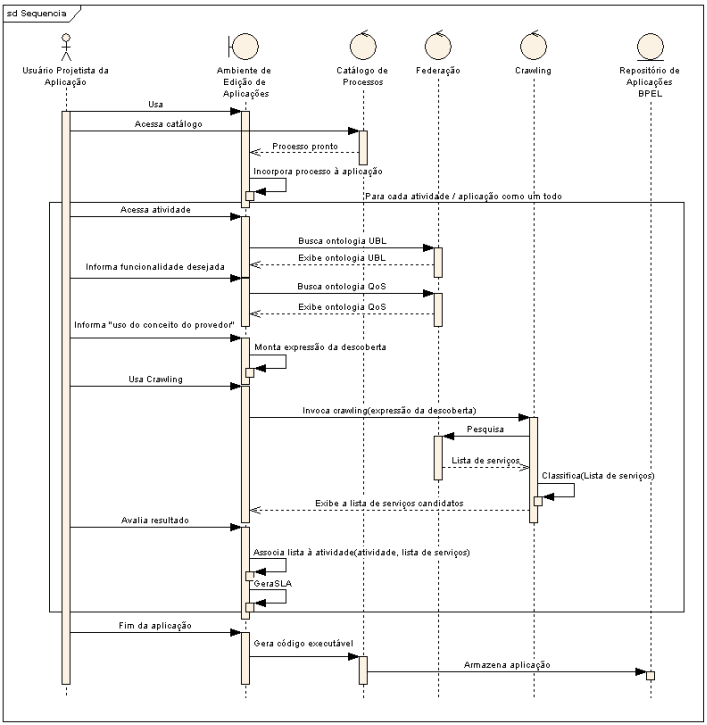


Figura 26 - Diagrama de sequência para projeto de aplicações.

O Quadro 18 resume os papéis e funções desempenhadas pelo projetista e pelo ambiente BPM na abordagem assistida no projeto de aplicações.

Aspecto	Abordagem: Assistida
Quem identifica e informa necessidade	Ambiente BPM e Projetista
Quem avalia (desejo x resultado)	Ambiente BPM "*crawling*" e Projetista
Interatividade	Alta
Comportamento/ Atitude	Projetista (ativo) e Ambiente BPM (reativo)

Quadro 18 – No projeto de aplicações o ambiente BPM assume características pertencentes à abordagem assistida.

A justificativa para a adoção da abordagem assistida na fase de projeto de aplicações reside em dois aspectos chave. O primeiro aspecto tem a ver com o fato do ambiente de edição de aplicações BPM fornecer o contexto dos processos de negócios (o tipo de processo, seu fluxo e atividades) e a semântica dos aspectos de QoS a serem usados e considerados na descoberta. Isto é feito através do uso de duas ontologias, a UBL e a de QoS. Elas dão maior riqueza semântica à descoberta e otimizam a concepção de aplicações SOA, pois resolvem problemas semânticos e reduzem problemas de interoperabilidade, já que clientes e provedores usam um vocabulário comum quando interagem com o mecanismo de descoberta, seja para publicar ou obter serviços.

O segundo aspecto refere-se ao apoio dado ao projetista no desenvolvimento da sua aplicação. Este apoio está diluído em dois pontos básicos. Um deles está associado ao uso do catálogo de processos de negócios baseado no padrão UBL. Seu uso permite ao projetista economizar tempo e esforço para definir a sua aplicação, pois basta que ele navegue no catálogo e, a partir dele, use processos prontos, seja para construir ou customizar alguma parte da aplicação. O outro ponto está associado ao *crawling* que assiste o projetista desde a escolha dos atributos de QoS, com possíveis valores associados a eles, até o resultado da descoberta. Por exemplo, em uma situação de exceção, o *crawling* informa quais características de QoS dificultam a descoberta, sinalizando-as para que o projetista da aplicação possa reavaliar o uso das mesmas.

Estes dois aspectos somados permitem um alinhamento mais próximo entre o nível de negócios e o nível de sistemas, possibilitando uma mais transparente e ágil integração BPM&SOA.

4.2.2.3 Fase de Execução de Aplicações SOA (*on-line*)

Concluído o projeto de uma aplicação, a mesma fica à disposição para ser executada. Os passos referentes à execução de aplicações são resumidos no Quadro 19.

Resumo descritivo dos passos pertencentes à Fase de execução de aplicação	
1..9	Passos referentes ao projeto de aplicações (fase *off-line*)
10.	Um usuário usa um ambiente para executar aplicações.
11.	O usuário navega no repositório de aplicações e identifica a aplicação desejada. 11.1 0 ambiente de execução verifica se cada um dos serviços *web* está disponível e acessível Situações: a) Serviço disponível e acessível: executa serviço; b) Serviço indisponível ou inacessível: usa a lista de serviços candidatos e verifica qual serviço está disponível e acessível 11.2 Caso nenhum serviço candidato esteja disponível e acessível, invoca mecanismo de descoberta para selecionar um (1) novo serviço 11.3 Descobre novo serviço; 11.4 e 11.5 Escolhe serviço, usando informações sobre a reputação do provedor do serviço; 11.6 Vincula serviço a aplicação. 11.7 Gera SLA e requisita a assinatura do contrato pelo cliente e provedor de serviço selecionado.
12.	Através de um motor WS-BPEL, a aplicação é executada.

Quadro 19 – Resumo dos passos executados durante a fase de execução de aplicações.

O conjunto de mensagens trocadas entre os diversos elementos conceituais para a execução de aplicações SOA é apresentado no diagrama de sequência mostrado na Figura 27, conforme conjunto de passos descritos no Quadro 19.

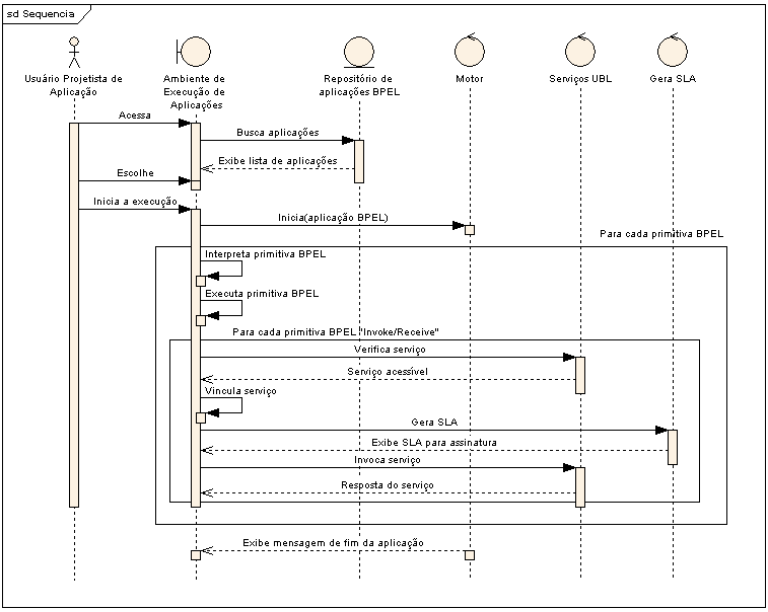


Figura 27 – Diagrama de sequência para execução de aplicações SOA.

No Quadro 19 (passo 11) e na Figura 27 (mensagem *Verifica serviço*) uma exceção pode ocorrer quando algum serviço previamente vinculado à aplicação não estiver disponível e acessível.

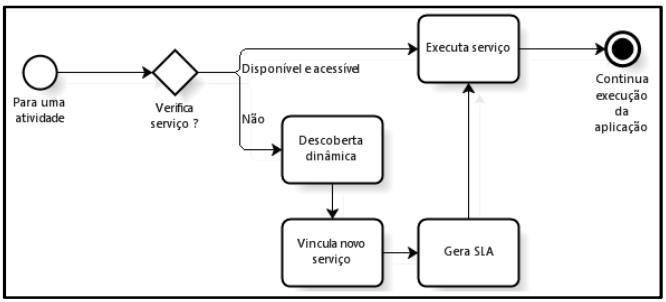


Figura 28 – Diagrama de atividades da UML que representa a exceção associada à execução de aplicação SOA.

Neste caso, um algoritmo de descoberta dinâmica é invocado para descobrir um novo serviço. O algoritmo inicia a busca pela lista de serviços candidatos associada à aplicação e, se necessário, faz uma

descoberta efetiva nos repositórios de serviços gerenciados pela federação. O diagrama de atividades da UML da Figura 28 mostra como este fluxo ocorre.

No instante em que uma aplicação SOA está em execução, o ambiente BPM assume características presentes na abordagem automática, conforme mostra o Quadro 20. Nesta fase, o ambiente de execuções de aplicações SOA é uma entidade com comportamento altamente ativo. É ela quem avalia resultados, informa necessidades e toma decisões (podendo relaxar critérios para encontrar serviços).

Aspecto	Abordagem: Automática
Quem identifica e informa necessidade	Ambiente BPM
Quem avalia (desejo x resultado)	Ambiente BPM
Interatividade	Nenhuma
Comportamento/Atitude	Ambiente BPM (altamente ativo)

Quadro 20 – Na execução de aplicações o ambiente BPM assume características pertencentes à abordagem automática.

O uso da abordagem automática na fase de execução de aplicações SOA é justificado em função das características peculiares associadas à execução de aplicações SOA. Ou seja, nesta fase, um usuário executor de aplicações está interessado em usar o ambiente para executar alguma aplicação SOA pronta, sem estar preocupado se todos os serviços associados à aplicação estão disponíveis ou se para todas as atividades presentes na aplicação, há serviços associados e vinculados. Portanto, nesta fase, o usuário executor de aplicações teoricamente não tem competência e conhecimento, seja para alterar a lógica da aplicação ou para sugerir algum serviço a ser vinculado na execução da aplicação SOA, ficando a cargo do ambiente tomar as decisões para que a aplicação SOA seja executada com sucesso. Portanto, o uso da abordagem automática na fase de execução de aplicações contribui para tornar mais transparente e ágil a integração BPM&SOA.

4.2.2.4 Exemplo

Com objetivo de elucidar o funcionamento operacional do modelo proposto, um exemplo passo a passo foi elaborado envolvendo

as fases de projeto e execução de aplicações SOA, conforme descrito nas Seções 4.2.2, 4.2.2.2 e 4.2.2.3.

O exemplo trata de uma aplicação para gerenciar a manufatura de automóveis. Inicialmente, o projetista da aplicação utiliza um ambiente como suporte à edição de aplicações BPMN. Ele cria um novo projeto e inicia a elaboração da aplicação, conforme ilustra a Figura 29.

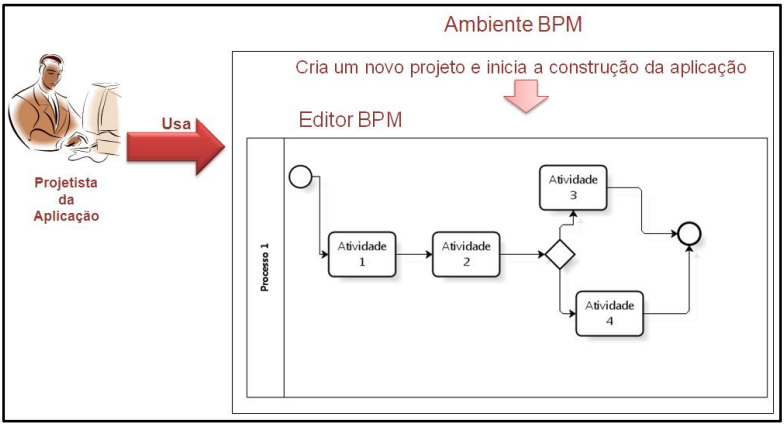


Figura 29 - Projetista na tarefa de criar aplicações.

Em seguida, o projetista identifica a necessidade de reposição de matéria-prima e decide realizar uma compra. O projetista ao invés de usar o ERP (*Enterprise Resource Planning*) tradicional para fazer o pedido, decide inserir, no projeto da aplicação, um conjunto de atividades prontas para a realização da compra de matéria-prima. Dessa forma, o projetista acessa e identifica o processo Pedido de Compra (*Ordering*) UBL presente no catálogo e integrá-o à aplicação em desenvolvimento.

Para um melhor entendimento dos processos UBL, a Figura 30 apresenta e descreve o funcionamento do processo de compra (*Ordering*) UBL. No processo de compra UBL, duas entidades estão envolvidas: a parte solicitante (*BuyerParty*) e a parte fornecedora (*SellerParty*). As caixas centrais com descrições sublinhadas (*Order*, *Order Response Simple*, *Order Response*, *Order Change* e *Order Cancellation*) representam os documentos trocados entre as partes envolvidas no processo. Os documentos trocados são representados via XML e regidos por seus respectivos *Schemas* UBL. Há também os losangos, que são símbolos gráficos usados para tomada de decisão, enquanto que os retângulos, com cantos arredondados, configuram

atividades a serem executadas por um determinado participante do processo. As setas representam o fluxo de informações presentes no processo de negócio *Ordering* UBL. Por fim, os círculos concêntricos representam o final do processo.

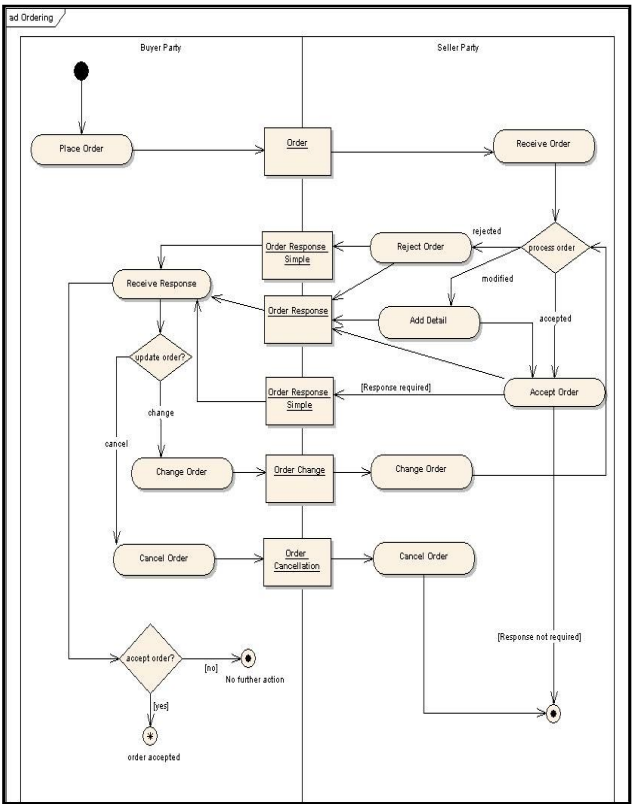


Figura 30 - Especificação do Processo de Solicitação de "Compra de Material" (*Ordering*) UBL.
Fonte: (OASIS, 2006).

O funcionamento do processo de *Ordering* UBL inicia com um pedido formulado pelo solicitante (lado superior esquerdo da Figura 30). Este pedido é enviado ao fornecedor, que analisa (losango *process order*) e envia uma resposta ao solicitante (lado superior direito da Figura 30). O solicitante pode recusar a transação, alterar o pedido ou

aceitar a negociação e concluir sua participação no processo. Do lado do fornecedor, a partir do momento que o pedido é aceito, ele finaliza sua participação na transação. Por outro lado, o fornecedor pode alterar o pedido ou mesmo cancelar o pedido de compra, havendo alguma modificação no mesmo.

A sequência do exemplo da construção da aplicação para manufatura de automóveis ocorre com a Figura 31, que mostra como fica o ambiente de edição de aplicações, após a incorporação do processo *Ordering* UBL à aplicação (parte inferior da Figura 31).

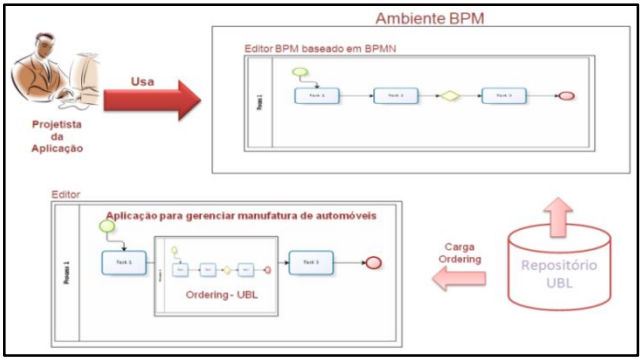


Figura 31 - Ambiente de edição de aplicações incorporando o processo *Ordering* UBL à aplicação de manufatura de automóveis.

Ainda, o projetista pode importar outros processos prontos ou fazer alguma customização na aplicação. A próxima tarefa do projetista é buscar serviços para implementar cada uma das atividades que compõem a aplicação. Para isso, ele escolhe uma atividade, informa o nome do serviço (por exemplo: *AcceptOrderBuyerParty*, usando a ontologia UBL), registra as características de QoS (usando a ontologia de QoS) e escolhe usar ou não a reputação do provedor para classificar serviços. No mesmo instante, o ambiente de edição monta a expressão da descoberta, supondo que a funcionalidade *AcceptOrderBuyerParty* é requerida para implementar a atividade escolhida e um conjunto de características de QoS, tal como *Performance* (característica defina a partir dos atributos *ResponseTime* e *ExecutionTime*) são desejados, conforme mostra a Figura 32.

F: Funcionalidade requerida

A ontologia UBL determina as capacidades para o serviço requerido
Exemplo:
Nome: *UBL/.../AcceptOrderBuyerParty*

Q: Características de QoS desejados

A ontologia QoS determina as características desejadas para o serviço requerido.
Exemplo:
Característica: *Peformance*
Atributo=*ResponseTime* valor=1s
Atributo=*ExecutionTime* valor=5s

Figura 32 - Instância da expressão de descoberta para o exemplo fictício.

Depois da expressão da descoberta estar formada, o projetista aciona o *crawling*, cuja finalidade básica é descobrir serviços candidatos e assistir o projetista na escolha do serviço mais adequado. Neste ponto, o *crawling* acessa repositórios de serviços (disponíveis na federação) e verifica quais serviços atendem os critérios presentes na expressão da descoberta.

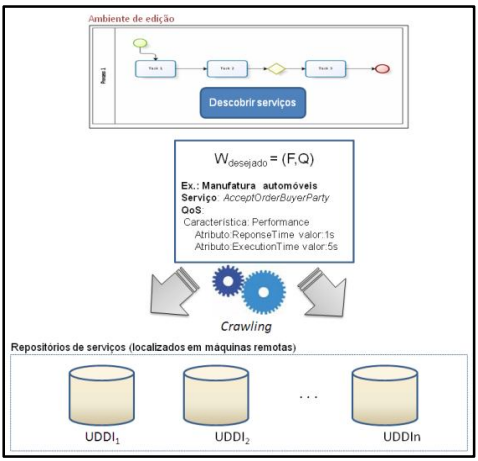


Figura 33 - Ilustração do *crawling* na busca por serviços.

No exemplo da aplicação para manufatura de automóveis, o *crawling* verifica sintaticamente quais serviços têm condições de implementar a atividade *AcceptOrderBuyerParty* UBL e, ao mesmo tempo, testa quais destes serviços respeitam os critérios de qualidade exigidos, conforme ilustra a Figura 33.

O resultado devolvido pelo *crawling* é organizado em uma lista denominada de serviços candidatos. A Figura 34 exemplifica um resultado hipotético fornecido pelo *crawling*.

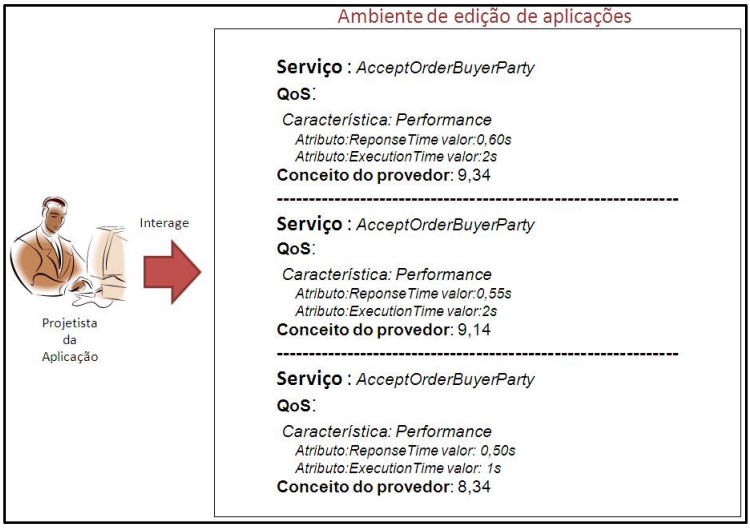


Figura 34 - Exemplo de retorno do *crawling*.

A lista de serviços candidatos apresentada (somente com serviços perfeitos), é classificada usando a reputação do provedor. Por este motivo, o serviço mais adequado é o primeiro da lista, ou seja, é o serviço com reputação mais alta que os demais, embora seus valores de QoS não sejam tão bons, comparados com os valores de QoS apresentados pelos outros serviços candidatos listados.

Esta mesma sequência de tarefas deve ser feita para todas as atividades presentes na aplicação. Caso o projetista prefera, ele pode informar um conjunto de características globais de QoS (para a aplicação com um todo) e acionar o *crawler*. Neste caso, o *crawling* sai a procura de serviços candidatos, os quais, após descobertos, são vinculados automaticamente a cada uma das atividades da aplicação. É possível que para uma dada atividade não haja serviços candidatos vinculados. Neste caso, a descoberta dinâmica, em tempo de execução de aplicação, fica com a responsabilidade de descobrir e vincular um serviço para que a aplicação seja executada com sucesso.

A conclusão do projeto da aplicação acontece quando o projetista solicita ao ambiente de edição de aplicações que realize o mapeamento

da aplicação para o formato WS-BPEL. Ele apenas informa o local onde será armazenada a aplicação e o nome do arquivo. O mecanismo de mapeamento faz automaticamente a conversão para WS-BPEL (detalhes da conversão da aplicação pronta para o formato WS-BPEL são explicados na Seção 4.2.3.2).

A sequência do exemplo ocorre com a fase de execução da aplicação de manufatura de automóveis. Neste momento, um usuário executor identifica e escolhe a aplicação manufatura de automóveis. O ambiente de execução de aplicações faz a carga da mesma e apresenta uma interface gráfica para que o usuário executor possa iniciar a execução da aplicação e acompanhar o seu estado. A Figura 35 ilustra os passos para execução da aplicação de manufatura de automóveis.

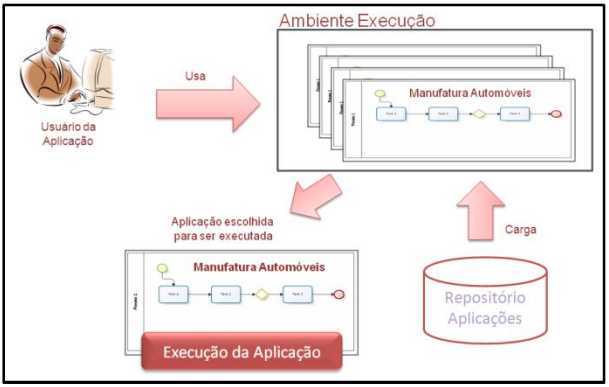


Figura 35 – Ilustração da execução de uma aplicação SOA.

O ambiente inicia a execução da aplicação. Para cada atividade associada a um serviço na fase de projeto, o ambiente de execução verifica a disponibilidade do serviço alocado. Caso algum serviço não esteja disponível e acessível, ele invoca o algoritmo de descoberta dinâmica para descobrir e vincular um (1) novo serviço à aplicação. Por fim, de forma semelhante à fase de projeto de aplicações, um contrato (SLA) entre cliente e provedor de serviço é gerado para que as partes possam assiná-lo.

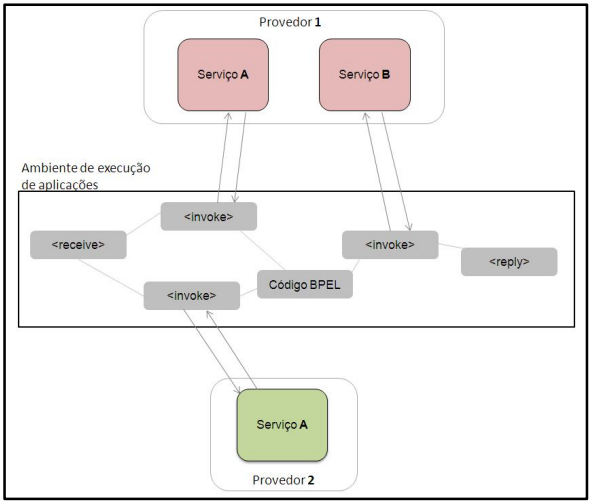


Figura 36 - Ilustração da execução da aplicação para manufatura de automóveis em WS-BPEL.

A Figura 36 ilustra a aplicação de manufatura de automóveis sendo executada com os serviços mais adequados para o momento. Nela, por exemplo, a primitiva WS-BPEL <*invoke*> solicita a chamada do Serviço A, ofertado pelo Provedor 1, que implementa uma dada atividade da aplicação para manufatura de automóvel.

4.2.3 Visão dos Módulos Conceituais

A visão dos módulos conceituais possibilita ter uma visão das unidades de código que formam a arquitetura conceitual do modelo proposto. Ela é composta por cinco módulos e um repositório interno de serviços, conforme ilustra a Figura 37.

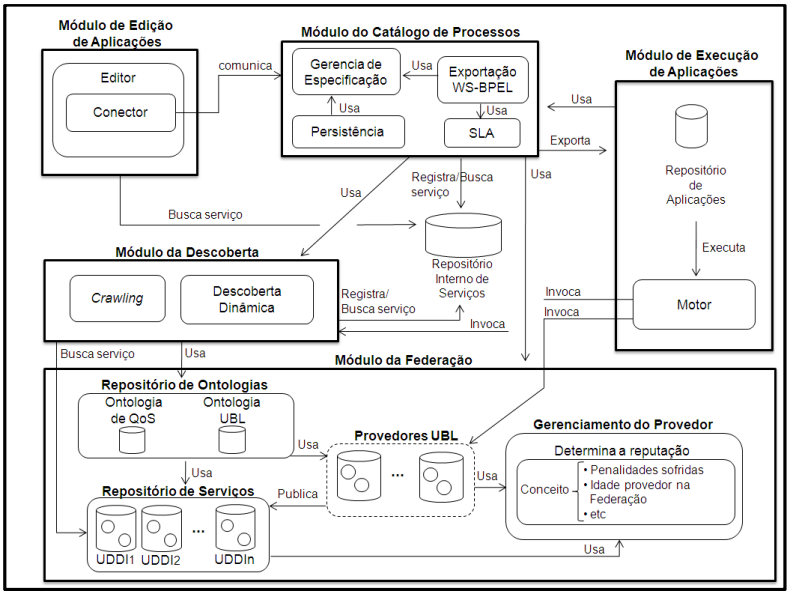


Figura 37 – Arquitetura conceitual do modelo proposto.

O módulo de *Edição de Aplicações* é o ponto de acesso do usuário projetista para projetar aplicações baseadas no padrão UBL. Dentro dele há um conector (*plug-in*) que permite acesso ao catálogo de processos de negócios.

O módulo do *Catálogo de Processos de Negócios* é outro elemento que compõe a arquitetura conceitual do modelo proposto. Ele possui quatro submódulos. O submódulo *Gerência de Especificações* responsável pela especificação usada no catálogo (neste caso a UBL). O submódulo *Exportação da Aplicação para um Formato Executável* faz a conversão da aplicação pronta para formato o executável (WS-BPEL), o submódulo de *Geração de SLA*, que gera um contrato de uso de serviço entre o cliente e o provedor do serviço escolhido para uso. O submódulo *Persistência* que gerencia o armazenamento em memória persistente das aplicações em desenvolvimento, ou seja, antes de serem convertidas para formato executável.

É importante salientar que o módulo que define o catálogo de processos de negócios agrega diversas funcionalidades, implementadas através de serviços *web*, conforme é comentado no Capítulo 5. A

justificativa para esta opção é tornar o modelo proposto independente da ferramenta de edição de aplicação escolhida (no caso IBM *WebSphere Business Modeler*) e usada como ambiente de edição de aplicações.

O módulo do *Ambiente de Execução de Aplicações* é composto pelo submódulo que representa um motor de aplicações WS-BPEL, cujo objetivo é executar aplicações no formato WS-BPEL, e por um repositório WS-BPEL com aplicações prontas para serem executadas.

O módulo da *Federação* é composto pelo repositório de ontologias (que armazena as ontologias UBL e QoS), por um conjunto de repositórios de informações sobre serviços, por um submódulo que determina a reputação (o conceito) dos provedores e mais um elemento lógico (representado por uma linha tracejada no centro do módulo Federação - Figura 37) que mostra a participação dos provedores de serviços na federação.

O módulo da *Descoberta de Serviços* é composto por dois submódulos: um submódulo de *Crawling* que busca serviços candidatos em tempo de projeto de aplicações e um submódulo de *Descoberta dinâmica* que realiza pesquisa e seleção em tempo de execução de aplicações.

No centro da Figura 37 há um *Repositório Central de Serviços* usado para publicar e descobrir serviços internos ao funcionamento do modelo proposto, já que todo o modelo é baseado no paradigma SOA, cujos detalhes de implementação são explicados no Capítulo 5.

As subseções que seguem detalham o funcionamento conceitual dos cinco módulos que compõem a arquitetura conceitual do modelo proposto.

4.2.3.1 Módulo de Edição de Aplicações

O módulo de edição de aplicações possui um conector que faz a ligação entre o editor de aplicações e o catálogo de processos de negócios. Dentro dele há quatro submódulos (Importar processos do catálogo, Informar QoS, Invocar *crawling* e Exportar Processo para Linguagem de Execução).

O submódulo, *Importar Processos do Catálogo*, é responsável por importar processos (aqueles escolhidos pelo usuário projetista) contidos no catálogo para dentro do editor BPM. Seu funcionamento é descrito pelo diagrama de sequência da UML, mostrado na Figura 38.

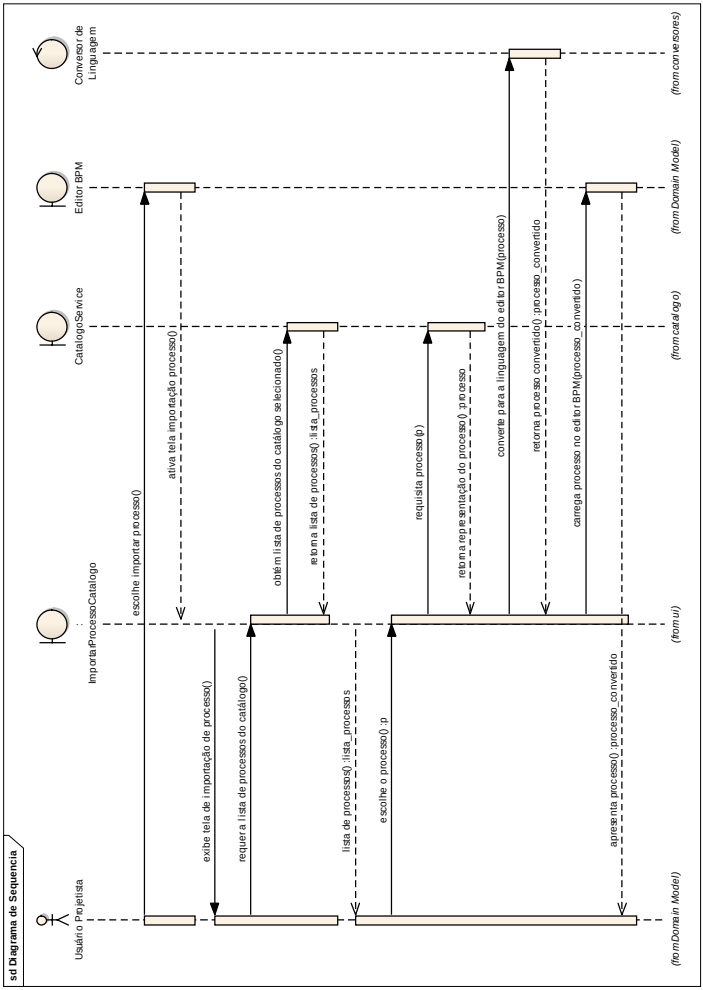


Figura 38 – Diagrama de sequência da UML que detalha o funcionamento conceitual do submódulo que mostra a importação de processos armazenados e disponíveis no catálogo.
Fonte: Bezerra (2011).

No diagrama apresentado na Figura 38, um usuário projetista requisita ao editor BPM a importação de um processo do catálogo. Este ativa o componente *ImportarProcessoCatalogo*, responsável pela

interface gráfica de importação. Em seguida, o usuário solicita a lista de processos, fazendo com que a interface gráfica se comunique com o catálogo e obtenha a lista de processos disponíveis. Com base nessa lista, o usuário escolhe qual deles deseja importar. O processo escolhido é requisitado ao catálogo de processos, que o envia à interface gráfica. Esta invoca o conversor de linguagem, que irá converter o processo obtido para a linguagem utilizada pelo editor BPM. Em seguida, o processo já convertido é carregado pelo editor e, finalmente, exibido ao usuário projetista.

O segundo submódulo, *Informar QoS*, permite a um usuário projetista indicar restrições de qualidade QoS, tanto para atividades individuais como para o conjunto de atividades pertencentes à aplicação como um todo, conforme mostra o diagrama de sequência da UML apresentado na Figura 39.

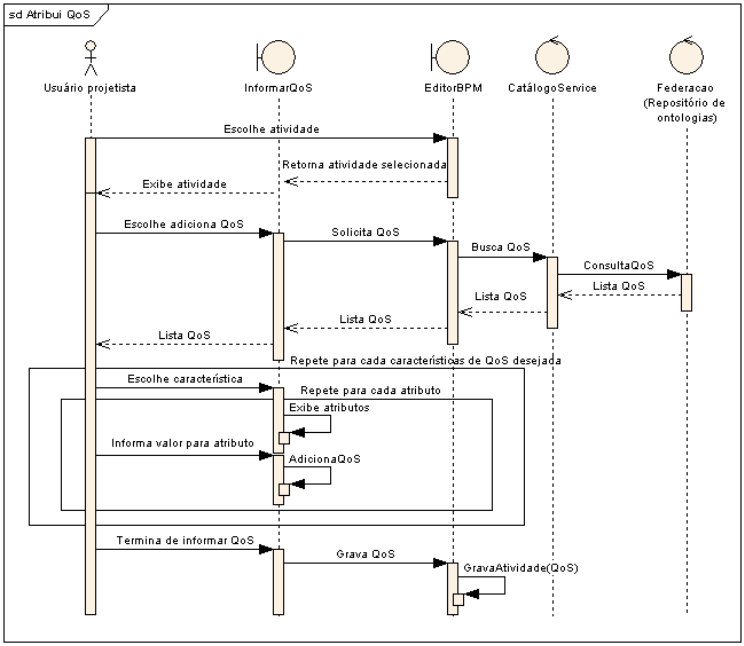


Figura 39 – Diagrama de sequência da UML que detalha o funcionamento conceitual do submódulo usado para informar QoS.

Após o processo ter sido importado, o usuário projetista seleciona alguma atividade dentro da aplicação. O editor de aplicações envia uma requisição ao catálogo que repassa à Federação, que consulta o

repositório de ontologias, e retorna um conjunto de características de QoS disponíveis. O usuário projetista escolhe quais características quer utilizar e informa valores para os atributos que compõem a característica de QoS desejada. Quando o usuário projetista termina de atribuir valores para os atributos de QoS, o editor associa e grava as informações de QoS à atividade selecionada.

O terceiro submódulo, *Invocar crawling*, solicita a execução da descoberta e associa serviços candidatos às atividades correspondentes, conforme mostra o diagrama de sequência da UML, apresentado na Figura 40. Após o usuário projetista definir a funcionalidade requerida, informar QoS e escolher usar ou não conceito do provedor para classificar serviços, o ambiente de edição de aplicações monta a expressão e, com base nela, solicita ao catálogo que invoque o *crawling*. O resultado do *crawling* (uma lista de serviços candidatos) é apresentada ao usuário projetista para análise.

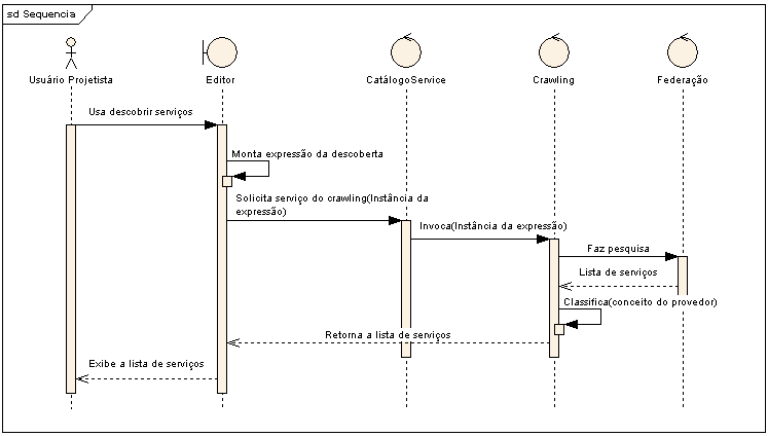


Figura 40 - Diagrama de sequência da UML que detalha o funcionamento conceitual do submódulo que invoca o *crawling*.

É importante salientar que o *crawling* pode ser usado pelo projetista diversas vezes, principalmente nas situações em que os serviços retornados não atendem as exigências requisitadas pelo projetista da aplicação. Esta forma de assistir o projetista permite que ele faça ensaios com o mecanismo de descoberta com objetivo de verificar se há serviços disponíveis que possam implementar uma dada atividade,

observando, por exemplo, se as restrições de QoS estão demasiadamente exageradas ou não. Somado a isso, o conhecimento prévio do conjunto de serviços candidatos a serem usados para implementar atividades de uma aplicação otimiza o desempenho da aplicação e dá mais agilidade na integração do nível de negócios com o nível dos sistemas.

No quarto submódulo, *Exportar Processo para Linguagem de Execução*, o usuário projetista, após a aplicação estar pronta, ou seja, após a orquestração da mesma estar definida e os serviços candidatos estarem vinculados às atividades, escolhe-se exportar para a linguagem de execução, a fim de poder executá-la posteriormente em um ambiente de excução de aplicações, conforme mostra a Figura 41.

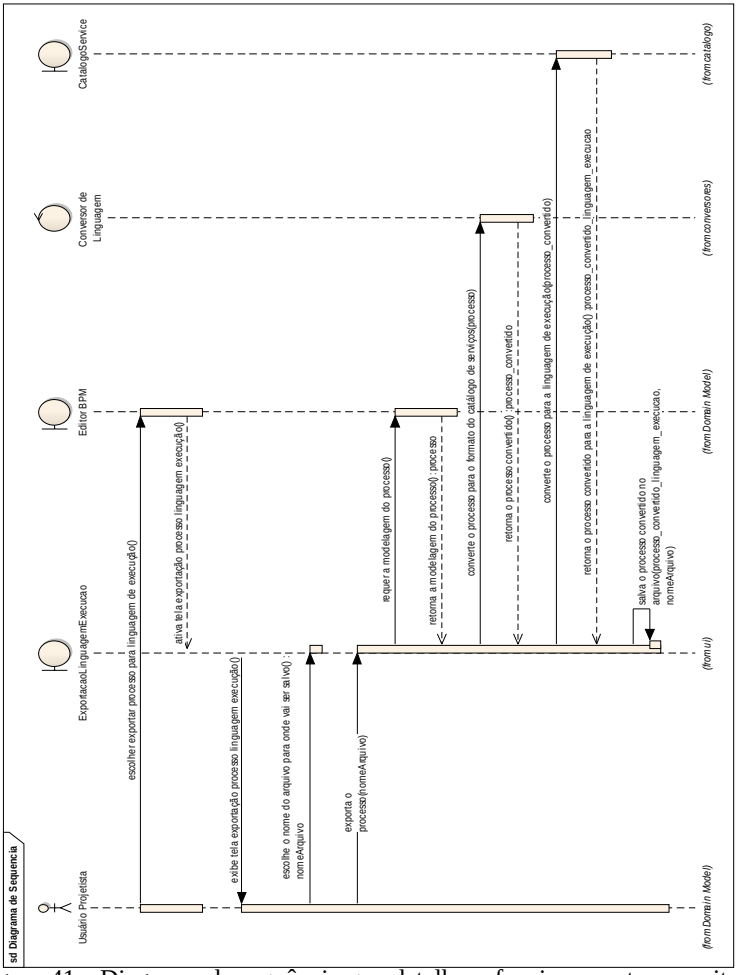


Figura 41 - Diagrama de sequência que detalha o funcionamento conceitual do submódulo Exportação para Linguagem de Execução.
Fonte: Bezerra (2011).

No diagrama de sequência da Figura 41, o ambiente de edição ativa a interface gráfica de Exportação de Processo para Linguagem de Execução, que é exibida ao usuário. Este escolhe qual o nome que deseja atribuir ao arquivo que será criado e procede com a exportação.

Nesse momento, o catálogo é requisitado para efetuar a conversão para o formato de execução. O retorno é então salvo no arquivo escolhido pelo usuário, conforme mostra o diagrama de sequência da UML apresentado na Figura 41.

O mecanismo efetivo que converte a aplicação em código executável é realizado mediante a invocação de um submódulo presente no catálogo. Por esta razão, o conjunto de passos para exportar aplicações prontas em código executável será descrito no Módulo do Catálogo de Processos de Negócios.

4.2.3.2 Módulo do Catálogo de Processos de Negócios

A arquitetura conceitual do Módulo do Catálogo de Processos de Negócios é composta por quatro submódulos (Gerência de Especificações, Persistência, Exportação e Geração de SLA) e um Modelo de dados.

O submódulo de *Gerência de Especificações* permite que o catálogo armazene as especificações de processos de negócio. No caso, a especificação UBL contém um conjunto de processos, sua categorização hierárquica e ontológica, os participantes envolvidos, os documentos trocados e as *interfaces* dos serviços que implementam as atividades dos processos.

O submódulo de *Persistência* é responsável por buscar, armazenar e recuperar os processos desenvolvidos pelos usuários. Também faz o armazenamento da localização dos arquivos que compõem as especificações de processos de negócio disponíveis no catálogo. A forma como os dados são armazenados, como por exemplo, num banco de dados relacional ou num arquivo XML, é de responsabilidade desse submódulo.

O *Modelo de dados* é uma abstração lógica responsável por representar processos, especificações e demais elementos contidos no catálogo.

A especificação conceitual detalhada do catálogo de processos de negócios não faz parte dos objetivos desta tese. Desta forma, para maiores detalhes da sua arquitetura conceitual e de implementação, consultar o trabalho de Bezerra (2011). Entretanto, o submódulo de *Exportação para Linguagem de Execução* é descrito para mostrar os pormenores do processo de conversão e facilitar o entendimento do funcionamento global do modelo e da desejada integração BPM&SOA.

O submódulo *Exportação para Linguagem de Execução* executa a operação *exportarProcesso* que tem como entrada um processo, no

formato reconhecido pelo Catálogo de Processos de Negócio. A saída é um processo executável, conforme mostra a Figura 42.

Primeiramente, uma aplicação pronta é lida, obtendo-se a lista dos seus documentos trocados e de suas atividades. Dependendo do elemento obtido (documento trocado ou atividade), o submódulo de Gerência de Especificações é requisitado a fim de fornecer as definições desses documentos e os serviços vinculados a cada atividade.

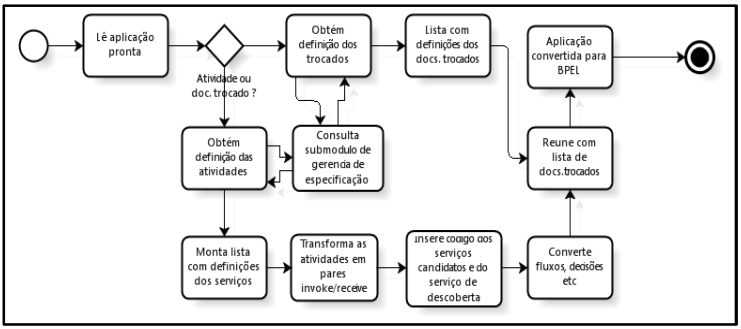


Figura 42 - Diagrama de atividades da UML que detalha o funcionamento conceitual do submódulo de conversão de aplicações.

Cada atividade da aplicação é transformada em um par <*invoke*> e <*receive*> WS-BPEL. Juntamente com este código, o mecanismo de exportação faz a geração do código WS-BPEL que armazena a lista de serviços candidatos associada à atividade e do código WS-BPEL que invoca a descoberta dinâmica, para o caso de nenhum dos serviços candidatos estarem disponíveis e acessíveis em tempo de execução de aplicação.

Continuando o fluxo da Figura 42, o mecanismo de conversão faz o mapeamento das estruturas de fluxo da aplicação para as equivalentes na linguagem WS-BPEL, permitindo que a aplicação seja executada obedecendo às mesmas regras de controle da aplicação original.

