Roque Oliveira Bezerra

Proposta de Catálogo Eletrônico de Processos de Negócio Baseados em UBL para Composição de Aplicações SOA

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

Florianópolis

2011

Catalogação na fonte pela Biblioteca Universitária
da
Universidade Federal de Santa Catarina

B574p Bezerra, Roque Oliveira
Proposta de catálogo eletrônico de processos de negócio baseados em UBL para composição de aplicações SOA [dissertação] / Roque Oliveira Bezerra ; orientador, Ricardo José Rabelo. - Florianópolis, SC, 2011.
267 p.: il., grafs., tabs.

Dissertação (mestrado) - 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. 3. Catálogos. 4. Arquitetura orientada a serviços. 5. Gerenciamento de processos de negócio. 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)

Roque Oliveira Bezerra

Proposta de Catálogo Eletrônico de Processos de Negócio Baseados em UBL para Composição de Aplicações SOA

Esta Dissertação foi julgada adequada para obtenção do Título de "Mestre em Engenharia de Automação e Sistemas" e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina

Florianópolis, 28 de fevereiro de 2011.

_____
Prof. Ricardo José Rabelo, Dr.
Orientador

_____
Prof. José Eduardo Ribeiro Cury, Dr.
Coordenador do Programa de Pós-Graduação em Engenharia de Automação e Sistemas

Banca Examinadora:

_____	_____
Prof. Ricardo José Rabelo, Dr.	Prof. Carlos Barros Montez, Dr.
Presidente	Membro
_____	_____
Prof. Frank Augusto Siqueira, Dr.	Profª. Flávia Maria Santoro, Dra.
Membro	Membro

À minha mãe, Dilma

AGRADECIMENTOS

Agradeço a Deus por ter me dado forças para continuar meu trabalho e iluminado meu caminho.

A minha família por ter me dado todo o apoio e as condições necessárias ao meu crescimento pessoal e profissional. Em especial, a minha mãe Dilma por ter desde pequeno me tratado com muito amor e carinho e me dado uma boa educação, com valores indispensáveis como o caráter, a perseverança, e os ensinamentos de vida.

A minha namorada Vivian, que sempre esteve do meu lado, apoiando-me, e por ser a pesosa maravilhosa que é. Agradeço a sua compreensão nos momentos em que precisei me dedicar ao trabalho.

Ao meu orientador, Ricardo José Rabelo, pelo seu zelo e dedicação com que orientou este trabalho, com vistas a obter um excelente resultado. Comportamentos como o seu engrandecem a profissão de Professor.

A todos os professores de Programa de Pós-Graduação em Engenharia de Automação e Sistemas, que de forma direta ou indireta me deram apoio desde o princípio, especialmente aqueles com quem tive contato direto, José Eduardo Ribeiro Cury, Max Hering de Queiroz, Joni da Silva Fraga, Marcelo Ricardo Stemmer, Jomi Fred Hübner, Werner Kraus Junior e Rômulo Silva de Oliveira.

Aos colegas do GSIGMA, Fabiano Baldo, Rui Jorge Tramontin Junior, Saulo Popov Zambiazi, Omir Alves,

Alexandre Perin de Souza, Daniel Mayer e Maiara Heil Cancian. Todos eles, de alguma forma, contribuíram para quer esta dissertação fosse concluída.

Aos meus colegas de trabalho, da Superintendência de Governança Eletrônica e Tecnologia da Informação e Comunicação da UFSC, em especial ao José Marcos da Silva, meu chefe, e ao Marcio Clemes, superintendente da unidade, por terem sempre me apoiado e me dado as condições necessárias para que eu pudesse assistir às aulas e me dedicar ao desenvolvimento do trabalho. Agradeço também aos meus colegas de sala, em especial ao Richard Henrique de Souza, por ter me apoiado e repassado, em partes, a sua experiência acadêmica.

Aos Membros da Banca Examinadora desta Dissertação, que têm todos os méritos inerentes à avaliação crítica e intervenção construtiva para a melhoria do documento final.

“Verás que um filho teu não foge à luta”
(Hino Nacional Brasileiro)

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

PROPOSTA DE CATÁLOGO ELETRÔNICO DE PROCESSOS DE NEGÓCIO BASEADOS EM UBL PARA COMPOSIÇÃO DE APLICAÇÕES SOA

Roque Oliveira Bezerra

Fevereiro, 2011

Orientador: Prof. Ricardo José Rabelo, Dr.
Área de Concentração: Sistemas Computacionais.
Palavras-chave: Gerenciamento de Processos de Negócio, Arquitetura Orientada a Serviços, Catálogo de Processos de Negócio.
Número de Páginas: 267

RESUMO: Cada vez mais as empresas têm enfrentado um ambiente de intensa concorrência e mudanças de mercado. Isso faz com que elas tenham que adaptar rapidamente seus processos de negócio, refletindo essas alterações nos seus sistemas computacionais. No lado da modelagem de processos de negócio, o uso de *Business Process Management* (BPM) permite fazer o gerenciamento de processos utilizando uma visão horizontal, que envolve sistemas, pessoas, departamentos etc. No lado da implementação, a arquitetura orientada a serviços (SOA) permite implementar esses processos através do uso de serviços *web*, que são projetados para serem reutilizáveis e interoperáveis. A integração entre BPM e SOA vem sendo defendida como uma forma de lidar com as crescentes mudanças nos requisitos de negócios, adaptando-se mais rapidamente a elas e permitindo que os sistemas computacionais acompanhem essas mudanças. Esta integração, porém, carece de agilidade, pois a transição entre a modelagem do processo de negócio, no nível BPM, e a sua implementação, no nível SOA, é feita de forma manual ou no máximo semi-automatizada. Isso impede que as empresas sejam mais ágeis na implementação dos seus processos. Este trabalho apresenta uma pesquisa acerca de como aumentar essa agilidade na integração entre BPM e SOA, propondo uma arquitetura

baseada num catálogo de processos de negócio, padronizado segundo a especificação *Universal Business Language* (UBL). Este catálogo é integrado a uma ferramenta de modelagem de processos de negócio de forma a permitir que o usuário-projetista possa compor suas aplicações, utilizando como base processos padronizados presentes no catálogo. A arquitetura permite buscar e vincular serviços *web* às atividades do processo, e também exportar esses processos para um ambiente que permita a sua posterior execução, a nível computacional. A proposta é fortemente baseada em padrões, de forma a mitigar questões de interoperabilidade e aproveitar as melhores práticas atualmente existentes.

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements of the degree of Master in Automation and Systems Engineering.

BUSINESS PROCESS ELECTRONIC CATALOGUE PROPOSAL BASED ON UBL FOR SOA APPLICATION COMPOSITION

Roque Oliveira Bezerra

February, 2011

Advisor: Prof. Ricardo José Rabelo, Dr.
Area of Concentration: Computer Systems.
Keywords: Business Process Management, Service Oriented Architecture, Business Process Catalogue.
Number of Pages: 267

ABSTRACT: Nowadays, organizations have to deal with a more competitive and changing market. So, they have to quickly adapt their business processes, reflecting these changes on their information systems. In the business modeling tier, the use of Business Process Management (BPM) enables the process management using a horizontal view that involves systems, people, departments etc. In the implementation tier, the Service Oriented Architecture (SOA) lets the implementation of these processes using web-services, which are projected to be reusable and interoperable. The BPM and SOA integration is being defended as a way to deal with the increasing change of business requisites, adapting faster to them and letting the computer systems accomplish these changes. This integration, however, has a lack of agility, because the transition between business process modeling, in BPM tier, and its implementation, in the SOA tier, occours in a manual or semi-manual way. This prevents companies to be more agile in the implementation of these processes. This work presents a research about how to increase agility of integration between BPM and SOA, proposing an architecture based on a business process catalogue that uses the Universal Business Language (UBL) specification. This catalogue is integrated with a business modeling tool, allowing application designers to compose their applications basing on

standardized processes contained in the catalogue. The architecture allows to search and bind web services with process tasks and to export these processes to an environment that lets their subsequent execution in a computational level. The proposal is strongly based on standards, as a way of mitigate interoperability issues and reusing the well-known best practices.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANSI – American National Standards Institute

API – Application Programming Interface

B2B – Business to Business

BD – Banco de Dados

BPEL – Business Process Execution Language

BPM – Business process management

BPMN – Business Process Modeling Notation

BPMO – Business Process Management Ontology

BPMS – Business Process Management Suite

CEO – Chief Executive Officer

CGI – Common Gateway Interface

CIO – Chief Information Officer

CORBA – Common Object Request Broker Architecture

DAS – Departamento de Automação e Sistemas

DTD – Document Type Definition

EJB – Enterprise Java Bean

EMF – Eclipse Modeling Framework

ERP – Enterprise Resource Planning

GSIGMA – Grupo de Sistemas Inteligentes de Manufatura

HTTP – Hypertext Transfer Protocol

IDE – Integrated Development Environment

IP – Internet Protocol

ISO – International Organization for Standardization

KPI – Key Performance Indicators

MIT – Massachusetts Institute of Technology

OASIS – Organization for the Advancement of Structured Information Standards

OMG – Object Management Group

PHP – Hypertext Preprocessor

PIP – Partner Interface Process

PPGEAS – Programa de Pós-Graduação em Engenharia de Automação e Sistemas

QoS – Quality of Service

RMI – Remote Method Invocation

RNIF – RosettaNet Implementation Framework

RPC – Remote Procedure Call

sBPEL – Semantic Business Process Execution Language

sBPMN – Semantic Business Process Modeling Notation

SCA – Service Component Architecture

SCDL – Service Component Definition Language

SOA – Service Oriented Architecture

SOAP – Simple Object Access Protocol

SWT – Standard Widget Toolkit

TCP – Transmission Control Protocol

TI – Tecnologia da Informação

UBL – Universal Business Language

UDDI – Universal Description, Discovery and Integration

UFSC – Universidade Federal de Santa Catarina

UI – User Interface

UN/CEFACT – United Nations Centre for Trade Facilitation and Electronic Business

CCTS – Core Component Technical Specification

UML – Unified Modeling Language

W3C – World Wide Web Consortium

WBM – Websphere Business Modeler

WSA – Web Services Architecture

WSDL – Web Services Description Language

WSML – Web Services Modeling Language

XML – Extensible Markup Language

XSD – XML Schema

SUMÁRIO

Capítulo 1

Introdução

1.1 Apresentação

Profundas transformações vêm ocorrendo na vida das pessoas e das organizações. Alguns dos aspectos responsáveis por isso são a convergência da base tecnológica, a dinâmica da indústria de tecnologia e o crescimento da internet. O primeiro aspecto diz respeito à representação da informação de uma única forma: a digital. O segundo aspecto leva em conta a capacidade das indústrias de criar máquinas cada vez mais poderosas e com preços menores, de forma a ficarem acessíveis a um público cada vez maior. Em terceiro lugar, o crescimento da internet, que de tão importante vem-se tornando um fator estratégico no desenvolvimento das nações. Esses fatores, quando inter-relacionados, criam um fenômeno global com potencial para transformar as atividades sociais, financeiras e de negócios. Exemplo disso são a competição acirrada nas empresas, o aumento dos custos operacionais e a necessidade cada vez maior de se adaptar às mudanças nos requisitos do negócio (TAKAHASHI, 2000).

Dentro desse contexto, o gerenciamento adequado dos processos de negócio tornou-se bastante importante (BALDAM *et al.*, 2007). Alguns dos fatores que corroboram com essa afirmação são (SIMCHI-LEVI *et al.*, 2003): i) a necessidade de transferência rápida de informações e tomada de decisão; ii) a necessidade de se adaptar rapidamente às mudanças e iii) a maior competição internacional. Tais fatores, se não tratados corretamente, podem trazer grandes prejuízos na rentabilidade e na própria sobrevivência das empresas. Portanto, estas

empresas precisam se adaptar a um ambiente cada vez mais ágil e dinâmico (SIMCHI-LEVI *et al.*, 2003).

Em um ambiente onde as mudanças são cada vez mais rápidas e profundas, uma série de barreiras surge. Nesse sentido, a empresa deve se perguntar (HURBEAN, 2007):

- Existe pessoal capacitado para ajustar os processos de negócio na velocidade exigida pelo mercado?
- Em termos de tecnologia da informação, tem-se como garantir que essas mudanças nos processos sejam implementadas adequadamente ao nível de *software?*
- Essas mudanças não impactarão negativamente nos demais sistemas de informação empresariais?
- Qual o custo e quais tecnologias necessárias para isso?

Como forma de respondê-las, torna-se cada vez mais necessário o alinhamento entre processos de negócio e a TI. O sucesso da empresa está cada vez mais dependente da habilidade de atualizar, integrar, personalizar e implantar aplicações rapidamente, de forma a fornecer condições para que seus clientes tenham um acesso rápido às informações (HURBEAN, 2007).

Uma das iniciativas criadas para tentar lidar com esses problemas é fazer o gerenciamento dos processos empresarias através do uso de sistemas computacionais. A essa iniciativa deu-se o nome de *Business Process Management* (BPM) (KO *et al.*, 2009). BPM proporciona uma visão organizacional em que a ênfase está na execução de tarefas através de um fluxo horizontal, que envolve sistemas, pessoas, departamentos etc. Isto contrasta com a visão clássica, que descreve o trabalho da empresa através de uma divisão vertical. O realce na visão horizontal envolve o gerenciamento de recursos alocados 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 *et al.*, 2008).

Para serem efetivas, as empresas devem gerenciar rigorosamente as mudanças nos processos de negócio. A integração dos processos é uma forma de lidar com a crescente complexidade dos sistemas de informação de uma forma economicamente viável (HURBEAN, 2007).

No que diz respeito a essa integração, a prática tem mostrado que aplicações computacionais são concebidas para cumprir seus requisitos de forma individual, sem se preocupar com as demais

necessidades da organização, ou mesmo com processos que envolvem diversos setores da mesma. Uma empresa, por exemplo, pode ter sistemas que lidam com as finanças, contabilidade, recursos humanos, vendas e outros, sem que haja nenhuma associação entre eles. Isso leva a problemas como a redundância da informação e outros erros que surgem de uma abordagem de desenvolvimento que não leva em conta questões de integração (HURBEAN, 2007).

Se no lado da modelagem do processo, o uso de BPM tem sido cada vez mais comum, na parte de implementação dos processos e da integração de sistemas, ao nível de TI, a arquitetura SOA *(Service Oriented Architecture*) vem se tornando uma tendência (DORN *et al.*, 2007). SOA é uma filosofia de desenvolvimento de sistemas distribuídos onde os componentes essenciais são serviços, que podem ser executados em computadores diferentes a partir de provedores de serviços distintos (SOMMERVILLE, 2006). Esses serviços ficam disponíveis na web, podendo ser acessados por sistemas que utilizem suas funcionalidades. A *The World Wide Web Consortium* (W3C) define um serviço *web* ou *web service* como um sistema projetado para suportar interação entre máquinas presentes numa rede de computadores. As funcionalidades que eles fornecem são descritas através de uma interface, num formato processável por máquinas. A comunicação entre esses serviços ocorre utilizando mensagens, tipicamente definidas pelo protocolo SOAP e sendo transportadas pelo protocolo *Hypertext Transfer Protocol* (HTTP).

Dentro desse contexto, torna-se cada vez mais necessário a integração dos processos de negócio com as tecnologias de informação. Nesse sentido, a combinação de BPM e SOA tem sido defendida como uma poderosa abordagem para permitir um alinhamento conciso entre os processos empresariais e os recursos tecnológicos (BPMINSTITUTE, 2006). O uso de BPM permite que os analistas de negócio modelem os processos da empresa enquanto que SOA age ligando-os ao nível de TI.

Segundo (KAMOUN, 2007), a combinação de BPM & SOA é sinergética e permite automatizar e otimizar processos de negócio através do uso de componentes reusáveis (serviços *web*), de forma a integrar sistemas complexos e heterogêneos. Isso permite que as empresas tenham maior agilidade e capacidade em responder melhor às mudanças nos seus requisitos de negócios.

Nessa combinação, BPM permite modelar o negócio em termos de atividades, que envolvem vários sistemas da empresa. Os conjuntos dessas atividades formam os processos de negócio, que são

representados em um formato processável por máquinas, o que permite que eles sejam executados. SOA age implementando as atividades como serviços, de forma que diferentes processos e aplicações possam compartilhá-los. Isso é possível, pois esses serviços possuem uma alta interoperabilidade e baixo acoplamento, que são advindos do próprio fundamento tecnológico dos serviços *web* (KAMOUN, 2007).

Segundo (NOEL, 2005), a junção BPM & SOA permite que as empresas possam se tornar mais competitivas, adaptando-se melhor às mudanças de mercados, melhorando a eficiência e a colaboração entre os departamentos da empresa, que tradicionalmente trabalhavam de forma isolada.

1.2 Problema de Pesquisa

Apesar de todo o avanço que o uso de BPM&SOA proporcionou, ainda existe uma lacuna entre a modelagem do processo e a sua implementação, através de serviços. Por mais que os processos sejam bem descritos e modelados, a vinculação deles com serviços no ambiente SOA ainda é feito de forma manual ou no máximo semi-automatizada, o que impede a empresa de ser mais ágil e flexível (ADAM *et al.*, 2008).

Hepp (2005) afirma, por exemplo, que a falta de semântica associada aos serviços e ao processo impede que a vinculação dos serviços seja feita de maneira mais eficiente. Por outro lado, Adam *et al.* (2005) acredita que a falta de sinergia entre os analistas de negócios, responsáveis pelos processos, e os analistas de TI, que criam os serviços, faz com que seja difícil fazer essa vinculação. Um serviço, por exemplo, pode ser excessivamente técnico (inserir um registro no banco de dados, por exemplo) a ponto de não ser utilizável por um analista de negócios, que não possui um conhecimento profundo de TI. Também pode ocorrer de o serviço ser tão específico a uma função, que não pode ser reutilizado em outros contextos. O gerenciamento do portfólio de serviços e a correta granularidade deles em termos de funcionalidade são aspectos que devem ser levados em conta num ambiente BPM & SOA.

Outro problema é que geralmente os processos não são modelados utilizando especificações padronizadas de processos de negócio, tais como UBL (*Universal Business Language*) (OASIS, 2006b) ou RosettaNet (ROSETTANET, 2008). O uso de tais padrões poderia reduzir problemas de interoperabilidade e aumentar o reuso de processos já existentes (NURMILAAKSO, 2007).

Com base no exposto acima, esse trabalho pretende contribuir para minimizar os seguintes problemas, que afetam uma melhor integração entre BPM e SOA:

- A falta de semântica associada aos processos, no nível de BPM e aos serviços, em nível de SOA (HEPP *et al.*, 2005);
- A falta de sinergia entre os envolvidos na modelagem dos processos de negócio e nos envolvidos na implementação dos serviços (ADAM *et al.*, 2008);
- O não uso de especificações padronizadas de processos de negócio (NURMILAAKSO, 2007).

1.3 Pergunta de Pesquisa

Esta dissertação visa conduzir um experimento exploratório de pesquisa que gere novos conhecimentos e que contribua para responder a seguinte pergunta:

O uso de um catálogo com processos de negócio padronizados pode agilizar o projeto de soluções SOA para melhor se integrarem ao nível BPM?

Nesta pergunta de pesquisa, o catálogo é um repositório que servirá como elo de ligação entre o analista de negócio (usuário-projetista, pessoa responsável por modelar o processo), e o ambiente SOA, que possui os serviços que implementam o processo. Esse catálogo possui processos de negócio padronizados segundo uma especificação de processos de negócio.

Por agilizar entende-se fazer uma integração mais transparente, interoperável e eficiente entre os processos de negócio e os serviços que os implementam, simplificando a vinculação das atividades do processo aos serviços *web*. Pretende-se dessa forma, fazer essa vinculação de uma forma transparente ao usuário, abstendo o mesmo de ter que se preocupar com questões técnicas, como por exemplo, a localização do serviço, suas operações etc.

Acredita-se que o uso do catálogo possui o potencial de diminuir o tempo necessário entre o projeto do processo de negócio e a sua efetiva implementação através de serviços *web*.

1.4 Hipótese de Pesquisa

Com base na pergunta de pesquisa, a hipótese de pesquisa é apresentada abaixo:

Pergunta	O uso de um catálogo padronizado de processos de negócio pode agilizar o projeto de soluções SOA para melhor se integrarem ao nível BPM?
Hipótese	A não existência de algo como um catálogo padronizado de processos tira muito das vantagens potenciais de uma ágil integração BPM&SOA. Portanto, um catálogo com processos padronizados e de fácil acesso pelo usuário-projetista BPM pode dar maior qualidade (em termos de aderência ao processo e confiabilidade) e rapidez (em termos temporais) no processo de composição de aplicações SOA e integração BPM-SOA

Considerando-se o método hipotético-dedutivo a ser aplicado nesta dissertação, 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 catálogos de processos de negócio (MENEZES *et al.*, 2005).

Mais concretamente, visa observar em que medida o uso de um catálogo de processos e o forte uso de padrões pode agilizar a integração entre os níveis BPM e SOA. Este modelo de integração é tido como a principal contribuição científica desta dissertação e, acredita-se, tem inclusive elementos de ineditismo em relação ao estado da arte.

1.5 Objetivo Geral

O objetivo desse trabalho é desenvolver um catálogo de processos de negócio, baseado numa instância de Modelo de Referência de Processos de Negócio – a especificação UBL – que auxilie na concepção de aplicações SOA, tornando o processo mais ágil.

Este catálogo será integrado a um editor de processos de negócio e permitirá ao usuário importar e editar processos, utilizando o padrão de linguagem *Business Process Modeling Language* (BPMN). Também permitirá vincular serviços *web* que implementem as

atividades dos processos e exportar esses processos para um ambiente onde os mesmos possam ser executados.

O papel do catálogo de processos de negócio, na união BPM&SOA, é constituir uma base de processos prontos para serem usados. O projetista poderá compor aplicações SOA usando como base alguma especificação presente no catálogo, ou customizar sua aplicação SOA usando como inspiração um modelo presente no catálogo.

1.6 Objetivos Específicos

Embasados no principal objetivo deste trabalho, a seguir são descritos os objetivos específicos:

- Concepção de um catálogo de processos de negócio, que será utilizado para a armazenagem de processos no padrão UBL;
- Criação de uma ontologia de processos no padrão UBL, de forma a permitir que os processos contidos no catálogo possam ser categorizados segundo esta e permitir a vinculação destes processos com os serviços *web* correspondentes;
- Integração do catálogo a um ambiente de edição de processos de negócio;
- Integração do catálogo a um ambiente de descoberta dinâmica de serviços;
- Criação de 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;
- Construção de um ambiente de execução de aplicações, de forma a permitir que os processos exportados possam ser executados e acompanhados.

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, dentro da linha de integração de sistemas.

Este trabalho se enquadra dentro de um trabalho mais amplo, de uma tese de doutorado do PPGEAS, que visa construir um modelo inteligente de descoberta dinâmica de serviços de software.

1.8 Enquadramento Metodológico

O propósito do enquadramento metodológico está em conhecer as características e métodos necessários para condução da pesquisa. Desta forma, a pesquisa científica será enquadrada nos seguintes aspectos descritos a seguir (MENEZES *et al.*, 2005).

Classificação quanto à Natureza da Pesquisa

Em relação à Natureza da Pesquisa, este trabalho toma a forma de uma **pesquisa aplicada**, visto que objetiva a geração de conhecimento para aplicações práticas. Para isso serão 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 etc., e alguns modelos e filosofias, tais como BPM e SOA, para o desenvolvimento do modelo.