O passo seguinte é agregar à aplicação convertida os demais arquivos necessários ao *deployment* no servidor de aplicações. Esses arquivos são:

- Arquivos com as definições das *interfaces* dos serviços invocados, juntamente com os endereços de localização dos mesmos;

- Arquivos que descrevem os documentos utilizados o pela aplicação (os XML Schemas dos documentos UBL trocados, pois os mesmos estão já em formato WS-BPEL).

O último passo faz uso do submódulo de *Geração de SLA*, responsável pela geração de um contrato de uso dos serviços. Este submódulo usa como inspiração o trabalho de Cancian, Rabelo e Von Wangenheim, (2009) para definir os elementos que fazem parte do contrato.

A exportação da aplicação BPMN para a linguagem de execução de processos é totalmente automatizada e transparente, de forma que o usuário projetista não precise ter conhecimento técnico para efetuar a exportação da aplicação SOA para o ambiente onde ela será executada. O ambiente é responsável por fazer a conversão e a vinculação dos serviços que irão executar em conjunto com a aplicação SOA.

4.2.3.3 Módulo que define a Federação

Embora não seja o foco do trabalho definir conceitualmente a federação, o quinto e último módulo descreve os elementos conceituais básicos para garantir o uso da federação no modelo proposto. Dessa forma, a federação é composta de três submódulos. Um submódulo que descreve conceitualmente o repositório de ontologias (formado pelas ontologias QoS e UBL). Um submódulo que define o comportamento dos repositórios de informações sobre serviços e um submódulo que define o gerenciamento de informações dos provedores de serviços.

4.2.3.3.1 Submódulo que define o Repositório de Ontologias

Fazem parte do repositório de ontologias, a ontologia de QoS e a ontologia de processos UBL. A ontologia de processos UBL é baseada na especificação *Universal Business Language* v. 2 (OASIS, 2006), sendo elaborada em três etapas macros. A primeira compreendeu o estudo e análise da especificação UBL. A segunda contemplou a identificação dos atores e a interpretação dos diagramas responsáveis pelo detalhamento dos processos de negócios UBL. A terceira etapa foi relativa à edição da ontologia, conforme é explicado na Seção 5.2.6.1. Ao final do estudo da especificação, criou-se uma representação, conforme mostra a Figura 43.

<nome_da_especificação>/<categoria_processo>/<nome_processo>/
<participante_executor_atividade>/<nome_atividade>

Figura 43 – Representação dos processos de negócios desenvolvida a partir dão padrão UBL.

Nela, o elemento <*nome_da_especificação*> representa o nome do padrão de processo usado (neste caso a UBL). O elemento <*categoria_processo*> indica o nome de uma categoria de processos dentro da especificação em questão. O elemento <*nome_processo*> diz qual processo de negócio está sendo descrito. O elemento <*participante_executor_atividade*> informa qual é o ator ou o participante que executa determinada atividade e o elemento <*nome_atividade*> determina o nome da atividade dentro de um processo. Por exemplo, o Quadro 21 ilustra a classificação proposta na Figura 43 para as instâncias dos processos *Ordering* e *Payment* pertencentes à especificação UBL.

Categoria	Processo	Participante	Atividade
Ordering	*OrderingProcess*	*SellerParty*	*AddDetail*
Ontologia: UBL/Ordering/OrderingProcess/SellerParty/AddDetail			
Ordering	*OrderingProcess*	*BuyerParty*	*PlaceOrder*
Ontologia: UBL/Ordering/OrderingProcess/BuyerParty/PlaceOrder			
Payment	*PaymentProcess*	*AccountingCustomer*	*AuthorizePayment*
Ontologia:UBL/Payment/Paymentprocess/AccountingCustomer/AuthorizePayment			

Quadro 21 - Exemplo de categorização resultante dos processos UBL.

Na primeira coluna do Quadro 21, representa-se a categoria do processo (*Ordering* e *Payment*). Na segunda coluna, mostra-se o nome processo (*OrderingProcess* e *PaymentProcess*). Na coluna *Participante* informam-se as partes envolvidas no processo (*SellerParty*, *BuyerParty* e *AccountingCustomer*). Finalmente, na coluna denominada *Atividade*, representam-se as atividades (*AddDetail*, *PlaceOrder* e *AuthorizePayment*) presente na execução de uma aplicação.

A construção da ontologia de QoS foi inspirada na ontologia proposta por Tondello (2008), a qual teve como base a definição de QoS (OMG, 2006). Como a ontologia proposta por Tondello (2008) não define o conjunto de características e atributos de QoS, e para evitar a criação de mais uma classificação, usou-se o documento intitulado *QoS*

for Web Services: Requirements and Possible Approaches do W3C (2003) como base para determinar as características e atributos de QoS no contexto deste trabalho. Outra razão que motivou o uso da relação de QoS para serviços do W3C foi a necessidade de minimizar problemas de interoperabilidade, através do uso de padrões e recomendações abertas, permitindo que uma mais transparente e ágil integração entre os elementos conceituais que formam a integração BPM&SOA. O Quadro 22 apresenta o conjunto de características definidas a partir da análise do documento *QoS for Web Services: Requirements and Possible Approaches* do W3C (2003).

Características	Descrição
Desempenho (*Performance*)	Representa quão rápido uma requisição pode ser completada. Ele pode ser medido ou determinado em termos de: *throughput* (que corresponde ao número de requisições atendidas pelo serviço em um intervalo de tempo), tempo de resposta (tempo para completar uma requisição), latência (*round-trip delay*, tempo entre o envio de uma requisição e o recebimento da respectiva resposta), tempo de execução (tempo que um serviço leva para processar sua sequência de atividades), tempo de transação (tempo total que uma transação leva para ser concluída) entre outros. Em geral, segundo W3C (2003), um serviço deve idealmente prover alto *throughput*, tempo de resposta baixo, latência baixa, tempo de execução baixo e baixo tempo para completar transações.
Confiabilidade (*Reliability*)	É capacidade de um serviço executar adequadamente um conjunto de operações solicitadas de acordo com um conjunto de condições estabelecidas. Em um nível mais específico, confiabilidade está relacionada à entrega garantida e ordenada de mensagens sendo transmitidas e recebidas. A medida global de confiabilidade de um serviço está relacionada ao número de falhas por dia, semana, mês ou ano.
Escalabilidade (*Scalability*)	Está relacionada ao aumento da capacidade de computação quando cresce o número de solicitações ou transações feitas por usuários em um intervalo de tempo.
Capacidade (*Capacity*)	Refere-se ao limite relacionado ao número de requisições simultâneas que devem ser atendidas com garantia de desempenho pelo serviço.
Robustez (*Robustness*)	Representa o grau em que um serviço pode funcionar corretamente mesmo na presença de entradas inválidas ou incompletas. Serviços devem ainda funcionar mesmo

	quando parâmetros incompletos sejam fornecidos para a invocação do mesmo.
Manipulação de Exceção (*Exception Handling*)	Está relacionada à forma como o serviço trata essas exceções, ou seja, os problemas que ocorrem durante a execução de um conjunto de operações presente em uma transação.
Precisão (*Accuracy*)	É definida como a taxa de erros gerados pelo serviço, sendo que o número de erros gerados em um intervalo de tempo deve ser minimizado.
Integridade (*Integrity*)	Diz respeito ao acesso não autorizado a dados e a programas. A integridade dos dados define se os dados transferidos são modificados em trânsito. Integridade transacional refere-se ao conjunto de medidas que garante preservar a integridade da base de dados após uma transação transcorrer.
Acessibilidade (*Accessibility*)	Representa se o serviço é ou não capaz de atender pedidos dos clientes.
Disponibilidade (*Availability*)	Refere à necessidade de o serviço estar disponível imediatamente quando é invocado.
Interoperabilidade (*Interoperability*)	Refere à necessidade de o serviço ser interoperável entre diferentes ambientes de execução.
Segurança (*Security*)	No âmbito dos serviços ela deve atender algumas propriedades: (1) autenticação (usuários ou serviços devem ser autenticados antes de acessar qualquer dado), (2) autorização (somente usuários ou serviços autorizados podem acessar serviços reservados), (3) confidencialidade (somente usuários ou serviços autorizados podem modificar algum dado), (4) rastreabilidade/auditoria (deve ser possível conhecer a história de um serviço desde quando invocado), (5) criptografia de dados, (6) não repúdio (a não negação da solicitação de um serviço).

Quadro 22 - Características de QoS para serviços segundo W3C (2003).
Fonte: (W3C, 2003).

O resultado da combinação entre a ontologia de alto nível proposta por Tondello (2008) e as características presentes em W3C (2003) gerou uma nova classificação conforme mostra a Figura 44, onde as classes *QoSContext*, *QoSCompoundContext* e *QoSSingleContext* são as de mais alto nível, provenientes de Tondello (2008). As classes *DiscoveryWS*, *QoSUnit*, *QoSCharacteristAttribute* e *QoSCharacterist*

foram definidas no contexto deste trabalho. Elas estão relacionadas entre si através de associações. Por exemplo, a classe *QoSCharacterist* se associa à classe *QoSCharacteristAttribute* através da relação *hasCharacteristAttribute.*

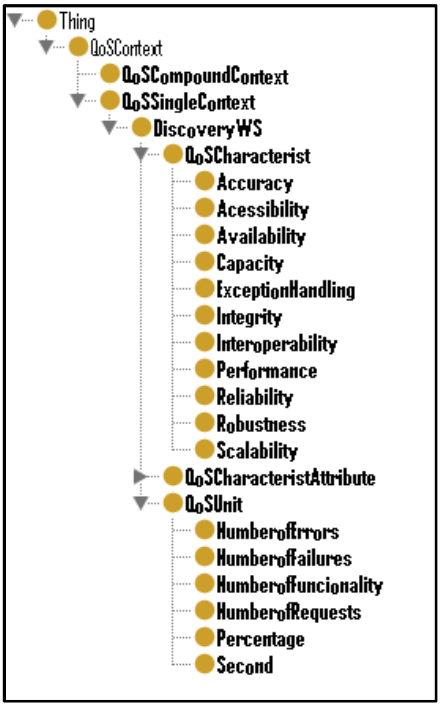


Figura 44 – Ilustração da classificação de QoS, usando o editor *Protégé*.

Na prática os relacionamentos foram criados para definir restrições e o reuso de classes para aproximar mais a ontologia da realidade a ser representada. Além disso, a estruturação das classes proposta visa facilitar futuros melhoramentos e incrementos à ontologia, pois o projeto da ontologia privilegiou aspectos de coesão e acoplamento entre as classes.

O Quadro 23 exemplifica a definição da característica de QoS *Availability* (disponibilidade), a qual é determinada em função da classificação de QoS apresentada na Figura 44.

Característica	Atributo(s) usado(s) para determinar uma dada característica
Availability (disponibilidade)	Através da relação *hasCharacteristAttribute* chega-se ao atributo *AvailabilityAttribute*, o qual possui como unidade associada o percentual, obtida através da associação *hasUnit* entre as classes *QoSCharacteristAttribute* e *QoSUnit.*

Quadro 23 – Exemplo de determinação da medida de características de QoS.

4.2.3.3.2 Submódulo que define os Repositórios de Serviços

Os repositórios de serviços são as entidades responsáveis pelo armazenamento das informações sobre serviços. Conceitualmente, eles usam a estrutura de dados do padrão UDDI (OASIS, 2004), seja para guardar informações sobre serviços ou para armazenar aspectos de qualidade (especificados através da ontologia de QoS) e informações sobre o contexto dos serviços (definidos através da ontologia UBL).

A seção 5.2.6.2 mostra em detalhes como estas informações foram armazenadas em uma UDDI concreta.

4.2.3.3.3 Submódulo que define a Publicação de Serviços

O processo de publicação de serviços parte do interesse de um provedor em ofertar serviços para uso. No modelo proposto, este procedimento é representado através de um diagrama de sequência da UML, apresentado na Figura 45. No diagrama, um fornecedor software (provedor) acessa a federação e informa que deseja publicar serviços. De posse desta informação, a federação exibe a ontologia UBL para o provedor. O provedor navega na ontologia UBL e define o nome do serviço a ser publicado (sua funcionalidade). O passo posterior se resume a informar as características de QoS atendidos pelo serviço. Isto é realizado mediante o acesso à ontologia de QoS, disponibilizada também pela federação. O passo final é o registro destas informações em um repositório de serviços (UDDI), que também guarda as informações cadastrais do provedor (por exemplo: nome, contato e endereço).

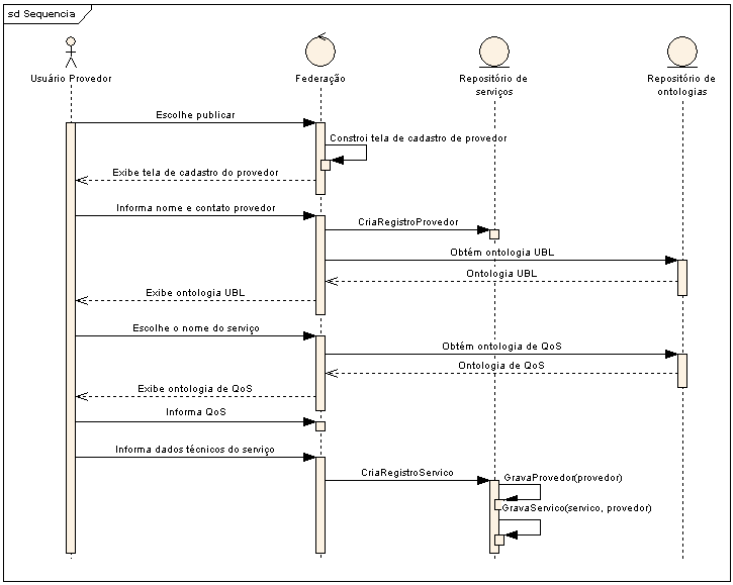


Figura 45 - Diagrama de sequência da UML que detalha o funcionamento conceitual do processo de publicação de serviço.

4.2.3.3.4 Submódulo que define a Reputação do Provedor

No modelo proposto usou-se como critério para classificar serviços candidatos um valor associado à reputação do provedor. Assim, preliminarmente, com intuito de viabilizar a construção completa do protótipo computacional, conforme características do modelo já mencionadas e descritas, e devido à complexidade para desenvolver e avaliar estratégias para determinar a reputação do provedor, optou-se por gerar valores de reputação através da geração de números aleatórios, entre 0 a 100. Isso caracteriza uma limitação do trabalho e abre oportunidade, em pesquisas futuras, para escolha de um método mais adequado para representar e determinar a reputação do provedor como para definir critérios de avaliação de serviços, conforme apresenta Li *et al.* (2009).

4.2.3.4 Módulo que define a Descoberta de Serviços

O módulo de *Descoberta de Serviços* está organizado em dois submódulos. Um deles denominado de *Crawling* e ou outro submódulo de *Descoberta dinâmica de serviços*.

O funcionamento básico do *crawling* é mostrado através de um diagrama de sequência da UML, conforme ilustra a Figura 46.

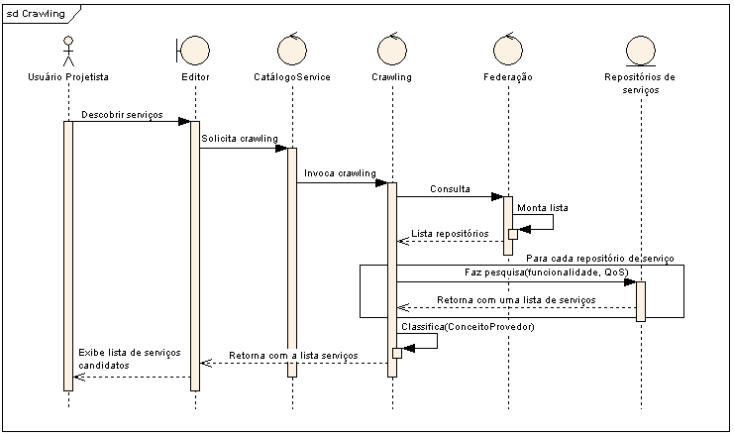


Figura 46 – Diagrama de sequência da UML detalha o funcionamento conceitual do *Crawling*.

O submódo do *crawling* usa como entrada o serviço desejado (usando a ontologia UBL, descrita na Seção 4.2.3.3.1), valores de QoS (usando a ontologia de QoS, descrita na Seção 4.2.3.3.1), uma informação (a reputação do provedor) que permite decidir pela classificação ou não dos serviços recuperados pelo *crawling*. Estes dados formam a uma instância da expressão da descoberta que é enviada como parâmetro para o *crawling*.

De posse da expressão da descoberta, o editor de aplicações solicita ao catálogo que invoque o *crawling*. O *crawling* solicita a federação à lista de repositórios de serviços disponíveis. A partir desta lista, o *crawling* dispara uma pesquisa em cada repositório e aguarda até que todos os repositórios devolvam a lista de serviços candidatos para proceder a classificação (se isto for desejado). Mostra a lista de serviços candidatos para o projetista de aplicações para análise. Como comentado anteriormente, caso o projetista não concorde com a resposta do *crawling*, ele tem a opção de fazer novas descobertas usando outras características de QoS, até encontrar o serviço que melhor atenda as suas necessidades.

O submódulo de descoberta dinâmica (DDS) também é representado por um diagrama de sequência da UML, conforme mostra a Figura 47.

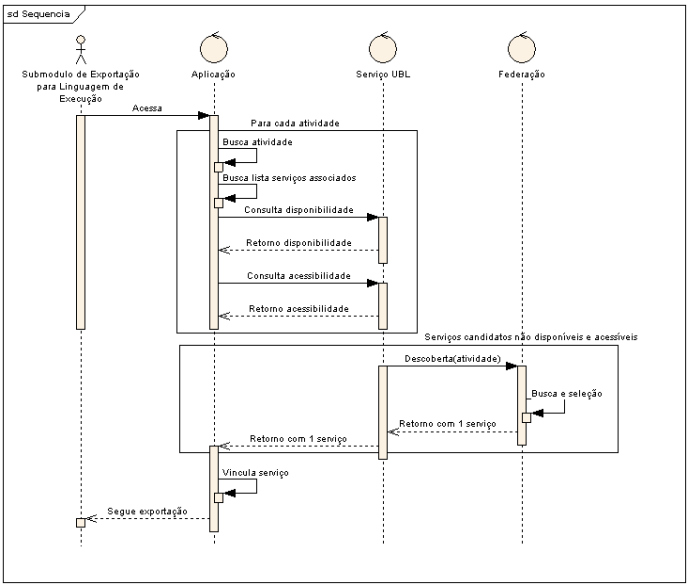


Figura 47 - Algoritmo usado pelo ambiente de execução para descobrir dinamicamente serviços.

A DDS parte da lista de serviços candidatos associada a uma dada atividade ou ao conjunto de atividades da aplicação. A descoberta dinâmica varre a lista de serviços candidatos, averiguando a disponibilidade de cada serviço. Caso um serviço esteja disponível e acessível, a descoberta é finalizada e o serviço é imediatamente vinculado a aplicação. Não havendo serviço candidato que possa ser usado na execução da aplicação, faz-se nova descoberta nos repositórios de serviços gerenciados pela Federação. A partir deste ponto, o processo de descoberta acontece de maneira semelhante ao algoritmo de *crawling*, ou seja, todos os serviços são analisados e o primeiro a atender os valores contidos na expressão da descoberta é escolhido e imediatamente vinculado à aplicação. A DDS termina no momento que encontrar o primeiro serviço que atenda os requisitos presentes na expressão da descoberta.

4.2.3.5 Módulo de Execução de Aplicações

O módulo de execução de aplicações é o local onde aplicações são executadas. Ele é composto de um Repositório de Aplicações WS-BPEL, um Motor de Execução (um exisitente), que interpreta a linguagem de execução das aplicações e as executa.

A fim de maximizar questões de interoperabilidade, o ambiente usa uma linguagem padronizada para a descrição dos processos executáveis, a especificação WS-BPEL (OASIS, 2007). O diagrama de sequência da UML mostrado na Figura 48 apresenta conceitualmente o funcionamento do módulo de execução de aplicações SOA.

Inicialmente um usuário executor de aplicações SOA ativa uma interface gráfica de monitoramento de aplicações. Após, navega no repositório de aplicações WS-BPEL, escolhe uma aplicação para ser executada e dá início a execução da aplicação. Enquanto o motor de execução interpreta e executa o código WS-BPEL, expresso através de primitivas básicas (<*invoke*>, <*receive*>, <*assign*> e outras) e de primitivas de controle de fluxo (<*sequence*>, <*while*> e outras), serviços UBL são invocados para executarem as atividades da aplicação WS-BPEL. Caso algum serviço UBL não esteja disponível e acessível, o serviço de descoberta dinâmica é chamado para buscar e selecionar um novo serviço.

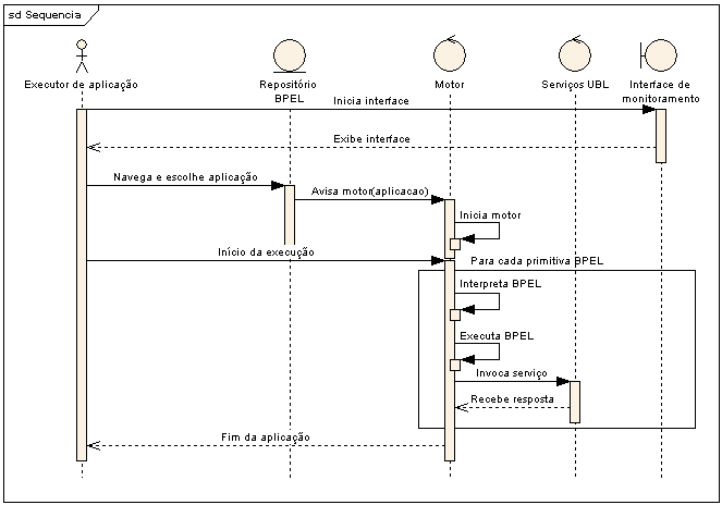


Figura 48 - Diagrama de sequência da UML que detalha o funcionamento conceitual do módulo de execução de aplicações SOA.

4.2.4 Visão de Implantação

A visão de implantação (*deployment view*) objetiva mostrar como o sistema será implantado dentro dos limites da infraestrutura física de equipamentos. Esta visão é representada através de um diagrama de implantação da UML que inclui vários tipos de artefatos, tais como máquinas, processos, arquivos e dependências (O'DOCHERTY, 2005).

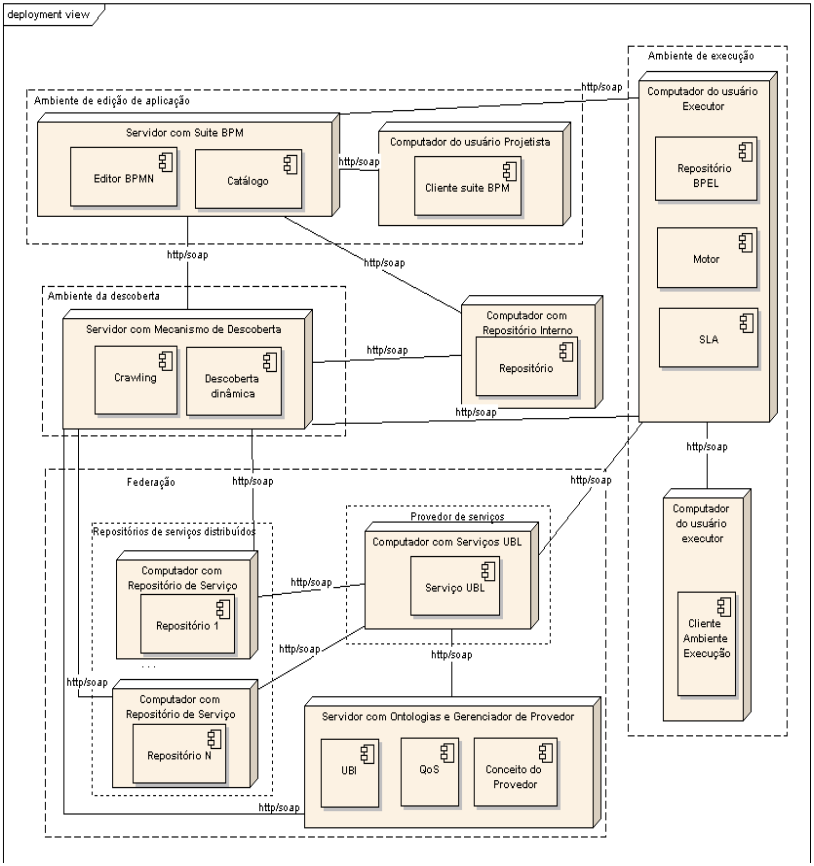


Figura 49 - Diagrama de implantação da UML para modelo proposto.

A Figura 49 apresenta o diagrama UML de implantação para o modelo de descoberta proposto. A parte superior esquerda do diagrama mostra o ambiente de edição de aplicações concentrado em uma máquina e um computador com acesso ao pacote BPM usado para editar aplicações. A parte central apresenta o ambiente de descoberta de serviços implantado em uma máquina e um computador contendo um repositório interno com informações dos serviços usados no escopo do modelo proposto. Na parte final do diagrama apresenta-se a federação de provedores de serviços, os diversos repositórios implantados em computadores remotos e em um servidor as ontologias (UBL e QoS) e o

módulo que gerencia a reputação dos provedores de serviços. No lado esquerdo do diagrama tem-se o ambiente de execução de aplicações SOA, contendo o motor para execução de aplicações e o repositórios de aplicações WS-BPEL e em um computador um cliente para acesso via http/soap aos diversos módulos do ambiente de execução.

4.3 **Pressupostos**

O modelo proposto está alicerçado em cinco pressupostos que o sustentam, sendo eles: conteúdo da pesquisa, adoção do padrão UBL, local da descoberta, composição da aplicação SOA e experiência do projetista de aplicações SOA.

O primeiro pressuposto diz respeito ao conteúdo da pesquisa. A consulta é por serviços *web* voltados a processos de negócios, que implementam atividades ou representam fluxos de trabalhos presentes em processos de negócios. Como apresentado no Capítulo 2, a tecnologia de serviços *web* vem sendo considerada como padrão de implementação do conceito de serviços de software e sobre o qual todas as iniciativas internacionais têm focado. Assim sendo, e também dada à grande quantidade de padrões associados aos serviços *web*, esta tese se restringe e assume que todos os serviços *web* disponibilizados por provedores na Internet adotam o padrão *web services*, onde suas interfaces são publicadas via padrão WSDL e UDDI (Seções 2.2.4.2 e 2.2.4.3). Com esta adoção, espera-se aproximar mais o trabalho à realidade das organizações, uma vez que as ferramentas de SOA do mercado estão sendo projetadas para lidar com serviços *web*, o mesmo ocorrendo com as ferramentas WS-BPEL, que dialogam com as de BPM para uma mais ágil aplicação SOA.

O segundo pressuposto refere-se ao uso do padrão UBL para modelar aplicações SOA. Como apresentado no capítulo 2, há alguns padrões de processos de negócios internacionais, como por exemplo: RosettaNet, UBL e ebXML. Independentemente dos seus propósitos e vantagens, isso reflete a enorme tendência – fruto de reais necessidades das organizações – do uso de padrões de especificação e de modelagem de processos de negócios. Portanto, o pressuposto de uso de um padrão parece ser realístico. Quanto à opção pelo padrão UBL, trata-se de um padrão aberto, gratuito e independente de domínio de aplicação, como não é o caso do RosettaNet, que é voltado para o setor eletro-eletrônico. Já o ebXML, mais antigo, reflete uma visão diferente sobre como duas organizações devem interoperar, criando-se “pares” de protocolos de transação de alto nível que devem ser previamente conhecidos pelas

partes envolvidas. O padrão UBL parte de um princípio mais amplo, no qual organizações têm o potencial de interoperar caso sigam um padrão geral de processo de negócio. Além disso, o padrão UBL foi concebido pensando-se na possibilidade da sua integração com ambientes BPM, que é um dos principais objetivos desta tese. Embora o modelo esteja preparado para considerar outros padrões, espera-se que haja um aumento no número de provedores e que empresas adotem padrões de processos. Desta maneira, espera-se que provedores de serviços tenham interesse em ofertar serviços alinhados aos padrões e via algum modelo de comercialização de software, tal como SaaS. O uso de especificações de processos minimiza problemas de interoperabilidade entre clientes e provedores de serviços, porque compartilha um vocabulário comum (através das ontologias UBL e QoS), reduzindo custos e aumentando o reuso de serviços.

No contexto do modelo proposto, o uso das ontologias UBL e QoS é um ponto chave, porque a descoberta é sintática, apesar de ela ser fundamentada em uma semântica. Ela compara informações presentes na expressão da descoberta com informações registradas sobre serviços nos repositórios.

O terceiro pressuposto prevê a publicação de serviços em repositórios instanciados através do padrão UDDI (OASIS, 2004). Considerando o cenário no qual há diversos provedores autônomos, heterogêneos e distribuídos, o modelo adota a abordagem de Federação (RABELO, 2008), adotando também características de outros trabalhos conforme foi apresentado na Seção 3.3.2. O uso da federação tem a função de agir como um ponto central e transparente para pesquisa e registro de provedores e serviços. Além disso, através de um guia (CANCIAN, 2009) a federação certifica provedores de serviços garantindo a qualidade dos serviços publicados e ofertados. Este pressuposto reforça o uso do padrão UDDI e, ao mesmo tempo, segue a tendência de deixar de ofertar serviços em um único local (em uma UDDI privada) (ELGAZZAR; HASSAN; MARTIN, 2010). A prática da integração BPM&SOA mostra que poucas organizações adotam o padrão UDDI. Todavia, ao se tentar compreender a razão disto, infere-se que isso pode ser reflexo da incipiência da adoção do paradigma SOA pelas organizações. As que já o adotam, mantêm seus serviços em repositórios locais e sem redundância funcional, ou seja, só há um serviço que implementa uma dada funcionalidade. Isso porque a

prioridade está “ainda” na reutilização de serviços, e não na flexibilidade de se escolher, dentre vários serviços equivalentes e disponíveis em inúmeros provedores externos e autônomos, os serviços mais adequados. Portanto, ainda é pouco comum encontrar empresas que usam o UDDI; porém, acredita-se que passará a ser uma tendência usá-lo na medida em que o paradigma SOA passar a ser adotado em grande escala, conforme inúmeros estudos apontam.

O quarto pressuposto é chamado de *relação 1:1* na composição da aplicação SOA. Isso significa que toda atividade presente na aplicação SOA será composta por e vinculada a um único serviço, embora diversos serviços descobertos possam ser associados a ela. Por exemplo, para a atividade *AcceptOrder*, realizada pela parte *BuyerPart*, pertencente ao processo *Ordering* UBL (OASIS, 2006) (conforme ontologia UBL), pode haver diversos serviços disponíveis e ofertados por diferentes provedores, mas somente um deles será escolhido e vinculado à atividade.

O uso deste pressuposto advém da própria adoção da ontologia UBL para especificar atividades dos processos de negócios. Como o nível de detalhamento de cada atividade a ser implementada por um serviço está previsto na ontologia UBL, tem-se uma granularidade fina para cada uma delas. Isso possibilita que cada atividade seja associada a um único serviço, potencializando o reuso e o desacoplamento do serviço.

Com a adoção da relação 1:1, problemas associados à composição e a execução de aplicações SOA são minimizados. Por exemplo, quando uma atividade é implementada por diversos serviços, uma orquestração precisa definir a responsabilidade de cada serviço de maneira a refletir a necessidade da atividade em questão. Já do lado dos provedores, acredita-se que com a crescente adoção de padrões de processos de negócios, estes passem gradualmente a oferecer serviços de software compatíveis aos padrões, incrementando seus negócios enquanto a oferta de serviços equivalentes cresceria para os clientes SOA. Assim, cada serviço corresponderia a uma atividade de processo UBL, de baixa granularidade, e este seria publicado numa dada UDDI, através do padrão WSDL, passíveis de serem localizadas por mecanismos de descobertas de serviços (como o proposto nesta tese), que de forma “inteligente”, se encarregaria de encontrar o conjunto de serviços mais adequados para um dado processo BPM/SOA.

Por outro lado, esta abordagem tem fortes implicações em termos de desempenho e segurança, comparada a soluções de alta granularidade. Evidentemente, uma vez que os provedores e cada

serviço esteja fisicamente implantado em algum servidor largamente distribuído, haverá potencialmente uma larga coleção de serviços *web* sendo invocados em execução, o que demanda uma grande confiança e robustez global da rede. Sabe-se, no entanto, que garantir 100% de disponibilidade da rede e dos vários serviços é praticamente impossível. A concepção de algoritmos e mecanismos para suportar tal garantia está fora do escopo desta tese. De qualquer forma, é importante esclarecer que neste cenário haverão *n* serviços vinculados a cada processo *m*. Portanto, num cenário comum empresarial, um dado processo de compra, por exemplo, envolverá um conjunto de inúmeros serviços, que deverão ser devidamente orquestrados. Com isso, comparando-se com soluções monolíticas tradicionais, logicamente a execução desta tenderá a ser muito mais rápida em relação a composta por *n* serviços e que estes serão encontrados e invocados em tempo de execução. Porém, esta é uma desvantagem intrínseca do paradigma SOA, que por sua vez traz potenciais vantagens em termos de reuso e flexibilidade de projeto e execução de aplicações (entre outras).

Assim, esta questão da relação 1:1 é, na verdade, sujeita a algumas particularidades quando se confronta isto na prática. Isso porque se assume que todos os processos (suas atividades[17]) seriam, necessariamente, associados a um serviço *web* e este estaria implantado (provido) por um provedor externo. Há que se considerar que no mundo real das organizações isso não seria necessariamente assim. Quando se modelam processos e estes estão vinculados à execução de software, há inúmeros casos onde se estaria invocando uma funcionalidade de um sistema ERP, de tempo real, ou outro. Ou seja, de sistemas que necessariamente estão localmente implantados dentro da empresa. Portanto, não seriam resultados de descoberta externas, mas sim de vinculações fixas, que eventualmente podem ser "encapsuladas" (*wrapped*) como serviços *web*.

Outro aspecto importante sobre a estratégia 1:1 é a preparação para se trabalhar sobre o modelo SaaS. Como comentado na Seção 2.5, o conceito de SaaS corresponde a um modelo arquitetural e de negócios onde usuários (tipicamente) SOA acessam serviços (*web*) sob demanda,

[17] Efetuadas somente por serviços *web*.

pela Internet, e só pagam pelo que usam (acessam). Uma baixa granularidade potencializa uma maior utilização do serviço e uma maior garantia de que o cliente só pagará pelo código efetivamente útil às suas necessidades, contrariamente a um serviço de alta granularidade (com muitas funcionalidades) ou a um sistema monolítico. Esta base para suportar SaaS adiciona uma outra perspectiva interessante de descoberta, ligada a requisitos não-funcionais, que é, por exemplo, a econômica. Ou seja, um dos critérios de escolha de um serviço por parte do cliente pode ser o preço, da mesma forma que, do lado do provedor, múltiplos modelos de negócios podem ser ofertados sobre um mesmo serviço, conforme o cliente ou tipo de cliente.

Somam-se a isto o problema da confecção e assinatura dos contratos, ou seja os SLAs gerados para cada serviço acessado. Nesta tese assume-se a possibilidade disso ser feito automaticamente, a partir de um modelo de referência proposto na literatura. Todavia, isso pode ser considerado ainda como estado-da-arte, pois na prática acredita-se que tal cenário de automatismo em relação a contratos deve ser de mais demorada adoção, principalmente em se considerando que a adoção em si de SOA e SaaS são ainda relativamente incipientes nas organizações.

Esses pressupostos, em menor ou em maior grau, estão relacionados ao fato de que esta tese tem um cunho também exploratório, onde se enfrenta um cenário que pode ser considerado ainda como “futurístico”. Porém, tentou-se sempre ter um embasamento junto ao estado-da-arte na literatura e no estado-da-prática, de forma que esses pressupostos, na opinião deste autor, não são vistos como demasiados fortes ou limitadores do trabalho.

Embora o pressuposto 1:1 pareça simplificar o modelo, o mesmo não tem esta intenção essencial, uma vez que se trata mais de uma estratégia encontrada para maximizar o reuso, tirando maior proveito do desacoplamento e de como o modelo proposto deveria enxergar os inúmeros serviços (UBL) que são disponibilizados por diversos provedores. Portanto, casualmente esta estratégia tornou o modelo mais simples neste quesito. No modelo proposto, o serviço associado a uma atividade pode se tornar um serviço abstrato (sem implementação) e agregar outros serviços, permitindo que questões de orquestração, desempenho e outras sejam devidamente analisadas e tratadas em um momento oportuno. Além disso, o modelo prevê que todas as atividades presentes na aplicação serão implementadas através serviços.

O quinto e último pressuposto considera que o projetista da aplicação é uma pessoa bastante experiente em processos de negócios e

que este terá a ajuda (pelo menos nas primeiras vezes) de uma pessoa de TI, especialista em QoS, quando da especificação destes requisitos.

A adoção dos paradigmas de BPM e SOA têm grandes impactos de várias naturezas em uma organização. Um deles impacta a qualificação das pessoas para trabalharem nos ambientes computacionais associados. Por outro lado, isso não é uma particularidade do BPM ou SOA, mas sim uma regra geral nas empresas. Em outras palavras, a adaptação das pessoas às novas tecnologias é uma situação corriqueira nas empresas, a despeito do custo e dos problemas que isso apresenta. Além disso, atualmente qualquer empresa que trabalhe com a noção de modelagem de processos de negócios (por exemplo, usando ferramentas como *Microsoft Visio* ou o Aris) é naturalmente muito familiarizada (e supostamente experiente) com essa tarefa, conhecendo os processos da empresa, as terminologias, os fluxos de informação e outros detalhes. E em se considerar que UBL é um padrão, espera-se que as empresas gradativamente migrem seus modelos proprietários de processos para padrões e ambientes modernos como os de BPM, ainda mais considerando que cada vez mais as empresas estão tendo que trabalhar entre si e interoperar. Para o modelo proposto nesta tese, essencialmente o que importa é o formato de saída da aplicação de modelagem de processos de negócios. No caso desta tese, espera-se uma saída no padrão BPMN. A partir disso, o modelo de descoberta é disparado automaticamente, após haver a conversão deste padrão de processos de negócios (BPMN) e posteriormente a conversão deste no padrão WS-BPEL de execução de serviços *web*, para executar a aplicação SOA composta, com os serviços selecionados e vinculados.

Do ponto de vista ontológico, esta abordagem assume, portanto, que todos os clientes e provedores SOA adotariam um padrão, o UBL no caso. Por outro lado, isso pode ser visto como um pressuposto demasiado forte. Porém, como afirmado, existe uma forte tendência pela adoção de processos de negócios padrão pelas empresas. Não que se acredite que todas as empresas passariam a adotar a UBL, mas o modelo e *framework* desenvolvidos são abertos para contemplar outros padrões. Além disso, o uso de ontologias pode suportar uma interoperação entre diferentes padrões, por exemplo, através de mapeamentos. Desta forma, um dado serviço pode ser ofertado sob vários padrões (ou mesmo sem padrão) e uma ontologia UBL poderia ser utilizada como “ponte” entre o serviço e o mecanismo de busca.

Já do ponto de vista de critérios de QoS, a tarefa é um pouco mais complicada. Isso porque, além de se exigir que o projetista/analista de negócios conheça bem o que significa cada critério de QoS e tenha alguma sensibilidade prática sobre seus valores, não há métricas padrão. Portanto, aqui há certo espaço de subjetividade e experiência requerida, de forma que ele não indique nem valores de QoS muito baixos (e com isso haveria uma potencial enorme quantidade de serviços associados, vindos do algoritmo de descoberta) e nem muito altos (e assim pouquíssimos ou mesmo nenhum serviço seria encontrado). Porém, esta questão de QoS não dispõe de "padrões" e, portanto, acaba sendo um problema intrínseco à sua consideração no modelo.

4.4 **Diferencial da Proposta**

Embora existam diversos trabalhos que promovam a integração BPM&SOA (ADAM; DOERR, 2008) e (INAGANTI; BEHARA, 2007), percebe-se a falta de uma solução mais ampla (PAPAZOGLOU *et al.*, 2007), que enfatize a descoberta, de forma dinâmica, com geração de SLAs, considerando a semântica de processos e serviços, QoS e uma larga distribuição dos provedores, em um único artefato, completamente integrada com padrões abertos, e tenha as bases para suportar o modelo SaaS.

Portanto, são todos esses elementos unidos e integrados num único artefato, inseridos num cenário ubíquo de fornecimento de serviços de software, que representam o ineditismo deste trabalho, cuja finalidade é ajudar para que ocorra uma mais transparente e ágil integração entre os ambientes (ou camadas) BPM&SOA.

Cabe salientar que o modelo desenvolvido é, nele mesmo, uma aplicação SOA. Todos os seus componentes são modelados como serviços e invocados internamente. Como tal, no extremo, o modelo proposto pode ser visto, conceitualmente, como um "meta-modelo" de descoberta, pois os próprios serviços ora implementados no modelo podem ser descobertos e/ou trocados por outros. Por exemplo, um novo algoritmo de descoberta pode ser "plugado" no modelo de forma transparente ao seu funcionamento, desde que ele esteja preparado para se comunicar com os padrões usados e requeridos.

4.5 **Considerações**

Este capítulo apresentou o conjunto de elementos conceituais usados como alicerce na concepção do modelo proposto. Foi

apresentado o conjunto de características que o definem, a sua arquitetura conceitual, o seu funcionamento operacional, os pressupostos que sustentam o modelo e o diferencial frente a outras iniciativas estudadas.

Em relação ao avanço científico que o modelo proposto carrega e apresenta, há que se destacar alguns pontos. O primeiro está ligado à visão operacional do modelo. Nas fases de projeto e execução de aplicações, um conjunto operacional de passos foi definido e descrito. Essas operações são essenciais para o entendimento de como ocorre o processo de descoberta e, consequentemente, como isto possibilita uma mais transparente e ágil integração BPM&SOA. Além de servirem para compreensão do processo global de integração, as operações subsidiam o desenvolvimento de novos projetos que se destinam a integrar o nível de negócios ao nível dos sistemas.

O segundo ponto é a definição e uso de um *crawling* para atuar na fase de projeto de aplicações SOA. Ele assiste o projetista na tarefa de definir restrições, seja em termos funcionais (usando a ontologia UBL), seja em termos de requisitos de qualidade (usando a ontologia de QoS). Além disso, o uso do *crawling* visa poupar tempo e esforço na fase de execução de aplicações, porque antecipa descobertas evitando que pesquisas exaustivas sejam realizadas.

O terceiro ponto leva em conta o contexto dos processos de negócios na descoberta, o qual é capturado a partir do uso de um catálogo eletrônico de processos baseado no padrão UBL e representado através da ontologia UBL. A finalidade da ontologia UBL é determinar serviços em termos funcionais. Isso dá maior riqueza semântica à descoberta e otimiza a concepção da aplicação SOA, pois clientes e provedores usam um vocabulário comum quando interagem com o mecanismo de descoberta, seja para publicar ou obter serviços.

O quarto elemento de destaque é a adoção do modelo de disponibilização de software SaaS. Com base na revisão do estado da arte em descoberta, percebeu-se a falta de atenção dada pelos trabalhos estudados em como os serviços são comercializados. Este é um aspecto importante de ser considerado, principalmente se considerar o cenário de aumento no número da oferta de serviços similares, fato que naturalmente gera a necessidade de controles por quem usa e comercializa serviços.

O quinto ponto importante é o uso intenso de padrões e recomendações para minimizar problemas de interoperabilidade, seja na tarefa de incorporar novos elementos (para acompanhar e acomodar demandas específicas) ou para substituir algum elemento do modelo proposto por outro.

O sexto ponto diz respeito ao uso de uma federação de provedores, que além de prover interoperabilidade através das ontologias UBL e QoS, é o local que congrega logicamente provedores (independentes, autônomos e heterogêneos) de serviços que, mediante uma política de qualidade de software, publicam e ofertam serviços. Além disso, a federação é uma entidade que avalia provedores e gerencia um conjunto de repositórios de informações sobre serviços usados como fonte para a descoberta de serviços, fornecendo conectividade entre clientes e provedores de serviços.

Diante do exposto, o modelo proposto ganha força em relação às demais propostas, porque além de ser integrado, aberto e flexível, incorpora diversos elementos em um único artefato. Isso é relevante porque o modelo trata e enxerga o problema da descoberta sob várias dimensões (por exemplo: negócios, seleção de serviços, interoperabildiade) e permite responder aos cinco aspectos listados no início do capítulo, possibilitando aproximar mais os ambientes BPM&SOA.

O próximo capítulo descreve como os elementos conceituais apresentados neste capítulo foram instanciados na forma de software.

Capítulo 5

Protótipo Computacional

Este capítulo apresenta a implementação do modelo proposto descrito no capítulo anterior, como um procedimento metodológico de avaliação da hipótese de pesquisa e de suporte à pergunta de pesquisa, ambas descritas no Capítulo 1.

O capítulo inicia apresentando a lista de tecnologias e ferramentas usadas na implementação da arquitetura conceitual apresentada na Figura 37. A seção 5.2 justifica e explica o uso do paradigma SOA na implementação do protótipo computacional, mostra como os elementos conceituais do modelo proposto foram instanciados e como eles assumiram a forma de um protótipo computacional. A seção 5.3 apresenta um exemplo de funcionamento do protótipo computacional. A seção 5.4 trás as considerações do capítulo.

5.1 Tecnologias Utilizadas

Esta seção apresenta, descreve e justifica o uso das principais ferramentas e tecnologias computacionais utilizadas no desenvolvimento do protótipo computacional.

5.1.1 Eclipse Plataform

A plataforma Eclipse é um ambiente *open source* para desenvolvimento de software. Ele suporta diversas linguagens de

programação, podendo ser executado em várias plataformas, como: Windows, Linux, Solaris e MacOS, oferecendo aos desenvolvedores de sistemas ferramentas para criar aplicativos profissionais, por exemplo, para *desktop*, *web* (BEZERRA, 2011).

Um dos principais atrativos da plataforma é o fato dela ser extensível. O Eclipse possui uma estrutura modular que é formada por centenas de *plug-ins* que executam em cima de um pequeno núcleo e de um módulo de carregamento (CLAYBERG; RUBEL, 2006).

Assim, devido a sua extensibilidade o Eclipse tem sido uma das IDE (*Integrated Development Enviroment*) mais utilizadas profissionalmente e em trabalhos acadêmicos (LÜER, 2003).

Utilizou-se nesse trabalho a versão 3.5 (*Galileo*) do Eclipse, pela experiência do autor no uso dessa ferramenta e por ser a última versão disponível no momento do desenvolvimento do protótipo.

5.1.2 IBM Websphere Business Modeler

O *Websphere Business Modeler* (WBM) é uma ferramenta desenvolvida pela IBM (*International Business Machines*) para modelagem de processos de negócios com suporte para BPMN (BEZERRA, 2011). A ferramenta dispõe de recursos para modelagem, simulação e análise, permitindo aos analistas de negócios conceber, documentar e implantar processos de negócios (IBM, 2010).

O WBM simplifica a modelagem do processo através o uso de um ambiente estruturado que provê interação com o projetista na fase de concepção do processo, permitindo incorporar mudanças em modelos existentes, geralmente sem a necessidade de esforços de desenvolvimento (NOEL, 2005).

A escolha do *Websphere Business Modeler*, como o editor de aplicações neste trabalho, se deu em razão da ferramenta ser largamente conhecida, ser baseada na plataforma Eclipse, que permite sua extensão através do uso de conectores (*plug-ins*), e pelo fato da disponibilidade da mesma aos usuários do GSigma, conforme convênio firmado entre UFSC/DAS e IBM.

5.1.3 Business Process Execution Language

É um padrão (OASIS, 2007) para definir a execução de processos de negócios em nível de implementação (DIJKMAN; DUMAS; OUYANG, 2008).

A execução de um processo WS-BPEL é semelhante a de um sistema de *workflow*, no qual um serviço é invocado um após o outro. As tarefas WS-BPEL são equivalentes a uma invocação numa linguagem de programação. A atividade *receive* aguarda o recebimento de uma mensagem de entrada, enquanto que a atividade *invoke* transmite uma mensagem de saída (MARGOLIS, 2007).

5.1.4 Service Component Architecture

A *Service Component Architecture* (SCA) é um padrão para a composição e implantação de aplicações que utilizam a Arquitetura Orientada a Serviços (SOA). O seu uso permite que vários detalhes sejam abstraídos de tecnologias de implementação de forma que um ambiente SCA faça o gerenciamento e a ligação dos vários componentes heterogêneos presentes (MARGOLIS, 2007). Um componente, em uma SCA é um pedaço de código escrito em alguma linguagem. Esses componentes podem estar distribuídos na mesma máquina local ou remotamente. Conjuntos de componentes formam composições, que são descritas através da linguagem SCDL (*Service Composition Description Language*). O arquivo SCDL descreve como os componentes interagem entre si, quais são as dependências de um com os outros e também quais são os serviços expostos por esses componentes.

Outro conceito importante numa arquitetura SCA é o domínio. Domínio é um agrupamento lógico de composições, que podem estar distribuídas em várias máquinas. Um domínio é gerenciado por uma implementação SCA de algum fornecedor, também chamado de *middleware*.

O uso da SCA neste trabalho é motivada por duas razões fundamentais. A primeira é que todos os módulos e submódulos conceituais apresentados e descritos no Capítulo 4 são implementados via serviços *web*, ou seja, todo o modelo proposto de integração BPM&SOA funciona a partir de chamadas a serviços *web*. A segunda razão é pelo fato da SCA ser uma metodologia padrão para a concepção de aplicações SOA.

5.1.5 Apache Tuscany

O Apache *Tuscany* é um projeto *open source* da fundação Apache que permite o uso da *Service Component Architecture* (SCA). Ele é a implementação considerada referencia para aplicações que usam a SCA (LAWS *et al.*, 2011). Além disso, o *Tuscany* provê uma integração transparente com diversas tecnologias, permitindo o desenvolvimento de aplicações baseadas em SOA, é gratuito e implementado em Java. Portanto, foram estas as características que determinaram a escolha desta ferramenta no uso deste trabalho.

5.1.6 Apache ODE

O Apache ODE (*Orchestration Director Engine*) é um motor de aplicações escritos na linguagem WS-BPEL. Ele é responsável por interagir com os serviços, enviando e recebendo mensagens, lidando com a manipulação de dados e recuperação de erros (APACHE, 2010b).

Optou-se pelo seu uso nessa tese por se tratar de uma implementação madura do padrão WS-BPEL e por ser uma ferramenta gratuita e com um bom suporte em termos de documentação.

5.1.7 Intalio BPMS

O BPMS (*Business Process Managment Suite*) é uma suíte de ferramentas de gerenciamento de processos de negócios desenvolvida pela empresa Intalio. Ela é construída sobre diversas ferramentas *open source* como o *Eclipse*, Apache ODE e o sistema de *workflow Tempo* (INTALIO, 2010).

A *suíte* possui um editor visual que permite a modelagem dos processos e um servidor para executar aplicações baseado no Apache ODE. Através de uma interface *web*, por exemplo, foi possível iniciar e monitorar o estado das aplicações, analisando quais estão em execução, quais foram completadas.

O Intalio BPMS foi utilizado nesse trabalho por prover uma interface administrativa intuitiva, servindo como um complemento ao Apache ODE e permitindo acompanhar aplicações de uma forma amigável.

5.1.8 Apache jUDDI

O Apache jUDDI é uma implementação em Java do padrão *Universal Description Discovery and Integration* (OASIS, 2004). Ele é um projeto *open source* mantido pela fundação Apache. Junto ao seu pacote, um cliente UDDI é fornecido, o qual pode ser usado para conexão com implementações do padrão UDDI.

Escolheu-se o Apache jUDDI como implementação concreta dos repositórios de serviços. A motivação se deu pelo fato de ser um projeto consolidado, recentemente promovido a um projeto de primeiro nível (*TLP – Top Level Project*) da fundação Apache e por ser umas das únicas implementações encontradas em Java da especificação UDDI v3.

5.1.9 Protégé

O *Protégé* é uma plataforma *open source* escrita em Java desenvolvida pelo grupo de pesquisa da *Stanford Medical Informatics* (Escola de Medicina da Universidade de Stanford, nos Estados Unidos). É uma ferramenta que permite a edição, visualização e armazenamento de ontologias em diversos formatos.

A escolha do uso do *Protégé* v. 4.0.2 para edição das ontologias recaiu sobre três fatores principais. O primeiro por ser um editor com funcionalidades intuitivas e estáveis. O segundo por ser um editor *open source*, gratuito e largamente usado na construção de ontologias. O terceiro por fornecer suporte à linguagem OWL 2 (*Web Ontology Language*) (W3C, 2009), padrão usado para escrita de ontologias.

5.1.10 Bibliotecas de Suporte

Esta subseção apresenta a relação das principais bibliotecas de suporte utilizadas na implementação do protótipo, conforme mostra o Quadro 24.

Biblioteca	Versão	Finalidade	Site
Apache CXF	2.2.10	Suporte a manipulação de *webservices*	cxf.apache.org
Freemarker	2.3.16	Ferramenta de *templating* e geração de código	freemarker.sourceforge.net
Hibernate	3.4	Persistência de objetos em bases relacionais	www.hibernate.org
HSQLDB	1.8.1	Banco de dados relacional de pequeno porte	hsqldb.org
Jetty	6.1.21	Servidor *web*	jetty.codehaus.org/jetty
SimpleXML	2.3.5	Serialização de objetos em XML	simple.sourceforge.net
Jena	2.6.4	Leitura e escrita de ontologias em OWL	jena.sourceforge.net
Jdom	1.1.1	Leitura e escrita XML	www.jdom.org

Quadro 24 – Lista das principais bibliotecas de suporte usadas na implementação do protótipo computacional.

Apesar dessas bibliotecas não desempenharem um papel essencial na implementação do protótipo, elas foram importantes na codificação de algumas funcionalidades que permitiram poupar esforço e tempo de desenvolvimento.

5.2 **Arquitetura de Implementação**

Esta seção descreve a implementação dos cinco módulos que formam a arquitetura conceitual do modelo proposto (descrita no Capítulo 4), bem como comenta e justifica o uso de ferramentas e de tecnologias de informação na implementação do protótipo computacional.

5.2.1 Motivação para uso da Arquitetura Orientada a Serviços

O modelo proposto foi concebido seguindo o paradigma de desenvolvimento de sistemas distribuídos SOA. As funcionalidades presentes nos diversos módulos foram implementadas via serviços *web*,

usando como metodologia a SCA (*Service Component Architecture*) e a ferramenta Apache *Tuscany 2.0* (APACHE, 2010d). Assim, justifica-se o uso de um repositório interno de serviços, uma vez que antes de um serviço ser invocado, o mesmo precisa estar publicado e ser descoberto. O repositório interno de serviços foi instanciado via Apache jUDDI e inicializado (publicados) com todos os serviços que compõem o conjunto de funcionalidades que garantem a operacionalização do protótipo computacional.

O uso de SOA na implementação do modelo proposto é motivada por dois aspectos importantes. O primeiro deles ratifica a viabilidade técnica de soluções computacionais distribuídas construídas através de serviços *web*. O segundo aspecto, o mais importante deles, está associado à agilidade para produzir mudanças. Ou seja, este aspecto tem a ver com a flexibilidade para substituir serviços por outros; pois, devido a dinâmica imposta pelo mercado, nem sempre há tempo e recursos para modelar, implementar e testar software. Neste caso, é preciso ter flexibilidade para dar respostas rápidas. A chave para atingi-la surge a partir do fraco acoplamento existente entre os serviços e pela alta interoperabilidade existente no emprego dos mesmos. O fraco acoplamento entre os serviços permite que existam poucas dependências entre eles e que modificações ou falhas tenham menos impacto em outros serviços. Já a alta interoperabilidade decorre da capacidade de conectar serviços diferentes com um menor esforço.

5.2.2 Implementação do Módulo de Edição de Aplicações

A implementação do módulo de edição de aplicações usou a ferramenta da IBM *WebSphere Business Modeler* (WBM). Nela foi incorporado um conector, cuja finalidade foi integrar o ambiente de edição ao módulo de catálogo de processos de negócios, desenvolvido por Bezerra (2011).

5.2.2.1 Implementação do Conector de Integração

Um conector (*plug-in*) de integração foi desenvolvido em Java e incorporado ao ambiente de edição de aplicações SOA (ao *WebSphere*) para acesso ao catálogo de processos. A finalidade da integração do catálogo ao ambiente de edição de aplicações é possibilitar o uso de padrões prontos de processos na construção de aplicações SOA (evitando que o projetista da aplicação tenha que construir a aplicação completa), bem como capturar o contexto dos processos para ser repassado a outros módulos do protótipo computacional.

A implementação em Java do conector é composta por seis pacotes, cada um responsável por uma funcionalidade específica dentro do conector. A Figura 50 apresenta estes pacotes:

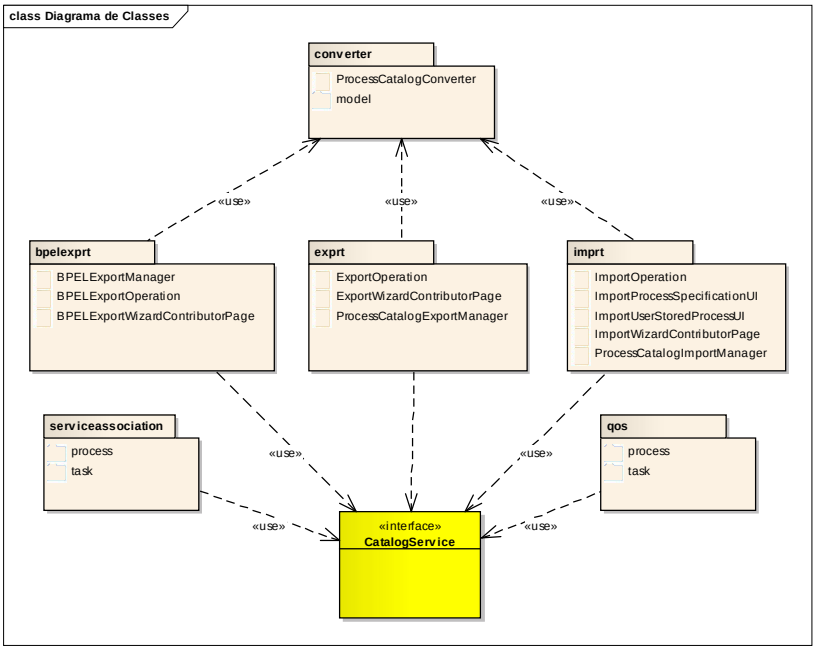


Figura 50 – Pacotes, classes e interfaces usadas na edição de aplicações com acesso ao catálogo.
Fonte: (BEZERRA, 2011).

- *converter* – possui a função de fazer a conversão entre o modelo de dados usado pelo *Websphere Business Modeler* e o usado pelo Catálogo de Processos de Negócio;
- *bpelexprt* – responsável pela exportação do processo para a linguagem de execução WS-BPEL;
- *imprt* – responsável por importar o processo do catálogo para o editor;
- *exprt* – faz a exportação (a transferência) da aplicação para o catálogo;

- *serviceassociation* – faz a busca e associação dos serviços candidatos às atividades da aplicação. Ele possui dois subpacotes: *process* e *task*. O primeiro faz a busca e a associação de uma só vez a todas às atividades da aplicação. O segundo faz a descoberta individualmente para cada atividade da aplicação;
- *qos* – permite a associação de critérios de QoS às atividades da aplicação. Ele possui dois subpacotes *process* e *task*, que respectivamente permitem definir critérios de QoS para todas as atividades da aplicação de uma vez só, ou individualmente.

Os pacotes *bpelexprt*, *exprt*, *imprt*, *serviceassociation* e *qos* interagem com a interface de serviço *CatalogService*, permitindo que o conector execute as operações do módulo do catálogo de processos de negócio. O endereço do serviço é obtido através do acesso (da descoberta do serviço) no repositório interno de serviços, instanciados via jUDDI. A Figura 51 apresenta a classe *WebServiceLocator*, usada para fazer a consulta ao repositório.

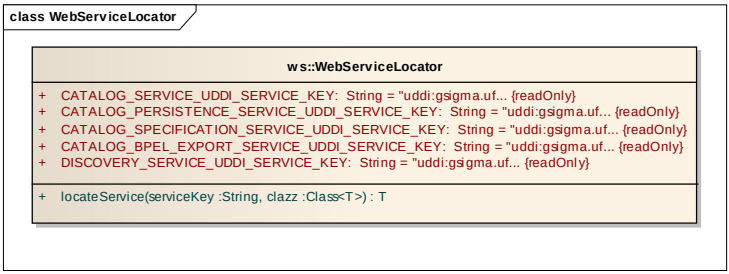


Figura 51 – Classe *WebServiceLocator*.
Fonte: (BEZERRA, 2011).

O método *locateService* recebe como parâmetro a chave do serviço pelo qual o mesmo foi registrado no repositório e a classe da interface que possui as operações do serviço. O retorno é um *proxy* que implementa a interface fornecida e contém a referência do serviço desejado.

O conector de integração com o catálogo possui várias interfaces gráficas que permitem ao usuário interagir com o ambiente de edição de aplicações. Exemplos destas interfaces gráficas podem ser observados na Seção 5.3. Nela, um exemplo de funcionamento do protótipo

computacional é apresentado. A tecnologia escolhida para a implementação dessas interfaces gráficas foi o *SWT - Standard Widget Toolkit* (ECLIPSE, 2009). Este *framework* faz parte da plataforma Eclipse e permite a construção de interfaces gráficas utilizando componentes nativos do sistema operacional. Geralmente esta abordagem possui um melhor desempenho do que tecnologias que utilizam emulação, como, por exemplo, o padrão SWING (CLAYBERG; RUBEL, 2006).

5.2.3 Implementação do Módulo do Catálogo de Processos de Negócios

A arquitetura conceitual do catálogo de processos de negócios apresentada no Capítulo 4, define e mostra os principais submódulos pertencentes ao catálogo e descreve como eles se associam de forma a produzirem as funcionalidades necessárias para o seu funcionamento. A implementação destes submódulos não faz parte do escopo desta tese. Informações mais detalhadas sobre a implementação de cada submódulo devem ser obtidas em Bezerra (2011).

5.2.4 Implementação do Módulo de Descoberta de Serviços

A implementação do módulo de descoberta de serviços foi feita usando a linguagem Java e está representada por um diagrama de pacotes contendo as classes que operacionalizam o módulo de descoberta, conforme mostra a Figura 52.

No diagrama, o pacote *bootstrap* contém a classe *RunDiscoveryService* que é responsável pela criação e implantação do serviço de descoberta, deixando-o disponível para que outro módulo (funcionalidade) da arquitetura possa consumi-lo. O pacote *interface* abriga a interface *DiscoveryService*, que define o comportamento para o *crawling* e para o algoritmo de descoberta dinâmica. O pacote *discoveryImplementation* armazena o código que implementa a descoberta de serviços. Nele há duas classes. A classe *Crawler* responsável pela implementação da descoberta de serviços em tempo de projeto de aplicações e a classe *DynamicDiscovery* dedicada a descobrir um serviço mais adequado no momento em que a aplicação estiver em execução. O pacote *model* contém um conjunto de classes que representam estruturas de classes de informação utilizadas pelas classes

Crawler e *DynamicDiscovery*. A classe *QoSInformation* contém a estrutura conceitual que permite a manipulação das características de QoS obtidas a partir da ontologia de QoS. A classe *Expression* representa o conjunto de informações presentes na expressão da descoberta, usada para a descoberta de serviços. A classe *DiscoveryResult* representa a lista de serviços retornados pelo mecanismo de descoberta e a classe *DiscoveredService* contém a estrutura responsável por armazenar informações inerentes a um serviço.

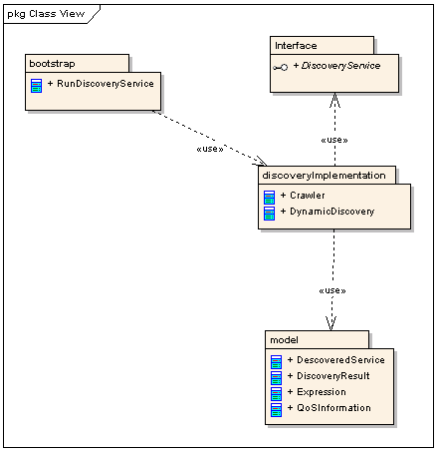


Figura 52 - Implementação do módulo de descoberta descrito através de pacotes de classes.

O algoritmo básico usado na implementação do *crawler* é apresentado na Figura 53. O algoritmo usa como entrada o serviço desejado (definido a partir da ontologia UBL, descrita na Seção 4.2.3.3.1) e um conjunto de características de QoS (definido a partir da ontologia de QoS, descrita na Seção 4.2.3.3.1).

Para cada UDDI presente na federação, o algoritmo cria e lança *threads* (Figura 53, linhas 1-4). Elas verificam cada serviço comparando capacidades funcionais (i.e se o nome do serviço desejado pertence à ontologia UBL) e checam valores de QoS com os valores desejados. Se um serviço satisfizer a expressão da descoberta, o mesmo é incluído em uma lista de serviços candidatos (Figura 53, linha 12). Esta lista é retornada pelo *crawling* quando todas as *threads* finalizam suas pesquisas.

```
Algoritmo Crawling
  Entradas:
        Lista UDDIs;
        Serviço desejado (a partir da ontologia UBL);
        Conjunto de características de QoS desejadas (a partir da ontologia QoS);
  Saída: lista de serviços candidatos.
Início
(1)   thread.cria <- listaUDDI.get()
(2)   Para as (n) UDDIs disponíveis na Federação faça
(3)       thread.dispara
      Fimpara
(4)   MostraListaServiçosCandidatos()
//
//    Início código thread
(5)   Para os (m) serviços (s) armazenados em cada UDDI faça
(6)       Se serviço desejado (sd) faz matchingUBL com serviço (s) armazenado UDDI então
(7)           Enquanto valorQoS(sd) atendido por (s) e tem atributos analisar (sd) faça
(8)               valorAtributoQoS <- listaQoS.obtemAtributo(sd);
              Fimenquanto
(9)           Se todos atributos (sd) foram atendidos por (s) então
(10)              listaServiçosCandidatos.adiciona(s)
              Fimse
          Fimse
      Fimpara
//
//    Fim código thread
(11)  Retorne (listaServiçosCandidatos)
Fim
```

Figura 53 - Algoritmo básico do *crawling*.

O uso de *threads* na implementação dos algoritmos de descoberta (tanto no *crawling* como no algoritmo de DDS, este último descrito nos próximos parágrafos) se justifica em razão de dois pontos chave. O primeiro ponto está relacionado ao cenário de distribuição considerado para os repositórios de serviços (em computadores remotos). Assim, para cada UDDI, disponível na federação, foi criada uma *thread* para executar a descoberta. Isto facilitou a análise isolada dos resultados das descobertas efetuadas e a identificação de erros de conectividade entre o mecanismo de descoberta e uma determinada UDDI.

O segundo ponto se refere ao compartilhamento de recursos. Como a descoberta em uma UDDI é independente das demais, optou-se por criar *thread*s para favorecer o compartilhamento de recursos, já que enquanto uma *thread* atua na descoberta propriamente dita (fazendo acesso e pesquisa em uma UDDI), outra pode estar usando o processador para outra atividade.

A Figura 54 mostra um exemplo de resultado produzido pelo *crawling*. Nela, a primeira coluna informa a relação dos nomes dos serviços descobertos (tais como: *AuthorizePayment* e *Notify of Payment*), a classificação da atividade dentro da ontologia UBL

(*.../accountingCustomer/authorizePayment*) é apresentada na segunda coluna, a quantidade de características de QoS desejados para uma dada atividade é mostrada na terceira coluna, identificada por *QoS Constraints*. Nessa coluna, tem-se o total de características de QoS requeridas na descoberta (neste caso apenas uma (1) característica foi informada). Ainda na Figura 54, na quarta coluna, a expressão identificada por *Matching Services* revela o total de serviços descobertos que satisfazem à expressão da descoberta.

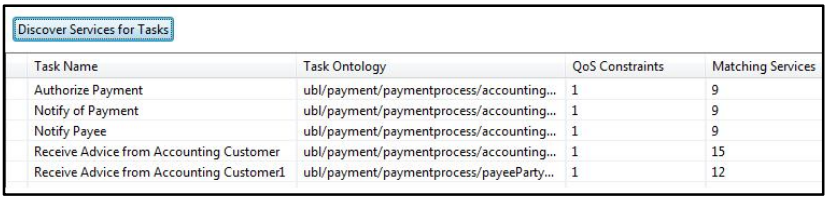


Figura 54 - Exemplo de serviços retornados pelo *crawling*.

Como o *crawling* foi implementado através de um serviço *web*, sua invocação requer o conhecimento da sua definição, a qual é expressa através de um arquivo no formato WSDL, disponível em: *http://localhost:8070/services/DiscoveryService/DiscoveryService.wsdl*. De posse da definição do serviço, um módulo que deseja usar o *crawling* deve proceder da seguinte forma:

- Instanciar um objeto da classe *Crawler*. Junto a este objeto, três parâmetros devem ser informados. O primeiro é a lista de UDDIs presentes e disponíveis na federação. O segundo parâmetro é a funcionalidade desejada para implementar a atividade (obtida a partir da ontologia UBL). O terceiro parâmetro é o conjunto de características de QoS requeridas (obtido a partir da ontologia de QoS);
- Criar um objeto pertencente à classe *DiscoveryResult* para receber a lista de serviços candidatos retornada pelo *crawling*.

É importante salientar que, caso um usuário projetista de aplicações informe o uso da reputação do provedor, a lista de serviços candidatos retornada pelo *crawling* é classificada e apresentada de forma decrescente em relação à reputação do provedor.

O algoritmo de DDS (o submódulo que faz a descoberta em tempo de execução de aplicações) parte da lista de UDDIs disponíveis

na federação e da lista de serviços candidatos associada à atividade em execução.

```
Algoritmo DescobertaDinâmicaServiço
  Entradas: Lista de UDDIs, Atividade (serviço desejado, restrições de QoS e lista de serviços candidatos).
  Saída: (1) um serviço.
Início
(1)   Enquanto listaServiçosCandidatos() não vazia faça  //   Varre a lista de serviços candidatos
(2)     Retira servico (s) <- listaCandidatos()
(3)     Se serviço (s) está disponível e acessível então
(4)        Vincular serviço (s) à aplicação
(5)        Retorne (s)     // ponto de término da descoberta dinâmica
(6)     Fimse
(7)   FimEnquanto
(8)   Para as (n) UDDIs disponíveis na Federação faça // Faz uma efetiva descoberta
(9)       thread.cria <- listaUDDI.get()
(10)      thread.dispara
(11)  Fimpara
//    Início da execução da thread
(12)  Para os (m) serviços (s) armazenados em cada UDDI faça
(13)     Se serviço desejado (sd) faz matching com serviço (s) armazenado na UDDI então
(14)        Enquanto valorQoS (sd) é atendido por (s) e houver atributos a serem analisados (sd) faça
(15)           valorAtributoQoS <- listaQoS.obtemAtributo(sd);
(16)        Fimenquanto
(17)        Se todos atributos (sd) foram atendidos por (s) então
(18)           Se serviço (s) está disponível e acessível então
(19)              Vincular serviço (s) à aplicação
(20)              Retorne (s)     // ponto de término da descoberta dinâmica
(21)           Fimse
(22)        Fimse
(23)     Fimse
(24)  Fimpara
//    Fim da execução da thread
Fim
```

Figura 55 - Algoritmo usado pelo ambiente de execução para descobrir dinamicamente serviços.

O algoritmo, nas linhas 1 a 7 (ver Figura 55) varre a lista de serviços candidatos da atividade corrente, averiguando a disponibilidade de cada serviço alocado previamente. Caso o serviço esteja disponível e acessível, a descoberta é finalizada (linha 7 da Figura 55), sendo que o serviço é imediatamente vinculado à atividade e a aplicação continua sua execução. Não havendo serviço candidato que possa ser usado na execução da aplicação (da atividade corrente), ocorre a descoberta dinâmica (linhas 8 a 24 da Figura 55). *Threads* são criadas e disparadas para cada uma das UDDIs disponíveis e ativas na Federação (linhas 8 a 11 da Figura 55). A partir deste ponto, o processo de descoberta acontece de maneira semelhante ao algoritmo de *crawling*. Todos os serviços são analisados, entretanto o primeiro a atender os valores contidos na expressão da descoberta é escolhido e imediatamente vinculado à aplicação. A DDS termina no momento que encontrar o

primeiro serviço que atenda aos critérios funcionais, de QoS desejados e esteja disponível e acessível (linha 20 da Figura 55).

Fundamentalmente, o algoritmo de DDS difere do algoritmo do *crawling* em dois aspectos. Antes de fazer uma busca efetiva, o algoritmo de DDS varre a lista de serviços candidatos até encontrar um primeiro serviço disponível e acessível. O segundo aspecto é referente ao término da descoberta. A descoberta dinâmica é finalizada quando encontra um (o primeiro) serviço que satisfaz a expressão da descoberta.

O algoritmo da descoberta dinâmica também foi implementado através de um serviço *web*. A sua invocação requer o conhecimento do arquivo que contém a sua definição (WSDL), disponível em *http://localhost:8069/services/DiscoveryService/DynamicDiscoveryService.wsdl*. O mecanismo de execução WS-BPEL é o responsável pela invocação do serviço de descoberta dinâmica, fazendo isto através das primitivas <*invoke*> e <*receive*> da linguagem WS-BPEL.

De forma similar à descoberta via *crawling*, na execução da descoberta dinâmica também ocorre a geração do contrato de uso do serviço (SLA) entre as partes envolvidas. Como forma de ilustrar a geração do SLA, um arquivo no formato XML foi criado usando como base a estrutura proposta por (CANCIAN; RABELO; VON WANGENHEIM, 2009). A Figura 56 mostra um fragmento do SLA gerado para um serviço hipotético. O elemento XML <*AcordoGeral*> informa as bases do contrato. O elemento XML <*Objetivo*> lista e descreve os objetivos do contrato. O elemento XML <*Responsáveis*> apresenta as partes envolvidas no contrato.

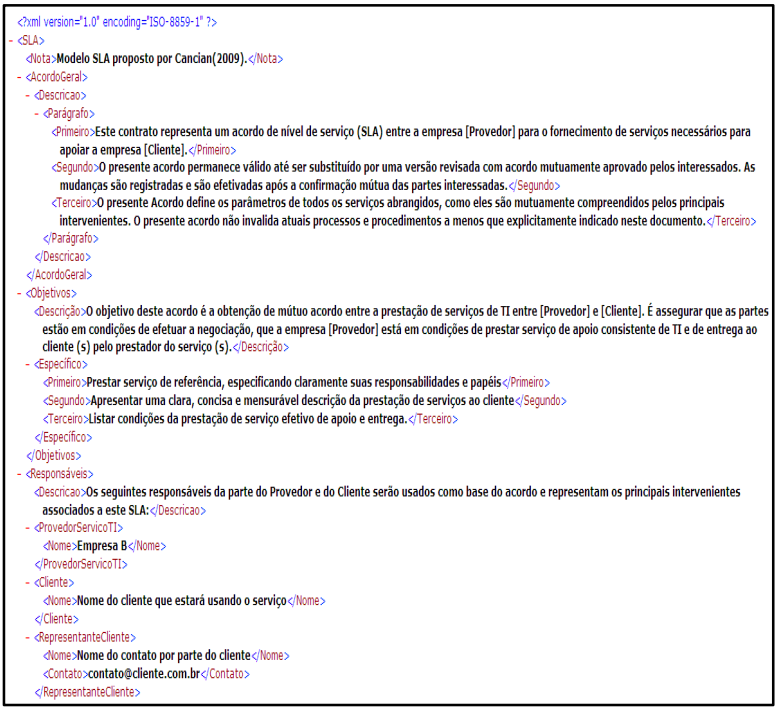


Figura 56 - Fragmento XML do SLA gerado para documentar o uso serviços.

5.2.5 Implementação do Módulo do Ambiente de Execução de Aplicações

O ambiente de execução de aplicações é o local onde as aplicações prontas (no formato WS-BPEL) são armazenadas e executadas. Nele, conforme comentado no Capítulo 4, há um repositório de aplicações WS-BPEL, ou seja, uma pasta onde as aplicações WS-BPEL ficam armazenadas para serem executadas por um motor. O motor de execução de aplicações usado foi o Apache ODE (APACHE, 2010b). Em razão da necessidade de visualizar o estado das aplicações em execução, funcionalidade não oferecida pelo Apache ODE, houve a necessidade de usar a *suíte* BPMS *Intalio* v. 6.0.3.050, que já incorpora o Apache ODE. Com isso, bastou armazenar as aplicações prontas na pasta de aplicações do *Intalio* para que o mesmo, através de uma interface gráfica, permitisse o início da execução de aplicações (usando

o Apache ODE) e a visualização do estado das mesmas. Um exemplo da interface gráfica disponibilizada pelo *Intalio* para gerenciamento de aplicações pode ser visualizado em detalhes na Seção 5.3, apresenta um exemplo do funcionamento do protótipo computacional.

5.2.6 Implementação do Módulo da Federação

Embora o desenvolvimento da federação não seja o foco principal do trabalho, foi necessário implementar algumas funcionalidades básicas para garantir a operação da mesma e, consequentemente, a operacionalização completa do protótipo computacional. As subseções a seguir descrevem este conjunto mínimo de funcionalidades desenvolvidas.

5.2.6.1 Repositório de Ontologias

Fazem parte do repositório de ontologias, a ontologia de QoS e a ontologia de processos UBL. Conforme comentado no Capítulo 4, conceitualmente a construção da ontologia de processos UBL foi baseada na especificação *Universal Business Language* v. 2 (OASIS, 2006) e ocorreu em três etapas macros. A primeira compreendeu o estudo e análise da especificação UBL. A segunda contemplou a identificação dos atores e a interpretação dos diagramas responsáveis pelo detalhamento dos processos de negócios UBL.

Na terceira etapa, da construção da ontologia UBL, a ferramenta usada para edição foi o *Protégé* v. 4.0.2. A escolha do uso do *Protégé* recaiu sobre três fatores. O primeiro por ser um editor com funcionalidades intuitivas e estáveis. O segundo por ser um editor *open source*, gratuito e largamente usado na construção de ontologias. O terceiro por fornecer suporte à linguagem OWL 2 (*Web Ontology Language*) (W3C, 2009), padrão usado para descrição de ontologias.

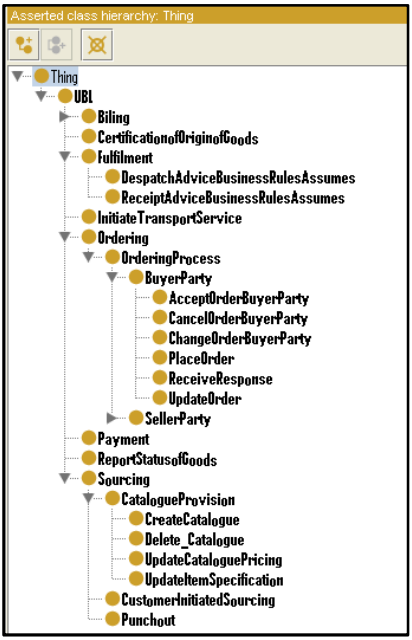


Figura 57 - Classificação criada a partir do padrão UBL e escrita através do *Protégé*.

A Figura 57 mostra a classificação criada a partir da especificação UBL. Nela, a classe *UBL* representa o nome da especificação de processos de negócio usada. A partir da classe *UBL*, diversas subclasses foram definidas (*Ordering* e *Biling*, por exemplo), representando a categoria do processo dentro da ontologia, conforme comentado na Seção 4.2.3.3.1. Para cada classe que define a categoria de um processo, criou-se uma subclasse que representa o nome do processo (*OrderingProcess*). Dentro de um processo, há atores que representam as partes que interagem com as atividades de um processo de negócios. Por exemplo, no processo *OrderingProcess* há dois atores *BuyerParty* e *SellerParty*, também representados por uma classe dentro da ontologia. No último nível hierárquico, há classes que representam as atividades executadas dentro de um processo, como por exemplo: *AcceptOrderBuyerParty* dentro do processo *OrderingProcess*.

A definição da subclasse *OrderingProcess*, que tem como super-classe *Ordering*, é exemplificada em OWL 2, conforme mostra a Figura

58. Para mais detalhes da versão da ontologia UBL em OWL, consultar o Apêndice A deste documento.

```
<owl:Class rdf:about="#Ordering">
    <rdfs:subClassOf rdf:resource="#UBL"/>
</owl:Class>


<!-- http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#OrderingProcess -->

<owl:Class rdf:about="#OrderingProcess">
    <rdfs:subClassOf rdf:resource="#Ordering"/>
</owl:Class>
```

Figura 58 - Definição das classes *Ordering* e *OrderingProcess* em OWL 2.

Como comentado na Seção 4.2.3.3.1, a construção da ontologia de QoS foi inspirada na ontologia proposta por Tondello (2008), a qual teve como base a definição de QoS (OMG, 2006).

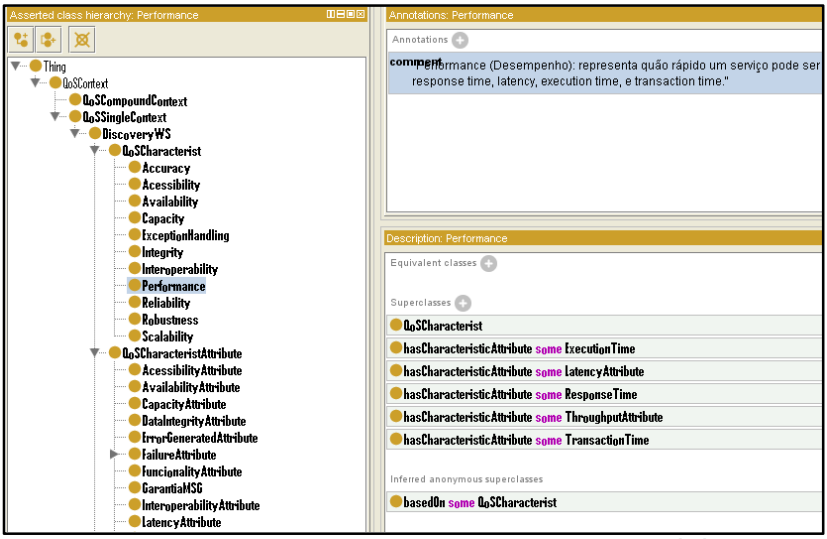


Figura 59 – Ilustração da edição da ontologia de QoS usando *Protégé*.

A Figura 59 ilustra a edição da ontologia de QoS através do *Protégé*. No lado esquerdo, tem-se um conjunto de classes que formam a hierarquia da ontologia de QoS. No lado direito, mostra-se um exemplo de relacionamento entre a classe *QoSCharacteristic* e a classe *QoSCharacteristicAttribute*. O relacionamento *hasCharacteristic* faz a

associação de uma característica de QoS (no caso *Performance*) com os atributos que a formam (*ExecutionTime*, *LatencyAttribute*, *ResponseTime*, *ThroughputAttribute* e *TransactionTime*). Cabe salientar que, conforme já mencionado na Seção 4.2.3.3.1, há outros tipos de relacionamentos existentes que definem a ontologia de QoS. Por exemplo, o relacionamento que permite ligar um atributo de QoS a uma unidade de medida (segundos, percentual, etc).

A definição completa em OWL 2 da ontologia de QoS 2 pode ser consultada no Apêndice B deste documento.

5.2.6.2 Repositórios de Serviços

Os repositórios de serviços foram instanciados na forma de UDDIs. Concretamente, o projeto jUDDI da Apache (APACHE, 2010a) foi usado como implementação dos repositórios de serviços.

Além das informações sobre serviços, cada uma das jUDDIs criadas armazenam informações sobre os aspectos de qualidade (descritos na ontologia de QoS) dos serviços e informações sobre o contexto dos serviços (através da ontologia UBL). Na prática, foram criadas instâncias do elemento <*tModel*>, presente na estrutura de dados das jUDDIs, para representar e armazenar aspectos de QoS e o contexto associado aos serviços.

A Figura 60 descreve como estas instâncias representam valores de QoS. A primeira instância do elemento <*tModel*> armazena informações sobre uma dada característica de QoS (*Accuracy*). Dentro dele, tem-se o elemento <*name*> que informa o nome da característica, o elemento <*description*> que registra uma breve descrição da característica e o elemento <*keyedReference*> que armazena informações de controle usadas para representar informações de QoS.

A segunda instância do elemento <*tModel*>, ainda na Figura 60, armazena informações sobre o atributo *GeneratedError*, usado para determinar a característica *Accuracy*. As instâncias do elemento <*tModel*> se conhecem através da existência de um propriedade (*keyvalue=”Accuracy”*) presente no elemento <*keyedReference* > do elemento <*CategoryBag*>.

No elemento <*CategoryBag*>, da segunda instância do elemento <*tModel*> (Figura 60), agrupam-se os atributos que definem a característica *Accuracy*, determinada a partir do valor armazenado no

atributo *GeneratedError* (erros gerados) que, por sua vez, tem como unidade *ErrorNumber* (número de erros gerados).

```
<!-- Accuracy -->
- <tModel tModelKey="uddi:gsigma.ufsc.br:qos:accuracy">
    <name>gsigma-ufsc-br:qos:accuracy</name>
    <description>Accuracy: defined as the service generated error rate. * TotalGeneratedError: number of errors that a WS generate in some time space
      minimum).</description>
  - <categoryBag>
      <keyedReference tModelKey="uddi:gsigma.ufsc.br:qos:item" keyName="uddi:gsigma.ufsc.br:qos:accuracy" keyValue="Accuracy" />
      <keyedReference tModelKey="uddi:gsigma.ufsc.br:qos:item:id" keyName="uddi:gsigma.ufsc.br:qos:item:id" keyValue="11" />
      <keyedReference tModelKey="uddi:uddi.org:categorization:types" keyName="uddi:gsigma.ufsc.br:qos:item" keyValue="QoSItem" />
    </categoryBag>
  </tModel>
- <tModel tModelKey="uddi:gsigma.ufsc.br:qos:accuracy:generatederror">
    <name>gsigma-ufsc-br:qos:accuracy:generatederror</name>
    <description>QoS -> Accuracy -> Generated Error</description>
  - <categoryBag>
      <keyedReference tModelKey="uddi:gsigma.ufsc.br:qos:item" keyName="uddi:gsigma.ufsc.br:qos:accuracy" keyValue="Accuracy" />
      <keyedReference tModelKey="uddi:gsigma.ufsc.br:qos:attribute" keyName="uddi:gsigma.ufsc.br:qos:accuracy:generatederror" keyValue="GeneratedError" />
      <keyedReference tModelKey="uddi:gsigma.ufsc.br:qos:unit" keyName="uddi:gsigma.ufsc.br:qos:unit:errornumber" keyValue="ErrorNumber" />
      <keyedReference tModelKey="uddi:uddi.org:categorization:types" keyName="uddi:gsigma.ufsc.br:qos:attribute" keyValue="QoSAttribute" />
    </categoryBag>
  </tModel>
```

Figura 60 - Exemplo de representação de QoS em uma jUDDI.

A Figura 61 ilustra valores da ontologia UBL representados em uma jUDDI através de instâncias do elemento <*tModel*>. O elemento <*name*> armazena e descreve a classificação UBL para um dado serviço. O elemento <*description*> representa esta mesma classificação em um formato que humanos podem compreender. O elemento <*overviewDoc*> informa a localização da interface que descreve o serviço e o elemento <*categoryBag*>, além das informações de controle, informa o contexto do serviço através do elemento <*keyedReference*> que representa e informa os efetivos valores que determinam a funcionalidade do serviço em relação à ontologia UBL.

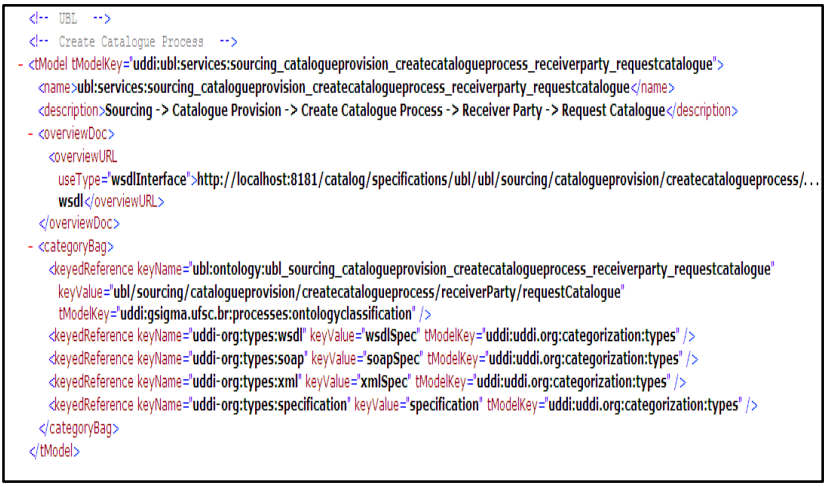


Figura 61 - Exemplo de valores da ontologia UBL em uma jUDDI.