Classificação quanto aos Objetivos

Quanto aos seus Objetivos, trata-se de uma **pesquisa exploratória**, pois visa proporcionar maior familiaridade com o problema com vistas a torná-lo explícito. Neste 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 uso de catálogos de processos padronizados como elo de integração entre BPM e SOA.

Classificação quanto à Abordagem de Pesquisa

Em relação à Abordagem de Pesquisa, trata-se de uma **pesquisa qualitativa**, cujo objetivo é a criação de um catálogo de processos de negócio e de um modelo de integração. Como principais indicadores qualitativos a serem aplicados na avaliação do trabalho e comprovação da hipótese de pesquisa serão usados os de usabilidade e confiabilidade no processo de uso do catálogo e vinculação entre as atividades do processo de negócio aos serviços *web* que as implementam. Não serão focos de atenção neste trabalho os aspectos relacionados à segurança de sistemas e à tolerância a faltas.

Classificação quanto ao Método de Pesquisa

Quanto ao Método de Pesquisa, este trabalho se caracteriza 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.

Classificação quanto ao Paradigma de Pesquisa

Em relação ao Paradigma de Pesquisa, este trabalho classifica-se como sendo **positivista**, pois a realidade – o contexto do problema – deve ser tangível. Com este tipo de paradigma será buscado identificar as causas que afetam o processo de modelagem e a integração de processos de negócio com o mundo SOA, como, por exemplo, a falta de um ambiente adequado de modelagem, falta de uma integração semântica entre processos de negócio e serviços web. Disso deverá resultar a identificação de, por exemplo, requisitos para tais sistemas, as abordagens mais adequadas de integração BPM&SOA, entre outros.

Classificação quanto aos Procedimentos Técnicos

Do ponto de vista dos procedimentos técnicos, a pesquisa se caracteriza como **pesquisa bibliográfica e experimental**. Ela será

elaborada em função de materiais bibliográficos já publicados, de padrões existentes associados aos domínios de BPM e SOA, e de ferramentas de software já disponíveis. Serão realizados experimentos envolvendo os vários aspectos presentes na modelagem de processos e na integração destes com serviços *web*, visando avaliar o protótipo de catálogo de processos de negócio no padrão UBL.

Classificação quanto ao Tipo de Observação

Na dimensão Tipo de Observação da pesquisa, esta pesquisa ainda se caracteriza como sendo **sistemática e factual**. Isto decorre do fato de que a coleta de dados será realizada em um ambiente com condições controladas. Os experimentos serão desenvolvidos, conduzidos e analisados a nível de protótipos computacionais, com dados (modelos e normas de processos, serviços etc.) previamente introduzidos – e portanto conhecidos à priori. Ao longo dos experimentos, serão observadas as respostas do protótipo quanto ao problema de modelagem e integração, com uma ênfase voltada principalmente à qualidade e à velocidade da modelagem / integração.

Classificação quanto ao Tempo de Pesquisa

Em relação à dimensão Tempo de Pesquisa, este trabalho tem característica de **estudo longitudinal**. Sendo assim, especificações desejadas para os processos e para os serviços serão introduzidos gradualmente e observados ao longo disto para fins de confirmação ou refutação da hipótese de pesquisa.

1.9 Organização do Documento

Esta dissertação está dividida em seis capítulos. Neste primeiro capítulo, foi descrito o contexto onde este trabalho está inserido, incluindo o seu objetivo e a abordagem proposta.

O segundo capítulo apresenta uma revisão bibliográfica sobre temas relacionados ao gerenciamento de processos de negócio e tecnologias relacionadas à arquitetura orientada a serviços. Também é apresentada uma revisão do estado da arte acerca de catálogos de processos de negócio. Este estudo traz embasamento para que se possa construir um modelo conceitual.

O terceiro capítulo apresenta o modelo conceitual proposto. A arquitetura do catálogo é descrita em termos de módulos e de seus relacionamentos. As funcionalidades necessárias ao modelo são explicitamente detalhadas.

O quarto capítulo detalha a implementação do protótipo computacional desenvolvido.

O quinto capítulo dedica-se a fazer a verificação, avaliação e análise dos resultados do modelo conceitual e do protótipo implementado.

No sexto capítulo são apresentadas as conclusões deste trabalho e as perspectivas para futuros melhoramentos.

Capítulo 2

Revisão Bibliográfica

Esta seção apresenta a revisão bibliográfica referente aos temas fundamentais para o entendimento do trabalho. Em seguida, são expostas algumas considerações a respeito destes temas e as suas relações com o trabalho.

2.1 Business Process Management (BPM)

Business Process Management (BPM) é um conjunto de metodologias, ferramentas e tecnologias usadas para construir, analisar e controlar processos empresariais. É uma abordagem centrada no processo, que reúne tecnologia da informação (TI) e governança a fim de melhorar o desempenho das organizações. BPM envolve a colaboração das pessoas de negócio e de TI como forma de viabilizar processos eficientes e transparentes (GARIMELLA *et al.*, 2008). BPM também pode ser definido como uma técnica que suporta processos de negócio usando *software* para especificar, controlar, executar e analisar processos empresariais que envolvem pessoas, empresas, aplicações, documentos, e outras fontes de informações (WESKE, 2007).

BPM é baseado na observação de que cada produto que uma organização produz é o resultado da realização de um conjunto de atividades. Nesse sentido, processos de negócio são instrumentos-chave

para entender o funcionamento dessas atividades e de que forma elas se relacionam (WESKE, 2007).

O significado de BPM pode ser melhor compreendido a partir da análise da suas três dimensões: *Business*, *Process* e *Management*, conforme realça a Figura 1 (GARIMELLA *et al.*, 2008).

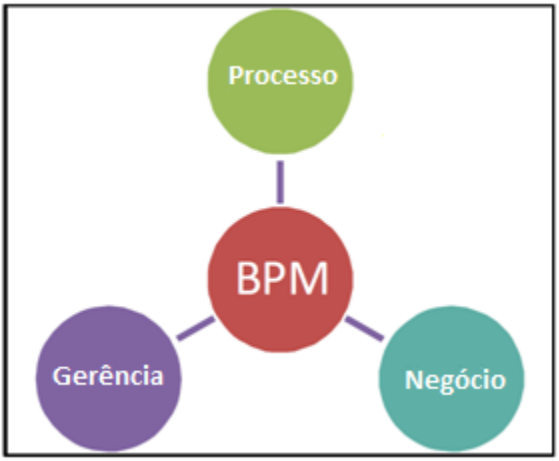


Figura 1 – Dimensões de BPM
Fonte: (GARIMELLA *et al.*, 2008)

- Negócio: algo que interessa ao cliente e que tem um valor agregado para a empresa;
- Processo: transforma entradas em saídas, sendo caracterizado pela necessidade de possuir: (i) agilidade; (ii) transparência e (iii) produtividade.
- Gerenciamento: se aplica ao uso de ferramentas que permitam produzir ou se chegar ao produto desejado;

A dimensão negócio é responsável pela criação de valor, tanto para os clientes como para a empresa. BPM facilita o alcance dos objetivos das empresas, através do aumento da inovação e produtividade. Também permite obter um alinhamento entre as atividades operacionais junto às diretrizes e objetivos estratégicos da empresa.

A dimensão processo é responsável por agregar valor através de atividades estruturadas chamadas de processos. Processos operacionais,

por exemplo, transformam recursos e materiais em produtos e serviços, disponibilizados para os clientes e consumidores da empresa.

A dimensão gerência é a responsável por orquestrar sistemas e pessoas colocando os processos em ação, também para alcançar os objetivos do negócio. Os processos, nesse contexto, são a ferramenta utilizada para se atingir o sucesso no negócio.

Segundo (BALDAM *et al.*, 2007), várias pesquisas vêm apontando BPM como uma tendência. Algumas justificativas para isso são:

- A *"hipercompetição"* – com o advento da globalização, a competição entre empresas extrapolou as fronteiras nacionais. Isso fez com que elas tivessem que se preocupar com a qualidade de seus produtos e a redução de custos, como forma de conquistar clientes cada vez mais sensíveis a esses fatores;
- *Controle de Complexidade* – a pressão por resultados ocorre mesmo em ambientes com pouca competição. Empresas de capital aberto precisam prestar conta a seus acionistas, que exigem lucratividade em seus negócios. Fusões entre empresas ocorrem com frequência, o que leva a um complexo gerenciamento de diferentes culturas, procedimentos, funções etc.;
- *Agilidade no desenvolvimento de produtos* – o tempo de desenvolvimento do produto está diminuindo, bem como o seu ciclo de vida. O foco passa a ser o produto e o cliente, este último querendo produtos cada vez mais customizados. Isso demanda o uso de tecnologias adaptáveis, que possam ter regras e fluxos alterados sem grande envolvimento de recursos;
- *Responsabilidade Social e Governança Corporativa* – há uma cobrança cada vez maior em relação à transparência e ao papel social das organizações. Isso é um comportamento valorizado, inclusive pelo mercado financeiro. Essa iniciativa exige mudanças profundas nos processos, o que corrobora com o uso de BPM.

Neste cenário, (GARIMELLA *et al.*, 2008) ratifica a enorme atenção dada a BPM, uma vez que ela representa o conjunto das melhores ideias e experiências obtidas em gerenciamento de processos de negócio nos últimos anos. Da mesma forma, (WESKE, 2007)

considera BPM o próximo passo após os sistemas de *workflow* dos anos 90. Enquanto sistemas tradicionais de *workflow* têm seu foco principal somente na execução de processos de negócio, BPM amplia o tratamento dado aos processos de negócio agindo, principalmente, na fase de diagnóstico. A Figura 1 ilustra o ciclo de vida BPM e as principais diferenças em relação a sistemas de workflow, na visão (WESKE, 2007).

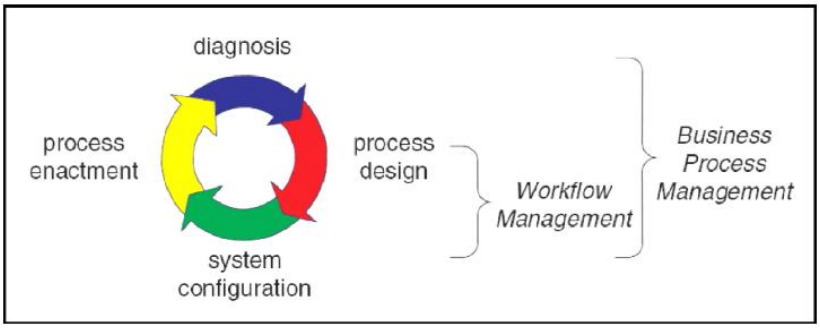


Figura 2 – Diferenças entre BPM e workflow
Fonte: (WESKE, 2007)

- Fase de projeto (*process design*): permite a especificação do processo de negócio, feita em ferramentas de modelagem BPM;
- Configuração (*system configuration*): as informações sobre o processo são inseridas e ajustadas para que o processo possa ser executado;
- Execução do Processo (*process enactment*): os processos de negócio, modelados eletronicamente, são implantados em sistemas computacionais;
- Diagnóstico (*diagnosis*): é feita a análise em função dos problemas e das melhorias que podem ser efetuadas ao processo;

Enquanto que sistemas de *workflow* se preocupam somente com a execução do processo (que é uma fase do ciclo de visa), BPM amplia o escopo, permitindo fazer também o monitoramento e a melhoria contínua do processo.

Outro ponto de vista sobre essa diferença entre BPM e *workflow* é dado pelo Gartner (HILL *et al.*, 2008) que afirma que BPM é

uma disciplina de gerenciamento orientada em processos, e não uma tecnologia. Já *workflow* é uma tecnologia de gerenciamento de fluxo de trabalho que é encontrada em suítes de gerenciamento de processos (BPMS). Isto é, BPM é uma filosofia, enquanto *workflow* é uma tecnologia utilizada para a execução de processos.