Definida a forma de armazenamento dos valores de QoS e dos valores da ontologia UBL nas jUDDIs, o passo posterior foi em direção a como representar os vários repositórios de serviços presentes na federação. A estratégia usada foi a de criar um arquivo no formato XML com nome de *uddifederation_BusinessEntity.xml*. O arquivo fica armazenado na jUDDI identificada por *ubl-uddi*, porque ela é a primeira jUDDI a ser criada e disponibilizada pela federação. A Figura 62 ilustra uma típica entrada no arquivo. Nele, cada elemento <*businessService*> contém informações sobre uma específica jUDDI. Além do nome e de uma breve descrição do repositório, o elemento <*bindingTemplate*> e <*accessPoint*> informam, respectivamente, o endereço e a porta de acesso à WSDL que especifica a interface dos serviços ofertados pela API da jUDDI, já que todas as jUDDIs são disponibilizadas como serviços *web*.

```
...
- <businessServices>
  - <businessService serviceKey="uddi:uddifederation:uddiprovider1" businessKey="uddi:uddifederation:uddi-federation">
      <name xml:lang="en">UDDI Inquiry Service - Provider 1</name>
      <description xml:lang="en">Web Service supporting UDDI Inquiry API</description>
    - <bindingTemplates>
      - <bindingTemplate bindingKey="uddi:juddi.apache.org:servicebindings-uddiprovider1-ws" serviceKey="uddi:uddifederation:uddiprovider1">
          <description>UDDI Inquiry API V3</description>
          <accessPoint useType="wsdlDeployment">http://localhost:8079/juddiv3/services/inquiry?wsdl</accessPoint>
        - <tModelInstanceDetails>
            <tModelInstanceInfo tModelKey="uddi:uddi.org:v3_inquiry" />
            <tModelInstanceInfo tModelKey="uddi:uddifederation:uddiserviceprovider" />
          </tModelInstanceDetails>
        - <categoryBag>
            <keyedReference keyName="uddi-org:types:wsdl" keyValue="wsdlDeployment" tModelKey="uddi:uddi.org:categorization:types" />
          </categoryBag>
        </bindingTemplate>
      </bindingTemplates>
    </businessService>
...
```

Figura 62 – Código XML que registra o acesso a uma jUDDI na federação.

5.2.6.3 Publicação de Serviços

O processo de publicação de serviços foi realizado através de um programa específico desenvolvido na linguagem Java. Nele, o primeiro passo ocorre com o registro de provedor. O método *createPublisers* é o responsável por esta tarefa, conforme ilustra o fragmento de código Java apresentado na Figura 63.

O código Java, mostrado na Figura 63, cria um usuário para acessar a jUDDI e posteriormente registra este usuário (linhas 5 a 12). Concluído o registro, usa-se um laço de repetição para criar diversos provedores (linhas 14 e 16 a 19).

```
1 private static void createPublishers(Transport transport, int k) throws Exception {
2
3       UDDISecurityPortType security = transport.getUDDISecurityService();
4
5       GetAuthToken getAuthTokenRoot = new GetAuthToken();
6       getAuthTokenRoot.setUserID("root");
7       getAuthTokenRoot.setCred("");
8
9       AuthToken authToken = security.getAuthToken(getAuthTokenRoot);
10
11      SavePublisher savePublisher = new SavePublisher();
12      savePublisher.setAuthInfo(authToken.getAuthInfo());
13
14      for (int i = 1; i <= k; i++) {
15
16              Publisher publisher = new Publisher();
17              publisher.setAuthorizedName("ubl" + i);
18              publisher.setPublisherName("Company " + i);
19              savePublisher.getPublisher().add(publisher);
20
21      }
22
23      transport.getJUDDIApiService().savePublisher(savePublisher);
24 }
```

Figura 63 – Fragmento de código Java que faz o registro de provedor.

Após a criação dos provedores, o passo posterior foi a publicação de serviços. O método *populateServices* presente na Figura 64 é invocado (linha 1) informando dados do registro do provedor e a jUDDI alvo da publicação.

```
1 private static void populateServices(Transport transport, String authLogin, String authPassword, String businessKey,
2       String businessEntityName) throws Exception {
3       if (businessKey == null) {
4          BusinessDetail businessDetail = createBusinessEntity(transport, authLogin, authPassword, businessEntityName);
5               if (businessDetail.getBusinessEntity() != null && businessDetail.getBusinessEntity().size() > 0) {
6                       businessKey = businessDetail.getBusinessEntity().get(0).getBusinessKey();
7               }
8
9               // Ordering Process
10              publishService(transport, authLogin, authPassword, businessKey, //
11                      "UBLOrderingProcessBuyerPartyPlaceOrder", //
12                      "UBL Payment -> Ordering Process -> Buyer Party -> Place Order", //
13                      "http://localhost:8082/services/ubl/orderingprocess/buyerParty/placeOrder", //
14                      new TModelInstanceInfo[] {
15                      getTModelInstanceInfo("uddi:ubl:services:ordering_orderingprocess_buyerparty_placeorder") }, //
16                      buildRandomQos());
17
18                      publishService(transport, authLogin, authPassword,
19                        businessKey, //
20                        "UBLOrderingProcessSellerPartyReceiveOrder", //
21                        "UBL Payment -> Ordering Process -> Seller Party -> Receive Order", //
22                        "http://localhost:8082/services/ubl/orderingprocess/sellerParty/receiveOrder", //
23                        new TModelInstanceInfo[] {
24                        getTModelInstanceInfo("uddi:ubl:services:ordering_orderingprocess_sellerparty_receiveorder") }
25                        buildRandomQos());
...
```

Figura 64 - Fragmento de código Java usado para publicar serviços.

Posteriormente, ainda dentro do método *populateServices*, chama-se o método *publishService* (Figura 64, linha 10) que efetivamente realiza a publicação do serviço. Ainda na Figura 64, nas linhas 10 a 18, publicam-se dois serviços pertencentes ao processo *OrderingProcess*. O primeiro chamado de *PlaceOrder* e o segundo *ReceiveOrder* (linhas 10 a 16 e 18 a 25, respectivamente).

5.2.6.4 Reputação do Provedor

Como comentado na Seção 4.2.3.3.4, optou-se por determinar a reputação do provedor através da geração de números aleatórios entre 0 a 100. Por um lado, isso caracteriza uma limitação do trabalho. Por outro, abre oportunidade para que pesquisas futuras possam identificar e escolher um método mais adequado para representar e determinar a reputação do provedor ou definir outros critérios de avaliação de serviços (LI *et al.*, 2009).

5.2.6.5 Serviços UBL

Os serviços UBL fornecem a implementação concreta para os serviços *web* usados durante a execução da aplicação WS-BPEL. Eles foram implementados utilizando o *framework Apache CXF*, que fornece a estrutura para a concepção de serviços *web*. Utilizou-se a ferramenta *wsdl2java* fornecida com o CXF para fazer a importação dos arquivos WSDL. O resultado da execução da ferramenta é um conjunto de arquivos Java que descrevem as operações e tipos de dados trocados pelos serviços. Um desses arquivos é a interface de serviço, que é referenciada pela classe Java que implementa o mesmo. No Apêndice C pode-se ver um extrato do arquivo WSDL utilizado para descrever as interfaces dos serviços UBL.

É possível que as aplicações geradas em WS-BPEL permitam a participação humana, ou seja, que certas atividades pertencentes a aplicação dependam da decisão humana para retornar a resposta. Como forma de resolver esse problema, foi criada a especificação BPEL4People (OASIS, 2010), que lida com esses aspectos de como modelar a interação humana dentro de um processo WS-BPEL. Nesse trabalho, modela-se a requisição e resposta dos serviços *web* de forma assíncrona. Na visão deste autor, isto faz sentido, pois são atividades que necessitam da intervenção humana para retornar uma resposta. Neste sentido, é difícil estimar o tempo de resposta do serviço, que inclusive pode ser muito grande, o que acaba inviabilizando o uso do mecanismo de comunicação síncrono dos serviços. Portanto, as operações que esses serviços *web* provêm são executadas de forma assíncrona. Quando o serviço recebe uma requisição, um dos parâmetros informa qual é o endereço de resposta do requisitante e qual o identificador associado. Isso permite que o serviço, logo quando tiver condição de responder a requisição, isto é, quando a pessoa por trás dele responder, o faça ao destinatário correto. Como forma da pessoa poder responder a essa requisição, utilizou-se o artifício de sempre mostrar uma mensagem de confirmação ao usuário quando esta requisição é feita. Só depois de o usuário confirmar essa mensagem é que a resposta é enviada ao requisitante (a aplicação WS-BPEL). Apesar de ser simples, esse artifício permite ao mecanismo de execução de aplicações e aos serviços vinculados estarem aptos a suportar a interação humana.

Soluções reais disponíveis no mercado modelam a interação humana geralmente implementando sistemas de *workflow* completos, onde, por exemplo, o responsável pela atividade recebe uma lista de tarefas pendentes, geralmente associadas a formulários que precisam ser

preenchidos com os dados requeridos pela aplicação. Essa lista de tarefas é organizada numa fila, de forma que o usuário possa estar envolvido com várias aplicações. O mecanismo de *workflow* cuida de encaminhar as respostas do usuário às instâncias corretas das aplicações que as requereram.

No que tange à execução dos serviços UBL implementados, utilizou-se o servidor *web Jetty* (CODEHAUS, 2010), que é um pequeno servidor HTTP que executa em conjunto com o *Apache CXF*. Com o seu uso, evita-se implantar serviços em um *container web* completo, como o Apache *Tomcat* (APACHE, 2010c), poupando recursos de hardware e agilizando o desenvolvimento do protótipo.

Os serviços UBL foram publicados nos repositórios de serviços distribuídos e gerenciados pela federação usando o mesmo código Java apresentado na Seção 5.2.6.3.

5.2.7 Visão geral da Arquitetura de Implementação

A visão geral da arquitetura de implementação do protótipo computacional está estruturada em camadas, sendo que cada camada possui um conjunto de módulos (já descritos nas seções anteriores) que correspondem a unidades de código usadas na implementação do modelo proposto.

A Figura 65 mostra a arquitetura de implementação do protótipo computacional através de um diagrama de pacotes de classes da UML. Nele é possível observar a ligação entre cada uma das camadas que formam o protótipo computacional, sendo elas: (1) Conector de integração com o ambiente de edição de aplicações, (2) Catálogo de processos de negócios, (3) Ambiente de Descoberta de Serviços, (4) Federação de Provedores e (5) Ambiente de Execução de Processos.

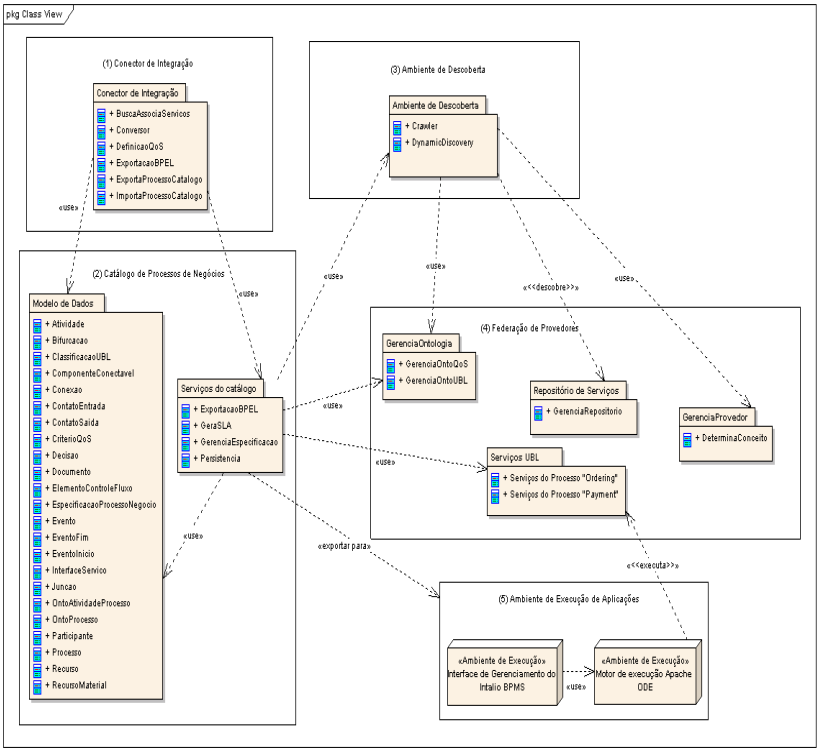


Figura 65 - Arquitetura de Implementação do Protótipo Computacional.

5.3 **Exemplo de Funcionamento**

Esta seção apresenta um exemplo do funcionamento do protótipo computacional. O início do exemplo começa com o usuário projetista iniciando a ferramenta de edição de processos, no caso o *IBM WebSphere Business Modeler*. Em seguida, o projetista escolhe a opção de importar um processo do catálogo para o editor. O conector de integração se conecta ao catálogo e lista todos os processos que estão disponíveis. Na Figura 66, o projetista escolhe (no caso *PaymentProcess*) um deles e solicita a importação.

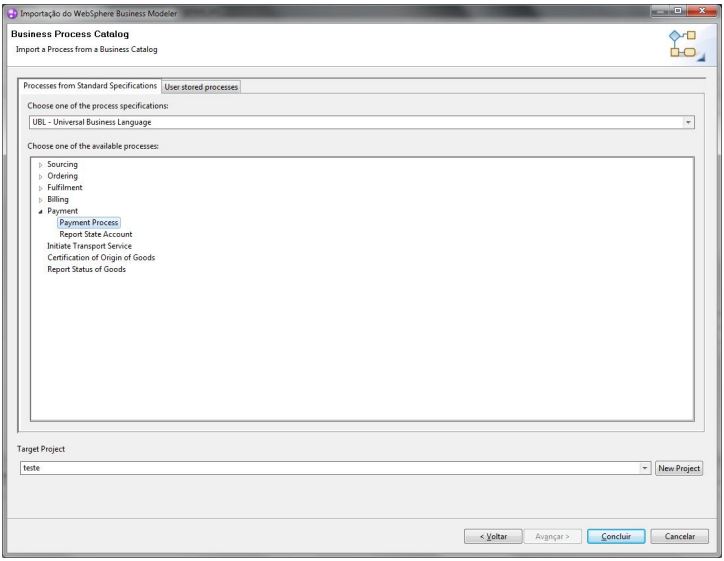


Figura 66 - Importação de processo do catálogo.

Após a importação ser concluída e incorporada à aplicação em desenvolvimento, conforme mostra a Figura 67, o projetista pode fazer customizações ou iniciar o processo de descoberta, selecionando uma atividade da aplicação.

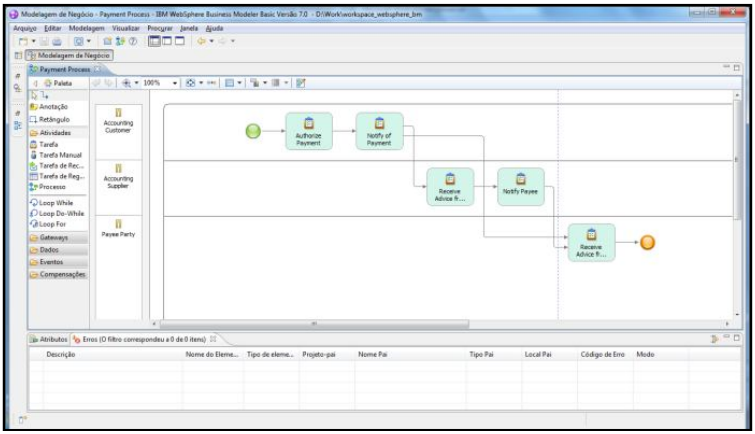


Figura 67 – Processo importado para o editor de aplicações.

Escolhida a atividade, o projetista informa o conjunto de restrições de QoS desejadas. Para isso, ele usa o botão *Add QoS Constraint* que mostra uma interface gráfica com as características de QoS disponíveis com respectivos atributos permitidos, conforme ilustra a Figura 68.

Figura 68 – Interface com características e atributos de QoS.

O projetista seleciona atributos, informa valores desejados, escolhe a unidade de medida e informa um designador de comparação (maior, menor, igual, maior igual ou menor igual) que determina a direção da checagem dos valores de QoS, conforme ilustra a Figura 69, para a característica *Performance* e atributo *ResponseTime*.

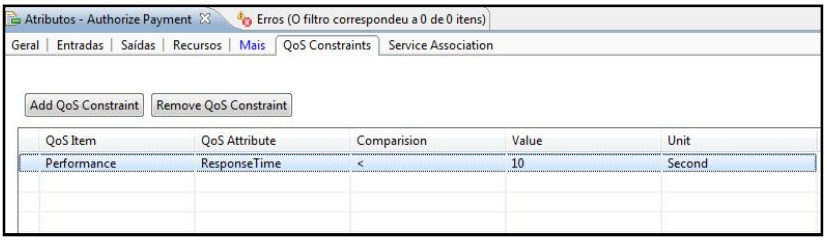


Figura 69 – Definindo atributos para uma dada característica de QoS.

O próximo passo é usar a opção *Discover Services* (Figura 70), que invoca o mecanismo de descoberta (neste caso o *crawling*), levando consigo a classificação ontológica da atividade (o nome do serviço procurado que neste caso é *authorizePayment*, bem como as entradas e saídas do serviço) e a lista de critérios de QoS requeridos.

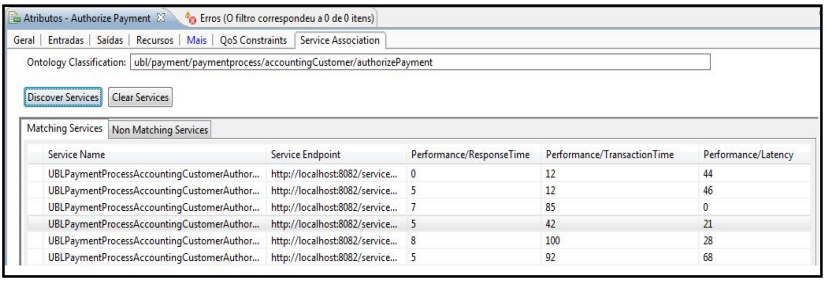


Figura 70 – Interface que permite informar a atividade a ser descoberta.

O retorno do *crawling* é apresentado na Figura 70, do lado esquerdo e abaixo. O resultado é dividido em duas partes (separadas pelas abas *Matching Services* e *Non Matching Services*). A aba *Matching Services* lista os serviços perfeitos (que atendem a expressão da descoberta) e a aba *Non Matching Services* lista os serviços que atendem parcialmente a expressão da descoberta (atendem ao critério funcional – são serviços que implementam a atividade *authorizePayment* – mas não atendem QoS). Ainda na Figura 70, a lista de serviços candidatos que satisfazem os requisitos da descoberta, para atividade *authorizePayment*, são apresentados na coluna *Service Name*. A coluna *Service Endpoint* lista o endereço de utilização do serviço e as colunas três, quatro e cinto mostram o número de serviços retornados que atendem, para a característica *Performance*, valores dos atributos *ResponseTime*, *TransactionTime* e *Latency*.

No caso de nenhum serviço ser retornado pelo mecanismo de descoberta, o projetista tem a opção de relaxar QoS a fim de encontrar algum serviço ou correr o risco de tentar fazer essa descoberta em tempo de execução (quando a aplicação estiver em execução no ambiente de execução de aplicações).

O término do desenvolvimento da aplicação ocorre quando o projetista exporta a aplicação para a linguagem de execução WS-BPEL, conforme ilustra a Figura 71. Nela, o projetista informa a pasta que

deseja armazenar a aplicação pronta. Esta pasta é copiada para o Ambiente de execução de aplicações, local onde a aplicação fica à disponível para ser executada.

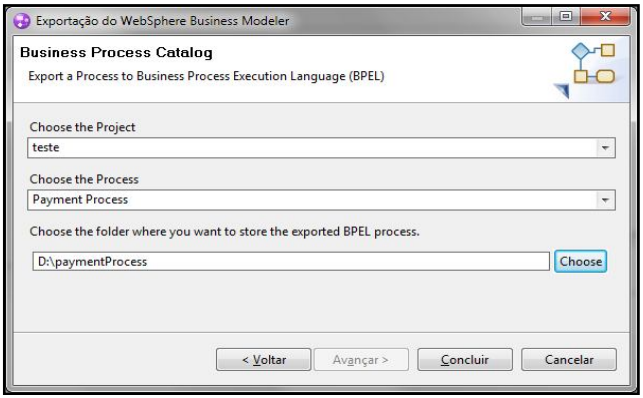


Figura 71 – Exportação para WS-BPEL.

No ambiente de execução de aplicações, conforme mostra a Figura 72, a aplicação *PaymentProcess* está disponível para execução. O usuário executor de aplicações acessa a interface apresentada pela ferramenta *Intalio* BPM e solicita a execução da aplicação WS-BPEL. O motor de execução Apache ODE é acionado e iniciam-se a orquestração dos serviços.

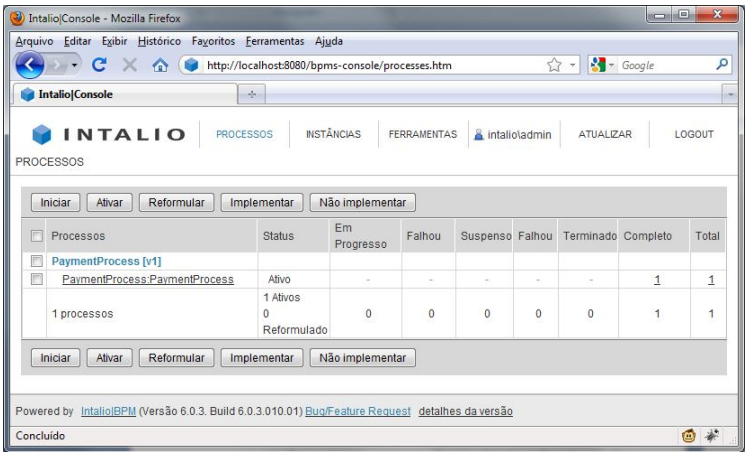


Figura 72 – Ambiente de Execução – Intalio BPMS.

Como forma de simular a interação humana nos processos, cada um desses serviços (que neste exemplo apenas mostram mensagens) solicita uma confirmação visual antes de executar seu processamento (Figura 73). Enquanto a confirmação não for recebida, o processo fica parado.

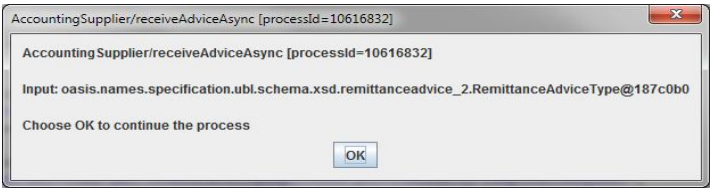


Figura 73 – Mensagem de confirmação – Serviço UBL.

Depois de todos os serviços envolvidos no processo retornarem com uma resposta, o processo é finalizado. Isto pode ser visualizado através da interface disponível no *Intalio* BPM.

5.4 **Considerações**

O capítulo apresentou um conjunto de aspectos relacionados à implementação dos elementos conceituais do modelo proposto. A arquitetura de implementação do protótipo foi apresentada e descrita, bem como comentou-se sobre as tecnologias usadas no processo de desenvolvimento do protótipo computacional.

Para ilustrar o funcionamento do protótipo e como os elementos conceituais do modelo proposto concretamente interagem, um exemplo do funcionamento do protótipo foi reproduzido.

Na implementação do protótipo computacional, cabe destacar a ferramenta da IBM *WebSphere Business*. Além de possuir um robusto e poderoso editor BPMN a ferramenta é baseada na plataforma *Eclipse*. Esse aspecto permite estendê-la através do uso de *plug-ins* usados para acesso a componentes de software, tais como o catálogo de processos. Isso poupou tempo e esforço na etapa de implementação do protótipo.

Outro ponto importante na implementação foi o uso do projeto Apache jUDDI para armazenamento de informações de serviços. Através do uso de algumas estruturas de dados prontas foi possível estabelecer uma lista de repositórios disponíveis para serem usados tanto na publicação como na descoberta de serviços. Além de usar estas

estruturas para representar e armazenar aspectos de QoS e o contexto dos serviços.

No que tange as ontologias, percebe-se que a ontologia UBL determina serviços em termos funcionais, dá maior riqueza semântica à descoberta e otimiza a concepção da aplicação SOA. Ademais, ela resolve problemas semânticos e reduz problemas de interoperabilidade, já que clientes e provedores usam um vocabulário comum quando interagem com o mecanismo de descoberta, seja para publicar ou obter serviços. Nesta dimensão, ela cria uma efetiva ponte (por ser baseada em um padrão) entre requisitos dos clientes e provedores heterogêneos, fornecendo maior qualidade na descoberta de serviços, uma vez que ela garante somente o retorno de serviços semanticamente alinhados aos processos UBL (ao nível de negócios).

Em relação à ontologia de QoS, a mesma serve para definir termos comuns a serem usados tanto na publicação de serviços como na descoberta dos mesmos. Além disso, ela permite que serviços sejam selecionados na descoberta através da averiguação dos seus atributos permitindo que seja realizada a escolha do mais adequado serviço. O uso de classes no topo da ontologia QoS deixou-a preparada para absorver melhorias futuras, seja de acréscimo de características, atributos ou unidades, além de possibilitar definir e representar contextos de aplicações diferentes através da classe *QoSContext*.

Em relação ao uso de recomendação e padrões na implementação do protótipo computacional, a principal vantagem obtida foi a compatibilidade entre os diversos artefatos que compõem o protótipo, permitindo reduzir problemas de integração entre os vários componentes. Entretanto, a busca e o entendimento de cada um dos padrões de mercado, usados na construção do protótipo, foi um obstáculo difícil de ser transposto. Cada padrão, antes de ser usado, foi estudado e verificado em relação à aderência aos requisitos exigidos para o modelo de descoberta proposto. Neste contexto, cabe destacar o nível de maturidade do modelo proposto. Isso possibilitou que muito esforço e tempo não fossem desperdiçados com re-trabalhos.

A importância da implementação dos elementos conceituais presentes no modelo proposto, expressas na forma de software, possibilita que o modelo seja avaliado frente a alguns critérios. O próximo capítulo é dedicado a apresentar a avaliação do modelo de descoberta.

Capítulo 6

Avaliação do Modelo

O foco deste capítulo está em avaliar o modelo proposto sob dois aspectos importantes. O primeiro deles avalia o modelo proposto frente à sua especificação funcional. Para isso, testes de unidade e de integração são aplicados nos vários elementos de software que compõem a arquitetura do protótipo computacional. Além disso, para o mecanismo de descoberta proposto determina-se a sua complexidade computacional visando observar a sua eficiência no processo de integração BPM&SOA. O segundo aspecto é avaliar o modelo proposto em relação à sua utilização. Objetivo está em identificar a efetiva, ágil e transparente desejada integração BPM&SOA. Para atingir este objetivo, foram realizadas duas atividades básicas. Uma pesquisa de campo foi realizada com especialistas usando o protótipo computacional tanto na tarefa de criar como de executar aplicações e, como segunda atividade, ao longo do desenvolvimento da tese, um conjunto de publicações foram realizadas sobre o modelo proposto. Os resultados obtidos a partir dos dois aspectos descritos são fundamentais para responder a duas questões fundamentais: o cumprimento dos objetivos e a comprovação da hipótese de pesquisa.

O capítulo inicia descrevendo o processo de verificação na seção 6.1. A seção 6.2 apresenta e comenta sobre o processo de validação adotado. A seção 6.3 comenta sobre as publicações realizadas

envolvendo a proposta. A seção 6.4 fecha o capítulo com as considerações finais sobre o mesmo.

6.1 **Verificação**

A verificação visa mostrar que um software atende a sua especificação (SOMMERVILLE, 2007). Ela faz parte de um processo que inicia com revisões de requisitos, revisões de projeto e inspeções, até chegar aos testes. Durante este processo de verificação, duas técnicas são classicamente empregadas: as inspeções e os testes de software (SOMMERVILLE, 2007).

As inspeções de software objetivam analisar e verificar representações do software: diagramas, requisitos, código-fonte, etc. As inspeções tipicamente são aplicadas durante o processo de desenvolvimento e constituem uma técnica estática de verificação e validação. Os testes envolvem executar uma instância do software com os dados de teste e examinar as saídas produzidas a fim de verificar se ele está sendo executado conforme o esperado. Os testes são uma técnica dinâmica de verificação e validação porque trabalham com uma representação executável do software (SOMMERVILLE, 2007).

Com base no exposto acima, aplicou-se nesse trabalho apenas os testes de software. Como o escopo desse trabalho não é o protótipo em si, mas sim o modelo conceitual, considerou-se desnecessário fazer inspeções de software. Portanto, esta seção apresenta a verificação do protótipo computacional implementado. Ela está dividida em duas subseções: testes dos módulos da arquitetura e teste de integração. O teste dos módulos da arquitetura tem como objetivo comprovar que eles, de forma isolada, estão corretos, sem erros, e estão de acordo com a especificação. No teste de integração pretende-se testar esses módulos em conjunto, num ambiente mais realista, onde haja interação entre eles.

6.1.1 Teste dos Módulos da Arquitetura

Testes de Unidade são utilizados na verificação dos módulos ou componentes de um sistema. O seu uso permite detectar erros dentro das fronteiras de um módulo. A complexidade dos testes, como por exemplo, quais erros que este pode detectar, é determinada pelo escopo do teste. O teste de unidade é focado no processamento lógico e nas estruturas de dados dentro do módulo (PRESSMAN, 2001).

Os testes de unidade foram efetuados para os seguintes módulos do protótipo computacional:

- Conector de integração do ambiente de edição de aplicações com o catálogo de processos de negócios;
- Catálogo de Processos de Negócio;
- Federação de Provedores, especificamente nos serviços UBL publicados e ofertados;
- Ambiente de Execução de Aplicações;
- Ambiente de Descoberta de Serviços.

O detalhamento dos testes de unidade está descrito no Apêndice E.

6.1.2 Teste de Integração

Testes de integração é uma técnica sistemática para a construção de arquiteturas de software a fim de garantir a perfeita coesão entre os módulos (PRESSMAN, 2001). O objetivo é, após os testes de unidade, garantir que essas unidades possam trabalhar de maneira conjunta e correta.

Os testes de integração foram efetuados na forma de experimentos realizados com o protótipo computacional. A estratégia para a condução dos experimentos foi a seguinte. Inicialmente, os módulos da arquitetura foram implantados em um ambiente de testes. Apesar de estes módulos terem sido projetados para funcionar de maneira distribuída (visto que se trata de uma arquitetura SOA), optou-se, primeiramente, por instalá-los numa única máquina. Escolheu-se essa abordagem, pois facilita a execução dos testes, visto que não é necessário uma infraestrutura (de rede e máquinas) para sua execução.

Em um segundo momento, repositórios de serviços foram implantados em máquinas remotas. Como os componentes se comunicam utilizando serviços *web*, infere-se que o comportamento dos testes será muito parecido, tanto com os componentes implantados localmente, como de maneira distribuída. Desta forma, é importante salientar que essa afirmação se assenta sobre o ponto de vista lógico, da sequencia de invocações, fluxo de controle e dados, etc., e não do ponto de vista de desempenho, pois isto depende de aspectos de qualidade da rede usada para conectar os vários componentes de software do protótipo.

Foram realizados três experimentos. O primeiro deles visou verificar a precisão, cobertura e o funcionamento integrado dos diversos

módulos do protótipo computacional. O segundo experimento testou o mecanismo de descoberta do protótipo computacional, quando ocorre o aumento tanto do número de serviços publicados como dos repositórios de serviços. O terceiro experimento foi caracterizado pela distribuição de UDDIs em máquinas remotas e considera também o aumento no número de serviços e a quantidade de UDDIs. Neste experimento, pretendeu-se chegar mais próximo de um cenário real de integração entre o nível de negócios e o nível de sistemas, onde diversos provedores heterogêneos publicam seus serviços em repositórios de serviços remotos.

Em razão da dificuldade de implantar todos os módulos da arquitetura em diversas organizações para poder testar e validar o modelo (pois muitos elementos estão em nível de protótipo computacional, sem as devidas condições de serem usados em produção), entende-se que os experimentos são representativos. Esta afirmação é reforçada por dois fatores. O primeiro deles tem relação com os pressupostos adotados para a concepção do modelo proposto. Eles definem um cenário ainda "futurístico"/exploratório da integração do nível de negócios com o nível de sistema. O segundo aspecto está relacionado às operações que sustentam e permitem que a desejada integração BPM&SOA ocorra, ratificando a viabilidade técnica da proposta.

Os três experimentos, descritos nas próximas subseções, foram realizados em um ambiente controlado e são compostos de quatro partes distintas: o objetivo, a descrição do cenário considerado, a execução e uma análise geral dos resultados obtidos a partir dos ensaios com o mecanismo de descoberta.

6.1.2.1 Primeiro Experimento

6.1.2.1.1 Objetivo

O primeiro experimento visou observar o comportamento do mecanismo de descoberta considerando as métricas de precisão e cobertura e verificar o funcionamento do protótipo computacional de forma integrada.

6.1.2.1.2 Cenário

O cenário do primeiro experimento é composto por três (3) repositórios de serviços, instanciados na forma de UDDIs, implantados em (1) uma máquina. Os repositórios foram denominados: *ubl-uddi*, *ubl-uddi2* e *ubl-uddi3*, compondo o conjunto de repositórios presentes e disponíveis na federação.

Como comentado na Seção 5.2.6.2, em um destes repositórios (ubl-uddi) há um arquivo XML designado por *uddifederation_BusinessEntity.xml* que contém diversas entradas, as quais representam as UDDIs usadas durante a descoberta. Antes do início da descoberta, o mecanismo obtém a lista de UDDIs e, logo após, procede à consulta. Após definida a lista de UDDIs participantes da federação, foi necessário editar o arquivo *server.xml*, armazenado dentro da pasta *config*, de cada uma das UDDIs, para configurar o endereço e a porta de acesso, já que cada UDDI foi disponibilizada através de um serviço *web*. A Figura 74 mostra o endereço e a porta de acesso de uma das UDDIs (ubl-uddi) presente na federação. Na prática, isto quer dizer que a UDDI foi iniciada e seu conjunto de funcionalidades está pronto para ser usado.

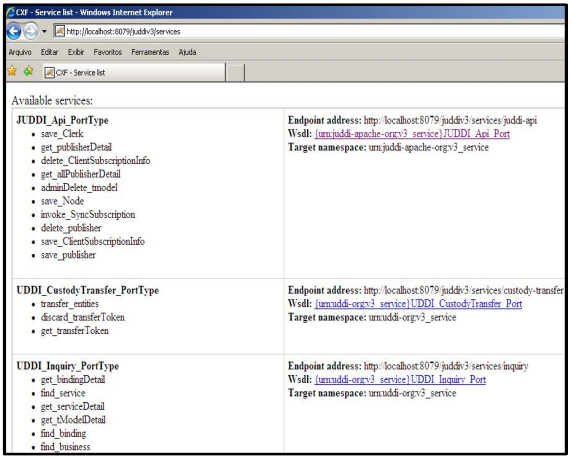


Figura 74 – Exemplo de endereço e porta de acesso da *ubl-uddi* presente na Federação.

O Quadro 25 apresenta a lista de UDDIs presentes na federação utilizadas neste experimento, bem como informa a identificação, o endereço (URL) e a porta de acesso de cada UDDI usada.

Identificação UDDI	URL e Porta
ubl-uddi	http://localhost:8079
ubl-uddi2	http://localhost:8078
ubl-uddi3	http://localhost:8077

Quadro 25 - Identificação e endereço de acesso das UDDIs para o primeiro experimento.

O próximo passo se restringiu à publicação e a descoberta de serviços. Inicialmente, a publicação envolveu a criação de 10 (dez) provedores hipotéticos identificados por: ubl1 até ubl10. Para isso, com base na API fornecida pela UDDI (ubl-uddi), um cliente Java foi criado e usado para registrar os provedores (para detalhes do cliente Java, ver Seção 5.2.6.3). A Figura 75 mostra a relação de provedores criados e disponíveis na federação, além de outros usados como teste.

Figura 75 - Relação de provedores criados e disponíveis na federação.

Após o registro dos provedores, foi realizada a efetiva publicação dos serviços. Nesta etapa, provedores foram escolhidos aleatoriamente,

sendo que para cada provedor eleito foram publicados 10 (dez) serviços relacionados ao processo *Ordering* UBL[18] (para detalhes do programa Java que implementa a publicação de serviços, ver Seção 5.2.6.3).

Processo UBL	Ontologia UBL	Nome do serviço
Ordering	*UBL/…/SellerParty/ PlaceOrder*	*PlaceOrder*
	UBL/… /BuyerParty/ ReceiveOrder	*ReceiveOrder*
	UBL/…/SellerParty/ ProcessOrder	*ProcessOrder*
	UBL/…/SellerParty/ AcceptOrder	*AcceptOrder*
	UBL/…/SellerParty/ RejectOrder	*RejectOrder*
	UBL/…/SellerParty/ AddDetail	*AddDetail*
	UBL/…/BuyerParty/ ReceiveResponse	*ReceiveResponse*
	UBL/…/BuyerParty/ AcceptOrder	*AcceptOrder*
	UBL/…/BuyerParty/ChangeOrder	*ChangeOrder*
	UBL/…/BuyerParty/CancelOrder	*CancelOrder*

Quadro 26 – Relação de serviços publicados relacionados ao processo *Ordering* UBL.

O Quadro 26 apresenta a lista de serviços UBL registrados. A coluna *Ontologia UBL* informa a classificação do serviço publicado dentro do processo *Ordering* UBL. A coluna *Nome do serviço* apresenta o nome do serviço publicado. O processo de registro de serviços levou em consideração a ontologia de processos UBL e a ontologia de QoS. Além disso, valores para QoS de cada serviço foram gerados de forma aleatória no intervalo [0,100].

Como forma de criar um conjunto de serviços inicial para o experimento, o processo de publicação foi repetido 20 (vinte) vezes para cada repositório (*ubl-uddi*, *ubl-uddi2*, *ubl-uddi3*), totalizando 600 serviços publicados nos repositórios da federação. O número de serviços publicados nos repositórios da federação é maior do que o número de elementos (o tamanho da amostra) que uma coleção deve possuir para ser usada como teste em experimentos de avaliação de sistemas de recuperação de informação (MANNING; RAGHAVAN; SCHÜTZE, 2008).

Uma ilustração do cenário descrito para o primeiro experimento é apresentada na Figura 76. Nela, têm-se os provedores de serviços (lado

[18] O processo *Ordering* UBL foi escolhido para ser usado no experimento pela familiaridade do autor com o mesmo e por ser o mesmo usado no exemplo conceitual apresentado no capítulo 4.

direito), o conjunto de repositórios presentes na federação de provedores (centro), cada um contendo 200 serviços *Ordering* UBL e o mecanismo de descoberta (lado esquerdo), que munido da lista de repositórios disponíveis na federação (obtida via *ubl-uddi*) executa a descoberta.

O cenário descrito para este primeiro experimento foi implantado em (1) um computador com processador Intel Centrino *Duo Core* de 1.66GHz, memória RAM de 2GB e Microsoft Windows XP *Professional Service Pack* 3.

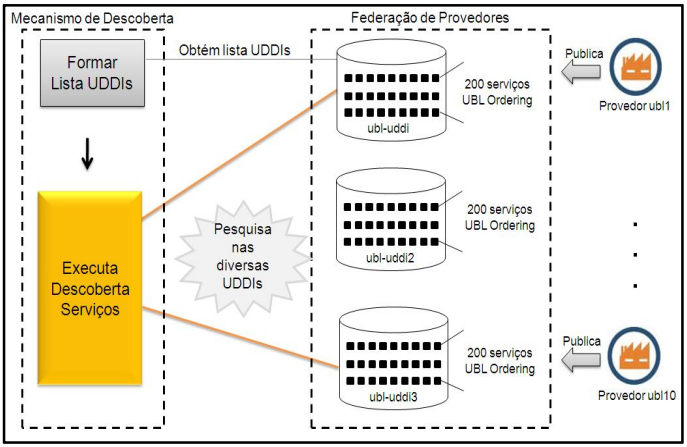


Figura 76 – Ilustração do cenário para o primeiro experimento.

6.1.2.1.3 Execução

Considerando o cenário descrito, o primeiro experimento usou como instância da expressão da descoberta um conjunto de valores conforme ilustra a Figura 77.

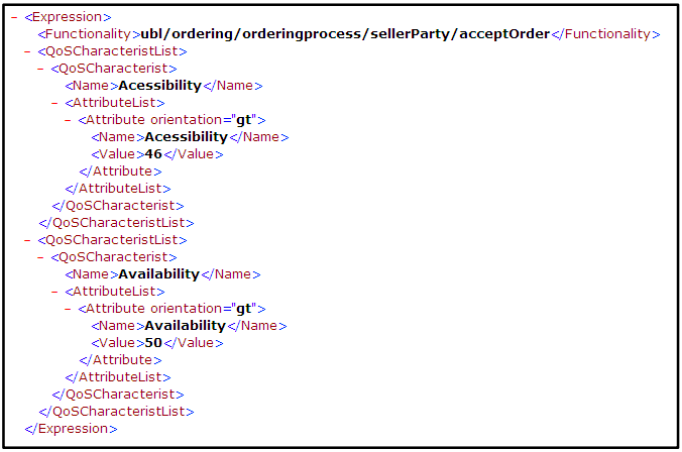


Figura 77 - Instância da expressão da descoberta para o primeiro experimento.

Nela, a marcação XML designada pelo elemento <*Functionality*> informa o nome do serviço requerido, isto é *ubl/ordering/orderingprocess/sellerParty/acceptOrder*. As marcações <*QoSCharacterist*> listam características de QoS desejadas. Para cada característica de QoS desejada um elemento <*attribute*> é registrado. Ele define o nome do atributo que compõe a respectiva característica, o valor e a orientação (como: maior que, menor que) usada na comparação entre valores de atributos pelo mecanismo de descoberta. No caso, tem-se a característica de QoS *Acessibility* composta pelos atributos *Acessibility* e *Availability* com valor igual a *46* e *50* respectivamente e orientação igual a *gt* que representa o designador maior que.

6.1.2.1.4 Resultados

Concluída a execução do primeiro experimento, um arquivo no formato XML foi gerado para documentar os resultados obtidos, conforme ilustra a Figura 78.

Figura 78 - Resultados registrados em um arquivo XML de resultados.

Na Figura 78, o elemento <*Date*> registra a data de execução do experimento. O elemento <*RunTime*> registra o tempo total da descoberta. O elemento <*Expression*> apresenta informações usadas no processo de descoberta. O elemento <*RetrievedServices*> mostra o total de serviços recuperados pelo mecanismo e o elemento <*Service*>, localizado dentro do elemento <*ServiceList*>, apresenta cada serviço recuperado que satisfaz a expressão da descoberta (ver Figura 77).

Considerando a expressão da descoberta (ilustrada na Figura 77), foram efetuados 10 (dez) ensaios com o mecanismo de descoberta. Os dados obtidos após a execução desses ensaios, usando 3 UDDIs, implantadas em 1 máquina, são apresentados no Quadro 27. Nele, (TSF) corresponde ao número total de serviços armazenados nos repositórios da federação por ensaio. O número de serviços recuperados por ensaio é representado por (NSR). O percentual de serviços recuperados em relação ao total de serviços armazenados na federação é representado por (PSRA). O percentual do aumento de serviços em cada teste

efetuado é representado por (PAST)[19]. O tempo total da descoberta e a variação percentual do tempo de descoberta em relação ao tempo da descoberta anterior são representados, respectivamente, por (T) e (VTD).

Ensaio	TSF (3 UDDIs locais)	NSR	PSRA (%)	PAST (%)	T (s)	VTD (%)
1ª	600	17	2,83	25,00	5	16,67
2ª	750	19	2,53	24,00	6	0,00
3ª	930	25	2,69	25,81	6	16,67
4ª	1170	29	2,48	25,64	7	0,00
5ª	1470	34	2,31	26,53	7	14,29
6ª	1860	41	2,20	25,81	8	37,50
7ª	2340	52	2,22	25,64	11	36,36
8ª	2940	70	2,38	25,51	15	13,33
9ª	3690	92	2,49	25,20	17	41,18
10ª	4620	109	2,36	25,00	24	-

Quadro 27 – Resultados obtidos para o primeiro experimento.

Em relação às métricas precisão e cobertura, o Gráfico 1 apresenta o comportamento da precisão média considerando os diversos níveis de cobertura [0,1] para o primeiro experimento.

[19] É importante frisar que o aumento no número de serviços é praticamente constante a cada teste efetuado, variando de 24% a 25% em relação à quantidade anterior de serviços na federação. A finalidade é fixar um incremento que é praticamente constante, aplicado ao número total de serviços, para permitir a comparação de resultados entre diferentes testes.

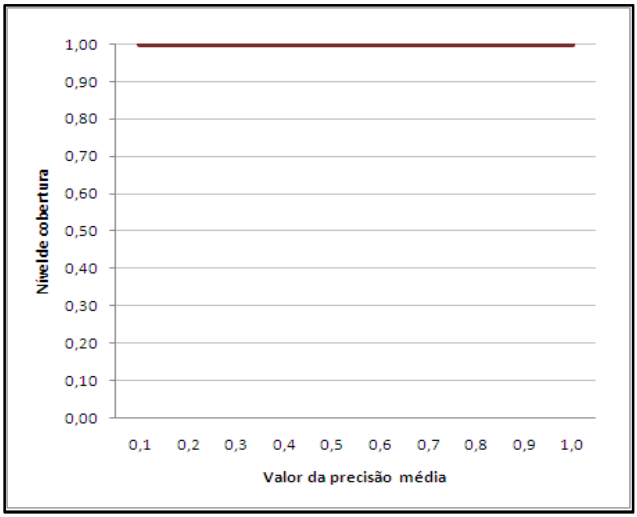


Gráfico 1 - Comportamento da precisão média considerando os diversos níveis de cobertura [0,1] para o primeiro experimento.