O trabalho de (KO, 2009) cita alguns dos benefícios de se adotar BPM junto às empresas:

- Aumenta a visibilidade e o conhecimento das atividades da empresa;
- Aumenta a capacidade de se encontrar gargalos no processo;
- Aumenta a identificação de potenciais áreas que possam ser otimizadas;
  Diminui o *lead time*, isto é, o tempo que decorre entre o início de uma atividade e o seu término;
- Melhora a definições das atribuições dentro da empresa;
- É uma boa ferramenta para prevenção de fraudes e auditoria.

No mesmo sentido, (GARIMELLA *et al.*, 2008) cita alguns preceitos presentes na abordagem BPM:

- **Centrar no Processo** – o foco no processo permite unificar a visão de negócios e TI, através do uso de convenções e notações padrões para a modelagem do processo;
- **Alinhamento entre Negócios e TI** – BPM facilita a colaboração direta entre esses profissionais no desenvolvimento, implementação e otimização de processos. O mesmo modelo de processo provê a visão de negócios para o analista de negócios e a visão de TI para o analista de sistemas;
- **Melhoramento contínuo do processo** – através de KPI's (*key performance indicators*) é possível monitorar e controlar o processo a fim de melhorá-lo continuamente;
- **Composição de soluções** – BPM facilita a elaboração, construção e o *deployment* da aplicação. Um analista de negócios pode vincular sistemas de TI e serviços no mesmo modelo de processo desenvolvido pelo analista de negócios;
- **Transparência** – permite ver quais atividades são atribuídas a cada participante do processo além de permitir que tanto o analista de negócios quanto o analista de sistemas monitorem o processo, cada um sob sua perspectiva;

- **Sistemas Existentes e Camada de Processos** – torna possível o uso de recursos e sistemas existentes, os quais são anexados a uma camada de processo que expõe suas funcionalidades. Dessa forma, eles podem ser usados e coordenados pelos analistas que elaboram o processo.

No contexto deste trabalho, a importância de BPM se dá por ser a camada de mais alto nível usada para projetar aplicações. Especificamente, a camada BPM será responsável por permitir que projetistas especifiquem seus processos de negócio utilizando um ambiente que permita modelá-los graficamente e especificar todos os requisitos necessários a eles.

2.1.1 Padrões Associados

Esta seção apresenta algum dos padrões utilizados no contexto de BPM. Segundo (KO *et al.*, 2009), existe pelo menos dez grandes grupos trabalhando em padrões BPM. Essa grande heterogeneidade tem um aspecto negativo, visto que dificulta uma padronização, o que é ruim para questões, por exemplo, de interoperabilidade (MENDLING *et al.*, 2004).

Como forma de contornar esse problema, buscou-se utilizar padrões que sejam bem suportados e largamente utilizados pela indústria. No contexto desse trabalho, analistas de negócios modelam os processos de negócio através de uma linguagem de modelagem de alto nível. Posteriormente, esses processos são convertidos para um formato passível de ser executado por computadores, isto é, uma linguagem de execução. Portanto, padrões que ajam nesses dois níveis são necessários. Nesse sentindo, os seguintes padrões foram escolhidos:

- **para a modelagem dos processos**: *Business Process Modeling Notation* (BPMN);
- **para a execução dos processos**: *Business Process Execution Language* (BPEL);

A justificativa para a escolha de BPMN está em ser largamente utilizada em ferramentas comerciais, principalmente aquelas que unem a filosofia BPM junto com SOA (WESKE, 2007). Ouyang (2006) afirma, por exemplo, que mais de trinta ferramentas comerciais suportam o padrão BPMN. Portanto, é considerado um padrão para modelagem de processos de negócio (WANG *et al.*, 2007).

No mesmo sentido, BPEL é considerado o padrão de *facto* para a execução de processos em ambientes que unem BPM com SOA.

Assim, BPEL é suportado por um grande número de ferramentas. (OUYANG *et al.*, 2006; KAMOUN, 2007).

2.1.1.1 Business Process Modeling Notation (BPMN)

BPMN é uma linguagem de notação para modelagem de processos de negócio desenvolvida sobre a coordenação da OMG (JURIC *et al.*, 2007). O principal objetivo do BPMN é prover uma notação que seja facilmente compreensível por todos os envolvidos na concepção dos processos, desde os analistas de negócio, responsáveis por modelar o processo, até os analistas de TI, que irão implementá-los com auxílio de soluções tecnológicas (OMG, 2006).

Em BPMN, os modelos de processos de negócio são expressos através de diagramas (WESKE, 2007). Os elementos utilizados para se fazer a modelagem são divididos em quatro categorias, conforme a Figura 3.

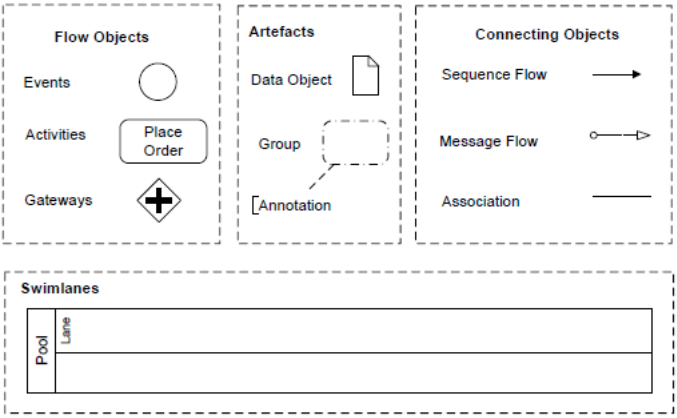


Figura 3 – Categorias dos Elementos de BPM
Fonte: (WESKE, 2007)

- Objetos de Fluxo (*flow objects*): definem atividades, eventos e *gateways*. Atividades podem ser tarefas ou sub-processos. Eventos podem ser de início, intermediários e de fim. Os *gateways* controlam a quebra de fluxos sequenciais em fluxos paralelos e a sua posterior convergência novamente em fluxos sequenciais;

- Objetos de Conexão (*connecting objects*): são usados para conectar objetos de fluxo entre si;
- Raias (*swimlanes*): são usadas para organizar atividades em categorias visuais, de forma a ilustrar claramente as diferentes responsabilidades e papéis dos envolvidos na execução do processo;
- Artefatos (*artifacts*): são usados para adicionar contextos específicos no processo que está sendo modelado. Objetos de dados são utilizados para mostrar documentos que são produzidos ou consumidos pelo processo. Grupos são usados para agrupar atividades semelhantes e anotações são usadas para anexar descrições textuais junto ao diagrama do processo.

Segundo (JURIC *et al.*, 2007), BPMN pode ser usado para se modelar partes de um processo ou processos inteiros, com diferentes níveis de detalhamento. Também é adequado para se modelar processos internos da empresa ou processos colaborativos, que envolvem diferentes parceiros comerciais (B2B). Na Figura 4, por exemplo, apresenta-se um exemplo um processo de compra descrito em BPMN.

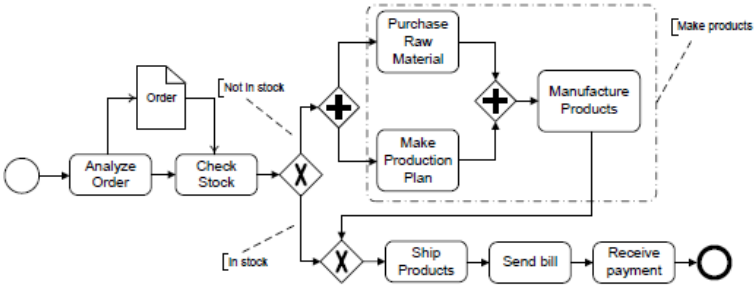


Figura 4 – Processo de Compra em BPMN
Fonte: (WESKE, 2007)

2.1.1.2 Business Process Execution Language (BPEL)

BPEL é um formato para a definição de processos de negócio a nível de implementação (DIJKMAN *et al.*, 2008). É um padrão da OASIS (OASIS, 2007), baseado em XML, que vem se tornado predominante na orquestração de serviços *web*, sendo usado junto com soluções BPM&SOA.

A execução de um processo BPEL é semelhante a de um sistema de *workflow*, onde um serviço vai sendo invocado um após o outro. Cada tarefa BPEL é equivalente a uma invocação numa linguagem de programação. A atividade, por exemplo, "*receive*" aguarda o recebimento de uma mensagem de entrada, enquanto que a atividade *"invoke"* invoca um serviço web (MARGOLIS, 2007).

Os seguintes elementos estão disponíveis na linguagem BPEL (WESKE, 2007):

- *Invoke*: invoca uma operação fornecida por um *web service*. Esta operação pode ou não ter uma resposta;
- *Receive*: espera uma mensagem chegar;
- *Reply*: manda a resposta de uma determinada mensagem recebida;
- *Wait*: espera por um determinado período de tempo;
- *Assign*: atribui valores a variáveis utilizadas pelo processo;
- *Throw*: utilizada para lançar exceções, quando ocorrem erros no processo;
- *Terminate*: encerra o processo.

O uso de BPEL permite a implementação de complexos processos de negócio com base em serviços *web*, previamente existentes. O código BPEL representa o fluxo do processo de negócio descrito no nível de BPM. As funcionalidades dos serviços que são invocados pelo BPEL são especificadas através de linguagens como o WSDL, incluindo, por exemplo, as definições do tipos de mensagens trocadas e de tipo de dados (SCHEITHAUER *et al.*, 2008).

A linguagem BPEL é suportada pela maioria das plataformas e ferramentas de desenvolvimento de sistemas e provê suporte para a abstração de processos de negócio, tornando-os executáveis (JURIC *et al.*, 2009).

2.1.2 Exemplo de Modelagem e Implantação de BPM

Esta seção apresenta um exemplo de como a abordagem BPM é aplicada tradicionalmente nas empresas. Segundo (KO, 2009), a implantação e modelagem processos BPM é feita geralmente pelo departamento de TI da empresa. Essa implantação se dá, segundo o autor, através de seis etapas, apresentadas na Figura 5.

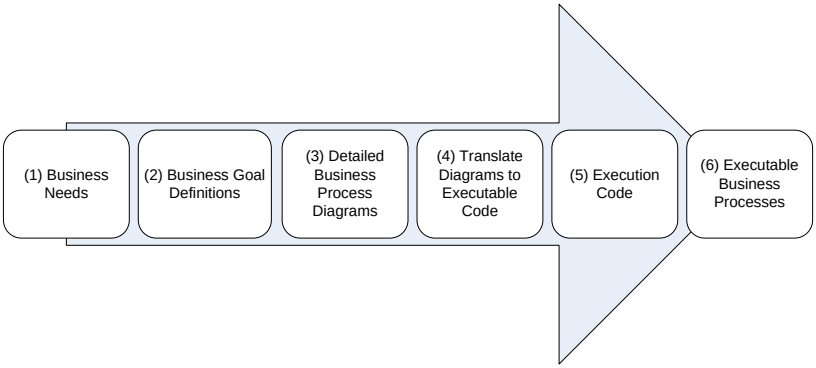


Figura 5 – Modelagem e implantação de processos utilizando BPM
Fonte: (KO, 2009)

As fases desse processo de implantação, que é o que ocorre na prática nas empresas, estão descritas a seguir (KO, 2009):

- **Passo 1: Identificação das Necessidades do Negócio**: Nesse passo, identifica-se qual é a necessidade num nível mais amplo do processo que será modelado. Por exemplo, num processo de compra, a necessidade maior é poder efetuar uma compra;
- **Passo 2 – Definição dos Requisitos**: Nesse passo a gerência irá detalhar aos analista de negócios de que forma o processo ocorre, quais são os seus requisitos etc. Esse detalhamento é equivalente à fase de levantamento de requisitos na engenharia de software. O objetivo é que o analista de negócios consiga ter informação suficiente para fazer a modelagem do processo de negócio;
- **Passo 3 - Detalhamento do Processo em Diagramas**: nessa fase, o analista de negócios já tem um conhecimento do processo, assim podendo modelá-lo numa notação gráfica (como BPMN). Esse passo é feito com auxílio de ferramentas, como um editor BPM;
- **Passo 4 - Conversão do Processo para um formato executável**: Com o auxílio de uma ferramenta automatizada, o processo descrito em BPMN é transformado numa linguagem técnica e que seja executável, como por exemplo, o BPEL;

- **Passo 5 – Codificação**: nessa fase, a equipe de TI da empresa irá fazer os ajustes no código executável, adicionando, se necessário, mais lógica de programação. Em seguida um protótipo do processo executável será apresentado à gerência, que irá validá-lo;
- **Passo 6 - Execução do Processo**: após o processo ter sido validado, ele será implantado num *software* chamado servidor de aplicações, que contém um motor de execução capaz de executar o processo de negócio.

Com relação às etapas de implantação, algumas limitações impedem que ela seja mais ágil. Na primeira e na segunda etapa, por exemplo, é necessário haver reuniões e encontros com a gerência e os analistas de negócio a fim de se levantar os requisitos e saber exatamente como o processo deve ser modelado. Esse procedimento pode ser demorado, diminuindo a velocidade que a implantação do processo poderia ser feita. Nessa abordagem tradicional não se prevê que esses processos já poderiam estar bem definidos e detalhados num catálogo de processos de negócio, por exemplo.

Já no passo cinco é necessário que a equipe de TI faça alterações no código executável gerado para que o processo possa ser, de fato, implementado. Nesse sentido, a conversão de BPMN para BPEL não é transparente, visto que é necessária essa etapa intermediária de ajustes. Uma especificação padronizada de processos de negócio poderia permitir que os artefatos utilizados para a execução do processo (por exemplo, serviços) já estivessem disponíveis na "nuvem", sendo descobertos e vinculados ao processo automaticamente, através de algum mecanismo automatizado de busca.

2.2 Service Oriented Architecture (SOA)

Atualmente, uma abordagem que vem adquirindo grande suporte na indústria é baseada na visão de que as soluções e processos de negócio das organizações são vistos como serviços, que são ligados através de *interfaces* bem definidas. Essa abordagem é conhecida como Arquitetura Orientada a Serviços ou SOA (*Service Oriented Architecture*) (BOOTH *et al.*, 2004).

Arquiteturas baseadas em SOA permitem o desenvolvimento de sistemas em que os principais componentes são serviços, que são encapsulados e reusados de forma que diferentes aplicações os utilizem com um baixo acoplamento e alta disponibilidade (KAMOUN, 2007).

Isso permite fazer uma comunicação eficaz entre empresas e plataformas heterogêneas através do uso de padrões abertos.

Além disso, SOA utiliza artefatos como especificações de processos, descritores de *workflow* e esquemas de documentos que são compreensíveis por sistemas computacionais e que permitem uma melhor adaptação a ambientes que mudam frequentemente (DORN *et al.*, 2007).

As organizações vêm atualmente gastando muito tempo tentando atingir uma integração rápida e flexível dos seus sistemas de TI, dentro do contexto dos seus negócios. SOA permite definir uma abordagem arquitetural que permite a integração dos sistemas de maneira rápida e flexível (DUDLEY, 2007). As motivações para tal incluem:

- Aumento na agilidade de implementação de novos produtos e serviços ou alteração dos já existentes;
- Redução dos custos de integração e implementação dos sistemas, tanto internos como inter-organizacionais;
- Possibilidade de reuso de sistemas e aplicações legadas;
- Alcançar um melhor aproveitamento da TI e do retorno sobre o investimento.

Nesse sentido, SOA fornece uma abordagem arquitetural, baseada em serviços, para que se possa atingir essa integração de forma rápida e flexível. Essa arquitetura deve lidar com os seguintes aspectos (LEYMANN *et al.*, 2002; BOOTH *et al.*, 2004):

- Criação de serviços;
- Descrição do serviço: a descrição da forma que ele pode ser invocado, incluindo os formatos e protocolos usados para tal;
- Publicação do serviço: o serviço de ser publicado em repositórios de forma que possa ser localizado por usuários em potencial;
- Descoberta de serviços: incluindo a busca por serviços que atendam determinados critérios, principalmente requisitos de negócios;

Uma arquitetura SOA consiste fundamentalmente de três componentes, que são o Provedor de Serviço, o Diretório de Serviços e o Consumidor do Serviço. Estes componentes são descritos na Figura 6 (LEYMANN *et al.*, 2002):

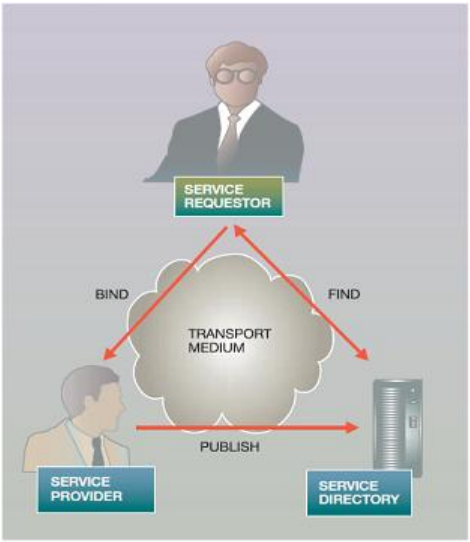


Figura 6 – Componentes SOA
(Fonte: LEYMANN et al., 2002)

- **Provedor do Serviço**: O provedor do serviço é a entidade que provê as aplicações de *software*, em forma de serviços. Os provedores publicam, removem e atualizam seus serviços de forma que fiquem disponíveis aos consumidores dos serviços. O provedor é o dono do serviço, mantendo sua lógica de negócio e implementação invisíveis do consumidor;
- **Consumidor do Serviço**: O consumidor do serviço é a entidade que possui uma necessidade que pode ser satisfeita por um serviço disponível. O consumidor pode ser um usuário, uma aplicação ou mesmo um outro serviço. O serviço que ele requisita é localizado através do uso do diretório de serviços.
- **Diretório de Serviços**: O diretório de serviços provê um repositório onde é possível se localizar descrições e interfaces de serviços. Os provedores de serviços publicam os seus serviços no diretório, e os consumidores de serviço o utilizam para localizar e obter informações necessárias para a invocação do serviço.

A filosofia SOA é vantajosa em relação às abordagens tradicionais, pois provê meios de se projetar aplicações fortemente coesas e fracamente acopladas através de módulos especializados, distribuídos, acessados remotamente, interoperáveis e reutilizáveis, chamados de serviços. Neste cenário, serviços determinam o comportamento de um sistema associado a um dado processo de negócio.

Em relação a este trabalho, a contribuição de SOA está em permitir que a lógica de negócios seja implementada através de serviços, usados para implementar o comportamento desejado dos processo de negócio e permitir o reuso destes serviços em diferentes contextos.

2.2.1 Web Services e Padrões Associados

Arquiteturas SOA utilizam serviços como elementos fundamentais para o desenvolvimento de aplicações (BREIVOLD *et al.*, 2007). Assim, um serviço *web* ou w*eb service* é definido pelo Grupo de Trabalho, ligado ao W3C, responsável pela especificação da *Web Services Architecture* (WSA), como:

> [Definition: A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.] (W3C, 2004).

Cada vez mais as aplicações precisam acessar recursos *web* de forma automatizada, garantindo maior integração dos processos de negócio. Os w*eb services* foram criados para constituir aplicações interoperáveis através da internet. Um w*eb service* é um programa ou procedimento remoto que pode ser acessado e executado via protocolos da *web*.

O trabalho de (PIAZZA, 2007) comenta sobre algumas das características que estão presentes nos *web services*:

- **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 a implementações. 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;
- **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.

Os serviços *web* são constituídos de um conjunto de padrões, que permitem a localização e invocação dos serviços, de maneira independente da plataforma e dos sistemas utilizados para a criação e para a publicação desses serviços. Diversos órgãos padronizadores e da indústria estão envolvidos na padronização dos serviços *web*. Entre eles destacam-se a W3C (W3C, 2010), a OASIS (OASIS, 2010) e a WS-I (WS-I, 2010).

Segundo (BAILEY, 2006), os principais padrões associados aos serviços *web* são:

- *Web Services Description Language* (WSDL);
- Protocolo SOAP;
- *Universal Description, Discovery and Integration* (UDDI).

Apresenta-se a seguir, em maiores detalhes cada um desses padrões.

2.2.1.1 WSDL

WSDL é uma linguagem XML utilizada para descrever a forma de acesso a um serviço *web*. Essa linguagem permite especificar a *interface* do serviço, que fornece as informações necessárias para que este possa ser invocado corretamente (MARGOLIS, 2007).

A descrição dos serviços via WSDL provê uma documentação para sistemas distribuídos e serve como uma forma de automatizar a comunicação entre as aplicações, pois contém todos os detalhes do serviço necessários para que ele possa ser utilizado.

Os elementos que fazem parte da estrutura de um documento WSDL são mostrados na Figura 7, seguidos de sua descrição (MARGOLIS, 2007).

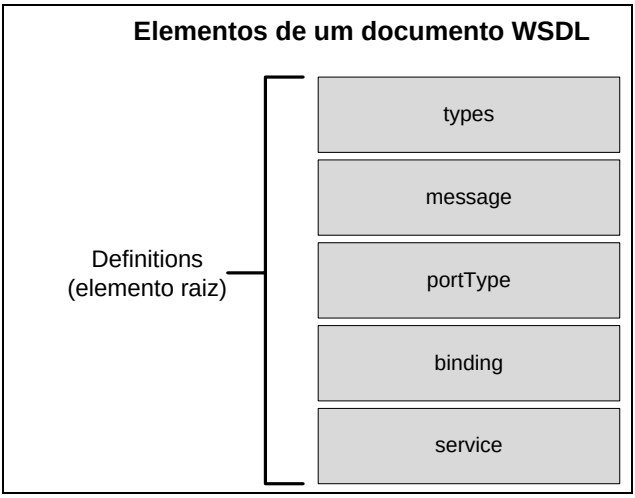


Figura 7 – Elementos de um documento WSDL

- ***definitions***: é o elemento raiz de um documento WSDL, contendo todas as definições globais;
- ***types***: indica as definições dos tipos de dados usados em declarações ao longo do documento WSDL;
- ***message***: este elemento descreve de forma abstrata a organização das mensagens utilizadas nas operações de envio e recebimento de uma mensagem;
- ***portType***: este elemento agrupa o conjunto de operações disponíveis no *web service;*
- ***binding***: descreve o mecanismo usado por um serviço para se comunicar com o cliente. Este elemento é utilizado para definir o formato das mensagens e mapear os elementos *operations* para cada *portType*;
- ***service***: define a localização (endereço) real do serviço, levando em conta que o mesmo pode conter várias portas e cada uma delas é específica para um tipo de ligação, descrita no elemento *binding*.

Na Figura 8 é apresentado um exemplo de um documento WSDL, onde estão realçados cada um dos elementos que o compõem.