6.1.2.1.5 Análise Geral dos Resultados

No que tange à precisão e cobertura, percebe-se que o comportamento do mecanismo de descoberta apresenta precisão máxima em qualquer nível de cobertura (ver Gráfico 1, linha horizontal em vermelho escuro). Isto advém da forma com que a expressão da descoberta foi projetada e da maneira como o modelo proposto foi concebido, pois a ontologia UBL é usada para definir a semântica dos serviços e a funcionalidade dos serviços desejados. Como ela é usada pelos provedores quando publicam seus serviços, isso garante que somente serviços UBL sejam recuperados, desde que seus atributos de QoS sejam compatíveis com os atributos de QoS presentes na expressão da descoberta.

Além disso, após a realização dos dez ensaios iniciais, percebeu-se que cada módulo da arquitetura do protótipo computacional funciona corretamente de forma integrada.

6.1.2.2 Segundo Experimento

6.1.2.2.1 Objetivo

O bjetivo deste experimento foi de coletar e analisar valores para o tempo de resposta da descoberta quando ocorre aumento no número de serviços, aumento no número de UDDIs na federação e aumento na quantidade de máquinas usadas para armazenar as UDDIs. Em segundo plano, visou-se confirmar a precisão do mecanismo em qualquer nível de cobertura.

6.1.2.2.2 Cenário

O cenário de testes para o segundo experimento foi construído no laboratório de informática da Universidade do Planalto Catarinense (UNIPLAC), em Lages, Santa Catarina. Foram usadas 10 (dez) máquinas (cada uma delas com um processador *Pentium Dual Core* de 2.5GHz com 3Gbytes de memória principal e sistema operacional *Windows 7 Ultimate*). Nelas foram instaladas diversas UDDIs e vários serviços foram publicados (ver Quadro 28 para detalhes). Em outra máquina (com um processador Intel Centrino *Duo Core* de 1.66GHz, memória RAM de 2GB e Microsoft Windows XP *Professional Service Pack* 3.), localizada no mesmo laboratório, foi instalado o mecanismo de descoberta. Todos os computadores foram conectados via rede local e a mesma instância da expressão da descoberta do primeiro experimento foi utilizada neste experimento.

6.1.2.2.3 Execução

Neste experimento a federação foi composta conforme mostra o Quadro 28. A coluna *Ensaio* representa os ensaios efetuados com o mecanismo de descoberta. A coluna *Total de serviços publicados na Federação* informa o total de serviços disponíveis para serem descobertos. A coluna *Número de UDDIs* diz respeito à quantidade de UDDIs usadas para formar a federação. A coluna *Número de serviços publicados em cada UDDI* corresponde ao total de serviços armazenados em cada UDDI e a coluna *Número de máquinas* informa a quantidade de computadores usados em cada ensaio. É importante salientar que devido à quantidade de memória RAM usada por cada UDDI (cerca de 300Mbytes), foi possível instalar no máximo 10 (dez) UDDIs em cada máquina.

Ensaio	Total de serviços publicados na Federação	Número de UDDIs	Número de serviços publicados em cada UDDI	Número de máquinas
1º	15400	10	1540	1
2º	30800	20	1540	2
3º	46200	30	1540	3
4º	61600	40	1540	4
5º	77000	50	1540	5
6º	92400	60	1540	6
7º	107800	70	1540	7
8º	123200	80	1540	8
9º	138600	90	1540	9
10º	154000	100	1540	10

Quadro 28 - Configuração inicial para a federação.

6.1.2.2.4 Resultados

O Quadro 29 apresenta os resultados dos (10) dez ensaios efetuados: o total de serviços na federação (TSF), o Número de UDDIs usadas, o tempo da descoberta (T) em segundos, a variação do tempo de descoberta por ensaio (VTD), o número de serviços recuperados (NSR), o percentual de serviços recuperados em relação ao total de serviços (PSRA) e o número de máquinas usados em cada ensaio.

Ensaio	TSF	Número de UDDIs	T(s)	VTD (%)	NSR	PSRA (%)	Número de máquinas
1º	15400	10	34	17,65	347	2,253	1
2º	30800	20	40	15,00	629	2,042	2
3º	46200	30	46	8,70	1049	2,271	3
4º	61600	40	50	12,00	1383	2,245	4
5º	77000	50	56	10,71	1770	2,299	5
6º	92400	60	62	9,68	2128	2,303	6
7º	107800	70	68	7,35	2503	2,322	7
8º	123200	80	73	31,51	2840	2,305	8
9º	138600	90	96	22,92	3125	2,255	9
10º	154000	100	118	-	3449	2,240	10

Quadro 29 - Resultados obtidos para o segundo experimento.

Com base nos números registrados no Quadro 29 foi possível desenvolver um gráfico (ver Gráfico 2). O Gráfico 2 é composto por diversos pontos (*n*,*t*). Cada ponto indica que, para uma entrada de tamanho *n* (número de serviços na federação), o mecanismo de descoberta teve um tempo de execução de *t* (segundos).

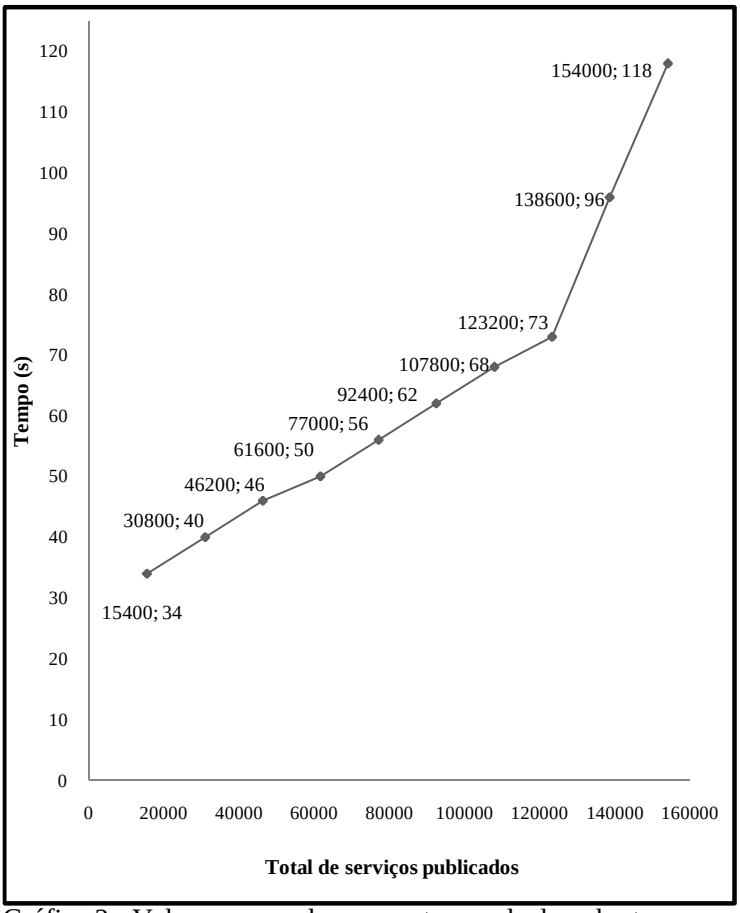


Gráfico 2 - Valores agregados para o tempo de descoberta *versus* quantidade de serviços publicados na federação.

6.1.2.2.5 Análise Geral dos Resultados

No que tange a cobertura e precisão, em todos os ensaios efetuados, confirmou-se o valor máximo (1) para precisão, em qualquer nível de cobertura, uma vez que o *matching* é sintático do nome do serviço. Em relação ao tempo de resposta do mecanismo de descoberta, observou-se que para uma pequena quantidade de serviços e para uma quantidade pequena de UDDIs, a variação do tempo de descoberta fica entre 8 e 18% (ver coluna VTD do Quadro 29, ensaios 1 a 7). Isto era esperado, pois quando se tem poucas entradas, tanto no número de serviços como de UDDIs, estes fatores devem afetar pouco o desempenho (em termos de tempo de resposta) do mecanismo de descoberta. Por outro lado, quando há um aumento no número de serviços, no número de UDDIs por máquina e na quantidade de máquinas a variação do tempo de descoberta tende a também se tornar maior (ver coluna VTD do Quadro 29, ensaios 9 e 10).

É importante frisar que os resultados dos experimentos obtidos em relação ao tempo de processamento podem variar. Em um cenário de testes com máquinas mais lentas, o tempo total para a descoberta sofrerá, ou seja, deve ser maior comparado ao desempenho registrado nos ensaios realizados. Isto porque o tempo de processamento também é afetado não só pelo hardware (processador, memória, disco, etc), mas também pelo software (sistema operacional) sobre o qual o mecanismo de descoberta é executado.

6.1.2.3 Terceiro Experimento

6.1.2.3.1 Objetivo

O terceiro experimento visou estabelecer um cenário de distribuição de repositórios para poder se aproximar do cenário idealizado para o funcionamento do modelo proposto.

6.1.2.3.2 Cenário

No experimento, o número total de UDDIs usadas e implantadas foi três. Uma delas implantada localmente em um (1) computador localizado na cidade de Lages/SC. Outra UDDI implantada no Laboratório de Redes e Sistemas Distribuídos da Universidade do

Planalto Catarinense, também em Lages/SC e a última UDDI implantada no servidor do Grupo de Pesquisa GSigma/DAS/UFSC localizado em Florianópolis/SC, todas conectadas e acessadas via Internet.

6.1.2.3.3 Execução

O estado inicial de cada um dos repositórios para este experimento é apresentado no Quadro 30, sendo que a mesma instância da expressão da descoberta dos experimentos 1 e 2 foi usada neste experimento.

Ensaio	Total de serviços na Federação com 3 UDDIs	Total de serviços em cada UDDI	Percentual de aumento no número de serviços
1º	600	200	25,00
2º	750	250	24,00
3º	930	310	25,81
4º	1170	390	25,64
5º	1470	490	26,53
6º	1860	620	25,81
7º	2340	780	25,64
8º	2940	980	25,51
9º	3690	1230	25,20
10º	4620	1540	-

Quadro 30 – Configuração inicial para a federação com 3 UDDIs (1 local e 2 distribuídas).

6.1.2.3.4 Resultados

O Quadro 31 resume os dados obtidos nos (10) dez ensaios para as 3 UDDIs. Ele apresenta o total de serviços em cada ensaio na federação (TSF), o tempo da descoberta (T) em segundos, o percentual de aumento de serviços em cada teste (PAST), a variação do tempo de descoberta por ensaio (VTD), o número de serviços recuperados (NSR) e o percentual de serviços recuperados em relação ao total de serviços (PSRA).

Ensaio	TSF com 3 UDDIs (1 local e 2 remotas)	T (s)	PAST (%)	VTD (%)	NSR	PSRA (%)
1º	600	9	25,00	11,11	11	1,83
2º	750	10	24,00	20,00	13	1,73
3º	930	12	25,81	16,67	18	1,94
4º	1170	14	25,64	14,29	23	1,97
5º	1470	16	26,53	37,50	30	2,04
6º	1860	22	25,81	18,18	41	2,20
7º	2340	26	25,64	23,08	55	2,35
8º	2940	32	25,51	25,00	72	2,45
9º	3690	40	25,20	27,50	85	2,30
10º	4620	51	25,00	-	105	2,27

Quadro 31 – Resumo dos dados obtidos nos (10) dez ensaios para 3 UDDIs (1 local e 2 distribuídas).

O Gráfico 3 apresenta de forma agregada os valores de tempo de descoberta *versus* serviços armazenados na federação, levando em conta os dados do primeiro experimento realizado e também considerando os dados colhidos neste experimento. Os dados do segundo experimento não foram considerados, já que foram usadas máquinas com espeficação diferente das máquinas usadas neste experimento.

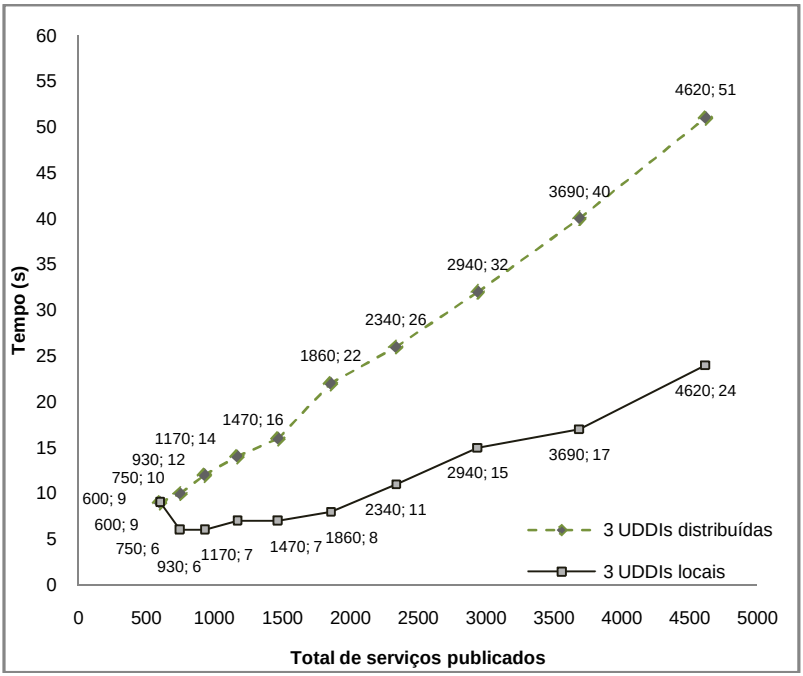


Gráfico 3 - Valores agregados de tempo de descoberta *versus* serviços na federação para primeiro e terceiro experimentos.

A relação completa com todos os pontos (serviços, tempo), considerando os experimentos 1 e 3 é apresentada no Quadro 32.

Ensaio	Experimento 1: Federação com 3 UDDIs locais		Experimento 3: Federação com 3 UDDIs (1 local e 2 remotas)	
	Serviços (Tot/R)	T (s)	Serviços (Tot/R)	T (s)
1º	600/17	9	600/11	9
2º	750/19	6	750/13	10
3º	930/25	6	930/18	12
4º	1170/29	7	1170/23	14
5º	1470/34	7	1470/30	16
6º	1860/41	8	1860/41	22
7º	2340/52	11	2340/55	26
8º	2940/70	15	2940/72	32
9º	3690/92	17	3690/85	40
10º	4620/109	24	4620/105	51

Quadro 32 – Síntese dos resultados obtidos com os experimentos (1 e 3).

No Quadro 32, a coluna *Ensaio* lista o conjunto de descobertas efetuadas. Para cada experimento realizado há uma sub-coluna *Serviços* dividida em: total de serviços na federação (Total) e quantidade de serviços recuperados em uma descoberta (R). Na outra sub-coluna *Tempo*, mostram-se valores obtidos para o tempo gasto por uma descoberta (em segundos).

6.1.2.3.5 Análise Geral dos Resultados

Em relação às métricas de precisão e cobertura, novamente elas se confirmam, pois o valor obtido para a precisão em todos os níveis de cobertura sempre foi igual a um (1), valor máximo. No que tange ao funcionamento integrado dos módulos da arquitetura do modelo, percebeu-se que eles apresentaram um comportamento correto durante os testes realizados.

É importante frisar que os experimentos efetuados não envolveram a coleta do tempo de execução do algoritmo da descoberta dinâmica (invocado quando aplicações SOA estão em execução), porque o *crawling* caracteriza o pior caso. Ou seja, o *crawling* sempre busca e seleciona serviços, enquanto que o algoritmo da descoberta dinâmica, inicialmente, varre a lista de serviços candidatos (já obtida via *crawling*) e, caso não encontre, faz a descoberta efetiva na federação por 1 (um) novo serviço.

A análise dos valores para os experimentos 1 e 3 (apresentados no Quadro 32) mostra que o aumento na quantidade de UDDIs locais e na quantidade de serviços na federação ocasiona um aumento natural no tempo da descoberta. Por outro lado, o maior impacto no tempo é quando UDDIs remotas são usadas (ver Quadro 32, 4º Ensaio). Neste cenário, o tempo é (praticamente) 2 vezes maior que o tempo obtido com três UDDIs locais, fato causado pela rede que conecta as UDDIs. Além disso, é importante comentar que os resultados apresentados no Quadro 32 (em relação aos serviços recuperados, ver sub-coluna Serviços) são dependentes da adoção de padrões para processos de negócios e da relação 1:1 (segundo pressuposto usado na concepção do modelo proposto, Seção 4.3). Observa-se, também, uma maior precisão (qualidade) na descoberta, pois a lista de serviços candidatos retornadas pelo *crawling* é composta somente de serviços alinhados aos requisitos

desejados. O uso de QoS minimiza o número de serviços retornados enquanto o seu uso é maximizado, considerando o processo corrente. A utilização da ontologia UBL resolve problemas semânticos e reduz problemas relacionados à interoperabilidade. Isso também se estende a QoS, já que clientes e provedores usam um vocabulário comum quando executam a descoberta ou a publicação de serviços.

6.1.3 Análise da Complexidade do Algoritmo da Descoberta

Um dos resultados esperados desta tese é integração BPM&SOA mais ágil, transparente e eficiente. Tendo em vista que o mecanismo de descoberta é um dos elementos fundamentais para obter o resultado descrito acima, principalmente no viés da eficiência, é que se justifica a análise do mecanismo de descoberta sob o ponto de vista das suas operações computacionais.

Esta análise é essencial para responder o quanto o mecanismo de descoberta proposto é eficiente e em quais condições, ou seja, com quais quantidades de dados isto se verifica.

6.1.3.1 Determinação da Complexidade Computacional do Algoritmo da Descoberta

Esta seção determina a complexidade computacional para o algoritmo do *crawling*. Ele representa o caso em que sempre a descoberta ocorrerá de forma efetiva, ao contrário do algoritmo da descoberta dinâmica, que pode ou não fazer esta busca exaustiva.

Assim, ao algoritmo do *crawling,* (apresentado na Seção 5.2.4) substituíram-se a numeração das linhas por número sequenciais postos no início de cada linha de código para identificar os comandos que determinam o custo computacional do algoritmo para a descoberta de serviços.

A Figura 79 apresenta o algoritmo da descoberta já com os comandos alvos da análise de complexidade. O algoritmo inicia com um comando de atribuição, comando (1) na Figura 79), com custo computacional constante e igual a 1, portanto $O(1)$.

Na linha sinalizada pelo comando (2), um laço é inaugurado. Ele dispara uma série de *threads* conforme a quantidade de UDDIs presentes na federação, comando (3). Seu custo computacional é igual ao total de UDDIs (n) somado uma unidade em razão do teste final do laço, ou seja, $O(n+1)$. Antes de calcular o custo computacional associado ao comando (4), como são iniciadas várias *threads*, é necessário calcular o custo

computacional do código executado por cada uma delas, comandos (5) a (10).

```
Algoritmo Crawling
 Entradas:
        Lista UDDIs;
        Serviço desejado (a partir da ontologia UBL);
        Conjunto de características de QoS desejadas (a partir da ontologia QoS);
 Saída: lista de serviços candidatos.
Início
(1)  thread.cria <- listaUDDI.get()
(2)  Para as (n) UDDIs disponíveis na Federação faça
(3)      thread.dispara
     Fimpara
(4)  MostraListaServiçosCandidatos()
//
//   Início código thread
(5)  Para os (m) serviços (s) armazenados em cada UDDI faça
(6)     Se serviço desejado (sd) faz matchingUBL com serviço (s) armazenado UDDI então
(7)        Enquanto valorQoS(sd) atendido por (s) e tem atributos analisar (sd) faça
(8)           valorAtributoQoS <- listaQoS.obtemAtributo(sd);
           Fimenquanto
(9)        Se todos atributos (sd) foram atendidos por (s) então
(10)          listaServiçosCandidatos.adiciona(s)
           Fimse
        Fimse
     Fimpara
//
//   Fim código thread
(11) Retorne (listaServiçosCandidatos)
Fim
```

Figura 79 – Algoritmo do *crawling* com marcações para determinar o custo computacional.

Ziviani (2004) comenta que, em alguns casos, a análise da complexidade de algoritmos deve partir dos comandos internos para os comandos mais externos. Esta forma de análise possibilita focar os anéis internos, abstraindo os externos, facilitando a obtenção da ordem de complexidade do algoritmo.

Neste caso, optou-se por iniciar a análise a partir dos comandos internos ao comando (6), ou seja, os comandos de (7) a (10).

O comando (7) sinaliza um laço de repetição. Dentro dele, está o comando (8), responsável por retirar um elemento da lista de QoS e atribuir o valor a uma variável que será testada no laço identificado pelo comando (7). A retirada de um elemento de uma lista, bem como a atribuição de valores leva um tempo constante para ser executado, igual a $O(1)$.

Como o número de iterações para executar o comando (7) corresponde ao total de características e atributos de QoS a serem verificados (23), somado a um último teste para finalizar o anel, tem-se: $O(24x1) = O(24)$.

Os comandos (6), (9) e (10) possuem tempo computacional constante, uma vez que realizam testes e atribuições. Em síntese, o valor agregado destes comandos é $O(\max(1,1,1)) = O(1)$.

Logo, o valor computacional gasto nos comandos (7) a (10) é $O(\max(24,1)) = O(24)$.

Considerando que o comando (6) realiza um teste, portanto, gasta $O(1)$. Nele, tem-se: $O(\max(24,1)) = O(24)$, como tempo total gasto para a execução do comando (6) e comandos aninhados a ele.

O comando (5) representa um anel cuja complexidade é igual ao total de repetições que o anel faz, mais uma unidade para testar o final do laço. Logo, tem-se: $O((m+1)x24)$, pois o comando (6) será executado conforme o número de repetições prevista no comando (5). Assim, o valor agregado para o comando (5) é $O(24m+24)$, que pode ser representado por $O(m)$, uma vez que 24 é um valor constante dentro do algoritmo e pode ser ignorado na análise, já que seu valor permanece inalterado durante a execução do algoritmo.

O anel, identificado pelo comando (2), possui tempo computacional igual a $O(n+1)$. Logo, tem-se como valor agregado para os comandos (2) e (5): $O((n+1) \times (m)) = O(n \times m + m)$. Este valor pode ser representado, no pior caso, como $O(n \times m)$.

Por fim, soma-se ao tempo gasto com os comandos (2) e (5), o valor $O(1)$ para o comando (4), obtendo-se: $O(\max((nxm),1) = O(nxm)$.

Como n e m representam valores de entradas do algoritmo, o tempo computacional final para a descoberta, considerando o pior caso, é de ordem quadrática, igual a $O(n^2)$.

Algoritmos desta ordem de complexidade ocorrem quando suas entradas são processadas aos pares (UDDIs e serviços), em anéis aninhados.

6.1.3.1.1 Comentários sobre a Determinação da Complexidade Computacional do Algoritmo da Descoberta

Algoritmos cuja ordem de complexidade é quadrática servem para resolver problemas com entradas pequenas (ZIVIANI, 2004). Assim, em razão do algoritmo de descoberta ser um elemento fundamental no modelo de descoberta proposto, é importante ressaltar a limitação que o mesmo possui para entradas grandes, isto é diversos

serviços publicados e várias UDDIs espalhadas remotamente. Isto pode restringir seu uso na prática quando há uma quantidade muito grande de repositórios remotos com inúmeros serviços publicados, acessados via Internet.

6.2 **Validação**

A literatura apresenta vários métodos que podem ser utilizados para a avaliação e validação de trabalhos relacionados com tecnologia (ZELKOWITZ, 2008). Nesse trabalho utilizou-se o método de avaliação *expert panel*, que segundo Zelkowitz (2008), utiliza a avaliação baseada no consenso de especialistas. Algumas de suas características são:

- Contexto é controlado – o ambiente onde a avaliação é efetuada é controlado. Os procedimentos utilizados para se apresentar o objeto de avaliação são, portanto, previamente estabelecidos;
- Os dados são coletados a partir dos especialistas – a fonte de dados da pesquisa é única e exclusivamente a dos especialistas;
- Aplicação no contexto real – deve-se efetuar a avaliação num ambiente que reproduza as condições de uso reais do objeto da avaliação.

A escolha da abordagem de avaliação baseada na opinião de especialistas foi motivada pela natureza qualitativa do trabalho e da necessidade de ter a opinião de pessoas envolvidas com a integração de processos de negócio junto à tecnologia de informação. Estas pessoas são, portanto, aptas a julgar a abordagem proposta neste trabalho. Assim, o uso do *expert panel* permite fazer uma avaliação através da opinião consensual dos especialistas.

O instrumento utilizado na avaliação foi um questionário composto de um conjunto ordenado de perguntas que devem ser respondidas pelos entrevistados (MENEZES; SILVA, 2005). O questionário deve ser objetivo, limitado em extensão e estar acompanhado de instruções. Estas devem esclarecer o propósito de sua aplicação e ressaltar a importância da colaboração do entrevistado (MENEZES; SILVA, 2005). O questionário é uma das formas mais rápidas e eficientes de coletar dados. As perguntas que o compõe

geralmente estão atreladas aos objetivos específicos do trabalho (GIL, 2010). Nesse sentido, o questionário provê uma forma eficiente de alinhar os objetivos do trabalho junto às perguntas que o compõem, dessa forma facilitando a avaliação do trabalho.

As respostas possíveis a cada pergunta do questionário foram formuladas de acordo com a escala de *Likert* (LIKERT, 1932). Essa escala se caracteriza por definir diferentes graus de intensidade para as possíveis alternativas de resposta associadas a cada pergunta (ALEXANDRE *et al.*, 2003).

A escala de *Likert* é largamente utilizada em pesquisas de opinião e permite extrair diferentes níveis de concordância para cada uma das afirmativas. Esses níveis permitem auferir em qual medida o entrevistado concorda com a afirmativa apresentada. Visto que as perguntas do questionário estão alinhadas aos objetivos do trabalho, o uso dessa escala permite quantificar a concordância dos entrevistados com os objetivos. Por esse motivo, optou-se por utilizar essa escala. As seguintes alternativas foram estabelecidas para cada pergunta:

- Concordo Fortemente;
- Concordo;
- Indiferente;
- Discordo;
- Discordo Fortemente.

O uso de cinco alternativas tem o objetivo de balanceá-las. Se por exemplo não existisse a alternativa “Indiferente”, o entrevistado poderia marcar a alternativa para o lado em que está mais “inclinado”, mesmo não tendo certeza de sua resposta (ALEXANDRE *et al.*, 2003). Portanto, é importante sempre que haja um ponto de equilíbrio no questionário, caso contrário pode-se provocar uma avaliação enviesada (GÜNTHER, 1999).

6.2.1 Aplicação do Questionário

Um questionário foi aplicado a um grupo de 9 (nove) especialistas, com conhecimento da área de gerenciamento de processos de negócio e de serviços *web*. Desses nove, três foram pessoas de empresas (duas do setor privado e uma do setor público). As seis pessoas restantes são do meio acadêmico. O objetivo da aplicação do questionário é usar o seu resultado como uma evidência que comprova que os objetivos gerais e específicos do trabalho foram atingidos e a

hipótese do trabalho foi confirmada. O questionário é mostrado no Quadro 33.

Pergunta 1	Na sua opinião o problema da limitada agilidade na integração entre BPM e SOA é relevante de ser melhorado ou resolvido?
Pergunta 2	Você considera que o modelo do catálogo de processos de negócio proposto tem potencial para agilizar (no sentido de tornar a integração mais transparente, interoperável e eficiente) a concepção de aplicações BPM & SOA, resolvendo ou amenizando os problemas de (cada item é respondido separadamente): a. falta de semântica associada entre os processos de negócio e os serviços web; b. falta de sinergia entre os envolvidos na modelagem dos processos de negócio e os envolvidos na implementação dos serviços; c. falta de padrões de processos de negócio.
Pergunta 3	Você concorda que a interface que o usuário-projetista interage no editor BPM, e que possibilita a vinculação de serviços e critérios QoS aos processos de negócio, permite a ele interagir facilmente com o catálogo?
Pergunta 4	Você concorda que a ontologia desenvolvida nesse trabalho consegue organizar os processos da especificação UBL de forma a transparentemente permitir a vinculação destes com os serviços *web* correspondentes?
Pergunta 5	Você concorda que o mecanismo de exportação dos processos permite fazer a conversão da linguagem BPMN para WS-BPEL, simplificando a posterior execução no ambiente de execução de processos?
Pergunta 6	Você concorda que o Ambiente de Execução de Processos permite ao usuário executar e acompanhar os processos em WS-BPEL?
Pergunta 7	Você acredita que no futuro as empresas passarão a adotar largamente padrões de processos de negócio (tais como UBL e RosettaNet) nas suas transações internas e com outras empresas?

Quadro 33 - Perguntas do questionário.

Para cada entrevistado, uma breve apresentação do trabalho desenvolvido foi feita e em seguida apresentou-se o protótipo

computacional. Após, cada entrevistado utilizou o sistema, ficando o autor deste trabalho ao lado de cada entrevistado para sanar eventuais dúvidas. Após a utilização do sistema, cada entrevistado respondeu o questionário mostrado no Quadro 33. A seção segiunte apresenta o resultado da aplicação deste questionário, juntamente com a análise dos resultados.

6.2.2 Análise dos Resultados

A etapa posterior à aplicação do questionário é a coleta dos dados. Esta deve estar relacionada com o problema e a hipótese do trabalho. O objetivo da coleta é obter elementos que comprovem que os objetivos propostos pelo trabalho foram alcançados (MENEZES; SILVA, 2005).

É importante frisar que devido à dificuldade de implantar os diversos componentes de software do modelo em várias empresas (pois muitos deles ainda estão em nível de protótipo computacional), da necessidade da criação de uma federação de repositórios de serviços e da necessidade do conhecimento do padrão UBL e das ontologias pelos usuários do modelo proposto, um número maior de empresas não puderam avaliar o modelo desenvolvido. Assim, tendo que vista que a pesquisa é exploratória e o cenário previsto é “futurístico”, os noves especialistas pesquisados dão indícios da importância e da relevância do modelo proposto na integração BPM&SOA.

O resultado do questionário aplicado é apresentado e, em seguida, apresenta-se a vinculação deste com o problema, a hipótese e os objetivos da tese.

1ª pergunta - Na sua opinião, o problema da limitada agilidade na integração entre BPM e SOA é relevante de ser melhorado ou resolvido?

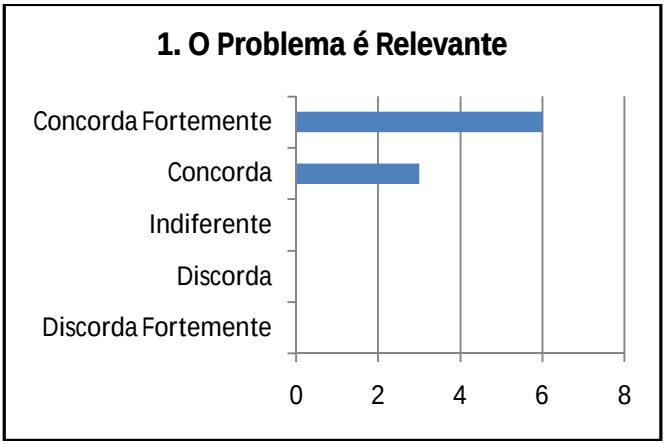


Gráfico 4 - Relevância do problema.

Com um nível de concordância variando de concorda fortemente a concorda, os participantes consideraram que o problema exposto pelo trabalho, isto é, que a limitada agilidade na integração entre BPM e SOA, é relevante de ser melhorado ou resolvido. Esse resultado é positivo no escopo dessa tese, visto que fornece uma fundamentação consistente no que diz respeito à justificativa do problema tratado.

O Quadro 34 lista algumas impressões dos avaliadores.

Algumas impressões dos avaliadores
"Sim, porque quando as duas iniciativas são aplicadas juntas, elas tornam-se sinérgicas. Ambas fornecem uma estratégia para tratar os vários desafios impostos pelas mudanças constantes dos requisitos relacionados às aplicações que implementam processos de negócio. O principal ganho, na minha opinião, desta cooperação, está em permitir a redução de custos, o aumento da eficiência e uma grande flexibilidade no desenvolvimento ou manutenção de aplicações"
"Concordo integralmente que a falta de integração entre a modelagem de processos e sua implementação existe, e que para a solução deste problema é necessário o desenvolvimento de metodologias, que mostrem como essa integração pode ser alcançada, e ferramentas, que façam essa integração o mais transparente possível para o usuário final. Entretanto, essa conversão de modelos em implementações deve estar baseada na Engenharia Guiada por Modelos (MDE), pois assim podem ser garantidos os princípios fundamentais da tradução de modelos em aplicações computacionais"

“Certamente é um gargalo e precisa ser solucionado, agilizando o processo da modelagem do negócio até a sua real utilização na prática e em nível de implementação. A automatização/facilitação disso só vem trazer benefícios aos envolvidos”

Quadro 34 - Impressões dos avaliadores – pergunta 1.

2ª pergunta - Você considera que o modelo do catálogo de processos de negócio proposto tem potencial para agilizar (no sentido de tornar a integração mais transparente, interoperável e eficiente) a concepção de aplicações BPM & SOA, resolvendo ou amenizando os problemas de:

a. falta de semântica associada entre os processos de negócio e os serviços *web*;
b. falta de sinergia entre os envolvidos na modelagem dos processos de negócio e os envolvidos na implementação dos serviços;
c. falta de padrões de processos de negócio.

Cada um dos problemas apresentados acima foi respondido individualmente pelos entrevistados. O resultado é apresentado, respectivamente nos Gráficos 5, 6 e 7.

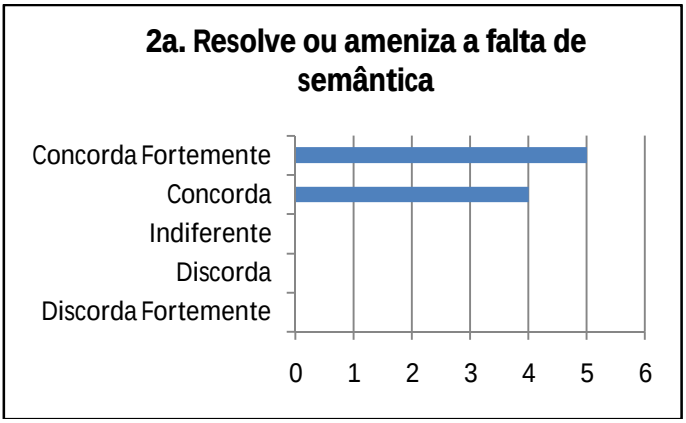


Gráfico 5 - Gráfico sobre a resolução ou amenização da falta de semântica.

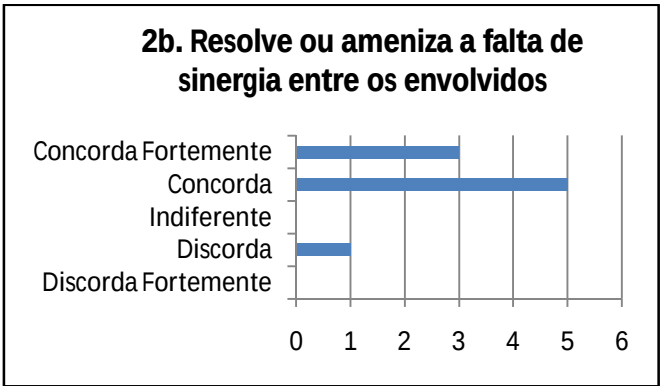


Gráfico 6 - Gráfico sobre a resolução ou amenização da falta de sinergia entre os envolvidos.

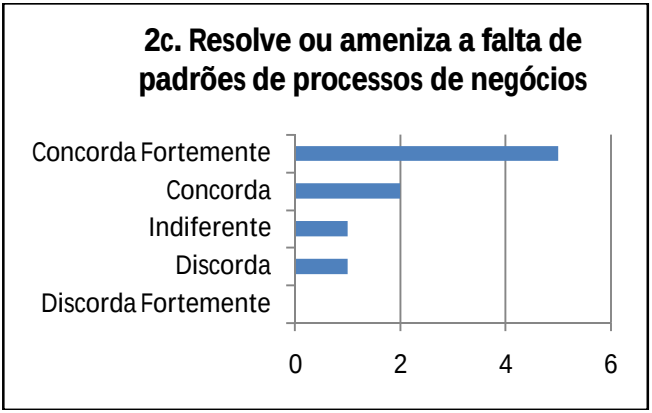


Gráfico 7 - Gráfico sobre a resolução ou amenização da falta de padrões de processos de negócio.

No que diz respeito à questão da falta de semântica associada aos processos de negócio e aos serviços *web*, os entrevistados concordam em maior ou menor grau que a abordagem proposta consegue resolver ou amenizar esse problema. Isso é possível, pois o modelo proposto possui uma ontologia que permite classificar de maneira suficientemente precisa os processos de negócio e os serviços *web* que podem ser vinculados a eles.

No que tange o problema da falta de sinergia entre os envolvidos na modelagem dos processos de negócio e os envolvidos na implementação destes a nível de software, a maioria dos entrevistados concorda em maior ou menor grau que a abordagem proposta ajuda a minimizar o problema. A única opinião contrária foi expressa por um entrevistado de uma empresa privada. Segundo ele, a falta de sinergia é mais de ordem organizacional do que tecnológica. Nesse sentido, considera-se relevante essa opinião, visto que o enfoque do problema da falta de sinergia neste trabalho é realmente no nível tecnológico. Por outro lado, discorda-se da opinião deste entrevistado. Primeiro porque várias referências bibliográficas (muitas delas citadas ao longo dos Capítulos 1, 2 e 3) ressaltam o problema tecnológico entre os níveis BPM e SOA. E segundo, porque o modelo proposto permite que a modelagem do processo de negócio e a sua posterior implementação, em SOA, sejam realizadas, tornando desnecessário que equipe de TI interprete o processo descrito em alto nível (em BPMN) para que ele seja implementando computacionalmente. Portanto, evita-se essa etapa que geralmente ocasiona discordâncias entre essas duas equipes, visto que a abordagem proposta permite fazer a implementação do processo de forma automática. Evidentemente que isso não implica dizer que os impactos organizacionais sejam pequenos.

No último item da questão, que trata sobre o problema da falta de especificações de processos de negócio, a maioria dos entrevistados concorda em maior ou menor grau que a abordagem proposta consegue resolver ou amenizar tal problema. Um único entrevistado (de uma empresa privada) é indiferente nessa questão. Ele acredita que no caso de processos considerados estratégicos pela empresa, isto é, os que representam um diferencial competitivo em relação aos concorrentes, não se aplica uma especificação padronizada. Portanto não há o interesse de se compartilhar tais processos. Por outro lado, o entrevistado concorda que no caso de processos que não garantem vantagens competitivas, por exemplo: processos de compras ou finanças, a abordagem proposta consegue minimizar o problema de falta de padrões de processos de negócio.

Outro entrevistado (de uma empresa pública) discorda dessa questão, argumentando que a falta de padrões só será resolvida se houver a mobilização dos diversos setores empresarias, no sentido de criar estes padrões. Contrapõe-se essa afirmação justificando que já existem especificações padronizadas de processos de negócio, como por exemplo a UBL e a RosettaNet, sendo que o trabalho proposto utiliza essa primeira. Concorda-se que nesse atual momento, essas

especificações ainda não conseguem contemplar todos os processos de uma organização, porém elas caminham nesta direção. E isso acontece com a mobilização dos diversos setores empresarias, o que de fato está ocorrendo.

O Quadro 35 lista algumas impressões dos avaliadores.

Algumas impressões dos avaliadores
"A maior contribuição da criação do catálogo é resolver a questão da falta de padronização na especificação de processos (semântica), sendo que esse é um aspecto fundamental para a utilização de serviços na implementação de processos de negócio modelados em BPM, por exemplo. No que diz respeito à sinergia entre projetistas de processos e implementadores de serviços, o maior benefício está na utilização de nomenclaturas padronizadas de processos, atividades, e principalmente das entradas e saídas de cada atividade contida em um processo, fornecida pela UBL. Isso não interfere diretamente na sinergia da equipe, mas diminui consideravelmente os problemas de comunicação. Sobre a falta de padronização, de fato, qualquer iniciativa que venha a propor maior padronização nos processos de negócio comumente realizados pelas empresas merece ser incentivada. Isso não traz benefícios apenas à área de TI, mas sim a todas as áreas das empresas que estão diariamente tendo que lidar com fornecedores e clientes, e outros parceiros comerciais, que implementam processos de negócio dos mais diversos tipos".
"Aparentemente o item "a" teria maior potencial de agregar valor a setores/negócios nos quais os processos mapeados não exijam uma interação humana muito intensa (ex: processos da indústria automobilística e similares)".
"Na minha opinião, o catálogo é um grande aliado para o sucesso da integração BPM&SOA, pois fornece especificações prontas e já testadas passíveis de serem usadas, seja no desenvolvimento de novas aplicações ou na melhoria de algumas já prontas. Por outro lado, também fornece informações que permitem definir o significado dos elementos usados durante o desenvolvimento de uma aplicação. Isto, de imediato, evita problemas relacionados à semântica e possibilita que a integração ocorra de forma mais transparente".

Quadro 35 - Impressões dos avaliadores – pergunta 2.

3ª pergunta - Você concorda que a *interface* que o usuário projetista interage no editor BPM, e que possibilita a vinculação de serviços e critérios QoS aos processos de negócio, permite a ele interagir facilmente com o catálogo?

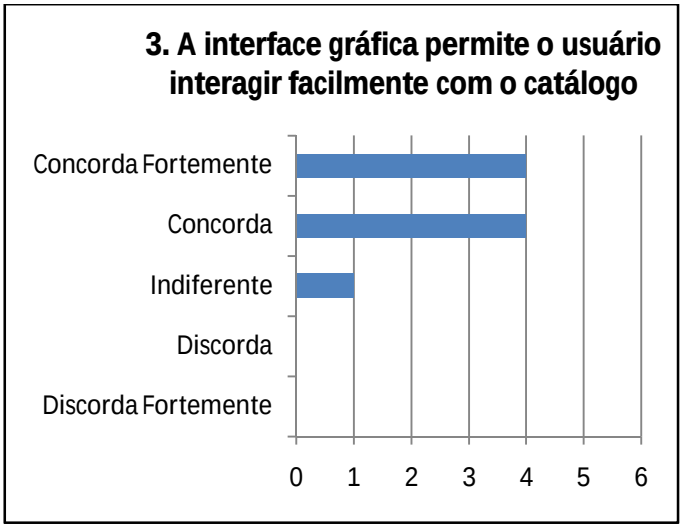


Gráfico 8 - Gráfico sobre a interface gráfica permitir o usuário interagir facilmente com o catálogo.

No que diz respeito ao fato de a interface gráfica que o usuário projetista interage permitir a ele utilizar as funcionalidades do catálogo de uma maneira fácil e intuitiva, a maioria dos entrevistados concorda em maior ou menor grau. Um entrevistado (de uma empresa privada) se mostrou indiferente nesse quesito, visto que ele sugere que o termo "facilmente" seja mensurado explicitamente, pois o que pode ser fácil para um, pode não ser para outro. Nesse sentido, a intenção da questão era de avaliar se o usuário conseguiria interagir com o catálogo de processos de negócio utilizando a interface gráfica sem ter dificuldades. Como o questionário foi aplicado para nove pessoas, e boa parte delas concordou que a interface gráfica permite interagir facilmente com o catálogo, considera-se, de fato, que tal quesito foi cumprido satisfatoriamente.

O Quadro 36 lista algumas impressões dos avaliadores.

Algumas impressões dos avaliadores
“Sim, o conjunto das funcionalidades oferecidas pela interface segue um padrão visual e possui boa navegação (com opções de avanço e retorno etc.). Em síntese, é bastante intuitiva”.
“Acreditamos que sim, desde que o usuário receba treinamento para isso”.
“A forma de implementação da solução para o problema de integração entre BPM e SOA facilita substancialmente o trabalho do projetista da aplicação SOA, pois a ferramenta apresenta de forma integrada, tanto a parte de busca por serviços que implementem as atividades contidas na modelagem do processo, quanto à transformação desse modelo em linguagem de execução de processos, já com os respectivos serviços associados”.

Quadro 36 - Impressões dos avaliadores - pergunta 3.

4ª pergunta - Você concorda que a ontologia desenvolvida nesse trabalho consegue organizar os processos da especificação UBL de forma a transparentemente permitir a vinculação destes com os serviços *web* correspondentes?

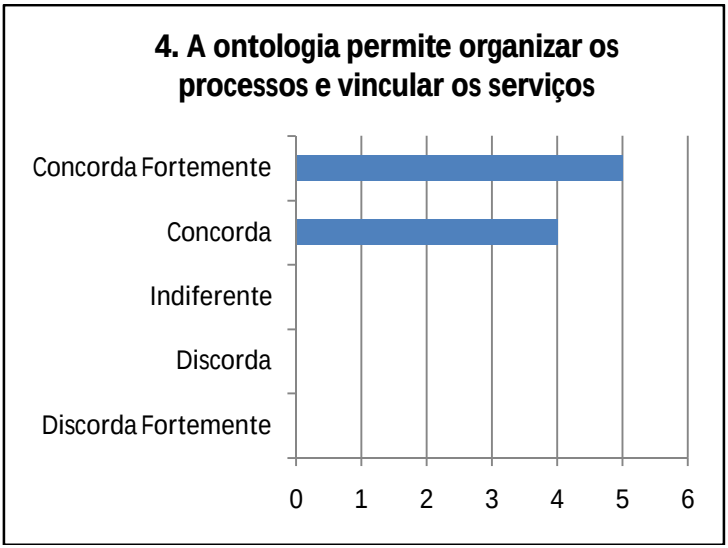


Gráfico 9 - Gráfico sobre a ontologia permitir organizar os processos e vincular os serviços.

No que se refere à ontologia UBL, os entrevistados responderam com unanimidade, em maior ou menor grau, que ela permite organizar os processos de negócio de forma transparente, permitindo que os serviços *web* relacionados sejam vinculados aos processos. A ontologia UBL desenvolvida provê uma taxonomia que é precisa o bastante para fazer essa classificação e permitir a vinculação. Os serviços *web*, providos por diferentes empresas (os provedores de serviços) seguem estritamente essa classificação, que padroniza aspectos como operações, entradas e saídas de dados dos serviços. Portanto, esses serviços são funcionalmente equivalentes, de forma que qualquer um deles é um candidato a executar a atividade do processo que possui a mesma classificação ontológica dele. Por outro lado, estes serviços diferem entre si somente em aspectos não funcionais, por exemplo, aqueles descritos pelos critérios de qualidade de serviço (QoS), localização geográfica, tecnologia de implementação, etc.

O Quadro 37 lista algumas impressões dos avaliadores.

Algumas impressões dos avaliadores
"Concordo fortemente, porém, ressalto que esta especificação da ontologia deve realmente ser "realista" para que os serviços na "nuvem" realmente sejam encontrados".
"Sim, a ontologia UBL retrata de forma fiel os processos UBL, seus atores, casos de uso, documento trocados etc. Sua fácil visualização, disponibilizada através da interface da ferramenta, permite uma rápida vinculação aos serviços".
"A ontologia desenvolvida para categorizar os serviços web é fundamental para que o processo de vinculação dinâmica aconteça. A forma como ela foi projetada evidencia que a mesma consegue abranger todos os elementos necessários para classificar os serviços de forma que os mesmos sejam recuperados adequadamente no momento da busca. Entretanto, há que se fazer uma ressalva sobre a nomenclatura utilizada na denominação do produto dessa classificação, pois da forma como a ontologia foi projetada, a mesma se enquadra mais adequadamente como uma taxonomia, ao invés de uma ontologia propriamente dita".
"Creio que esse seja o ponto forte da pesquisa".

Quadro 37 - Impressões dos avaliadores - pergunta 4.

5ª pergunta - Você concorda que o mecanismo de exportação dos processos permite fazer a conversão da linguagem BPMN para WS-BPEL, simplificando a posterior execução no ambiente de execução de processos?

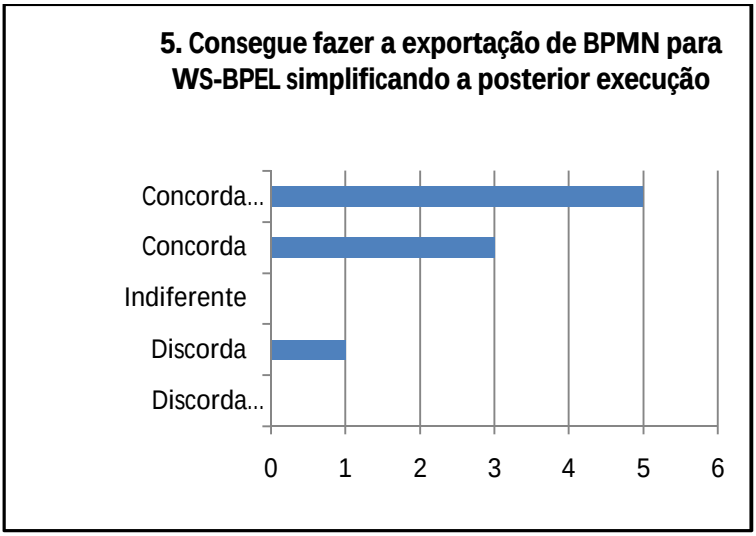


Gráfico 10 - Gráfico sobre o exportador conseguir fazer a exportação BPMN para WS-BPEL simplificando a execução do processo.

O resultado do questionário, no que diz respeito ao mecanismo de exportação de BPMN para WS-BPEL, mostra que a maioria dos entrevistados concorda em maior ou menor grau que de fato o mecanismo permite fazer essa conversão, viabilizando a posterior execução do processo pelo Ambiente de Execução de Processos. Um único entrevistado (de uma empresa privada) discordou dessa pergunta, usando como justificativa o fato de que as ferramentas que permitem fazer a conversão de BPMN para WS-BPEL serem proprietárias, dessa forma fazendo parte da propriedade intelectual das empresas que as desenvolveram. É importante salientar que conversor desenvolvido, embora limitado a alguns elementos presentes nos processos *Ordering* e *Payment* UBL, é baseado em padrões abertos (como BPMN e WS-BPEL) e é disponibilizado gratuitamente. Assim, espera-se que ele contribua para mudar um pouco esse cenário de soluções proprietárias. Além do mais, existem trabalhos acadêmicos (WHITE, 2005) e (OUYANG *et al.*, 2006) que propõem métodos para se efetuar essa conversão. Portanto, esse conhecimento de conversão não está restrito somente a empresas, estando disponível também à comunidade científica.

O Quadro 38 lista algumas impressões dos avaliadores.

Algumas impressões dos avaliadores
"Dado que a especificação do processo de negócio é feita utilizando uma notação padronizada para a especificação de processos, o BPMN, e que os serviços que implementam as atividades contidas no processo modelado já foram localizados e vinculados, o mecanismo de exportação tem os elementos necessários para a especificação no processo em linguagem de execução. Sabendo que o mecanismo de exportação trata de todas as especificidades necessárias para a conversão do processo em linguagem de execução, o mesmo pode ser considerado adequado para a conversão do processo em notação BPM para WS-BPEL, diminuindo substancialmente a complexidade de se fazer esse processo manualmente".
"Sim, principalmente em razão da estratégia usada na conversão e exportação. O processo se deu a partir dos processos mais complicados, aqueles contendo o maior número de elementos BPMN complexos. A partir deles, iniciou-se o processo de conversão e exportação dos demais processos".

Quadro 38 - Impressões dos avaliadores - pergunta 5.

6ª pergunta - Você concorda que o Ambiente de Execução permite ao usuário executar e acompanhar os processos em WS-BPEL?

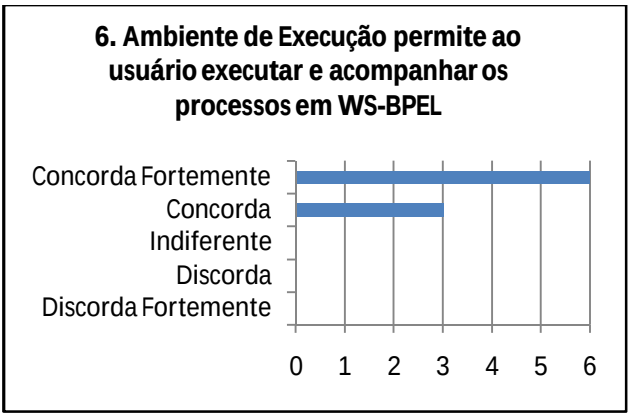


Gráfico 11 - Sobre o ambiente de execução de processos, ele permite executar e acompanha os processos WS-BPEL.

No que diz respeito ao fato de o Ambiente de Execução de Processos permitir ao usuário executar e acompanhar os processos em

WS-BPEL, todos os entrevistados concordaram em maior ou menor grau que isso de fato acontece. O resultado vai ao encontro da escolha feita, de se utilizar uma ferramenta de mercado como ambiente de execução de processos, no caso o *Intalio BPMS*. Portanto, a ferramenta provê uma interface avançada, que permite aos usuários executar e acompanhar as instâncias dos processos de negócio. Esse acompanhamento permite, por exemplo, avaliar em que estado o processo se encontra, por exemplo, em qual atividade o mesmo está parado, quais estão em andamento, quais foram concluídos.

O Quadro 39 lista algumas impressões dos avaliadores.

Algumas impressões dos avaliadores
"O ambiente de execução apresenta módulos de visualização do estado de execução das atividades, permitindo que o gestor de execução da aplicação possa identificar o estágio de execução de forma adequada".
É muito importante, em um processo de negócio, poder acompanhar o andamento da situação. Isso fornece ao usuário um mecanismo de avaliação dos fornecedores de serviços quanto ao tempo de resposta a uma invocação de serviço, quanto à agilidade do negócio. E, por fim, encontrar e solucionar gargalos".
O ambiente, formado pelo motor, por uma interface *web* etc., permite visualizar com tranqüilidade e em detalhes todos os processos em execução".

Quadro 39 - Impressões dos avaliadores - pergunta 6.

7ª pergunta - Você acredita que no futuro as empresas passarão a adotar largamente padrões de processos de negócio (tais como UBL e RosettaNet) nas suas transações internas e com outras empresas?

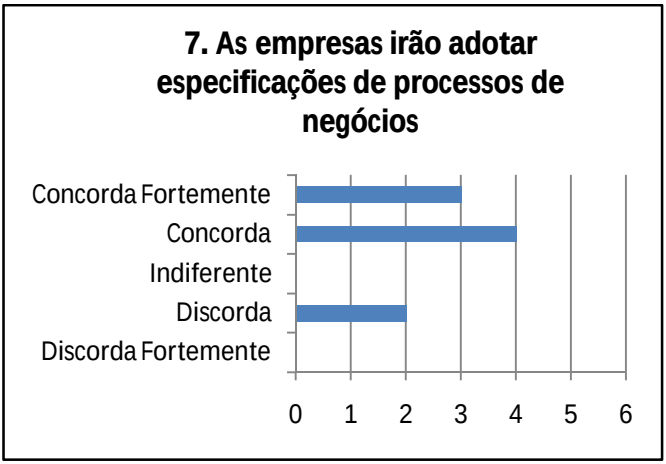


Gráfico 12 - Gráfico sobre se as empresas irão adotar especificações de processos de negócios.

A última pergunta indagou os entrevistados para o fato das empresas adotarem especificações de processos de negócio em suas transações internas e com outras empresas no futuro. No contexto desse trabalho, essa pergunta é importante, pois serve como um indicador da corretude da visão "futurista" da abordagem proposta neste trabalho, fortemente baseada no uso de padrões de processos de negócios pelas organizações. O catálogo é baseado numa especificação de processos de negócio (a UBL), sendo esta utilizada para fazer a classificação dos processos e também dos serviços que os implementam. Essa padronização permite fazer a busca e vinculação dos serviços, também serve como uma linguagem comum na comunicação entre as empresas nas suas transações comerciais. Nesse contexto de interoperabilidade e cooperação, é necessário que essas especificações sejam largamente aceitas e difundidas pelas empresas. Nesse sentido, os entrevistados concordam em consenso, em maior ou menor grau, que as empresas irão adotar essas especificações, portanto reforçando o grau de aplicabilidade futura deste trabalho.

Por outro lado, um dos entrevistados (de uma empresa privada) discorda que as empresas irão adotá-las, visto que elas possuem processos estratégicos, que fornecem um diferencial competitivo e, portanto, não é do interesse delas padronizá-los e disponibilizá-los para fazer parte de uma especificação. Segundo o mesmo entrevistado, essa padronização diminuiu a competitividade empresarial e só faz sentido

em *holdings* (empresas do mesmo grupo) ou em cadeias de suprimentos integradas.

Outro entrevistado (de uma empresa pública) discorda da afirmação, por considerar que esta adoção de padrões de processos de negócio depender de uma profunda e difícil mudança cultural. Nesse sentido, o entrevistado é cético em relação à adoção desses processos, principalmente com relação a outras empresas no que diz respeito a questões relacionadas à persistência de dados proprietários.

Embora se ache que essas observações façam sentido, não se concorda completamente com elas. O primeiro entrevistado que discorda, por exemplo, menciona que a padronização dos processos seria útil em cadeias de suprimentos integradas. Se for observado o que já ocorre, ou seja, cada vez mais as empresas estão trabalhando de forma integrada, em alianças, como um elemento de vantagem competitiva. Portanto, esta integração perpassa fronteiras intraorganizacionais e chega aos processos interoganizacionais, que precisam se comunicar e interoperar corretamente. Além disso, nos processos que efetivamente sejam considerados estratégicos e/ou que lidem com dados proprietários, os serviços a serem vinculados não precisam necessariamente vir de fora, podendo estar no repositório local da empresa. Na verdade, uma aplicação completa SOA pode, na prática, ter serviços que sejam direta e fixamente vinculados a aplicações tradicionais da empresa, como um módulo de ERP, por exemplo.

Contudo, em nível de processo e ambiente BPM, basta o usuário-projetista escolher o serviço desejado, serviço este que, no caso, encapsularia uma invocação ao ERP. Se no futuro a empresa desejar mudar isso, basta mudar no editor BPM. De qualquer forma, a UBL e demais padrões permitem especializações, instanciando certos processos para uma realidade particular e/ou para se inserir o que se consideraria um diferencial em nível de processo. Além disso, o catálogo pode ser usado apenas internamente, dentro de uma mesma empresa.

Outro contra-argumento em relação aos entrevistados é que apesar do processo em si ainda ser considerado um elemento potencial de diferenciação, observa-se cada vez mais que os elementos *produto* (preço, *design*, etc.) e *serviços* associados (não de *software*) são grandes diferenciais, portanto independendo de como as aplicações envolvidas implementam o processo.

Finalmente, não se desconsidera os impactos não tecnológicos da abordagem de integração BPM&SOA proposta. Como qualquer mudança, isso traz impactos; porém, este estudo tecnológico-exploratório visou avaliar em que medida um catálogo pode trazer melhorias, e não que ele resolveria todos os problemas organizacionais das empresas, estando assim fora do escopo desta tese.

O Quadro 40 lista algumas impressões dos avaliadores.

Algumas impressões dos avaliadores
"Sim. Com a quantidade de transações, pessoas e processos se avolumando, crescendo e o mercado exigindo cada vez mais das empresas, adotar um conjunto de processos prontos e já testados é fator estratégico para que empresas possam ser mais competitivas".
"Sim, é evidente a grande vantagem em padronizar os processos de negócio, mas ainda falta uma conscientização das empresas em reusarem as suas informações".
"Acredito que a adoção de padrões nos processos de negócio é uma tendência, mas nem sempre conseguimos ter a ideia completa de que caminhos são tomados na prática. Os padrões surgem para tentar deixar as coisas mais comunicáveis e, por outro lado, fazer a tarefa de desenvolver sistemas que devem trocar informações ser mais fácil para os desenvolvedores. Assim, viabiliza-se que essas aplicações sejam interoperáveis umas com as outras. Ou mesmo, no caso do UBL, uma forma padronizada de efetuar um processo de negócio que pode, em certa instância, ser automatizado. Num modelo de negócios é essencial que os envolvidos falem a mesma língua, ou possua algum mecanismo que faça essa tradução. No final, o que nós, pesquisadores e desenvolvedores podemos fornecer, são sugestões, mas se serão seguidas, é algo que muitas vezes é difícil se prever. Contudo, algo que ainda precisa ser bem trabalhado para que isso possa ter grandes chances de sucesso, é tornar a forma de se trabalhar com isso mais intuitiva, mais facilitada. Acho que é nesse ponto que seu trabalho entra e isso tem grande validade".
"Sem dúvida está é uma tendência de mercado fortíssima. Inclusive várias empresas desenvolvedoras de software já estão começando a montar esta estrutura de compartilhamento e padronização de modelos de negócio e de códigos fonte a fim de reduzir o tempo do ciclo de desenvolvimento de software, compartilhando modelos de negócio, e os códigos fonte".

Quadro 40 - Impressões dos avaliadores - pergunta 7.

6.3 **Publicações**

As publicações constituem-se num importante indicador de avaliação do trabalho proposto perante a comunidade científica especializada.

Procurou-se publicar em eventos considerados relevantes e de prestígio na área. Ao total, oito artigos foram publicados em anais de conferências, além de serem apresentados nos respectivos eventos.

A relação detalhada delas encontra-se no Apêndice D.

6.4 **Considerações**

O objetivo desse capítulo foi de apresentar os procedimentos metodológicos que foram efetuados para fazer a verificação e avaliação, bem como a análise geral dos resultados do modelo proposto.

A verificação se deu através das técnicas de engenharia de software de testes de unidade, testes baseados em casos de uso e testes de integração. Ferramentas de testes, como o *JUnit* (JUNIT, 2010) especializado em testes unitários, e o *SoapUI* (EVIWARE, 2010), que permite gerar relatórios de conformidade *WS-I* (WS-I, 2010), que atestam a interoperabilidade dos serviços *web*, foram utilizadas.

A avaliação teve como objetivo comprovar a hipótese apresentada na Seção 1.4 e mostrar que o modelo proposto consegue responder a pergunta de pesquisa apresentada no Capítulo 1. Por ser um trabalho qualitativo, para avaliar o modelo proposto e seus resultados, a estratégia foi a elaboração de um questionário, respondido por especialistas na área de processos de negócio e serviços *web*.

Em relação à confirmação da hipótese de pesquisa, da relevância do problema e dos cumprimentos dos objetivos da tese, algumas evidências foram coletadas e são apresentadas nos parágrafos que seguem.

A primeira delas tem como base o resultado da aplicação do questionário. Neste sentido, fez-se a ligação deste com o problema, a hipótese de pesquisa e os objetivos da tese, resgatados e mostrados no Quadro 41.

Problema	É possível descobrir de forma dinâmica o mais adequado serviço *web*, levando em conta seus aspectos funcionais e não-funcionais, o contexto dos processos de negócios e o cenário composto por diversos repositórios largamente distribuídos usado por provedores para ofertarem e disponibilizarem seus serviços?
Hipótese de Pesquisa	A concepção de um modelo que una o nível de processos de negócios da empresa e o nível no qual os serviços de software são globalmente disponibilizados, guiado pelas desejadas métricas de QoS e pelo contexto do processo, tem o potencial de garantir que o serviço web mais adequado para executar o processo de negócio em andamento seja descoberto.
Objetivo Geral	Esta tese tem como objetivo fundamental desenvolver um modelo integrado, flexível e aberto de descoberta dinâmica de serviços *web* é capaz de buscar e selecionar o serviço mais adequado consoante o contexto do processo de negócio em questão e os requisitos de qualidade vigentes. Esta descoberta seguiu critérios funcionais e não-funcionais em um cenário pervasivo composto por provedores ofertando e disponibilizando de serviços software
Objetivos Específicos	• Levantar e registrar o referencial teórico em função dos elementos de estudo (BPM, SOA, BPM&SOA, Serviços *web* etc.) e registrar o estado da arte relativo à descoberta de serviços *web*; • Integrar o catálogo de processos de negócios UBL proposto em Bezerra (2011) ao ambiente BPM de concepção de aplicações SOA. Esta integração permite que o projetista de aplicações use processos prontos e consolidados para compor suas aplicações; • Implementar um *crawling* para serviços. O *crawling* é usado na fase de concepção de aplicações, que munido da expressão de descoberta de serviços monta uma lista de serviços candidatos a implementarem determinada atividade. Além do *crawling*, um algoritmo de descoberta dinâmica foi implementado, o qual usa informações colhidas pelo *crawling*, tendo como objetivo definir um (1) serviço a ser utilizado quando aplicações SOA forem executadas; • Organizar um ente lógico denominado Federação (RABELO, 2008) a ser consultado em relação à descoberta de serviços. A Federação é fonte de publicação e pesquisa de serviços e conta com um conjunto de ontologias. Uma ontologia QoS criada e utilizada na especificação de QoS para serviços e outra ontologia (chamada UBL) criada e usada para definir o contexto dos processos de negócios a

	serem descobertos e publicados; • Criar um mecanismo de exportação do processo para um formato padrão de execução, de forma a permitir que esses processos ou aplicações possam ser executados em um ambiente de execução de aplicações; • Construir um ambiente de execução de aplicações, de forma a permitir que aplicações prontas possam ser executadas e acompanhadas; • Projetar um protótipo do modelo proposto que contemple todos os elementos conceituais a serem definidos e instanciados na forma de software.

Quadro 41 – Problema, hipótese e objetivos do trabalho.

Em relação ao problema de pesquisa, o mesmo é considerado relevante com base no resultado da pergunta 1 – "*Na sua opinião o problema da limitada agilidade na integração entre BPM e SOA é relevante de ser melhorado ou resolvido?*". A concordância dos entrevistados dá subsídios para afirmar que este problema é importante e relevante de ser minimizado ou resolvido.

No que tange ao objetivo específico "*Levantar e registrar o referencial teórico em função dos elementos de estudo e registrar o estado da arte relativo à descoberta de serviços web*" foi atingido através da construção dos Capítulos 2 e 3.

Em relação ao objetivo específico "*Integrar o catálogo de processos de negócios UBL proposto em Bezerra (2011) ao ambiente BPM de concepção de aplicações SOA*" cabe salientar que este objetivo específico foi alvo do trabalho de mestrado de Bezerra (2011), sendo avaliado através da pergunta 3 – "*Você concorda que a interface que o usuário projetista interage no editor BPM, e que possibilita a vinculação de serviços e critérios QoS aos processos de negócio, permite a ele interagir facilmente com o catálogo?*". Através do consenso dos entrevistados, estes atestam e concordam que a interface que permite a integração do editor de aplicações com o catálogo foi de fato construída, sendo considerada ainda de fácil uso pelos entrevistados. Mais detalhes sobre a integração do catálogo ao ambiente de edição de aplicação podem ser obtidos nas seções 4.2.3.1, 4.2.3.2 e 5.2.2.

Em relação ao objetivo específico "*Implementar um crawling para serviços. O crawling é usado na fase de concepção de aplicações,*

que munido da expressão de descoberta de serviços monta uma lista de serviços candidatos a implementarem determinada atividade. Além do crawling, um algoritmo de descoberta dinâmica foi implementado, o qual usa informações colhidas pelo crawling, tendo como objetivo definir um (1) serviço a ser utilizado quando aplicações SOA forem executadas". O *crawling* foi usado pelos entrevistados no momento do desenvolvimento de aplicações quando os mesmos selecionaram a opção "buscar serviços" para descobrir serviços candidatos. A descoberta dinâmica foi usada em algumas situações, justamente para que os entrevistados pudessem observar quando e como ocorre seu uso. Nesse sentido, como forma de comprovar que esse objetivo foi de fato cumprido, observar as seções 4.2 e 5.2, que mostram, respectivamente, em detalhes a concepção e a implementação do mecanismo de descoberta proposto.

O objetivo específico "*Organizar um ente lógico denominado Federação a ser consultado em relação à descoberta de serviços. A Federação é fonte de publicação e pesquisa de serviços e conta com um conjunto de ontologias. Uma ontologia QoS criada e utilizada na especificação de QoS para serviços e outra ontologia (chamada UBL) criada e usada para definir o contexto dos processos de negócios a serem descobertos e publicados"* é considerado atingido com base no resultado na pergunta 4 – "*Você concorda que a ontologia desenvolvida nesse trabalho consegue organizar os processos da especificação UBL de forma a transparentemente permitir a vinculação destes com os serviços web correspondentes?*". Com os entrevistados mostrando concordância com a afirmação da pergunta, considera-se que esse objetivo específico também foi atingido. Visto que a construção completa da federação não foi o foco desta tese, concentrou-se apenas na implementação das funcionalidades essenciais para o funcionamento do protótipo computacional, dentre elas as ontologias UBL e de QoS (apresentadas em mais detalhes na Seção 4.2.3.3.2) e no uso distribuído de repositórios de serviços (mais detalhes podem ser consultados na Seção 5.2.6.2).

No que diz respeito ao objetivo específico "*Criar um mecanismo de exportação do processo para um formato padrão de execução, de forma a permitir que esses processos possam ser executados num ambiente de execução*", a pergunta 5 – "*Você concorda que o mecanismo de exportação dos processos permite fazer a conversão da linguagem BPMN para WS-BPEL, simplificando a posterior execução no ambiente de execução de processos?*" permite avaliar se isso foi cumprido. Os entrevistados afirmaram em consenso que de fato o

mecanismo de exportação de processos para o formato de execução foi desenvolvido, permitindo, portanto fazer a conversão do processo descrito em BPMN para o formato executável WS-BPEL. Mais detalhes podem ser obtidos nas seções 4.2.3.1, 4.2.3.2 e 5.2.2.

O objetivo específico "*Construir um ambiente de execução de aplicações, de forma a permitir que os processos exportados possam ser executados e acompanhados*" é alcançado com base no resultado da pergunta 6 – "*Você concorda que o Ambiente de Execução de Processos permite ao usuário executar e acompanhar os processos em WS-BPEL?*". Os entrevistados concordam que o ambiente de execução de processos desenvolvido, baseado na solução *Intalio BPMS*, permite que os processos sejam executados e acompanhados pelos usuários. Mais detalhes podem ser obtidos na Seção 4.2.3.5 e 5.2.5.

O objetivo específico "*Projetar um protótipo do modelo proposto que contemple todos os elementos conceituais a serem definidos e instanciados na forma de software"* é atingido em razão de dois fatores importantes. Um deles é a especificação conceitual e de implementação presente nas seções 4.2 e 5.2. O segundo fator é a oportunidade que os entrevistados tiveram para interagir com o protótipo computacional em execução.

Por fim, em relação à desejada integração BPM&SOA, o resultado da pergunta 2 – "*Você considera que o modelo do catálogo de processos de negócio proposto tem potencial para agilizar (no sentido de tornar a integração mais transparente, interoperável e eficiente) a concepção de aplicações BPM & SOA, minimizando ou resolvendo problemas de:*" (que foi dividida em três itens, onde cada um deles apresenta a concordância dos entrevistados com a resolução dos problemas de falta de semântica associada aos processos e serviços, falta de sinergia entre os envolvidos na modelagem do processo e na sua implementação e a falta do uso de especificações de processos de negócio). Como respostas, os entrevistados concordaram que o catálogo consegue minimizar ou resolver tais problemas, o que corrobora com o objetivo do trabalho, com uma mais ágil e transparente integração BPM&SOA.

Capítulo 7

Conclusões e Trabalhos Futuros

7.1 **Conclusões**

Nesta tese foram apresentados os principais avanços e desafios enfrentados pela comunidade que trabalha na área de descoberta de serviços. A partir do estudo sistemático das várias iniciativas foi possível definir o tema, o problema e a hipótese de pesquisa. Com base nesses elementos foram traçados os objetivos desta tese que, resumidamente, envolvem a integração do mundo dos negócios (BPM) ao nível tecnológico dos sistemas (SOA), através de um modelo de descoberta dinâmico de serviços, focado na tecnologia de serviços *web*.

Considerando a metodologia de pesquisa adotada e os pressupostos assumidos, diversas conclusões puderam ser obtidas. A primeira aponta para a originalidade do modelo proposto. No Capítulo 3, o estudo comparativo das iniciativas relacionadas à descoberta demonstrou que o modelo difere dos demais, incorporando diversos elementos em um só artefato, fato não foi observado em nenhum trabalho analisado.

A segunda conclusão manifesta-se no uso de padrões e recomendações abertos. O uso deles traduziu-se em vantagem, porque permitiu que problemas de integração entre os vários componentes

usados e construídos fossem reduzidos. Além disso, eles criaram um alicerce para que o modelo esteja preparado para o acréscimo de novos elementos conceituais e de implementação, os quais podem ser incorporados com menor tempo e esforço. Por exemplo, devido ao modelo proposto ser SOA, outros serviços e algoritmos podem ser utilizados na descoberta de serviços.

A terceira conclusão aparece sobre o mecanismo de descoberta. Com base nos experimentos efetuados, percebe-se que os algoritmos para o *crawling*, como para descoberta dinâmica, são caracterizados como sistemas de recuperação de informações, pois atendem os critérios clássicos (precisão e cobertura) da área de Recuperação da Informação. Eles possuem precisão máxima em todos os níveis de cobertura, porque usa a ontologia UBL para definir a semântica dos serviços.

A quarta conclusão reside na concepção da ontologia de QoS e da ontologia UBL (ver Seção 5.2.6.1 para maiores detalhes). A ontologia baseada no padrão UBL possibilita incorporar o contexto do processo ao processo de descoberta e a ontologia QoS permite que serviços sejam escolhidos com base em seus valores de QoS. Cabe ressaltar que, conforme mencionado na Seção 4.3, o projetista da aplicação é uma pessoa experimente e conhecedora dos aspectos de QoS e que muitos desses aspectos de QoS podem ser intrinsicamente dinâmicos.

A quinta conclusão está relacionada à divisão do problema em duas partes, a fase de concepção de aplicações e a fase de execução de aplicações. Uma das vantagens da separação é possibilitar uma mais acurada especificação das restrições de QoS, uma vez que o usuário projetista da aplicação não conhece, por vezes, sequer o significado disso. Associado a isso, a estratégia de *crawling*, adotada na fase de concepção de aplicações, mostra-se relevante sob o aspecto de desempenho da aplicação em tempo de execução, porque agiliza a descoberta de serviços em tempo de projeto, de forma que fique mais rápida a seleção do serviço em tempo de execução de aplicações, sem perder a flexibilidade para selecionar o serviço mais adequado.

No que tange ao modelo SaaS e SLA, percebeu-se que os trabalhos estudados (registrados no Capítulo 3) praticamente não atacam este aspecto. O modelo proposto inclui as bases para que tais aspectos possam ser incorporados, pois é uma forma moderna, organizada e transparente de gerenciar a disponibilização e a comercialização de serviços. Isto é extremamente relevante, considerando que o número de

serviços similares e ofertados vem crescendo e a dificuldade no gerenciamento da oferta e comercialização dos mesmos tende também a ficar mais complexa. É importante frisar que a geração automática do SLA causa também diversos impactos, seja de ordem jurídica, de segurança e outros, além da necessidade do gerenciamento do SLA gerado, impactos não tratados pelo modelo proposto.

Além disso, cabe salientar que o modelo proposto não é uma solução totalmente completa para composições de aplicações SOA, embora o modelo contribua de forma efetiva, através da descoberta de serviços, para que a composição ocorra com sucesso.

Em relação à integração BPM&SOA, o uso do modelo proposto mostra que é possível diminuir as dificuldades na integração dos níveis de processos (BPM) e de TI (SOA) e fazer com que esta integração seja mais ágil e transparente. Todavia, a introdução de BPM e SOA numa empresa traz diversos impactos que não se restringem aos de TI, mas vão muito além. Mesmo os aspectos considerados como melhores práticas são muito difícieis de serem aplicados, por questões organizacionais, culturais e financeiras. Um bom exemplo disso é a plena utilização de padrões de TI e de processos pelas empresas, onde estudos empíricos mostram que a grande maioria delas ainda não os adota. Uma adoção implica também impactos, inclusive culturais.

Em síntese, tratou-se de uma pesquisa qualitativa (o modelo proposto) com avaliação quantitativa (através de indicadores da área de RI) e qualitativa (através de um questionário e de publicações, sobre os vários elementos que compõem o modelo proposto e o ineditismo), que visou responder como se pode dar maior agilidade à integração BPM&SOA em um cenário altamente distribuído, no qual provedores disponibilizam serviços e estes são ofertados em repositórios espalhados em diversos locais.

Por ser uma pesquisa de cunho essencialmente exploratório, naturalmente alguns aspectos não puderam ser focados em profundidade, como o de mais eficientes estratégias de busca em sistemas altamente distribuídos. Assim, o aspecto do tempo de resposta da descoberta, apesar de ser extremamente relevante num cenário real, foi considerado “menos crucial” neste estudo exploratório, que basicamente visou investigar: uma abordagem mais integrada para agilizar a ligação BPM&SOA, como o uso de padrões influenciava isso, a viabilidade e impacto da adoção de padrões em nível de processos de negócios e a captura do contexto, entre outros aspectos.

Portanto, o conjunto de experimentos realizados, as comparações efetuadas, os elementos na forma de software projetados e construídos,

são argumentos que, no conjunto, comprovam a veracidade da hipótese, com base em um conjunto de pressupostos de pesquisa desta tese, ou seja, mostram que um modelo integrado, aberto, flexível de descoberta de serviços tem a capacidade de melhor integrar o nível de negócios ao nível de sistemas nas organizações de forma ágil e transparente.

O principal avanço científico desta tese reside na concepção de um modelo de descoberta dinâmica de serviços que integra de forma transparente e ágil o nível de processos ao nível de sistemas. Ele difere das demais propostas estudadas ao aproximar os ambientes BPM e SOA, porque:

- Faz uso intenso de padrões abertos e consolidados;
- Define serviços em termos funcionais, usando a ontologia UBL;
- Usa a ontologia de QoS para informar restrições de QoS (tanto na publicação como na descoberta de serviços);
- Faz a descoberta considerando que os serviços são publicados por diversos provedores de serviços e em repositórios remotos;
- Cria as condições para que o modelo SaaS e SLA possam ser aplicados;
- Usa um *crawling* para auxiliar o projetista de aplicações na descoberta de serviços e na otimização da descoberta em tempo de execução;
- Usa um algoritmo de descoberta dinâmica para descobrir serviços mais adequados em tempo de execução de aplicações;
- Faz uso de uma entidade lógica chamada de federação para gerenciar provedores, ontologias e repositórios de serviços; e
- Permite a edição, conversão e execução de aplicação SOA, incorporando todos os elementos mencionados em um único artefato.

7.2 **Limitações do Modelo**

O modelo proposto possui algumas limitações. Uma delas se relaciona ao tempo para descobrir serviços. O cenário considerado para uso do modelo é composto por repositórios distribuídos e pelo crescente número de serviços sendo publicados e ofertados por diversos provedores. Assim, independentemente da complexidade dos algoritmos propostos e de uma possível otimização deles, isso causa pouca influência no tempo da descoberta, principalmente quando há muitos

repositórios altamente distribuídos e o número de serviços tende a crescer. A justificativa para esse fato reside no conjunto de variáveis intrínsecas às redes de computadores (Internet) que conecta as UDDIs. Isto porque elas, em última análise, prevalecem quando se determina o tempo total da descoberta. A análise e o tratamento dessas variáveis envolveriam o estudo de técnicas de Sistemas Distribuídos, atividade que está fora do escopo deste trabalho.

Outra limitação refere-se à política de classificação de serviços perfeitos usada pelo *crawling* e pelo algoritmo de descoberta dinâmica. O modelo usa um conjunto de atributos com valores fictícios para determinar a reputação de cada provedor. Em um ambiente real de integração e descoberta, essa restrição precisa ser resolvida, sob pena de a classificação (*ranking*) não refletir a realidade dos serviços disponíveis na federação.

Em relação ao mapeamento de aplicações escritas em BPMN para WS-BPEL (ver Figura 23, Capítulo 4), como não foi encontrada uma funcionalidade específica em uma ferramenta BPM de código aberto, o mapeamento BPMN para WS-BPEL foi realizado com base no exemplo proposto por White (2005) e na tradução explícita de elementos BPMN para WS-BPEL, considerando um conjunto pequeno e traduzível de elementos BPMN, todos relacionados aos processos *Ordering* e *Payment* UBL. Portanto, uma mais robusta e ampla estratégia de mapeamento de aplicações BPMN para WS-BPEL precisa ser elaborada, implementada e validada para garantir que qualquer elemento BPMN tenha sua correspondência em WS-BPEL.

O modelo proposto não trata das questões de segurança associadas aos locais onde a aplicação SOA estará sendo composta, nos canais de comunicação envolvidos no processo de descoberta, e nos locais onde estarão os repositórios de serviços. Este é um tema complexo, muito pesquisado na literatura, que se considerou fora do escopo desta tese. Além disso, em razão de BPM ser usada tipicamente com processos estruturados e devido às características intrínsicas do padrão UBL, processos de negócios que exigem a participação humana não são contemplados pelo modelo proposto.

Com relação aos objetivos propostos, conclui-se que foram alcançados, uma vez que o modelo de descoberta dinâmica de serviços foi definido, implementado e avaliado, o que permitiu comprovar a hipótese de pesquisa desta tese, de que é possível descobrir de forma dinâmica o mais adequado serviço *web*, levando em conta seus aspectos funcionais e não-funcionais, o contexto dos processos de negócios e o

cenário composto por diversos repositórios largamente distribuídos usado por provedores para ofertarem e disponibilizarem seus serviços.

Assim, espera-se que o resultado mais expressivo do uso do modelo proposto resida na possibilidade de potencializar um moderno ambiente de competitividade e sustentabilidade para empresas desenvolvedoras de software como serviço (modelo SaaS), sendo possível flexibilizar a escolha de serviços para aplicações SOA, sem ficar-se preso a um único fornecedor, considerando o contexto do processo de negócio e os aspectos desejáveis e existentes de QoS do serviço, tudo isso suportado por um intensivo uso de padrões.

Por fim, ainda há um caminho complexo a ser trilhado quando o objetivo é integrar o nível de negócios ao nível de TI, através da descoberta de serviços. Isso porque ainda não há uma solução geral e completa que incorpore e trate todos os problemas presentes nessa integração, principalmente quando o cenário desejado requer proximidade com a realidade vivenciada pelas organizações. Neste sentido, por exemplo, percebe-se que a adoção dos paradigmas de BPM e SOA têm grandes impactos de várias naturezas em uma organização. Um deles impacta a qualificação das pessoas para trabalharem nos ambientes computacionais associados. Por outro lado, isso não é uma particularidade do BPM ou SOA, mas sim uma regra geral nas organizações. Em outras palavras, a adaptação das pessoas às novas tecnologias é uma situação corriqueira nas organizações, a despeito do custo e problemas que isso tem.

7.3 **Trabalhos Futuros**

Em função de algumas limitações observadas no modelo proposto e no protótipo desenvolvidos e com base nos pressupostos assumidos, alguns pontos são considerados como de interesse científico de serem explorados em trabalhos futuros.

A manutenção e atualização das ontologias (QoS e UBL) são um desses pontos. Para a manutenção da ontologia de QoS, há que se definir: quais serão os atributos usados no âmbito dos serviços, como eles serão descritos, qual será o domínio dos valores usados e quando e como serão medidos. Isso passa, necessariamente, por um esforço da comunidade interessada em organizar e criar um padrão. Nesta direção, a análise da proposta de Qualidade para Serviços (OASIS, 2005) passa

ser um ponto de apoio e referência, mesma ainda encontrando-se sob análise.

No tocante à ontologia de processos UBL, sua manutenção e atualização constituem um desafio por si só, pois acompanhar mudanças nos processos de negócios e poder refletir estas mudanças na ontologia é uma tarefa trabalhosa e complexa, que exige um alinhamento próximo aos processos de negócios presentes nas organizações.

Em relação ao critério de classificação para serviços considerados perfeitos, usado pelo *crawling* e algoritmo de descoberta dinâmica, o estabelecimento de uma política, baseada em dados reais de reputação do provedor (e talvez considerando aspectos econômicos) deve garantir a confiança para que o serviço escolhido seja realmente o melhor. Portanto, sugere-se um estudo efetivo sobre políticas que permita se chegar a uma avaliação do provedor ou do serviço.

Outro ponto importante está relacionado ao desempenho do algoritmo de descoberta, já que sua ordem de complexidade computacional é quadrática. Este fator de complexidade é inadequado para aplicações cujos dados de entrada tendem a crescer bastante (ZIVIANI, 2004), como no caso do modelo proposto. Em função disso, percebe-se a necessidade de um aprofundamento que permita melhorar a eficiência do mecanismo de descoberta. Assim, em razão das características do modelo proposto, sugere-se o estudo e aplicação de técnicas ligadas à área de sistemas distribuídos (por exemplo: estratégia P2P) para contribuir com respostas que permitam melhorar a eficiência da descoberta.

Embora um pouco fora do escopo deste trabalho, todavia importante ao se desejar que ele possa ser aplicado num futuro próximo em cenários reais empresariais, a questão de suporte à segurança poderá ser atacada, num trabalho complementar.

Finalmente, o modelo de descoberta proposto adotou como premissa a utilização de uma federação, ente lógico, usado como fonte de pesquisa e publicação no modelo proposto. Tendo em vista a importância para o desenvolvimento deste trabalho, apenas desenvolveram-se as funcionalidades básicas para seu uso. Estas funcionalidades são descritas na Seção 4.2.3.3. Além da operacionalização básica, a federação ainda carece de diversos módulos para a sua completa operacionalização, como por exemplo um módulo que possa monitorar características de QoS. Outro destes módulos é o que deve tratar da gestão da federação, por exemplo, quem (um indivíduo ou um comitê formado por provedores) e com quais instrumentos (normas) serão usados para que a federação se mantenha.

Além disso, novas funcionalidades devem ser implementadas para um gerenciamento mais intuitivo dos atuais módulos (repositórios de ontologias, repositórios de serviços e gerenciamento da reputação do provedor) e o uso de um de modelo de qualidade de software que permita certificar provedores e serviços ofertados na federação.

Referências

AALST, V. D. W. M. P.; HOFSTEDE, A. H. M. T.; MATHIAS, W. **Business Process Management: A Survey**. Lecture Notes in Computer Science - Springer-Verlag, 2003. Disponivel em: <http://www.springerlink.com/content/9yh5wyawlwv20uae/fulltext.pdf >. Acesso em: 20 dez. 2010.

ADAM, S.; DOERR, J. **How to better align BPM & SOA - ideas on Improving the Transation between process design and deployment**. 9th Workshop on Business Process Modeling, Development, and Support BPMDS'08. Montpellier, France. 2008.

ALBRESHNE, A.; PASQUIER, J. **Semantic-Based Semi-Automatic Web**. Proceedings of the Fith Libyan Arab International Conference On Electrical and Electronic Engineering LAICEEE. Tripoli, Lybia: [s.n.]. 2010. p. 603-615.

ALEXANDRE, J. W. C. et al. **Análise do número de categorias da escala de Likert aplicada À gestão pela qualidade total através da teoria da resposta ao item**. XXIII Encontro Nacional de Engenharia de Produção. Ouro Preto, MG, Brasil: ABEPRO. 2003.

APACHE. **JUDDI v3**, 2010a. Disponivel em: <http://juddi.apache.org/>. Acesso em: 10 Novembro 2010.

APACHE. **ODE (Orchestration Director Engine)**, 2010b. Disponivel em: <http://ode.apache.org/>. Acesso em: 10 Março 2011.

APACHE. **Tomcat**, 2010c. Disponivel em: <http://tomcat.apache.org/>. Acesso em: 08 jun 2011.

APACHE. **Tuscany**, 2010d. Disponivel em: <http://tuscany.apache.org>. Acesso em: 2010 jul 18.

ARAUJO, R. B. D. **Computação Ubíqua: Princípios, Tecnologias e Desafios**. XXI Simpósio Brasileiro de Redes de Computadores. Natal: Sociedade Brasileira de Computação. 2003. p. 45-115.

AZEVEDO, L. G. et al. **Identificação de serviços a partir da modelagem de processos de negócios**. In: V Simpósio Brasileiro de Sistemas de Informação. Brasília: Sociedade Brasileira de Computação. 2009. p. 133-144.

BADR, Y. et al. **Enhancing Web Service Selection by User Preferences of Non-functional Features**. In Proceedings of the 2008 4th international Conference on Next Generation Web Services Practices. Washington, DC, USA: IEEE Computer Society. 2008. p. 60-65.

BAILEY, J.; ZHU, Z. **Fast Discovery of Interesting Collections of Web Services**. IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings). Hong Kong, China. 2006.

BALDAM, R. et al. **Gerenciamento de Processos de Negócios BPM - Business Process Management**. 2a. ed. São Paulo: Érica, 2007.

BASHA, S. J. et al. **Professional java web services**. Birmingham: Wrox Press, 2002. 588 p.

BEHARA, K. G. **BPM and SOA: A Strategic Aliance**. BPMTrends, Maio 2006.

BERNSTEIN, A.; KLEIN, M. **Discovering services: Towards High-Precision Service Retrieval**. International Workshop on Web Services, E-Business, and the Semantic Web. London, UK, UK: Springer-Verlag. 2002. p. 260-275.

BEZERRA, R. O. **Proposta de Catálogo Eletrônico de Processos de Negócio baseados em UBL para a Composição de Aplicações SOA**. Pós-graduação em Engenharia de Automação e Sistemas: Dissertação. UFSC. Florianópolis, p. 163. 2011.

BREITMAN, K. **Web Semântica: a Internet do Futur**o. 1a. ed. Rio de Janeiro: LTC, 2005.