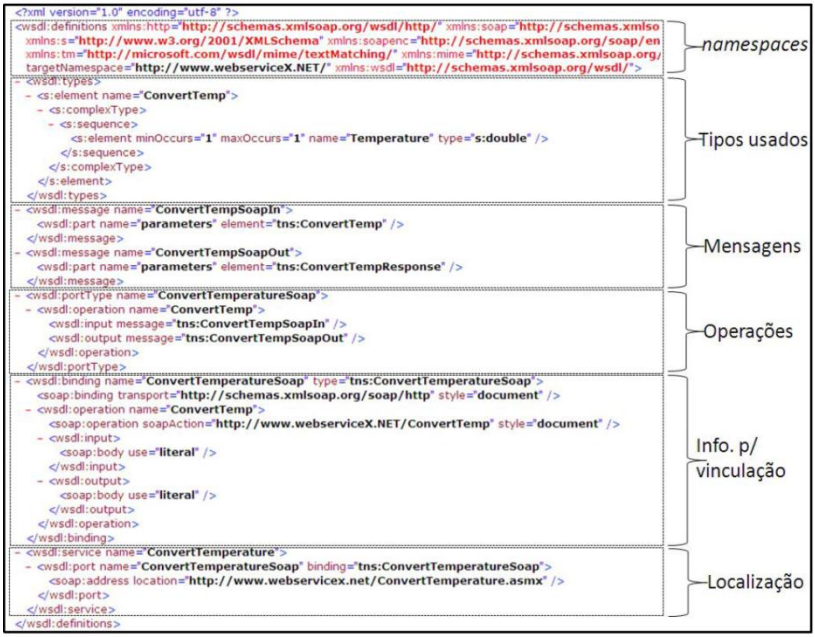


Figura 8 – Exemplo de um documento WSDL
Fonte: (SOUZA, 2009)

2.2.1.2 SOAP

SOAP é um protocolo que define um mecanismo para a comunicação de *web services* pela internet. Ele é responsável por especificar o formato das mensagens que são trocadas entre o cliente e o serviço *web*. SOAP também define como o protocolo HTTP pode ser usado para permitir chamadas a procedimentos remotes (RPC – *remote procedure call)* pela *internet*, através da combinação dos cabeçalhos HTTP com o corpo da mensagem SOAP (LEYMANN *et al.*, 2002).

Um dos objetivos do SOAP é permitir a troca de mensagens de aplicações desenvolvidas em diferentes linguagens, executando em plataformas heterogêneas e se comunicando pela internet, assim facilitando a interoperabilidade (PIAZZA, 2007). Ferramentas automatizadas são capazes de gerar requisições SOAP através da interpretação do WSDL do serviço (MARGOLIS, 2007).

A estrutura de uma mensagem SOAP é descrita no formato XML, tendo os seus elementos apresentados na Figura 9 (PIAZZA, 2007).

Figura 9 – Estrutura de uma mensagem SOAP
Fonte: (PIAZZA, 2007)

- **Envelope**: é o elemento raiz da mensagem XML, que geralmente possui as declarações de *namespace* usadas;
- **Cabeçalho**: é um elemento opcional que pode ser usado para declarar algum parâmetro adicional (por exemplo QoS);
- **Corpo**: é a parte principal da mensagem SOAP, onde estão armazenadas as informações necessárias para fazer a chamada ao procedimento remoto (RPC). Caso a requisição gere um erro no serviço, as informações a respeito do mesmo são anexadas junto ao corpo.

A Figura 10 e a Figura 11 apresentam respectivamente um exemplo de uma requisição e a sua resposta, no formato definido pelo protocolo SOAP.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope"/>
  <soap:Header/>
  <soap:Body>
    <getInformacaoAluno>
      <matricula>04132440</matricula>
    </getInformacaoAluno>
  </soap:Body>
</soap:Envelope>
```

Figura 10 – Exemplo de requisição SOAP

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns1:getInformacaoAlunoResponse
                xmlns:ns1="http://services.cagr.ufsc.br/"
                xmlns:ns2="http://output.services.cagr.ufsc.br">
      <ns1:informacaoAluno>
        <ns2:nome>Roque Oliveira Bezerra</ns2:nome>
        <ns2:matricula>4132440</ns2:matricula>
        <ns2:codigoCurso>208</ns2:codigoCurso>
        <ns2:nomeCurso>CIÊNCIAS DA COMPUTAÇÃO</ns2:nomeCurso>
        <ns2:codigoCentro>CTC</ns2:codigoCentro>
        <ns2:semestreIngresso>20041</ns2:semestreIngresso>
        <ns2:codigoSituacao>1</ns2:codigoSituacao>
        <ns2:nomeSituacao>formado</ns2:nomeSituacao>
        <ns2:modoInformacao>Reduzido</ns2:modoInformacao>
      </ns1:informacaoAluno>
    </ns1:getInformacaoAlunoResponse>
  </soap:Body>
</soap:Envelope>
```

Figura 11 – Exemplo de resposta SOAP

2.2.1.3 UDDI

A UDDI (Universal Description, Discovery and Integration) é uma especificação da OASIS que define uma forma padrão de publicar e localizar serviços em um registro (OASIS, 2005).

Um registro UDDI fornece um mecanismo padronizado de classificação, catalogação e gerenciamento de serviços, de forma a permitir que os serviços possam ser descobertos e consumidos por outras aplicações. Nesse sentido, uma organização que deseja publicar e disponibilizar seus serviços, isto é, um provedor de serviços, pode fazê-lo através da UDDI. Da mesma forma, uma organização (consumidora do serviço) pode usar a UDDI para encontrar serviços que atendam as suas necessidades e usar o repositório para obter os metadados necessários para invocá-los (PIAZZA, 2007).

Três tipos de informações estão disponíveis em uma UDDI:

- ***White Pages***: incluem informações básicas de uma organização, tais como: nome, endereço, contato etc.;
- ***Yellow Pages***: descrevem os serviços através de uma determinada categorização e taxonomia;
- ***Green Pages***: incluem informações técnicas (por exemplo, formatos de dados, parâmetros de entrada e saída etc.) relativas as funções disponíveis num determinado serviço.

A informação dentro de uma UDDI é armazenada num conjunto de estruturas de dados, dessa forma permitindo fazer publicações e pesquisas de serviços. A Figura 12 apresenta essas estruturas:

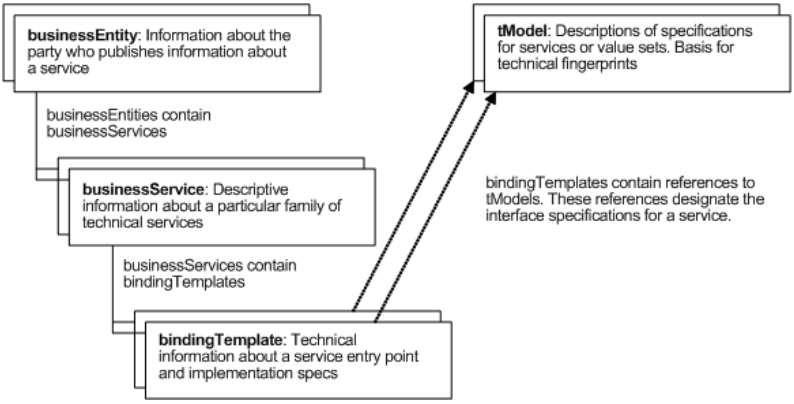


Figura 12 – Estruturas de um repositório UDDI
Fonte: (OASIS, 2005)

- **businessEntity**: fornece as informações sobre quem fornece o serviço;
- **businessService**: contém informações descritivas em termos de negócios (informação não técnica) sobre os serviço oferecido;
- **bindingTemplate**: contém informações sobre o acesso ao serviço, isto é, as informações técnicas necessárias para que o serviço possa ser invocado;
- **tModel**: tem a finalidade de prover informações de categorização e especificações técnicas do serviço. Um tModel pode descrever, por exemplo, como o serviço se comporta, quais convenções utiliza e com quais especificações ou padrões é compatível.

2.2.2 Descoberta de Serviços Web

Descoberta, segundo o grupo da W3C responsável pela especificação da *Web Services Architecture* (WSA), é a ação de

localizar um serviço *web* que atenda determinados critérios (W3C, 2004).

Num ambiente SOA a criação de um novo processo de negócio necessita de serviços reusáveis que estão expostos por diversos sistemas corporativos e provedores. Para que eles possam ser usados, eles precisam primeiro ser descobertos. (KOURTESIS *et al.*, 2007). A função da descoberta de serviços é, portanto, encontrar serviços web que sejam adequados aos requisitos funcionais e não funcionais de uma aplicação. Existem duas abordagens para a descoberta de serviços: a estática e a dinâmica (SOUZA, 2009).

Na abordagem estática ou clássica, o serviço é escolhido quando a aplicação está sendo projetada. Não há garantias de que no momento em que ela será executada, o serviço estará disponível. Já na abordagem dinâmica os serviços são automaticamente descobertos e vinculados à aplicação em tempo de execução. Da mesma forma que na abordagem estática, não há garantias de que esses serviços serão encontrados.

Com relação a essas abordagens, o trabalho (GAROFALAKIS *et al.*, 2006) comenta que o mecanismo de descoberta de serviços deve, em tempo de projeto (*off-line*), pesquisar em repositórios de serviços e encontrar candidatos aptos a executarem determinadas tarefas. Em tempo de execução (*on-line*), a aplicação consumidora deve usar o mecanismo de descoberta para identificar, escolher e vincular uma instancia de serviço mais adequada aos requisitos que a aplicação exige.

O problema de descoberta dinâmica de serviços *web* é complexo. A sua resolução envolve lidar com questões complicadas tais como: i) heterogeneidade tecnológica e baixa interoperabilidade; ii) ambiguidade de conceitos devido a diferentes domínios de aplicação e iii) limitação nas tecnologias utilizadas para se projetar serviços (GAROFALAKIS *et al.*, 2006).

Como forma de tentar resolver esse problema, os trabalhos propostos na literatura podem ser divididos em quatro dimensões (SOUZA, 2009): i) recuperação da informação; ii) arquitetura; iii) QoS e iv) padrões. Na dimensão de recuperação de informação, o enfoque está, grosso modo, em definir meios semânticos para melhorar a precisão na seleção dos serviços. Na arquitetura o enfoque está em questões como escalabilidade, segurança e disponibilidade. A dimensão de QoS (*Quality of Service*) se preocupa em definir atributos e métricas de QoS – qualidade de serviço – a serem usados no processo de descoberta de serviços; estabelecer *frameworks* mais completos e robustos para

representar, verificar, atualizar e gerenciar QoS e definir formas de seleção de serviços. Na dimensão de padrões, a preocupação maior é com a questão da interoperabilidade, onde o uso de padrões WS*, tais como UDDI (OASIS, 2005), SOAP (W3C, 2007a) e WSDL (W3C, 2007b) são bastante usados.

A descoberta de serviços num contexto BPM&SOA requer uma visão holística do problema, que vai além do mostrado nas quatro dimensões e que leve em conta a camada BPM e a sua semântica associada. Nesse sentido, a importância da descoberta de serviços no contexto desse trabalho está em permitir que os serviços que irão executar as atividades do processo possam ser descobertos e vinculados junto ao processo de negócio. Num contexto que visa a integração ágil entre BPM&SOA, é imprescindível que haja um mecanismo automatizado que faça esse papel.

2.3 Qualidade de Serviço (QoS)

O termo QoS (acrônimo de "Quality of Service") designa a capacidade de fornecer um serviço (não necessariamente um serviço *web*) conforme as exigências de alguns requisitos não funcionais, geralmente relacionados com tempos de resposta, largura de banda etc.

Os critérios de QoS exercem um papel muito importante quando relacionados no contexto de serviços *web*. Eles permitem, por exemplo, diferenciar serviços, atendendo melhor as necessidades dos consumidores e, no viés dos provedores de serviços, seus valores constituem fonte de informação para detecção de anomalias associadas aos serviços *web* (SOUZA, 2009).

No contexto de arquiteturas SOA e serviços *web* surgiram, a partir do trabalho de (SABATA *et al.*, 1997), novas frentes de trabalho, com o intuito de amadurecer quais aspectos deveriam ser considerados em um domínio de serviços *web* (CANCIAN, 2009). Neste sentido, (MANI *et al.*, 2005) propuseram uma relação de requisitos de qualidade a serem considerados no contexto de serviços *web*, e apresentados na Tabela 1.