BREOVOLD, H. P.; LARSSON, M. **Component-Based and Service-Oriented Software Engineering: Key Concepts and Principles**. Software Engineering and Advanced Applications, 2007. 33rd EUROMICRO. 2007. p. 13-20.

BUCCHIARONE, A.; GNESI, S. **A Survey on Services Composition Languages and Models**. International Workshop on Web Services Modeling and Testing (WS-MaTe 2006). Lucca, Italy. 2006.

CANCIAN, M. H. **Uma Proposta de Guia de Referência para Provedores de Software como um Serviço**. Pós-graduação em Engenharia de Automação e Sistemas: Dissertação. UFSC. Florianópolis, p. 196. 2009.

CANCIAN, M. H.; RABELO, R. J.; VON WANGENHEIM, C. G. **Uma proposta para elaboração de Contrato de Nível de Serviço para Software-as-a-Service (SaaS)**. 8th International Information and Telecommunication Technologies Symposium. Florianópolis: IEEE Computer Press. 2009.

CARENINI, A. et al. **Semantic Web Service Discovery and Selection: a Test Bed Scenario**. In Proc. of EON & SWS-Challenge 2008 (Sixth International Workshop on Evaluation of Ontology-based tools and the Semantic Web Service Challenge). Tenerife, Spain: CEUR-WS.org. 2008a.

CARENINI, A. et al. **GLUE2: A Web Service Discovery Engine with Non-Functional Properties.** In Sixth European Conference on Web Services. Dublin, Ireland: IEEE Computer Society. 2008b. p. 21-30.

CASTRO-LEON, E.; HE, J.; CHANG, M. **Scaling Down SOA to Small Businesses**. Service-Oriented Computing and Applications, 2007. SOCA '07. IEEE. 2007. p. 99-106.

CHAARI, S. et al. **Framework for Web Service Selection Based on Non-Functional Properties**. In International journal of Web Services Practices, Seoul, Korea, 3, 2008. 94-109.

CLAYBERG, E.; RUBEL, D. **Eclipse: Building Commercial-Quality Plug-ins (Eclipse)**. 2. ed. Addison-Wesley Professional, 2006.

CODEHAUS. **Jetty Web Server**, 2010. Disponivel em: <http://jetty.codehaus.org/jetty/>. Acesso em: 07 set. 2010.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. **Sistemas distribuídos: conceitos e projeto.** 4a. ed. Porto Alegre: Bookman, 2007. 784 p.

DAN, A.; JOHNSON, R.; ARSANJANI, A. **Information as a Service: Modeling and Realization**. International Workshop on Systems Development in SOA Environments, SDSOA '07. IEEE Computer Society. 2007.

DECKER, G. et al. **BPEL4Chor: Extending BPEL for Modeling Choreographies.** IEEE International Conference on Web Services. Salt Lake City, Utha, USA. 2007.

DEITEL, H. M.; DEITEL, P. J. **Java Web Services For Experienced Programmers (Deitel Developers Series)**. New Jersey: Prentice Hall Regents, 2003. 721 p.

DIJKMAN, R. M.; DUMAS, M.; OUYANG, C. **Semantics and analysis of business process models in BPMN**. Information and Software Technology, Newton, MA, USA, 50, 2008. 1281-1294.

DUSTDAR, S. **Service Composition**. International Conference on Service Oriented Computing (ICSoC). Trento, Itália. 2003.

ECLIPSE. **Eclipse Modeling Framework**, 2009. Disponivel em: <http://www.eclipse.org/modeling/emf/>. Acesso em: 21 set. 2010.

ELGAZZAR, K.; HASSAN, A. E.; MARTIN, P. **Clustering WSDL Documents to Bootstrap the Discovery of Web Services**. In Proceedings of the 2010 IEEE International Conference on Web Services (ICWS '10). Washington, DC, USA: IEEE Computer Society. 2010. p. 147-154.

EVIWARE. **SoapUI 3.6.1**, 2010. Disponivel em: <http://www.soapui.org>. Acesso em: 21 set. 2010.

EYHAB, A.-M.; MAHMOUD, Q. H. **Investigating web services on the world wide web**. Proceedings of the 17th international conference on World Wide Web. Beijing, China: ACM, New York, NY, USA. 2008. p. 795-804.

FOURNIER, G. **Essential Software Testing: A Use-Case Approach**. USA: Auerbach Publications, 2009.

GARIMELLA, K.; LEES, M.; WILLIAMS, B. **BPM Basic for Dummies**. Indiana: Wiley Publishing, 2008.

GAROFALAKIS, J. D. et al. **Contemporary Web Service Discovery Mechanisms**. In Journal of Web Eng, 2006. 265-290.

GIL, A. C. **Como elaborar projetos de pesquisa**. 5. ed. São Paulo: Atlas, 2010.

GOLDKUHL, G.; LIND, M. **Coordination and Transformation in Business Processes: towards an integrated view**. Business Process Management Journal (BPMJ), v. 14 n. 16, p. 761-777, 2008.

GONSALVES, E. P. **Iniciação à Pesquisa Científica**. 4a. ed. Campinas SP: Alínea, 2005.

GREER JR, M. B. **Software as a Service Inflection Point: Using Cloud Computing to Archieve Business Agility**. iUniverse, 2009.

GÜNTHER, H. **Como Elaborar um Questionário.** In PASQUALI, Luiz (org.). Instrumentos Psicológicos: manual prático de elaboração. Brasília: UnB, 1999.

IBM. Developer Works. **Dynamic Discovery and Invocation of Web services: How to dynamically discover and invoke a Web service using UDDI and Axis**, 2001. Disponivel em: <http://www.ibm.com/developerworks/webservices/library/ws-udax/index.html>. Acesso em: 04 Junho 2011.

IBM. **WebSphere Business Modeler**, 2010. Disponivel em: <http://www-01.ibm.com/software/integration/wbimodeler/advanced/features/>. Acesso em: 25 set. 2010.

IEEE. **IEEE Standard Glossary of Software Engineering Terminology**, 1990. Disponivel em: <http://www.idi.ntnu.no/grupper/su/publ/ese/ieee-se-glossary-610.12-1990.pdf>. Acesso em: 10 Maio 2011.

INAGANTI, S.; BEHARA, G. K. **Service Identification: BPM and SOA Handshake**. Business Process Trends, Março 2007.

INTALIO. **BPMS**, 2010. Disponivel em: <http://www.intalio.com/bpms>. Acesso em: 23 out. 2010.

JOSUTTIS, N. M. **SOA na prática: A arte de modelagem de sistemas distribuídos**. Rio de Janeiro: Alta Books, 2008.

JUNIT. **Resources for Test Driven Development**, 2010. Disponivel em: <http://www.junit.org/>. Acesso em: 23 nov. 2010.

KAMOUN, F. **A Roadmap Towards the Convergence of Business Process Management and Service Oriented Architecture**. Ubiquity Journal, Dubai, v. 8 n.14, p. 1-8, 2007.

KHURANA, R.; MANDKE, V. V. **Business process modeling with information integrity**. Business Process Management Journal, 15, 2009. 487 - 503.

KOKASH, N. **Web Service Discovery With Implicit QoS Filtering**. In Proceedings of the IBM PhD Student Symposium, in conjunction with the International Conference on Service Oriented Computing. Amsterdam, Netherlands: Springer-verlag. 2005. p. 61-66.

KOURTESIS, D. et al. **Web Service Discovery in a Semantically Extended UDDI Registry: The Case of Fusion**. In: CAMARINHA-MATOS, L., et al. IFIP International Federation for Information

Processing, Establishing the Foundation of Collaborative Networks. Boston USA: Springer, v. 243, 2007. Cap. Establishing The Foundation of Collaborative Networks, p. 547-554.

KOVÁCS, L.; MICSIK, A.; PALLINGER, P. **Two-phase Semantic Web Service Discovery Method for Finding Intersection Matches using Logic Programming**. In Proceedings of SemWS. Zurich, Switzerland. 2006.

KOWALSKI, G. **Information Retrieval Systems - Theory and Implementation**. Kluwer Academic Publishers, 1997.

KRITIKOS, K.; PLEXOUSAKIS, D. **Semantic QoS-based Web Service Discovery Algorithms for Over-Constrained Demands**. In International Journal of Web Services Practice, Seoul, Korea, 2008. 71-82.

KUEHNE, B. T. **Modelos e algoritmos para composição de web services com qualidade de serviço**. Instituto de Ciências Matemáticas e de Computação - ICMS-USP. São Carlos/SP, p. 96. 2009.

LAWS, S. et al. **Tuscany SCA in Action**. Stamford, USA: Manning Publications Co., 2011.

LI, P. et al. **Advanced Non-functional Property Evaluation of Web Services**. European Conference on Web Services. Eindhoven - The Netherlands: IEEE Computer Society. 2009. p. 27-36.

LIKERT, R. **A technique for the measurement of attitudes**. New York. 1932.

LÜER, C. **Evaluating the Eclipse platform as a composition environment**. In 3rd International Workshop on Adoption-Centric Software Engineering (ACSE 2003). Oregon, Portland. 2003.

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. **Introduction to Information Retrieval**, 2008. Disponivel em: <http://nlp.stanford.edu/IR-book/information-retrieval-book.html>. Acesso em: 13 jan. 2010.

MARGOLIS, B. **SOA for the Business Developer: Concepts, BPEL, and SCA (Business Developers series)**. Lewisville, TX, USA: Mc Press, 2007.

MAXIMILIEN, E. M.; SINGH, M. P. **A Framework and Ontology for Dynamic Web Services Selection**. In IEEE Internet Computing Magazine, v. 8, n. 5, p. 84-93, 2004.

MENÉNDEZ, A. I. M. **Uma Ferramenta de apoio ao Desenvolvimento de Web Services**. Dissertação (Programa de Pós-graduação em Informática). Universidade Federal de Campina Grande. Campina Grande, PB. 2002.

MENEZES, E. M.; SILVA, E. L. **Metodologia da Pesquisa e Elaboração de Dissertação**. Florianópolis: UFSC, 2005.

MERSON, P. **Minicurso: Como Documentar Arquitetura de Software**. Simpósio Brasileiro de Banco de Dados e Simpósio Brasileiro de Engenharia de Software. Uberlândia: SBC. 2005.

MILANOVIC, N.; MALEK, M. **Current solutions for Web service composition**. IEEE Internet Computing, NJ/USA, 2004. 51-59.

MILLER, J. et al. LSDIS Lab. University of Georgia. **WSDL-S: Adding Semantics to WSDL- White Paper**, 2004. Disponivel em: <http://lsdis.cs.uga.edu/library/download/wsdl-s.pdf>. Acesso em: 12 dez. 2010.

NESSI. Framing the future of the Service Oriented Economy. **NESSI - European Technology Platform dedicated to Software and Services**, 2006. Disponivel em: <http://www.nessi-europe.com/documents/>. Acesso em: 10 Junho 2011.

NEUBAUER, T. **An empirical study about the status of business process management**. Business Process Management Journal (BMPJ), v. 15 n.2, p. 166-183, 2009.

NOEL, J. **BPM and SOA: Better together**. Whitepaper. IBM Coporation. 2005.

OASIS. **UDDI Overview**, 2000. Disponivel em: <http://www.uddi.org/pubs/UDDI_Overview_Presentation.ppt>. Acesso em: 12 dez. 2010.

OASIS. **UDDI Specification V 3.0.2**, 2004. Disponivel em: <http://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm>. Acesso em: 10 dez. 2010.

OASIS. **Web Services Quality Model (WSQM)**, 2005. Disponivel em: <http://www.oasis-open.org/committees/download.php/15910/WSQM-ver-2.0.doc>. Acesso em: 28 Maio 2011.

OASIS. **Universal Business Language V2.0**, 2006. Disponivel em: <http://docs.oasis-open.org/ubl/os-UBL-2.0/UBL-2.0.html>. Acesso em: 20 dez. 2010.

OASIS. **Web Services Business Process Execution Language Version 2.0**, 2007. Disponivel em: <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>. Acesso em: 20 dez. 2010.

OASIS. **WS-BPEL Extension for People (BPEL4People) Specification Version 1.1**, 2010. Disponivel em: <http://docs.oasis-open.org/bpel4people/bpel4people-1.1.html>. Acesso em: 11 nov. 2010.

O'DOCHERTY, M. **Object-oriented analysis and design:** Understanding system development with UML 2.0. UK: JWE - JOHN WILEY, 2005.

OMG. **UML Profile for Modeling QoS and FT Characteristics and Mechan. Specification, v1.0**, 2006. Disponivel em: <http://www.omg.org/technology/documents/spec_catalog.htm>. Acesso em: 15 Fevereiro 2011.

OMG. **Business Process Model and Notation (BPMN)**, 2011. Disponivel em: <http://www.omg.org/spec/BPMN/2.0/>. Acesso em: 10 Março 2011.

ORACLE. **State of the Business Process Management Market. An Oracle White Paper**, 2008. Disponivel em: <http://www.oracle.com/technologies/bpm/docs/state-of-bpm-market-whitepaper.pdf>. Acesso em: 20 dez. 2010.

O'SULLIVAN, J.; EDMOND, D.; TER HOFSTEDE, A. **What is a Service? Towards Accurate Description of Non-Functional Properties**. Journal of Distributed and Parallel Databases, v. 12, p. 117-133, 2002.

OUYANG, C. et al. **Translating BPMN to BPEL**. BPM Center: a collaborative virtual research center in the area of BPM. BPM Center Report BPM-06-02. 2006.

PAPAZOGLOU, M. P. et al. **Service-Oriented Computing: State of the art Research Challenges**. IEEE Computer Society, 40(11), 2007. 38-45.

PATHAK, J. et al. **Framework for Semantic Web Services Discovery**. In Proceedings of 7th ACM International Workshop on Web Information and Data Management (WIDM'05). Bremen, Germany: ACM. 2005.

PÉREZ, M. et al. **Semi-automatic Discovery of Web Services Driven by User Requirements**. Proceedings of the 21st international conference on Database and expert systems applications: Part I. Bilbao, Spain: Springer-Verlag. 2010. p. 62-75.

PIAZZA, A. P. **Uma Abordagem para Interoperabilidade entre Plataformas Heterogêneas de Serviços Web para Redes Colaborativas de Organizações**. UFSC. Florianópolis. 2007.

PILIOURA, T.; TSALGATIDOU, A.; BATSAKIS, A. **Using WSDL/UDDI and DAML-S in Web Service Discovery**. In Proceedings of WWW 2003 Workshop on E-Services and the Semantic Web (ESSW' 03). Budapest, Hungary. 2003.

PIRES, P. F. **Arquitetura Orientada a Serviços & Gerência de Processos de Negócios**. Simpósio Brasileiro de Banco de Dados e Simpósio Brasileiro de Engenharia de Software. Campinas. 2008. Minicurso realizado no Simpósio Brasileiro de Banco de Dados e Simpósio Brasileiro de Engenharia de Software.

PRESSMAN, R. **Software Engineering: A Practitioner's Approach**. 5. ed. Boston: Mcgraw-Hill do Brasil Ltda, 2001.

RABELO, R. J. Advanced Collaborative Business ICT Infrastructures. In: CAMARINHA-MATOS, L.; AFSARMANESH, H.; OLLUS, M. **Methods and Tools for Collaborative Networked Organizations**. 1a Edição. ed. Nova Iorque: Springer Publishing Company, Incorporated, 2008. p. 337-370.

RAN, S. **A model for web services discovery with QoS**. Journal of SIGecom Exch., New York, NY, USA, 2003. 1-10.

RIJSBERGEN, C. J. V. **Information Retrieval**, London: Butterworths, 1979. Disponivel em: <http://www.dcs.gla.ac.uk/Keith/Preface.html>. Acesso em: 13 dez. 2010.

ROMPOTHONG, P.; SENIVONGSE, T. **A query federation of UDDI registries**. In ACM International Conference Proceeding Series - Proceedings of the 1st international symposium on Information and communication technologies. Dublin, Ireland: Trinity College Dublin. 2003.

ROSETTANET. **RosettaNet Partner Interface Processes (PIPs)**, 2010. Disponivel em: <http://www.rosettanet.org/cms/export/sites/default/RosettaNet/Downloads/RStandards/ClustersSegmentsPIPsOverview_10Oct2008.pdf>. Acesso em: 20 dez. 2010.

SENTANIN, O. F.; SANTOS, F. C. A.; JABBOUR, C. J. C. **Business process management in a Brazilian public research centre**. Business Process Management Journal (BPMJ), 14, n. 4, 2008. p. 483-496.

SHAIKHALI, A. et al. **UDDIe: An extended Registry for Web Services**. In Proceedings of the 2003 Symposium on Applicatrions and

the Internet Workshops. Washington, EUA: IEEE Computer Society. 2003.

SHEN, L. et al. **Web Services Dynamic Discovery Based on Modified CLIQUE Algorithm**. In Proccedings of International Symposium on Intelligent Information Technology Application Workshops. Washington, DC, USA: IEEE Computer Society. 2008. p. 379-382.

SILVA, E. L. D.; MENEZES, E. M. **Metodologia da Pesquisa e Elaboração de Dissertação**. 4a. ed. Florianópolis: UFSC, 2005.

SIVASHANMUGAM, K.; VERMA, K.; SHETH, A. **Discovery of Web Services in a Federated Registry Environment**. In Proceedings of the IEEE international Conference on Web Services. Washington, DC,: IEEE Computer Society. 2004. p. 270.

SOMMERVILLE, I. **Engenharia de Software**. 8a. ed. São Paulo, Brasil: Pearson Education, Inc., 2007.

STOLLBERG, M.; HEPP, M.; HOFFMANN, J. **A caching mechanism for semantic web service discovery**. Proceedings of the 6th international the Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference. Busan, Korea: Springer-Verlag. 2007. p. 11-15.

STORB, B. H. **Um modelo difuso de recuperação de documentos utilizando processamento morfologico**. Universidade Federal de Santa Catarina. Florianópolis, p. 107. 1997.

SYCARA, K. et al. **Automated discovery, interaction and composition of semantic web services**. In Journal of Web Semantic, 2003. 27-46.

TAKAHASHI, T. (Ed.). **Sociedade da Informação no Brasil: Livro Verde**. Brasília: Ministério de Ciência e Tecnologia, 2000.

TONDELLO, G. F. **Especificação Semântica de QoS: A Ontologia QoS-MO**. Programa de Pós-graduação em Ciências da Computação. UFSC. Florianópolis, p. 99. 2008.

TSALGATIDOU, A.; PILIOURA, T. **An Overview of Standards and Related Technology in Web Services**. Journal Distributed and Parallel Databases, Hingham, MA, USA, v. 12 n. 2-3, p. 135-162, 2002.

VITVAR, T.; ZAREMBA, M.; MORAN, M. **Dynamic Service Discovery Through Meta-interactions with Service Providers**. In Proceedings of the 4th European conference on The Semantic Web: Research and Applications. Innsbruck, Austria: Springer-Verlag. 2007. p. 84-98.

W3C. **Web Service Choreography Interface (WSCI)**, 2002. Disponivel em: <http://www.w3.org/TR/wsci/>. Acesso em: 10 março 2011.

W3C. **QoS for Web Services: Requirements and Possible Approaches**, 2003. Disponivel em: <http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/>. Acesso em: 15 Janeiro 2011.

W3C. **Web Services Architecture**, 2004. Disponivel em: <http://www.w3.org/TR/2004/NOTE-ws-arch-20040211>. Acesso em: 20 dez. 2010.

W3C. **Web Services Choreography Description Language Version 1.0**, 2005. Disponivel em: <http://www.w3.org/TR/ws-cdl-10/>. Acesso em: 01 ago. 2011.

W3C. **Semantic Annotations for WSDL**, 2006. Disponivel em: <http://www.w3.org/2002/ws/sawsdl/>. Acesso em: 02 Junho 2011.

W3C. **Dynamic Service Discovery**. A position paper for the W3C Workshop on Web Services for Enterprise Computing, 2007a. Disponivel em: <http://www.w3.org/2007/01/wos-papers/gestalt>. Acesso em: 1 Junho 2011.

W3C. **SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)**, 2007b. Disponivel em: <http://www.w3.org/TR/soap12-part1/>. Acesso em: 20 Julho 2011.

W3C. **Web Services Description Language (WSDL) Version 2.0 Part 1:** Core Language, 2007c. Disponivel em: <http://www.w3.org/TR/wsdl20/>. Acesso em: 23 Junho 2010.

W3C. **OWL 2 Web Ontology Language**, 2009. Disponivel em: <http://www.w3.org/TR/2009/REC-owl2-overview-20091027/>. Acesso em: 23 Março 2011.

WANG, H. et al. **A Solution to Intelligent Services Selection**. In Proceedings of the International Conference on Next Generation Web Services Practices. Washington, DC, USA: IEEE Computer Society. 2006a. p. 155-162.

WANG, H.; LU, T. **Modeling context information in mobile business process**. Advanced Management Science (ICAMS). Chengdu: IEEE. 2010. p. 448 - 452.

WANG, X. et al. **A QoS-aware Selection Model for Semantic Web Services**. In Proceedings of Service-Oriented Computing - ICSOC 2006, 4th International Conference. Chicago, IL, USA: Springer-verlag. 2006b. p. 390-401.

WAZLAWICK, R. S. **Análise e Sistemas de Informação Orientados a Objetos**. 1a. ed. Rio de Janeiro: Elsevier, 2004.

WHITE, S. A. **Using BPMN to Model a BPEL Process**, 2005. Disponivel em: <ttp://www.bptrends.com/publicationfiles/03-05%20WP%20Mapping%20BPMN%20to%20BPEL-%20White.pdf>. Acesso em: 30 Março 2010.

WOODLEY, T.; GAGNON, S. **BPM and SOA: Synergies and Challenges**. WISE 2005. Berlin Heidelberg: Spring-Verlag. 2005. p. 679-688.

WS-I. **The Web Services Interoperability Organization**, 2010. Disponivel em: <http://www.ws-i.org>. Acesso em: 14 nov. 2010.

ZANG, L.-J.; ZHOU, Q. Aggregate UDDI searches with Business Explorer for web services. IBM, 2002. Disponivel em:

<http://www.ibm.com/developerworks/webservices/library/ws-be4ws/>. Acesso em: 20 dez. 2010.

ZELKOWITZ, M. V. **An update to experimental models for validating computer technology**. The Journal of Systems and Software, 2008.

ZHOU, C. et al. **UX- An Architecture Providing QoS-Aware and federated support for UDDI**. In Proccedings of International Conference on Web Services (ICWS03). Las Vegas, Nevada, US: IEEE. 2003.

ZIVIANI, N. **projeto de Algoritmos com Implementações em Pascal e C**. 2. ed. São Paulo: Pioneira Thomson Learning, 2004.

ZONGYAN, Q. et al. **Exploring into the Essence of Choreography**. Technical Report, Peking Univ. China. 2007. http://www.math.pku.edu.cn:8000/var/preprint/7073.pdf.

Apêndice A

Ontologia UBL em OWL

Este apêndice apresenta a ontologia de UBL escrita em OWL 2.

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


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY Ontology1263487447828
"http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#" >
]>


<rdf:RDF
xmlns="http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#"

xml:base="http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.ow
l"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

xmlns:Ontology1263487447828="http://www.semanticweb.org/ontologies/2010/0/Ontol
ogy1263487447828.owl#">
    <owl:Ontology rdf:about=""/>

    <!--

///////////////////////////////////////////////////////////////////////////////
////////
    //
    // Object Properties
    //

///////////////////////////////////////////////////////////////////////////////
////////
     -->
```

```
    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#isExtend
edBy -->

    <owl:ObjectProperty rdf:about="#isExtendedBy"/>

    <!--

///////////////////////////////////////////////////////////////////////////////
////////
    //
    // Classes
    //

///////////////////////////////////////////////////////////////////////////////
////////
     -->

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#AcceptOr
derBuyerParty -->

    <owl:Class rdf:about="#AcceptOrderBuyerParty">
        <rdfs:subClassOf rdf:resource="#BuyerParty"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#AcceptOr
derSellerParty -->

    <owl:Class rdf:about="#AcceptOrderSellerParty">
        <rdfs:subClassOf rdf:resource="#SellerParty"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#AddDetai
l -->

    <owl:Class rdf:about="#AddDetail">
        <rdfs:subClassOf rdf:resource="#SellerParty"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#Biling -
->

    <owl:Class rdf:about="#Biling">
        <rdfs:subClassOf rdf:resource="#UBL"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#BilingBu
sinessRulesAssumed -->

    <owl:Class rdf:about="#BilingBusinessRulesAssumed">
```

```
        <rdfs:subClassOf rdf:resource="#Biling"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#BillingW
ithCreditNote -->

    <owl:Class rdf:about="#BillingWithCreditNote">
        <rdfs:subClassOf rdf:resource="#Traditional"/>
        <owl:disjointWith rdf:resource="#BillingWithDebitNote"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#BillingW
ithDebitNote -->

    <owl:Class rdf:about="#BillingWithDebitNote">
        <rdfs:subClassOf rdf:resource="#Traditional"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#BuyerPar
ty -->

    <owl:Class rdf:about="#BuyerParty">
        <rdfs:subClassOf rdf:resource="#OrderingProcess"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#CancelOr
derBuyerParty -->

    <owl:Class rdf:about="#CancelOrderBuyerParty">
        <rdfs:subClassOf rdf:resource="#BuyerParty"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#CancelOr
derSellerParty -->

    <owl:Class rdf:about="#CancelOrderSellerParty">
        <rdfs:subClassOf rdf:resource="#SellerParty"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#Catalogu
eProvision -->

    <owl:Class rdf:about="#CatalogueProvision">
        <rdfs:subClassOf rdf:resource="#Sourcing"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#Certific
ationofOriginofGoods -->

    <owl:Class rdf:about="#CertificationofOriginofGoods">
        <rdfs:subClassOf rdf:resource="#UBL"/>
```

```
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#ChangeOr
derBuyerParty -->

    <owl:Class rdf:about="#ChangeOrderBuyerParty">
        <rdfs:subClassOf rdf:resource="#BuyerParty"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#ChangeOr
derSellerParty -->

    <owl:Class rdf:about="#ChangeOrderSellerParty">
        <rdfs:subClassOf rdf:resource="#SellerParty"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#CreateCa
talogue -->

    <owl:Class rdf:about="#CreateCatalogue">
        <rdfs:subClassOf rdf:resource="#CatalogueProvision"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#Customer
InitiatedSourcing -->

    <owl:Class rdf:about="#CustomerInitiatedSourcing">
        <rdfs:subClassOf rdf:resource="#Sourcing"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#Delete_C
atalogue -->

    <owl:Class rdf:about="#Delete_Catalogue">
        <rdfs:subClassOf rdf:resource="#CatalogueProvision"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#Despatch
AdviceBusinessRulesAssumes -->

    <owl:Class rdf:about="#DespatchAdviceBusinessRulesAssumes">
        <rdfs:subClassOf rdf:resource="#Fulfilment"/>
        <owl:disjointWith rdf:resource="#ReceiptAdviceBusinessRulesAssumes"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#Freight
-->
```

```
    <owl:Class rdf:about="#Freight">
        <rdfs:subClassOf rdf:resource="#Biling"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#Fulfilme
nt -->

    <owl:Class rdf:about="#Fulfilment">
        <rdfs:subClassOf rdf:resource="#UBL"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#isExtendedBy"/>
                <owl:someValuesFrom
rdf:resource="#CertificationofOriginofGoods"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#isExtendedBy"/>
                <owl:someValuesFrom rdf:resource="#InitiateTransportService"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#isExtendedBy"/>
                <owl:someValuesFrom rdf:resource="#ReportStatusofGoods"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#Initiate
TransportService -->

    <owl:Class rdf:about="#InitiateTransportService">
        <rdfs:subClassOf rdf:resource="#UBL"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#Ordering
-->

    <owl:Class rdf:about="#Ordering">
        <rdfs:subClassOf rdf:resource="#UBL"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#Ordering
Process -->

    <owl:Class rdf:about="#OrderingProcess">
        <rdfs:subClassOf rdf:resource="#Ordering"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#Payment
-->

    <owl:Class rdf:about="#Payment">
```

```
        <rdfs:subClassOf rdf:resource="#UBL"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#isExtendedBy"/>
                <owl:someValuesFrom rdf:resource="#ReportStatusofGoods"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>


    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#PlaceOrd
er -->

    <owl:Class rdf:about="#PlaceOrder">
        <rdfs:subClassOf rdf:resource="#BuyerParty"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#ProcessO
rder -->

    <owl:Class rdf:about="#ProcessOrder">
        <rdfs:subClassOf rdf:resource="#SellerParty"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#Punchout
-->

    <owl:Class rdf:about="#Punchout">
        <rdfs:subClassOf rdf:resource="#Sourcing"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#ReceiptA
dviceBusinessRulesAssumes -->

    <owl:Class rdf:about="#ReceiptAdviceBusinessRulesAssumes">
        <rdfs:subClassOf rdf:resource="#Fulfilment"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#ReceiveO
rder -->

    <owl:Class rdf:about="#ReceiveOrder">
        <rdfs:subClassOf rdf:resource="#SellerParty"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#ReceiveR
esponse -->

    <owl:Class rdf:about="#ReceiveResponse">
```

```
        <rdfs:subClassOf rdf:resource="#BuyerParty"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#RejectOr
der -->

    <owl:Class rdf:about="#RejectOrder">
        <rdfs:subClassOf rdf:resource="#SellerParty"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#Reminder
ForPayment -->

    <owl:Class rdf:about="#ReminderForPayment">
        <rdfs:subClassOf rdf:resource="#Biling"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#ReportSt
atusofGoods -->

    <owl:Class rdf:about="#ReportStatusofGoods">
        <rdfs:subClassOf rdf:resource="#UBL"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#SelfBili
ng -->

    <owl:Class rdf:about="#SelfBiling">
        <rdfs:subClassOf rdf:resource="#Biling"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#SelfBill
ingWithCreditNote -->

    <owl:Class rdf:about="#SelfBillingWithCreditNote">
        <rdfs:subClassOf rdf:resource="#SelfBiling"/>
        <owl:disjointWith rdf:resource="#SelfBillingWithSelfBilledCreditNote"/>
    </owl:Class>



    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#SelfBill
ingWithSelfBilledCreditNote -->

    <owl:Class rdf:about="#SelfBillingWithSelfBilledCreditNote">
        <rdfs:subClassOf rdf:resource="#SelfBiling"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#SellerPa
rty -->

    <owl:Class rdf:about="#SellerParty">
        <rdfs:subClassOf rdf:resource="#OrderingProcess"/>
```

```
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#Sourcing
-->

    <owl:Class rdf:about="#Sourcing">
        <rdfs:subClassOf rdf:resource="#UBL"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#Traditio
nal -->

    <owl:Class rdf:about="#Traditional">
        <rdfs:subClassOf rdf:resource="#Biling"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#UBL -->

    <owl:Class rdf:about="#UBL">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#UpdateCa
taloguePricing -->

    <owl:Class rdf:about="#UpdateCataloguePricing">
        <rdfs:subClassOf rdf:resource="#CatalogueProvision"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#UpdateIt
emSpecification -->

    <owl:Class rdf:about="#UpdateItemSpecification">
        <rdfs:subClassOf rdf:resource="#CatalogueProvision"/>
    </owl:Class>

    <!--
http://www.semanticweb.org/ontologies/2010/0/Ontology1263487447828.owl#UpdateOr
der -->

    <owl:Class rdf:about="#UpdateOrder">
        <rdfs:subClassOf rdf:resource="#BuyerParty"/>
    </owl:Class>

    <!-- http://www.w3.org/2002/07/owl#Thing -->

    <owl:Class rdf:about="&owl;Thing"/>

    <!--

///////////////////////////////////////////////////////////////////////////
////////
```

```
    //
    // General axioms
    //

///////////////////////////////////////////////////////////////////////////////
////////
     -->

    <rdf:Description>
        <rdf:type rdf:resource="&owl;AllDisjointClasses"/>
        <owl:members rdf:parseType="Collection">
            <rdf:Description rdf:about="#CreateCatalogue"/>
            <rdf:Description rdf:about="#Delete_Catalogue"/>
            <rdf:Description rdf:about="#UpdateCataloguePricing"/>
            <rdf:Description rdf:about="#UpdateItemSpecification"/>
        </owl:members>
    </rdf:Description>
    <rdf:Description>
        <rdf:type rdf:resource="&owl;AllDisjointClasses"/>
        <owl:members rdf:parseType="Collection">
            <rdf:Description rdf:about="#CatalogueProvision"/>
            <rdf:Description rdf:about="#CustomerInitiatedSourcing"/>
            <rdf:Description rdf:about="#Punchout"/>
        </owl:members>
    </rdf:Description>
    <rdf:Description>
        <rdf:type rdf:resource="&owl;AllDisjointClasses"/>
        <owl:members rdf:parseType="Collection">
            <rdf:Description rdf:about="#BilingBusinessRulesAssumed"/>
            <rdf:Description rdf:about="#Freight"/>
            <rdf:Description rdf:about="#ReminderForPayment"/>
            <rdf:Description rdf:about="#SelfBiling"/>
            <rdf:Description rdf:about="#Traditional"/>
        </owl:members>
    </rdf:Description>
</rdf:RDF>

<!-- Generated by the OWL API (version 2.2.1.1138)
http://owlapi.sourceforge.net -->
```

Apêndice B

Ontologia QoS em OWL

Este apêndice apresenta a ontologia de QoS escrita em OWL 2.

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


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY unnamed "http://www.owl-ontologies.com/unnamed.owl#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>


<rdf:RDF xmlns="http://www.owl-ontologies.com/unnamed.owl#"
     xml:base="http://www.owl-ontologies.com/unnamed.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:unnamed="http://www.owl-ontologies.com/unnamed.owl#">
    <owl:Ontology rdf:about=""/>



    <!--

///////////////////////////////////////////////////////////////////////////////
////////
    //
    // Object Properties
    //

///////////////////////////////////////////////////////////////////////////////
////////
     -->



    <!-- http://www.owl-ontologies.com/unnamed.owl#AttributeType -->
```

```
    <owl:ObjectProperty rdf:about="#AttributeType">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    </owl:ObjectProperty>



    <!-- http://www.owl-ontologies.com/unnamed.owl#basedOn -->

    <owl:ObjectProperty rdf:about="#basedOn">
        <rdfs:comment rdf:datatype="&xsd;string"
            >Identifica a caracter&#237;stica ou as caracter&#237;sticas que
comp&#245;em o contexto de QoS definido.</rdfs:comment>
        <rdfs:domain rdf:resource="#QoSSingleContext"/>
    </owl:ObjectProperty>



    <!-- http://www.owl-ontologies.com/unnamed.owl#hasCharacteristicAttribute -
->

    <owl:ObjectProperty rdf:about="#hasCharacteristicAttribute">
        <rdfs:domain rdf:resource="#QoSCharacterist"/>
    </owl:ObjectProperty>



    <!-- http://www.owl-ontologies.com/unnamed.owl#hasUnit -->

    <owl:ObjectProperty rdf:about="#hasUnit">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    </owl:ObjectProperty>



    <!-- http://www.owl-ontologies.com/unnamed.owl#subContexts -->

    <owl:ObjectProperty rdf:about="#subContexts">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >Identifica os subcontextos de QoS que comp&#245;em o contexto
definido.</rdfs:comment>
        <rdfs:domain rdf:resource="#QoSCompoundContext"/>
    </owl:ObjectProperty>



    <!--

///////////////////////////////////////////////////////////////////////////
////////
    //
    // Data properties
    //

///////////////////////////////////////////////////////////////////////////
////////
```

```
     -->



    <!-- http://www.owl-ontologies.com/unnamed.owl#abbreviation -->

    <owl:DatatypeProperty rdf:about="#abbreviation">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >Representa&#231;&#227;o abreviada da unidade de
medida.</rdfs:comment>
        <rdfs:domain rdf:resource="#QoSUnit"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>



    <!-- http://www.owl-ontologies.com/unnamed.owl#descricao -->

    <owl:DatatypeProperty rdf:about="#descricao">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >Descreve caracter&#237;stica.</rdfs:comment>
        <rdfs:domain rdf:resource="#QoSCharacterist"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>



    <!-- http://www.owl-ontologies.com/unnamed.owl#direction -->

    <owl:DatatypeProperty rdf:about="#direction">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >Indica se os valores s&#227;o ordenados de forma crescente ou
decrescente.</rdfs:comment>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>



    <!-- http://www.owl-ontologies.com/unnamed.owl#id -->

    <owl:DatatypeProperty rdf:about="#id">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#QoSCharacterist"/>
        <rdfs:range rdf:resource="&xsd;int"/>
    </owl:DatatypeProperty>



    <!-- http://www.owl-ontologies.com/unnamed.owl#isDynamic -->

    <owl:DatatypeProperty rdf:about="#isDynamic">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >Indica se os valores das dimens&#245;es da caracter&#237;stica
podem ser atualizados dinamicamente.</rdfs:comment>
        <rdfs:domain rdf:resource="#QoSCharacterist"/>
```

```
        <rdfs:range rdf:resource="&xsd;boolean"/>
    </owl:DatatypeProperty>



    <!-- http://www.owl-ontologies.com/unnamed.owl#name -->

    <owl:DatatypeProperty rdf:about="#name">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >Nome da unidade de medida.</rdfs:comment>
        <rdfs:domain rdf:resource="#QoSUnit"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>



    <!-- http://www.owl-ontologies.com/unnamed.owl#nameContext -->

    <owl:DatatypeProperty rdf:about="#nameContext">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#QoSContext"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>



    <!-- http://www.owl-ontologies.com/unnamed.owl#nome -->

    <owl:DatatypeProperty rdf:about="#nome">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#QoSCharacteristAttribute"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>



    <!-- http://www.owl-ontologies.com/unnamed.owl#nomeCharacteristic -->

    <owl:DatatypeProperty rdf:about="#nomeCharacteristic">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#QoSCharacterist"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>



    <!-- http://www.owl-ontologies.com/unnamed.owl#statisticalQualifier -->

    <owl:DatatypeProperty rdf:about="#statisticalQualifier">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:comment rdf:datatype="&xsd;string"
            >Identifica o tipo de qualificador estat&#237;sti-co (por exemplo,
m&#233;
, mediana, desvio padr&#227;o, etc.) quando a dimens&#227;o repre-sentar um
valor estat&#237;stico.</rdfs:comment>
        <rdfs:range rdf:resource="&xsd;string"/>
```

```
    </owl:DatatypeProperty>



    <!-- http://www.owl-ontologies.com/unnamed.owl#valor -->

    <owl:DatatypeProperty rdf:about="#valor">
        <rdfs:range rdf:resource="&xsd;float"/>
    </owl:DatatypeProperty>



    <!--

///////////////////////////////////////////////////////////////////////////
////////
    //
    // Classes
    //

///////////////////////////////////////////////////////////////////////////
////////
     -->




    <!-- http://www.owl-ontologies.com/unnamed.owl#Accuracy -->

    <owl:Class rdf:about="#Accuracy">
        <rdfs:subClassOf rdf:resource="#QoSCharacterist"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
                <owl:someValuesFrom rdf:resource="#ErrorGeneratedAttribute"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:comment rdf:datatype="&xsd;string"
            >Accuracy: representa o n&#250;mero de erros gerados por um
servi&#231;o em
determinado espa&#231;o de tempo. &#201; mensurada em fun&#231;&#227;o da
quantidade ou n&#250;mero de erros gerados.</rdfs:comment>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#Acessibility -->

    <owl:Class rdf:about="#Acessibility">
        <rdfs:subClassOf rdf:resource="#QoSCharacterist"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
                <owl:someValuesFrom rdf:resource="#AcessibilityAttribute"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:comment rdf:datatype="&xsd;string"
            >Acessibility (Acessibilidade): capacidade que um servi&#231;o deve
ter em servir aos pedidos de clientes. &#201; medida com base em um
percentual.</rdfs:comment>
```

```
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#AcessibilityAttribute -->

    <owl:Class rdf:about="#AcessibilityAttribute">
        <rdfs:subClassOf rdf:resource="#QoSCharacteristAttribute"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnit"/>
                <owl:someValuesFrom rdf:resource="#Percentage"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#Availability -->

    <owl:Class rdf:about="#Availability">
        <rdfs:subClassOf rdf:resource="#QoSCharacterist"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
                <owl:someValuesFrom rdf:resource="#AvailabilityAttribute"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:comment rdf:datatype="&xsd;string"
            >Availability (Disponibilidade): &#233; a probabilidade de um
servi&#231;o estar pronto para ser usado.</rdfs:comment>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#AvailabilityAttribute -->

    <owl:Class rdf:about="#AvailabilityAttribute">
        <rdfs:subClassOf rdf:resource="#QoSCharacteristAttribute"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnit"/>
                <owl:someValuesFrom rdf:resource="#Percentage"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#Capacity -->

    <owl:Class rdf:about="#Capacity">
        <rdfs:subClassOf rdf:resource="#QoSCharacterist"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
                <owl:someValuesFrom rdf:resource="#CapacityAttribute"/>
```

```
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:comment rdf:datatype="&xsd;string"
            >Capacity (Capacidade): &#233; o limite de pedidos que podem ser
atendidos com garantia de performance por um servi&#231;o.</rdfs:comment>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#CapacityAttribute -->

    <owl:Class rdf:about="#CapacityAttribute">
        <rdfs:subClassOf rdf:resource="#QoSCharacteristAttribute"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnit"/>
                <owl:someValuesFrom rdf:resource="#NumberofRequests"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#DataIntegrityAttribute -->

    <owl:Class rdf:about="#DataIntegrityAttribute">
        <rdfs:subClassOf rdf:resource="#QoSCharacteristAttribute"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnit"/>
                <owl:someValuesFrom rdf:resource="#NumberofFuncionality"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#Day -->

    <owl:Class rdf:about="#Day">
        <rdfs:subClassOf rdf:resource="#FailureAttribute"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnit"/>
                <owl:someValuesFrom rdf:resource="#NumberofFailures"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#DiscoveryWS -->

    <owl:Class rdf:about="#DiscoveryWS">
        <rdfs:subClassOf rdf:resource="#QoSSingleContext"/>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#ErrorGeneratedAttribute -->
```

```
    <owl:Class rdf:about="#ErrorGeneratedAttribute">
        <rdfs:subClassOf rdf:resource="#QoSCharacteristAttribute"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnit"/>
                <owl:someValuesFrom rdf:resource="#NumberofErrors"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#ExceptionHandling -->

    <owl:Class rdf:about="#ExceptionHandling">
        <rdfs:subClassOf rdf:resource="#QoSCharacterist"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
                <owl:someValuesFrom rdf:resource="#FuncionalityAttribute"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:comment rdf:datatype="&xsd;string"
            >ExceptionHandling (Manipula&#231;&#227;o de exce&#231;&#227;o): se
o servi&#231;o possui funcionalidades manipular
exce&#231;&#245;es.</rdfs:comment>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#ExecutionTime -->

    <owl:Class rdf:about="#ExecutionTime">
        <rdfs:subClassOf rdf:resource="#TimeAttribute"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnit"/>
                <owl:someValuesFrom rdf:resource="#Second"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:comment rdf:datatype="&xsd;string"
            >ExecutionTime (baixo) = tempo que um web service leva para
executar uma sequ&#234;ncia de atividades.</rdfs:comment>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#FailureAttribute -->

    <owl:Class rdf:about="#FailureAttribute">
        <rdfs:subClassOf rdf:resource="#QoSCharacteristAttribute"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnit"/>
                <owl:someValuesFrom rdf:resource="#NumberofFailures"/>
            </owl:Restriction>
```

```
        </rdfs:subClassOf>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#FuncionalityAttribute -->

    <owl:Class rdf:about="#FuncionalityAttribute">
        <rdfs:subClassOf rdf:resource="#QoSCharacteristAttribute"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnit"/>
                <owl:someValuesFrom rdf:resource="#NumberofFuncionality"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#GarantiaMSG -->

    <owl:Class rdf:about="#GarantiaMSG">
        <rdfs:subClassOf rdf:resource="#QoSCharacteristAttribute"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnit"/>
                <owl:someValuesFrom rdf:resource="#Percentage"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#Integrity -->

    <owl:Class rdf:about="#Integrity">
        <rdfs:subClassOf rdf:resource="#QoSCharacterist"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
                <owl:someValuesFrom
rdf:resource="#TransactionalIntegrityAttribute"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
                <owl:someValuesFrom rdf:resource="#DataIntegrityAttribute"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:comment rdf:datatype="&xsd;string"
            >Integrity (Integridade): deve prever acesso autorizado a programas
e dados.</rdfs:comment>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#Interoperability -->

    <owl:Class rdf:about="#Interoperability">
```

```
        <rdfs:subClassOf rdf:resource="#QoSCharacterist"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
                <owl:someValuesFrom rdf:resource="#InteroperabilityAttribute"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:comment rdf:datatype="&xsd;string"
            >Interoperability (Interoperabilidade):&#233; a capacidade do
servi&#231;o ser interoper&#225;vel. Caracter&#237;stica medida atrav&#233;s de
um percentual.</rdfs:comment>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#InteroperabilityAttribute --
>

    <owl:Class rdf:about="#InteroperabilityAttribute">
        <rdfs:subClassOf rdf:resource="#QoSCharacteristAttribute"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnit"/>
                <owl:someValuesFrom rdf:resource="#Percentage"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#LatencyAttribute -->

    <owl:Class rdf:about="#LatencyAttribute">
        <rdfs:subClassOf rdf:resource="#QoSCharacteristAttribute"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnit"/>
                <owl:someValuesFrom rdf:resource="#Second"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:comment rdf:datatype="&xsd;string"
            >Latency = tempo entre o envio de um pedido e o recebimento da
resposta.</rdfs:comment>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#Month -->

    <owl:Class rdf:about="#Month">
        <rdfs:subClassOf rdf:resource="#FailureAttribute"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnit"/>
                <owl:someValuesFrom rdf:resource="#NumberofFailures"/>
            </owl:Restriction>
        </rdfs:subClassOf>
```

```
</owl:Class>



<!-- http://www.owl-ontologies.com/unnamed.owl#NumberofErrors -->

<owl:Class rdf:about="#NumberofErrors">
    <rdfs:subClassOf rdf:resource="#QoSUnit"/>
</owl:Class>



<!-- http://www.owl-ontologies.com/unnamed.owl#NumberofFailures -->

<owl:Class rdf:about="#NumberofFailures">
    <rdfs:subClassOf rdf:resource="#QoSUnit"/>
</owl:Class>



<!-- http://www.owl-ontologies.com/unnamed.owl#NumberofFuncionality -->

<owl:Class rdf:about="#NumberofFuncionality">
    <rdfs:subClassOf rdf:resource="#QoSUnit"/>
</owl:Class>



<!-- http://www.owl-ontologies.com/unnamed.owl#NumberofRequests -->

<owl:Class rdf:about="#NumberofRequests">
    <rdfs:subClassOf rdf:resource="#QoSUnit"/>
</owl:Class>



<!-- http://www.owl-ontologies.com/unnamed.owl#Percentage -->

<owl:Class rdf:about="#Percentage">
    <rdfs:subClassOf rdf:resource="#QoSUnit"/>
</owl:Class>



<!-- http://www.owl-ontologies.com/unnamed.owl#Performance -->

<owl:Class rdf:about="#Performance">
    <rdfs:subClassOf rdf:resource="#QoSCharacterist"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
            <owl:someValuesFrom rdf:resource="#LatencyAttribute"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
            <owl:someValuesFrom rdf:resource="#ExecutionTime"/>
        </owl:Restriction>
    </rdfs:subClassOf>
```

```
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
                <owl:someValuesFrom rdf:resource="#ThroughputAttribute"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
                <owl:someValuesFrom rdf:resource="#TransactionTime"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
                <owl:someValuesFrom rdf:resource="#ResponseTime"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:comment rdf:datatype="&xsd;string"
            >Performance (Desempenho): representa qu&#227;o r&#225;pido um
servi&#231;o pode ser completado, sendo mensurado em fun&#231;&#227;o de:
throughput, response time, latency, execution time, e transaction
time.</rdfs:comment>
    </owl:Class>


    <!-- http://www.owl-ontologies.com/unnamed.owl#QoSCharacterist -->

    <owl:Class rdf:about="#QoSCharacterist">
        <rdfs:subClassOf rdf:resource="#DiscoveryWS"/>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#QoSCharacteristAttribute -->

    <owl:Class rdf:about="#QoSCharacteristAttribute">
        <rdfs:subClassOf rdf:resource="#DiscoveryWS"/>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#QoSCompoundContext -->

    <owl:Class rdf:about="#QoSCompoundContext">
        <rdfs:subClassOf rdf:resource="#QoSContext"/>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#QoSContext -->

    <owl:Class rdf:about="#QoSContext"/>



    <!-- http://www.owl-ontologies.com/unnamed.owl#QoSSingleContext -->
```

```
    <owl:Class rdf:about="#QoSSingleContext">
        <rdfs:subClassOf rdf:resource="#QoSContext"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#basedOn"/>
                <owl:someValuesFrom rdf:resource="#QoSCharacterist"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#QoSUnit -->

    <owl:Class rdf:about="#QoSUnit">
        <rdfs:subClassOf rdf:resource="#DiscoveryWS"/>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#Reliability -->

    <owl:Class rdf:about="#Reliability">
        <rdfs:subClassOf rdf:resource="#QoSCharacterist"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
                <owl:someValuesFrom rdf:resource="#Day"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
                <owl:someValuesFrom rdf:resource="#GarantiaMSG"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
                <owl:someValuesFrom rdf:resource="#Month"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
                <owl:someValuesFrom rdf:resource="#Year"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:comment
            >Reliability (Confian&#231;a): representa o grau de confian&#231;a
proporcionado por um servi&#231;o. &#201; medido em termos do n&#250;mero de
falhas ocorridas na execu&#231;&#227;o do servi&#231;o no: dia, m&#234;s e
ano.</rdfs:comment>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#ResponseTime -->
```

```
    <owl:Class rdf:about="#ResponseTime">
        <rdfs:subClassOf rdf:resource="#TimeAttribute"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnit"/>
                <owl:someValuesFrom rdf:resource="#Second"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:comment rdf:datatype="&xsd;string"
            >ResponseTime = tempo requerido para completar um pedido (web
service request).</rdfs:comment>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#Robustness -->

    <owl:Class rdf:about="#Robustness">
        <rdfs:subClassOf rdf:resource="#QoSCharacterist"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
                <owl:someValuesFrom rdf:resource="#RobustnessAttribute"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:comment rdf:datatype="&xsd;string"
            >Robustness (Robustez): representa o grau com o qual um
servi&#231;o pode funcionar adequadamente, mesmo na presen&#231;a de
inv&#225;lido, incompleto ou entradas conflitantes.</rdfs:comment>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#RobustnessAttribute -->

    <owl:Class rdf:about="#RobustnessAttribute">
        <rdfs:subClassOf rdf:resource="#QoSCharacteristAttribute"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnit"/>
                <owl:someValuesFrom rdf:resource="#Percentage"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#Scalability -->

    <owl:Class rdf:about="#Scalability">
        <rdfs:subClassOf rdf:resource="#QoSCharacterist"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
                <owl:someValuesFrom rdf:resource="#ResponseTime"/>
            </owl:Restriction>
        </rdfs:subClassOf>
```

```
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
                <owl:someValuesFrom rdf:resource="#ExecutionTime"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
                <owl:someValuesFrom rdf:resource="#ThroughputAttribute"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
                <owl:someValuesFrom rdf:resource="#LatencyAttribute"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasCharacteristicAttribute"/>
                <owl:someValuesFrom rdf:resource="#TransactionTime"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:comment rdf:datatype="&xsd;string"
            >Scalability (Escalabilidade) : representa a capacidade de
crescimento do sistema computacional do fornecedor do servi&#231;o e a
habilidade para processar um maior n&#250;mero de pedidos, opera&#231;&#245;es
ou transa&#231;&#245;es em um dado per&#237;odo de tempo.</rdfs:comment>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#Second -->

    <owl:Class rdf:about="#Second">
        <rdfs:subClassOf rdf:resource="#QoSUnit"/>
    </owl:Class>

    <!-- http://www.owl-ontologies.com/unnamed.owl#ThroughputAttribute -->

    <owl:Class rdf:about="#ThroughputAttribute">
        <rdfs:subClassOf rdf:resource="#QoSCharacteristAttribute"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnit"/>
                <owl:someValuesFrom rdf:resource="#Second"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:comment rdf:datatype="&xsd;string"
            >Throughput = n&#250;mero de pedidos atendidos em um intervalo de
tempo.</rdfs:comment>
    </owl:Class>

    <!-- http://www.owl-ontologies.com/unnamed.owl#TimeAttribute -->

    <owl:Class rdf:about="#TimeAttribute">
        <rdfs:subClassOf rdf:resource="#QoSCharacteristAttribute"/>
        <rdfs:subClassOf>
            <owl:Restriction>
```

```
                <owl:onProperty rdf:resource="#hasUnit"/>
                <owl:someValuesFrom rdf:resource="#Second"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>



    <!-- http://www.owl-ontologies.com/unnamed.owl#TransactionTime -->

    <owl:Class rdf:about="#TransactionTime">
        <rdfs:subClassOf rdf:resource="#TimeAttribute"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnit"/>
                <owl:someValuesFrom rdf:resource="#Second"/>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:comment rdf:datatype="&xsd;string"
            >TransactionTime (baixo) = tempo para executar uma
transa&#231;&#227;o completa.</rdfs:comment>
    </owl:Class>



    <!-- http://www.owl-
ontologies.com/unnamed.owl#TransactionalIntegrityAttribute -->

    <owl:Class rdf:about="#TransactionalIntegrityAttribute">
        <rdfs:subClassOf rdf:resource="#QoSCharacteristAttribute"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnit"/>
                <owl:someValuesFrom rdf:resource="#NumberofFuncionality"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>

    <!-- http://www.owl-ontologies.com/unnamed.owl#Year -->

    <owl:Class rdf:about="#Year">
        <rdfs:subClassOf rdf:resource="#FailureAttribute"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty rdf:resource="#hasUnit"/>
                <owl:someValuesFrom rdf:resource="#NumberofFailures"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>
</rdf:RDF>



<!-- Generated by the OWL API (version 2.2.1.1138)
http://owlapi.sourceforge.net -->
```