Tabela 1 – Relação de requisitos de qualidade no contexto de serviços web
Fonte: (MANI *et al.*, 2002)

Requisito	Descrição
Disponibilidade (*Availability*)	Aspecto de qualidade que informa se o serviço está pronto para uso imediato. Este aspecto é representado por uma probabilidade. Quanto maior for o valor da probabilidade maior será a disponibilidade do serviço.
Acessabilidade (*Accessibility*)	Representa o grau que um serviço tem em prover determinado serviço. Podem ocorrer situações onde um serviço possa estar disponível, porém não acessível. Situações de não acessibilidade de um serviço podem ocorrer quando um serviço web não tem capacidade de absorver aumentos no número de requisições.
Integridade (*Integrity*)	Aspecto que faz referência ao comportamento de um serviço na execução de transações. Após a execução de uma transação, o estado da informação deve permanecer livre de inconsistências
Desempenho (*Performance*)	Aspecto medido em termo de *throughput* (número de requisições fornecidas em um dado tempo) e latency (tempo entre o envio de pedido e o recebimento de resposta). Altas taxas de *throughput* e baixas de latência representam bom desempenho de um serviço.
Segurança (*Security*)	Aspecto que fornece confiabilidade e não-repúdio das partes envolvidas, codificação de mensagens e controle de acesso. Devido a sua importância, já que serviços *web* são invocados via web, este aspecto, geralmente, envolve diferentes componentes e diferentes abordagens de implementação.

Complementando a lista de (MANI *et al.*, 2002), (LEE *et al.*, 2003) levantaram outros atributos de qualidade, apresentados na Tabela 2.

Tabela 2 – Relação de requisitos de qualidade no contexto de serviços web
Fonte: (LEE *et al.*, 2003)

Requisito	Descrição
Escalabilidade (*Scalability*)	Serviços *web* devem prover alto grau de escalabilidade. Este atributo refere-se ao aumento da capacidade de processar mais pedidos num mesmo dado intervalo de tempo sem comprometer o serviço web.
Capacidade (*Capacity*)	Capacidade limite de receber pedidos simultâneos, os quais devem ser fornecidos com garantia e desempenho.
Robustez (*Robustness*)	Serviços *web* devem ser providos com alto grau de robustez. Robustez representa até quanto um serviço *web* pode trabalhar mesmo na presença de dados de entrada inválidos ou incompletos.
Precisão (*Accuracy*	Serviços *web* devem fornecer alto grau de precisão. Precisão é definida como a taxa de erros gerada por um serviço *web*.
Interoperabilidade (*Interoperability*)	Serviços *web* devem possuir características que os tornem interoperáveis.
Confiabilidade (*Reliability*)	Garante a disponibilidade e confiabilidade dos recursos de TI, a fim de assegurar a satisfação do cliente e a reputação do negócio.

No contexto deste trabalho, QoS permite classificar e filtrar os serviços *web* que irão implementar as atividades do processo. Essa filtragem é necessária, no sentido de permitir que o usuário-projetista possa atribuir certos requisitos, em termos de qualidade, aos serviços que ele irá buscar, no momento em que a modelagem do processo é feita. Isso permite restringir o universo de serviços que serão candidatos a implementar as atividades do processo. Os atributos de QoS apresentados nesta seção servem como base para que o usuário estabeleça estas restrições de qualidade.

2.4 Especificações de Processos de Negócio

Esta seção dedica-se a apresentar as especificações de processos de negócio *Universal Business Language* (UBL), *RosettaNet* e ebXML.

No contexto deste trabalho, o estudo dessas especificações é importante, pois permite padronizar os processos de negócio no sentido de que o usuário-projetista, no momento em estiver concebendo sua aplicação BPM&SOA, possa se basear em processos já existentes e bem definidos pela especificação.

2.4.1 Universal Business Language (UBL)

A *Universal Business Language* (LAMPATHAKI *et al.*, 2009) é uma linguagem baseada em *XML* (W3C, 2003) para descrição de processos de negócio e dos documentos trocados por eles. A especificação é gerenciada pela OASIS (OASIS, 2006b) e é livre de royalties. Ela é a primeira implementação do padrão ISO 15000-5 (UN/CEFACT CCTS) (ISO, 2005), sendo baseada em modelos conceituais de componentes chamados de entidade de negócios. Esses componentes são encapsulados num formato específico como, por exemplo, *Order* ou *Invoice*, que são então transformados em *XML Schemas* (W3C, 2000).

A especificação UBL 2.0 (OASIS, 2006b) possui um total de 31 *XML Schemas*, que são especializados num conjunto de documentos que contém regras de formação de documentos trocados em transações eletrônicas ou por processos de negócio. Os *XML Schemas*, portanto, constituem o vocabulário trocado entre os parceiros de negócios, sendo caracterizados por serem modulares, reusáveis e extensíveis. Tais aspectos permitem que o padrão UBL evolua para outros domínios ou necessidades específicas, caracterizando um padrão consistente e genérico usado nas especificações das informações envolvendo transações eletrônicas.

A Figura 13 apresenta um diagrama de atividades UML para descrever um processo de compra:

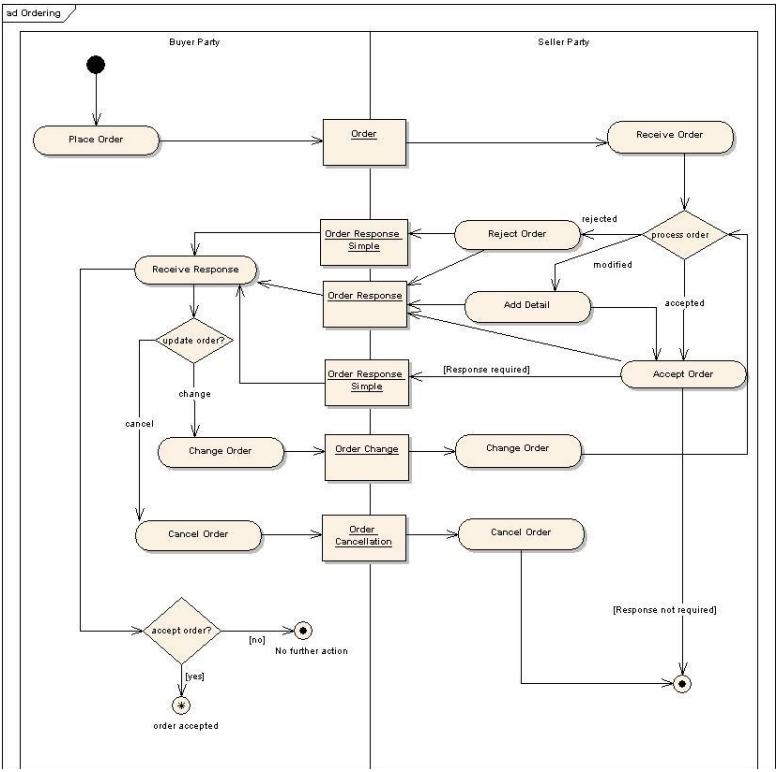


Figura 13 – Processo de Compra
Fonte: (OASIS, 2006b)

O processo ilustrado na Figura 13 possui dois participantes: o comprador (*buyer party*) e o vendedor (*seller party*).

Os retângulos com rótulos sublinhados são os documentos trocados pelos participantes. Eles são estruturados em XML e regidos pela especificação UBL. Os losangos representam etapas de tomada de decisão e os retângulos arredondados são as tarefas que compõem o processo. As setas representam o fluxo de informações e os círculos concêntricos o fim do processo.

De maneira simplificada, o processo inicia com uma requisição de compra (*Order* – canto superior esquerdo da Figura 13) que é enviada ao vendedor. Este analisa a requisição e responde ao comprador, que pode recusar, alterar ou aceitar a negociação. O vendedor, no momento em que aceita o pedido, encerra sua participação no processo.

2.4.2 RosettaNet

RosettaNet é um consórcio composto por empresas dos setores de tecnologia da informação, componentes eletrônicos, semicondutores, manufatura, comunicação e logística. O seu objetivo é a construção de padrões abertos que permitam a integração e interoperabilidade entre parceiros comerciais num contexto de *e-business*. A *RosettaNet* define processos inter-empresariais comuns e os documentos trocados por eles (KOTINURMI *et al.*, 2008).

A especificação foca-se em três grandes áreas de padronização, a fim de automatizar as interações no contexto B2B (MEDJAHED *et al.*, 2003).

A primeira área se dedica a especificar o vocabulário de dados trocados. Isso é feito através do *RosettaNet Business Dictionary*. Esse vocabulário possui as propriedades necessárias para descrever as características de produtos e serviços.

A segunda área trata da forma como as mensagens são encapsuladas e transmitidas aos outros participantes. O *RosettaNet Implementation Framework* (RNIF) especifica o conteúdo das mensagens, os protocolos de transporte (HTTP, CGI, email etc.) e mecanismos de segurança como, por exemplo, o uso de certificados digitais.

A terceira e última área trata da especificação dos processos e a sequência em que as mensagens serão trocadas pelos participantes. Isso é feito através dos *RosettaNet PIPs* (*Partner Interface Processes*). Os *PIPs* são conversações pré-definidas no formato *XML*. Uma conversação consiste num conjunto de documentos e numa lógica de intercâmbio de mensagens como, por exemplo, a sequência de ações que é efetuada num processo de compra.

O documento de especificação de um *PIP* define a sequência de troca de mensagens usando diagramas *Unified Modeling Language* (UML). Alguns dos diagramas usados são o de atividades, sequência e descrições textuais. Os papéis de cada participante dentro da transação comercial e as condições necessárias para que ela aconteça também são definidas pelos *PIP's* (KOTINURMI *et al.*, 2008).

Como exemplo, a Figura 14 apresenta o processo de Solicitação de Ordem de Compra (*Request Purchase Order*), onde um comprador (*Buyer*) emite uma ordem de compra e o vendedor (*Seller*) confirma se a mesma será aceita, rejeitada ou se ficará pendente.

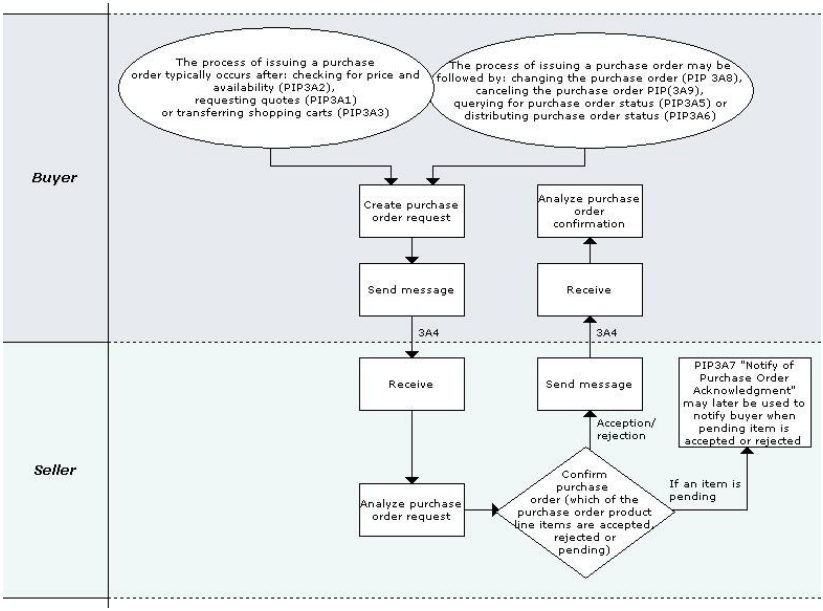


Figura 14 – PIP3A4 – "Solicitação de Ordem de Compra" (*Request Purchase Order*) (Fonte: ROSETTANET, 2008)

Na Figura 14 o fluxo das atividades é representado pelas setas direcionas. Os retângulos definem as atividades que são executadas, enquanto que losangos representam decisões a serem tomadas. As elipses representam descrições textuais bem como fatos presentes em outros *PIP's* que devem ser levados em conta para que a execução do PIP3A4 possa ser feita com sucesso.