Apêndice C

Interface WSDL do Serviço UBL

Este apêndice apresenta um trecho do arquivo de descrição de interface do serviço (WSDL) utilizado no serviço *AccountingSupplier/NotifyPayee*, do processo *Payment Process*.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="UBL_PaymentProcess_AccountingSupplier_NotifyPayee"
targetNamespace="http://ubl.oasis.services/payment/paymentprocess/accountingSup
plier/notifyPayee"
xmlns:tns=
"http://ubl.oasis.services/payment/paymentprocess/accountingSupplier/notifyPaye
e"
xmlns:ns1="urn:oasis:names:specification:ubl:schema:xsd:RemittanceAdvice-2"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:pctx="http://gsigma.ufsc.br/processContext"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
 <wsdl:types>
  <xsd:schema
targetNamespace="http://ubl.oasis.services/payment/paymentprocess/accountingSup
plier/notifyPayee"
xmlns:tns="http://ubl.oasis.services/payment/paymentprocess/accountingSupplier/
notifyPayee">
   <xsd:import namespace="http://gsigma.ufsc.br/processContext"
    schemaLocation="xsd/processContext.xsd" />
   <xsd:import
    namespace="urn:oasis:names:specification:ubl:schema:xsd:RemittanceAdvice-2"
    schemaLocation="xsd/ubl/maindoc/UBL-RemittanceAdvice-2.0.xsd" />
   <xsd:element name="notifyPayee">
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element name="input" nillable="true"
       type="ns1:RemittanceAdviceType" />
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
   <xsd:element name="notifyPayeeResponse">
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element name="output" nillable="true"
       type="ns1:RemittanceAdviceType" />
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
```

```
  <!-- Async -->
  <xsd:element name="notifyPayeeAsync">
   <xsd:complexType>
    <xsd:sequence>
     <xsd:element name="processContext" nillable="false"
      type="pctx:ProcessContext" />
     <xsd:element name="input" nillable="true"
      type="ns1:RemittanceAdviceType" />
    </xsd:sequence>
   </xsd:complexType>
  </xsd:element>
  <xsd:element name="notifyPayeeAsyncCallback">
   <xsd:complexType>
    <xsd:sequence>
     <xsd:element name="processContext" nillable="false"
      type="pctx:ProcessContext" />
     <xsd:element name="output" nillable="true"
      type="ns1:RemittanceAdviceType" />
    </xsd:sequence>
   </xsd:complexType>
  </xsd:element>
  <xsd:element name="notifyPayeeAsyncCallbackResponse">
   <xsd:complexType>
    <xsd:sequence>
     <xsd:element name="processContext" nillable="false"
      type="pctx:ProcessContext" />
    </xsd:sequence>
   </xsd:complexType>
  </xsd:element>
  <!-- Async -->
  <xsd:element name="alive">
   <xsd:complexType>
    <xsd:sequence />
   </xsd:complexType>
  </xsd:element>
  <xsd:element name="aliveResponse">
   <xsd:complexType>
    <xsd:sequence>
     <xsd:element name="output" nillable="true" type="xsd:boolean" />
    </xsd:sequence>
   </xsd:complexType>
  </xsd:element>
 </xsd:schema>
</wsdl:types>
<wsdl:message name="notifyPayeeRequestMsg">
 <wsdl:part element="tns:notifyPayee" name="payload" />
</wsdl:message>
<wsdl:message name="notifyPayeeResponseMsg">
 <wsdl:part element="tns:notifyPayeeResponse" name="payload" />
</wsdl:message>
<!-- Async -->
<wsdl:message name="notifyPayeeAsyncRequestMsg">
 <wsdl:part element="tns:notifyPayeeAsync" name="payload" />
</wsdl:message>
<wsdl:message name="notifyPayeeAsyncCallbackRequestMsg">
 <wsdl:part element="tns:notifyPayeeAsyncCallback" name="payload" />
</wsdl:message>
```

```
 <wsdl:message name="notifyPayeeAsyncCallbackResponseMsg">
  <wsdl:part element="tns:notifyPayeeAsyncCallbackResponse"
   name="payload" />
 </wsdl:message>
 <!-- Async -->
 <wsdl:message name="aliveRequestMsg">
  <wsdl:part element="tns:alive" name="payload" />
 </wsdl:message>
 <wsdl:message name="aliveResponseMsg">
  <wsdl:part element="tns:aliveResponse" name="payload" />
 </wsdl:message>
 <wsdl:portType name="UBL_PaymentProcess_AccountingSupplier_NotifyPayee">
  <wsdl:operation name="notifyPayee">
   <wsdl:input message="tns:notifyPayeeRequestMsg" name="notifyPayeeRequest" />
   <wsdl:output message="tns:notifyPayeeResponseMsg" name="notifyPayeeResponse"
/>
  </wsdl:operation>
  <!-- Async -->
  <wsdl:operation name="notifyPayeeAsync">
   <wsdl:input message="tns:notifyPayeeAsyncRequestMsg"
name="notifyPayeeAsyncRequest" />
  </wsdl:operation>
  <!-- Async -->
  <wsdl:operation name="alive">
   <wsdl:input message="tns:aliveRequestMsg" name="aliveRequest" />
   <wsdl:output message="tns:aliveResponseMsg" name="aliveResponse" />
  </wsdl:operation>
 </wsdl:portType>
 <wsdl:binding
  name="UBL_PaymentProcess_AccountingSupplier_NotifyPayeeServiceBinding"
  type="tns:UBL_PaymentProcess_AccountingSupplier_NotifyPayee">
  <soap:binding style="document"
   transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="notifyPayee">
   <soap:operation
soapAction="http://ubl.oasis.services/payment/paymentprocess/accountingSupplier
/notifyPayee/notifyPayee" />
   <wsdl:input name="notifyPayeeRequest">
    <soap:body use="literal" />
   </wsdl:input>
   <wsdl:output name="notifyPayeeResponse">
    <soap:body use="literal" />
   </wsdl:output>
  </wsdl:operation>
  <!-- Async -->
  <wsdl:operation name="notifyPayeeAsync">
   <soap:operation
soapAction="http://ubl.oasis.services/payment/paymentprocess/accountingCustomer
/notifyPayee/notifyPayeeAsync" />
   <wsdl:input name="notifyPayeeAsyncRequest">
    <soap:body use="literal" />
   </wsdl:input>
  </wsdl:operation>
  <!-- Async -->
  <wsdl:operation name="alive">
   <soap:operation
soapAction="http://ubl.oasis.services/payment/paymentprocess/accountingSupplier
/notifyPayee/alive" />
   <wsdl:input name="aliveRequest">
    <soap:body use="literal" />
```

```
    </wsdl:input>
    <wsdl:output name="aliveResponse">
     <soap:body use="literal" />
    </wsdl:output>
   </wsdl:operation>
  </wsdl:binding>
  <wsdl:service
   name="UBL_PaymentProcess_AccountingSupplier_NotifyPayeeService">
   <wsdl:port
    name="UBL_PaymentProcess_AccountingSupplier_NotifyPayeeServicePort"

binding="tns:UBL_PaymentProcess_AccountingSupplier_NotifyPayeeServiceBinding">
    <soap:address
location="http://localhost:8082/services/ubl/paymentprocess/accountingSupplier/
notifyPayee" />
   </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Apêndice D

Relação de Publicações

Este apêndice apresenta a relação de publicações originadas a partir do desenvolvimento desta tese.

D.1 Artigos Publicados em Eventos Nacionais

Em eventos nacionais foram publicados quatro artigos relacionados ao modelo proposto. O primeiro artigo *Um Modelo de Descoberta Dinâmica de Serviços de Software Baseado no Contexto de Processos de Negócios Empresariais e em QoS* apresentou e descreveu o modelo de descoberta. O artigo foi publicado nos anais do Simpósio Brasileiro de Sistemas Multimídia e *Web* - III Workshop de Gestão de Processos de Negócio em 2009.

O segundo artigo *Descoberta Dinâmica de Serviços: Um Modelo de Seleção Semântica de Serviços para ambientes BPM&SOA* teve como foco a apresentação do protótipo computacional construído com base no modelo proposto. O artigo foi publicado nos anais do Simpósio Brasileiro de Sistemas de Informação - IV Workshop de Gestão de Processos de Negócio em 2010.

O terceiro artigo *Uma solução aberta e flexível de descoberta de serviços para maior agilidade na integração BPM&SOA* registrou experimentos realizados e resultados obtidos com a pesquisa. O artigo foi publicado nos anais do Simpósio Brasileiro de Sistemas de Informação - V Workshop de Gestão de Processos de Negócio em 2011.

Um quarto artigo intitulado *Proposta de Catálogo Eletrônico de Processos de Negócio Baseados em UBL para Composição de Aplicações SOA* foi publicado em co-autoria, também no Simpósio Brasileiro de Sistemas de Informação - V Workshop de Gestão de Processos de Negócio em 2011. O artigo apresentou o desenvolvimento de um catálogo de processos de negócios, um dos principais elementos conceituais do modelo proposto nesta tese.

O Quadro 42 resume informações sobre os eventos nacionais, onde foram publicados os quatro artigos.

Evento	Ano	Promoção	Local
Simpósio Brasileiro de Sistemas Multimídia e *Web* - III Workshop de Gestão de Processos de Negócio	2009	Sociedade Brasileira de Computação	Fortaleza/CE
Simpósio Brasileiro de Sistemas de Informação - IV Workshop de Gestão de Processos de Negócio	2010	Sociedade Brasileira de Computação	Marabá/PA
Simpósio Brasileiro de Sistemas de Informação - V Workshop de Gestão de Processos de Negócio	2011	Sociedade Brasileira de Computação	Salvador/BA

Quadro 42 - Lista dos eventos nacionais com artigos publicados.

D.2 Artigos Publicados em Eventos Internacionais

Em eventos internacionais foram publicados quatro artigos relacionados ao modelo proposto. O primeiro artigo *An Approach for a more Agile BPM-SOA Integration Supported by Dynamic Services Discovery* (14th IEEE[20] *International Enterprise Computing Conference* – EDOC) apresentou resultados preliminares do protótipo computacional desenvolvido, neste ano casualmente realizada no Brasil.

O segundo artigo *Supporting Software Services Discovery and Sharing in Collaborative Networks* (PRO-VE'11 - 11th IFIP[21] *Working Conference on Virtual Enterprises*) visou mostrar o uso do modelo proposto no âmbito das redes colaborativas de organizações.

O terceiro artigo *Software services discovery as a mean to enhance collaboration in networked enterprises*, além de mostrar a aplicação do modelo proposto, descreveu o protótipo computacional construído e resultados de experimentos realizados com o mesmo, no

[20] IEEE *Computer Society*.
[21] *International Federation for Information Processing.*

âmbito das redes colaborativas de organizações, também apresentado na conferência PRO-VE'11.

O quarto artigo *A Model for Dynamic Services Discovery over largely distributed providers based on QoS and Business Processes contexts* foi submetido com a finalidade de mostrar os resultados finais obtidos a partir do modelo proposto. Cabe ressaltar que o artigo foi aceito no SERVICES 2011, evento realizado no âmbito do *International Conference on Web Services* (ICWS) 2011, mas que por falta de recursos não foi possível a participação no evento e a apresentação do artigo.

O Quadro 43 apresenta informações adicionais sobre os eventos internacionais onde foram publicados os quatro artigos.

Evento	Ano	Promoção	Local
14th IEEE EDOC *Conference - Workshop Service-oriented Enterprise Architecture for Enterprise Engineering*	2010	IEEE *Computer Society*	Vitória (ES)/ Brasil
11th IFIP *Working Conference on Virtual Enterprises*	2010	IFIP e outros	Saint-Etienne/ França
12th IFIP *Working Conference on Virtual Enterprises*	2011	IFIP e outros	São Paulo(SP)/Brasil
7th *World Congress on Services* IEEE *Services* 2011	2011	IEEE, IBM, SAP e outros	Washington (DC), USA

Quadro 43 - Lista dos eventos internacionais com artigos publicados.

D.3 Artigo submetido para um *Journal*

Os principais e mais recentes resultados obtidos a partir do desenvolvimento do modelo proposto foram organizados em um artigo científico e submetido ao *International Journal of Web Services Research* (IJWSR), considerado uma das revistas mais importantes da área e de grande impacto. O artigo apresenta o modelo de descoberta, o protótipo computacional desenvolvido, os experimentos realizados e comenta sobre os avanços e principais contribuições do modelo proposto.

Apêndice E

Testes de Unidade

Este apêndice apresenta o detalhamento dos testes de unidade efetuados em cada módulo presente na arquitetura de implementação do protótipo computacional.

E.1 Conector de Integração

O teste do conector de integração com o catálogo foi feito tomando como base os submódulos conceituais que compõem o conector de integração. O objetivo foi avaliar se esse módulo, implementado pelo protótipo, atende a especificação.

A abordagem utilizada para esses testes é a *baseada em cenários*, que segundo Pressman (2001) concentra-se nas ações que o usuário faz. Isso significa que as tarefas executadas pelo usuário, capturadas via casos de uso, servem como fonte para a definição dos casos de teste. Segundo O'Docherty (2005), o teste baseado em casos de uso é particularmente importante em sistemas que são dirigidos ao usuário. Como o conector de integração com o catálogo age numa camada em que há interação com o usuário, escolheu-se essa abordagem para fazer o teste deste módulo.

Para a definição dos casos de teste, utilizou-se a abordagem proposta por Fournier (2009), que descreve um caso de teste como um conjunto formado pelas seguintes informações:

- Nome: identifica o caso de teste. O nome geralmente está associado ao caso de uso relacionado;
- Descrição: o que o caso de teste irá verificar e o que acontece durante o teste;
- Precondições: o que é precisa estar condicionado para o caso de teste poder ser executado;
- Entradas: quais são as variáveis necessárias para a execução do teste;

- Resultado esperado: o que se espera da execução com êxito do teste.

De acordo com a Seção 4.2.3, que descreve a arquitetura conceitual do modelo proposto, criou-se os seguintes casos de testes:

- Importação do Processo do Catálogo;
- Exportação do Processo para o Catálogo;
- Definição de QoS às atividades do processo;
- Invocar a Descoberta de Serviços e associá-los às tarefas;
- Exportar o Processo para a Linguagem WS-BPEL.

O ambiente em que os testes foram realizados é descrito no Quadro 44.

Local/Ambiente	Laboratório de Tecnologia de Informação e Comunicação do Departamento de Automação e Sistemas da UFSC
Computador	• Processador: Intel Centrino Duo Core de 1.66GHz • Microsoft Windows XP Professional Service Pack 3 • Memória RAM de 2GB • Conexão com a Internet
Observações	O Catálogo de Processos de Negócio, acessado pelo conector, encontra-se na mesma máquina deste.

Quadro 44 - Características do ambiente de testes usado para testar o conector.

Os testes foram realizados no Laboratório de Tecnologia de Informação e Comunicação, do Departamento de Automação e Sistemas da UFSC, dentro do grupo de pesquisa GSigma, do qual o autor desse trabalho faz parte. Como o conector utiliza os serviços do catálogo para poder funcionar, é necessário que uma instância deste esteja em funcionamento, conforme salientado no item “Observações” do Quadro 44. Ressalta-se que o objetivo desse teste é garantir que o conector de integração funciona. Apesar deste conector requerer o Catálogo de Processos de Negócio para poder funcionar, este não é o enfoque do teste. A próxima subseção mostra o teste específico para o Catálogo de Processos de Negócio.

O Quadro 45, apresenta os casos de teste desenvolvidos, juntamente com os resultados obtidos.

Caso de Teste 1 – Importação do Processo do Catálogo	
Descrição	Esse caso de teste verifica se a importação de processos do catálogo está funcionando da maneira desejada.
Precondições	O editor BPM está executando no computador do cliente; O Catálogo de Processos de Negócio está *online* e acessível pelo conector.
Entradas	**Tipo do Processo** – se é um processo da especificação ou um processo previamente armazenado pelo usuário; **Nome da Especificação** – Se o usuário estiver querendo importar um processo de uma especificação, especifica-se o seu nome; **Nome do Processo** – nome do processo que será importado; **Nome do Projeto de Destino** - para qual projeto, dentro do editor BPM, o processo será importado.
Resultado Esperado	Após a invocação pelo usuário, a tela de importação de processos deve ser apresentada, com as opções de importação de processos da especificação e de processos previamente armazenados pelo usuário; Se a escolha for importar um processo da especificação, o sistema solicita ao usuário o nome da especificação, e em seguida escolhe o processo dentro da hierarquia de processos da especificação; Se a escolha for importar um processo previamente armazenado, o usuário escolhe o processo numa lista; O sistema acessa o Catálogo de Processos de Negócio e obtém o processo, em seguida o converte para o formato utilizado pelo editor BPM e o exibe ao usuário; O processo importado deve ser idêntico ao que está armazenado no catálogo.

Quadro 45 - Caso de Teste – Importação do Processo do Catálogo.

A análise do primeiro caso de teste aponta que o resultado da importação é compatível com o esperado. O processo é importado com êxito para a ferramenta de edição de aplicações e a sua estrutura, em termos de atividades, fluxos e demais elementos estruturais é mantida, exatamente da mesma forma que o processo está armazenado no catálogo. Neste caso de teste, os seguintes processos da especificação UBL foram importados:

- Processo de Criação de Catálogo (*Create Catalogue Process*);
- Processo de Ordem (*Ordering Process*);

- Processo de Pagamento (*Payment Process*).

Os demais processos não puderam ser importados, pois os mesmos não foram implementados no Catálogo de Processos de Negócio.

A análise do segundo caso de teste (detalhado no Quadro 46) aponta que o resultado da exportação é compatível com o esperado.

Caso de Teste 2 – Exportação do Processo para Catálogo	
Descrição	Esse caso de teste verifica se a exportação do processo para o catálogo está funcionando da maneira desejada.
Precondições	O editor BPM está executando no computador do cliente; Existe um processo dentro do editor que será usado na exportação ; O Catálogo de Processos de Negócio está *online* e acessível pelo *plug-in*.
Entradas	**Nome do Projeto de Origem** – o nome do projeto onde o processo que será exportado se encontra; **Nome do Processo** – nome do processo que será exportado;
Resultado Esperado	Após a invocação pelo usuário, a tela de exportação de processos deve ser apresentada; O usuário escolhe o projeto onde o processo se encontra. A lista dos processos contidos no projeto deve se apresentada; O usuário escolhe o processo que deseja exportar; O usuário escolhe qual o nome do processo que deseja usar na armazenagem do processo no catálogo; O sistema deve converter o processo escolhido para o formato do catálogo e o armazena; O processo que foi armazenado no catálogo deve ser idêntico ao que serviu como base para a exportação.

Quadro 46 - Caso de Teste – Exportação do Processo para o Catálogo.

O resultado do teste mostrou que somente as principais estruturas da linguagem BPMN foram reconhecidas, de forma que algumas estruturas que o editor *Websphere* utiliza não são suportadas na exportação. As estruturas são suportadas na exportação são:

- Tarefa;

- Decisão Simples (dois caminhos);
- Decisão Múltipla (vários caminhos);
- União, Bifurcação;
- Troca de Documentos.

As estruturas que não estão nessa lista não são exportadas, sendo "apagadas" do processo armazenado no catálogo.

Caso de Teste 3 – Definição de QoS às atividades do processo	
Descrição	Esse caso de teste verifica se o procedimento de atribuir restrições de QoS atribuídas às atividades do processo está funcionando de maneira adequada.
Precondições	O editor BPM está executando no computador do cliente; Existe um processo já construído no editor BPM, que será usado para a definição dos critérios de QoS; O Catálogo de Processos de Negócio está *online* e acessível pelo *plug-in*.
Entradas	**Nome da Atividade da Aplicação** – o nome da atividade do processo que se deseja atribuir à restrição; **Item de QoS** – o item de QoS usado para se estabelecer a restrição.
Resultado Esperado	O usuário seleciona a atividade que deseja atribuir restrições de QoS. Em seguida, escolhe a opção "Adicionar Restrição de QoS"; O sistema se conecta ao catálogo e obtém a lista de itens de QoS disponíveis; O usuário seleciona o item de QoS e solicita que o mesmo seja adicionado as restrições de QoS da atividade; O sistema adiciona o item de QoS à atividade; A lista atual de critérios de QoS é apresentada ao usuário; O usuário tem a opção de escolher qual valor o critério de QoS deve ter e também qual a comparação será utilizada (maior, maior ou igual, igual, menor, menor ou igual).

Quadro 47 - Caso de Teste – Definição de QoS às atividades da aplicação.

A análise do terceiro caso de teste (mostrado no Quadro 47) aponta que o resultado da definição de restrições de QoS às atividades do processo é compatível com o esperado. Os critérios de QoS foram atribuídos com êxito às atividades da aplicação.

Caso de Teste 4 – Invocar a Descoberta de Serviços e vinculá-los às tarefas	
Descrição	Esse caso de teste verifica se o procedimento de invocar a descoberta de serviços e associar esses serviços às atividades do processo está funcionando de maneira adequada.
Precondições	O editor BPM está executando no computador do cliente; Existe um processo já construído no editor BPM, com as restrições de QoS associadas e que será usado na invocação da descoberta e associação de serviços O Catálogo de Processos de Negócio está *online* e acessível pelo *plug-in*.
Entradas	**Nome da Atividade da Aplicação** – o nome da atividade do processo que se deseja invocar a descoberta de serviços;
Resultado Esperado	O usuário seleciona a atividade ao qual deseja atribuir serviços. Em seguida, escolhe a opção "Descobrir Serviços"; O sistema se conecta ao catálogo, que invoca o mecanismo de descoberta. Os parâmetros utilizados para a busca são a classificação ontológica da atividade e o conjunto de restrições de QoS; Os serviços descobertos são retornados, sendo divididos em duas categorias: serviços que tiverem correspondência total dos requisitos de QoS e serviços que tiveram correspondência parcial; O sistema deve associar os serviços que tiveram correspondência total à atividade do processo escolhida; Se nenhum serviço que atenda as restrições de QoS for encontrado, o sistema deve informar o fato ao usuário e sugerir que o mesmo relaxe esses critérios para poder efetuar uma nova busca ou se desejar, correr o risco de efetuar essa busca na fase de execução.

Quadro 48- Caso de Teste – Invocar a Descoberta de Serviços e vinculá-los às atividades.

A análise do quarto caso de teste (mostrado no Quadro 48) aponta que o resultado da invocação da descoberta de serviços e da associação destes às atividades do processo é compatível com o esperado. O Ambiente de Descoberta de Serviços foi invocado com sucesso, assim como a busca nos repositórios de serviços. Os serviços retornados pela

busca atendem aos critérios de QoS necessários, bem como a classificação ontológica do serviço.

Caso de Teste 5 – Exportar o Processo para a Linguagem WS-BPEL	
Descrição	Esse caso de teste verifica se o procedimento de exportação do processo para a linguagem WS-BPEL está funcionando de maneira adequada.
Precondições	O editor BPM está executando no computador do cliente; Existe um processo já construído no editor BPM, com as restrições de QoS e serviços vinculados e que será usado na exportação para WS-BPEL; O Catálogo de Processos de Negócio está *online* e acessível pelo *plug-in*.
Entradas	**Nome do Projeto de Origem** – o nome do projeto onde o processo que será exportado se encontra; **Nome do Processo** – nome do processo que será exportado; **Nome da Pasta** – nome da pasta onde os arquivos WS-BPEL da exportação serão gravados.
Resultado Esperado	Após a invocação pelo usuário, a tela de exportação de processos para WS-BPEL deve ser apresentada; O usuário escolhe o projeto onde o processo se encontra. A lista dos processos contidos no projeto deve se apresentada; O usuário escolhe o processo que deseja exportar; O usuário escolhe a pasta onde o processo WS-BPEL será salvo; O sistema deve converter o processo escolhido para o formato WS-BPEL e armazená-lo na pasta escolhida pelo usuário; O processo deve poder ser executado no Ambiente de Execução de Processos.

Quadro 49 - o Processo para a Linguagem WS-BPEL.

A análise do quinto caso de teste (mostrado no Quadro 49) aponta que o resultado da exportação do processo para WS-BPEL é compatível, na medida do possível, com o esperado. Os processos exportados e executados foram:

- Processo de Criação de Catálogo (*Create Catalogue Process*);
- Processo de Ordem (*Ordering Process*);
- Processo de Pagamento (*Payment Process*).

Com base na execução dos casos de testes acima, conclui-se que o conector de integração com o catálogo, implementado pelo protótipo, atende aos requisitos estabelecidos no modelo conceitual.

E.2 Catálogo de Processos de Negócios

Para testar o Catálogo de Processos de Negócio, utilizou-se um teste de unidade desenvolvido com o *framework JUnit* (JUNIT, 2010). O *JUnit* é um conjunto de classes Java utilizadas para gerar um ambiente de testes automatizado. Cada teste é implementado como um objeto, sendo que o mecanismo de execução de testes do *JUnit* os executa. O teste deve ser escrito de uma forma que indique se o sistema testado está se comportando de maneira esperada (SOMMERVILLE, 2007).

O teste desenvolvido é do tipo caixa preta é um teste que não se preocupa com detalhes internos de implementação do componente, mas somente com a *interface* de acesso que o mesmo disponibiliza (O'DOCHERTY, 2005). Dessa forma, atesta-se a corretude do módulo do catálogo analisando-se as entradas e as saídas desejadas. Com base nisso, utilizou-se a *interface* CatalogService para realizar os testes.

Com base nos testes efetuados em Bezerra (2011), conclui-se que o catálogo de processos de negócios desempenha suas funcionalidades conforme especificado.

E.3 Federação de Provedores

Conforme comentado e descrito na Seção 4.2, o módulo da Federação é formado de diversos submódulos. Com exceção dos serviços UBL, os demais submódulos usaram na sua implementação produtos prontos já devidamente testados, como é o caso dos repositórios de serviços, instanciados na forma de jUDDIs.

Portanto, somente os serviços UBL foram testados. Desta forma, utilizou-se a ferramenta *SoapUI* (EVIWARE, 2010), especializada no teste de serviços *web*. Essa ferramenta possui um analisador WS-I *Basic Profile*, que permite verificar a conformidade dos descritores WSDL dos serviços. Essa análise é importante, pois atesta que o serviço é interoperável, podendo ser invocado por clientes heterogêneos.

A WS-I (*Web Services Interoperability Organization*) é uma organização composta por diversas empresas, tais como Microsoft, Oracle, IBM, etc. responsável por promover a interoperabilidade entre os diversos serviços *web* heterogêneos presentes na Internet (WS-I, 2010). Uma de suas iniciativas é o *Basic Profile* que contém uma lista dos padrões básicos dos serviços *web*, que inclui o SOAP, WSDL, UDDI, XML e XML *Schema*, juntamente com direcionamentos, recomendações e esclarecimentos sobre esses padrões de forma a promover uma maior interoperabilidade.

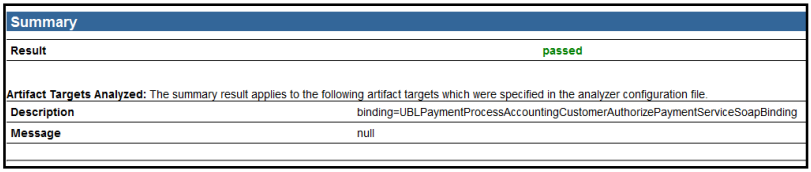


Figura 80 – Resultado do teste WS-I *Basic Profile*.

A Figura 80 apresenta o resultado do teste WS-I *Basic Profile* para o serviço *AuthorizePayment* do processo *Payment Process*. Isso atesta que o serviço é interoperável, no que diz respeito à troca de mensagens e a descrição do WSDL. O Quadro 50 apresenta todos os serviços UBL que foram testados.

Nome do Serviço	Resultado
Create Catalogue Process	
ReceiverParty/RequestCatalogue	Aprovado
ProviderParty/RespondToRequest	Aprovado
ProviderParty/ProcessCatalogueRequest	Aprovado
ProviderParty/SendRejection	Aprovado
ReceiverParty/ReceiveRejection	Aprovado
ProviderParty/SendAcceptanceResponse	Aprovado
ProviderParty/PrepareCatalogueInformation	Aprovado
ProviderParty/ProduceCatalogue	Aprovado
ProviderParty/DistributeCatalogue	Aprovado
ReceiverParty/ReceiveCatalogue	Aprovado
ReceiverParty/ReviewCatalogueContent	Aprovado
ReceiverParty/AcknowledgeAcceptance	Aprovado
ProviderParty/ReceiveAcknowledgeAcceptance	Aprovado
ReceiverParty/AcceptCatalogue	Aprovado
ReceiverParty/QueryCatalogueContent	Aprovado
ProviderParty/DecideOnAction	Aprovado

Nome do Serviço	Resultado
ProviderParty/CancelTransaction	Aprovado
ProviderParty/ReviseContent	Aprovado
Ordering Process	
BuyerParty/PlaceOrder	Aprovado
SellerParty/ReceiveOrder	Aprovado
SellerParty/ProcessOrder	Aprovado
SellerParty/AcceptOrder	Aprovado
SellerParty/RejectOrder	Aprovado
SellerParty/AddDetail	Aprovado
BuyerParty/ReceiveResponse	Aprovado
BuyerParty/AcceptOrder	Aprovado
BuyerParty/ChangeOrder	Aprovado
BuyerParty/CancelOrder	Aprovado
SellerParty/CancelOrder	Aprovado
SellerParty/ChangeOrder	Aprovado
SellerParty/ProcessOrderChange	Aprovado
Payment Process	
AccountingCustomer/AuthorizePayment	Aprovado
AccountingCustomer/NotifyOfPayment	Aprovado
AccountingSupplier/NotifyPayee	Aprovado
AccountingSupplier/ReceiveAdvice	Aprovado
PayeeParty/ReceiveAdvice	Aprovado

Quadro 50 - Testes Realizados nos Serviços UBL.

O resultado dos testes mostra que os serviços UBL implementados estão em conformidade com o WS-I *Basic Profile*. Assim, os serviços UBL utilizados no protótipo computacional são acessíveis e se comunicam de forma interoperável, como atesta o resultado desse teste.

E.4 Ambiente de Execução de Aplicações

Para se fazer a verificação do Ambiente de Execução de Aplicações aplicou-se a mesma estratégia utilizada na verificação do conector de integração. Essa escolha deve-se ao fato de, assim como o conector, o ambiente de execução também é voltado à interação com o usuário, de forma que o teste baseado em casos de uso mostra-se mais apropriado para a verificação.

Os casos de uso utilizados como base para os casos de teste foram, conforme apresenta a Seção 4.2.3.5 que descreve conceitualmente o funcionamento do ambiente de execução de aplicações:

- Acessar a Lista dos Processos Executáveis Disponíveis;
- Executar Processos;
- Monitorar o Andamento dos Processos.

O Quadro 51 mostra o ambiente onde os testes foram executados.

Local/Ambiente	Laboratório de Tecnologia de Informação e Comunicação do Departamento de Automação e Sistemas da UFSC
Computador	• Processador: Intel Centrino Duo Core de 1.66GHz • Microsoft Windows XP Professional Service Pack 3 • Memória RAM de 2GB • Conexão com a Internet
Observações	Ferramenta de Gerenciamento de Processos: • Intalio BPMS 6.0.3.010.01; o Motor de Execução Apache ODE O Ambiente de Descoberta de Serviços foi executado na mesma máquina, com cinco repositórios locais de serviços.

Quadro 51 - Ambiente de Testes – Execução de Aplicações.

O cenário de testes foi executado num único computador. A ferramenta *Intalio* BPMS é responsável pelo gerenciamento e execução dos processos. O Ambiente de Descoberta de Serviços pode ser invocado no momento da execução do processo, caso não existam serviços previamente atribuídos às atividades desse processo.

Caso de Teste 1 – Acessar a Lista de Processos Executáveis Disponíveis	
Descrição	Esse caso de teste verifica se os processos exportados para o Ambiente de Execução estão corretamente listados.
Precondições	Existe pelo menos um processo disponível no Ambiente de Execução; O Usuário Executor de Processos está autenticado dentro do Ambiente de Execução.
Resultado Esperado	Após estar autenticado no sistema, a lista dos processos que foram exportados para o Ambiente de Execução é apresentada; O usuário tem a opção de executar algum dos processos listados

Quadro 52 - Caso de Teste -Acessar a Lista de Processos Executáveis Disponíveis.

A análise do primeiro caso de teste (mostrado no Quadro 53) aponta que o resultado da listagem dos processos executáveis disponíveis é compatível com o esperado.

Caso de Teste 2 – Executar Aplicações	
Descrição	Esse caso de teste verifica se os processos exportados para o Ambiente de Execução estão sendo executados corretamente
Precondições	Pelo menos um processo foi exportado para o Ambiente de Execução; O Usuário Executor de Processos está autenticado dentro do Ambiente de Execução; O Ambiente de Descoberta de Serviços está disponível; Os serviços UBL necessários para a execução do processo estão disponíveis.
Entradas	**Nome do Processo** - o nome do processo que se deseja executar
Resultado Esperado	Após estar autenticado no sistema, a lista dos processos que foram exportados para o Ambiente de Execução é apresentada; O usuário escolhe um dos processos e solicita a sua execução; O Ambiente de Execução deve iniciar a execução do processo; A verificação dos serviços pré-atrelados às atividades do processo é iniciada. Caso nenhum serviço esteja disponível, o Ambiente de Descoberta é invocado, a fim de

	encontrar serviços candidatos; Se não existir pelo menos um serviço funcional atrelado a cada atividade, o processo é abortado; Caso contrário o processo inicia a invocação aos serviços UBL; Cada serviço UBL invocado exibe uma mensagem de confirmação no computador que o hospeda. Essa mensagem precisa ser aceita para que o processo possa continuar (simulação de interação humana no processo); O processo fica ocioso esperando a resposta dos serviços invocados; Quando todos os serviços invocados responderem, o processo é finalizado; O Ambiente de Execução marca a execução dessa instância do processo como "concluída com sucesso".

Quadro 53 - Caso de Teste – Executar Aplicações.

A análise do segundo caso de teste (mostrado no Quadro 53) aponta que o resultado da execução de aplicações é compatível com o esperado.

Caso de Teste 3 – Monitorar o Andamento das Aplicações	
Descrição	Esse caso de teste verifica se a funcionalidade de monitoramento do andamento dos processos está funcionando corretamente.
Precondições	Existe pelo menos um processo disponível no Ambiente de Execução de Processos; O Usuário Executor de Processos está autenticado dentro do Ambiente de Execução.
Resultado Esperado	Após estar autenticado no sistema, a lista dos processos que foram exportados para o Ambiente de Execução é apresentada; A listagem apresenta algumas informações, como o número de instâncias do processo que estão em execução, concluídas e que falharam; Selecionando uma instancia de algum processo, o usuário deve ter acesso a sua situação atual dele. Como por exemplo: • Em qual ponto do processo ele está parado; • Quais foram às atividades que já foram executadas; • As mensagens trocadas entre o processo e os serviços invocados

Quadro 54 - Caso de Teste – Monitorar o Andamento das Aplicações.

A análise do terceiro caso de teste (conforme mostra o Quadro 54) aponta que o resultado da funcionalidade de monitoramento do andamento dos processos é compatível com o esperado.

Com base na execução dos casos de testes acima, conclui-se que o Ambiente de Execução de Aplicações atende aos requisitos estabelecidos no modelo conceitual.

E.5 Ambiente de Descoberta de Serviços

De forma semelhante ao teste efetuado com os serviços UBL. Usou-se a ferramenta *SoapUI* (EVIWARE, 2010), especializada no teste de serviços *web*. O analisador WS-I *Basic Profile* da ferramenta verificou a conformidade dos descritores WSDL dos serviços (*crawling* e descoberta dinâmica), permitindo observar que ambos os serviços são interoperáveis, podendo ser invocados por clientes heterogêneos.

Para testar o mecanismo de descoberta de serviços, uma série de experimentos foram conduzidos. A Seção 6.1 apresenta e descreve os experimentos realizados.