2.4.3 ebXML

O *ebXML* foi criado como uma iniciativa conjunta entre duas organizações focadas em padronização, a UN/CEFACT (UN/CEFACT, 2010) e a OASIS (OASIS, 2010). O seu intuito está em prover um ambiente tecnológico que permita a colaboração num contexto B2B (DORN *et al.*, 2009). Dessa forma, ebXML define uma série de especificações que permitem a interação de empresas, tanto pequenas quanto grandes (MEDJAHED *et al.*, 2003). As especificações fornecidas são:

- *Messaging* (ebMS);
- *Registry* (ebRIM/ebRS);
- *Collaboration Protocol Profiles and Agreements* (CPP/A);
- *Core Components* (CC);
- *Business Process Specification Schemas* (BPSS);

Dentre essas especificações, duas estão focadas na especificação de processos de negócio: *Core Components* (CC) e *Business Process Specification Schemas* (BPSS).

Core Components são um conjunto de componentes que permitem o reuso de informações relativas a processos de negócio. Essas informações são livres de contexto, de forma que podem ser usadas em diferentes domínios de negócios (DORN *et al.*, 2007). Nesse sentido, ebXML é uma especificação que não trata de um domínio específico (padrão horizontal), ao contrário, por exemplo, da RosettaNet, que é uma especificação voltada à indústria de eletrônicos (padrão vertical) (KO *et al.*, 2009).

Business Process Specification Schemas capturam a coreografia envolvida em processos de negócio colaborativos num formato compreensível por computadores (DORN *et al.*, 2009). A especificação ebXML fornece um conjunto de processos de negócio que são compartilhados por diversas indústrias. Essas especificações são armazenadas numa biblioteca, podendo ser usadas para se customizar processos de negócio. A interação entre processos de negócio é feita através do uso de coreografias. Uma coreografia especifica a ordem e as transições que ocorrem na colaboração B2B. (MEDJAHED *et al.*, 2003).

A especificação ebXML possui um catálogo de processos de negócio chamado de *ebXML Catalog of Common Business Processes* (OASIS, 2001) que fornece um conjunto de processos que são independentes de indústrias específicas. Esse caráter genérico dos processos permite que eles sejam usados em variados negócios, através de pequenas adaptações. Algumas das categorias desses processos são (OASIS, 2001):

- Gestão de Compras;
- Administração;
- Gestão de Produtos;
- Finanças;
- Transporte e Logística;
- Gerenciamento de Marketing;
- Desenvolvimento de Produtos;

- Planejamento da Produção;
- Suporte.

Segundo (DORN *et al.*, 2009), apesar de especificação ebXML possuir muitas qualidades, ela atualmente está sendo muito pouco utilizada, principalmente pela falta de suporte da indústria. Esta, por sua vez, está mais focada no uso de serviços *web* e de seus padrões associados (WSDL, SOAP e UDDI). Isso se dá no sentido que ebXML fornece uma estrutura completa para B2B, que concorre com serviços *web* em muitas áreas. Apesar de tecnicamente superior em muitas delas, a especificação ebXML não obteve o sucesso esperado (DORN *et al.*, 2009).

No que diz respeito ao catálogo de processos de negócio, a especificação ebXML encontra-se desatualizada, sendo que a última versão desse catálogo é datada de 2001. Além do mais, em comparação com padrões como UBL e RosettaNet, o uso de ebXML é bastante pequeno (KO *et al.*, 2009).

2.5 Ontologia

Segundo (GUIZZARDI, 2000), em ciências da computação, o termo ontologia refere-se a um artefato constituído por um vocabulário de termos organizados em uma taxonomia, suas definições e um conjunto de axiomas formais usados para criar novas relações e para restringir as suas interpretações segundo um sentido pretendido.

Uma ontologia define um vocabulário comum, que permite o compartilhamento de informação, dentro de um domínio. Ela inclui definições, passíveis de serem interpretadas por computadores, de conceitos básicos e as suas relações (NATALYA *et al.*, 2002). Nesse sentido, (FALBO *et al.*) cita que uma ontologia pode ser vista como um modelo para um domínio de aplicações, sendo usada basicamente para especificá-las e desenvolvê-las, aumentando o reuso.

Com relação a sua estrutura, uma ontologia é formada por um conjunto de classes, propriedades, axiomas e instâncias (SILVA *et al.*, 2006).

- classes: podem ser abstratas ou concretas, elementares ou compostas, reais ou fictícias. Em uma classe podem existir taxonomias. Por exemplo, a classe homem pode ser uma sub-classe da classe pessoa;

- propriedades: representam os relacionamentos entre as classes, por exemplo, entre as classes de pessoa e carro, o relacionamento "ser dono";
- axiomas: são regras ou afirmações da verdade sobre classes. Um exemplo de axioma é afirmar que toda pessoa tem uma mãe;
- instâncias: são um conhecimento previamente existente na base de conhecimento. Por exemplo, "*Mustang*" pode ser uma instância da classe carro.

As classes são o foco da maioria das ontologias. Classes descrevem conceitos dentro do domínio da ontologia. Por exemplo, a classe bebidas representa todas as bebidas. Sub-classes representam classes que são mais específicos. Por exemplo, a classe bebida pode ser dividida nas sub-classes bebidas alcoólicas e não alcoólicas.

As propriedades permitem caracterizar as classes e as instâncias. Por exemplo, a bebida *Coca-Cola* pode ser do tipo com gás; produzida pela *Cola-Cola Company*. Nesse exemplo, tem-se duas propriedades: tipo, cujo valor é com gás; e produzido, cujo valor é *Coca-Cola Company*. A Figura 15 apresenta uma representação visual dessa pequena ontologia, cujo domínio é o de bebidas. As classes são representadas como retângulos pontilhados. As instâncias dessas classes são mostradas através dos retângulos preenchidos.

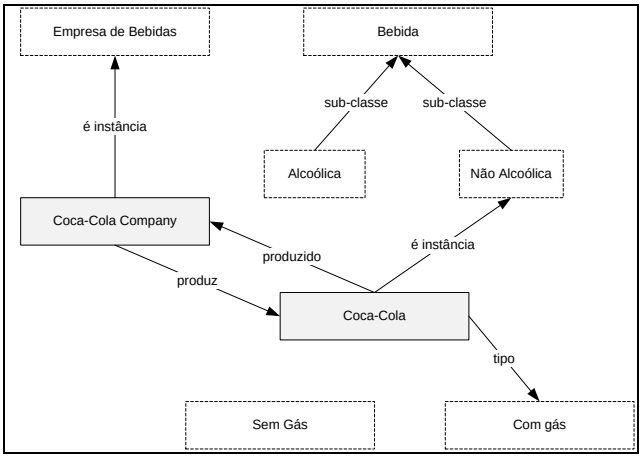


Figura 15 – Exemplo de Ontologia – Domínio: bebidas

No desenvolvimento de ontologias, (NATALYA *et al.*, 2002) cita alguns passos que devem ser tomados:

- definir as classes da ontologia;
- estruturar essas classes numa taxonomia hierarquizada, por exemplo as relações de sub-classe e super-classe;
- definir as propriedades das classes, e quais valores são permitidos para elas;
- definir o valor dessas propriedades para as instâncias das classes.

As ontologias, portanto, permitem a criação de bases de conhecimento, que podem ser usadas por mecanismos computacionais para fazer inferência em cima de determinadas situações. Sistemas especialistas, por exemplo, utilizam fortemente ontologias para que possam tirar suas conclusões. Um sistema de diagnóstico médico, por exemplo, é um sistema especialista que utiliza ontologias, nesse caso relativas às doenças existentes, seus sintomas característicos etc. (NATALYA *et al.*, 2002).

No contexto desse trabalho, o uso de ontologias é de fundamental importância, pois permite fazer a modelagem do conhecimento, nesse caso, os processos contidos na especificação UBL. A ontologia, portanto, permite atribuir uma camada semântica aos processos modelados, viabilizando a posterior busca e vinculação de serviços *web* a estes processos.

Optou-se por utilizar ontologias em detrimento de outras abordagens de classificação, pois esta provê uma forma bastante poderosa de representar o conhecimento. Apesar de no escopo deste trabalho, esse conhecimento ser relativamente simples, podendo, portanto ser modelado utilizando outras técnicas mais simples, o uso de ontologia permite que esse conhecimento representado possa ser expandido e refinado, por exemplo, atribuindo informações que necessitem deste sofisticado mecanismo de inferência fornecido pelas ontologias. Isto é particularmente útil no sentido de criar uma base que permita a expansão do trabalho proposto nesta dissertação.

2.6 Catálogos de Processos de Negócio

Este seção apresenta um estudo do estado da arte em catálogos de processos de negócio.

Apesar do crescente interesse no gerenciamento de processos, ainda são poucos os trabalhos direcionados ao desenvolvimento de catálogos de processos de negócio (CHOI *et al.*, 2007).

O trabalho de (YAN *et al.*, 2009) é um dos pioneiros em justificar o uso destes catálogos. Segundo ele, está se tornando bastante comum nas organizações a descrição de suas operações através de processos de negócio. Isso implica numa dificuldade em organizar essa grande quantidade de processos, o que leva a questões como: (i) dificuldade em achar um processo dentro de um conjunto; (ii) gerenciar diferentes versões; e (iii) garantir consistência quando várias pessoas trabalham sobre um processo. O catálogo de processos é, portanto, proposto como uma forma de lidar com esses problemas. O seu trabalho idealiza um modelo e uma arquitetura de referência que são usados para avaliar alguns catálogos já existentes. O autor conclui que a utilidade do catálogo de processos está em:

- Servir como meio de guardar modelos de processos de negócio e de encontrá-los posteriormente, através do uso de algum critério;
- Guardar os modelos de processos e também suas instâncias de execução (nesse caso se foca na integração dele com ferramentas de execução de *workflow*);
- Permitir armazenar modelos de referência, que podem ser reutilizados para se desenvolver processos específicos da empresa.

Além de definir os requisitos e funcionalidades gerais do catálogo, é importante focar no modelo de dados que irá armazenar os processos. Nesse sentido, (SHAHZAD, ELIAS, *et al.*, 2009) propõe um modelo de armazenagem independente da linguagem de modelagem, tais como BPMN (OMG, 2006), YAWL (YAWL, 2009) etc. Isto proporciona uma independência do modelo do processo e de sua representação visual. A motivação do trabalho leva em conta que a modelagem de processos é uma tarefa complexa e demorada, que poderia ser simplificada pelo reuso dos modelos de processo. Segundo os autores, os catálogos de processos disponíveis são de domínio específico ou proprietários e não extensíveis. Dessa forma, o desenvolvimento de um catálogo de processos universal teria capacidade de mitigar essas limitações.

Com base na motivação exposta, pesquisou-se na literatura alguma das propostas existentes de catálogos de processos de negócio. De acordo com o que foi pesquisado, existem aproximadamente quinze

grandes propostas de catálogos, sendo que o (i) Process Handbook; (ii) SBPR – Semantic Business Process Repository; (iii) IBM BPEL Repository e (iv) RepoX têm uma maior relevância entre elas (YAN *et al.*, 2009). Uma análise detalhada desses catálogos, incluindo algumas considerações, é mostrada a seguir.

2.6.1 Process Handbook Project

O Process Handbook Project (MIT, 2005) é um projeto de catálogo de processos de negócio liderado pelo *Center of Coordination Science* do MIT.

O foco do Process Handbook está em organizar o conhecimento sobre os processos e não em fornecer modelos detalhados dos mesmos. Desta forma, o catálogo é baseado em descrições textuais, ao invés de modelos (YAN *et al.*, 2009). O catálogo conta com mais de três mil processos.

A categorização dos processos é feita com base no *business compass* (ver Figura 16) que possui quatro dimensões: (i) generalização, (ii) uso, (iii) especialização e (iv) parte. Cada um dos processos presentes no catálogo pode ser descrito através dessas dimensões.

A generalização (i) permite definir processos que capturam a informação comum entre um conjunto de processos. Dessa forma é possível definir processos que sirvam como base para outros.

A dependência que um processo pode ter em relação a outros é modelado através da dimensão uso (ii). Um processo de compra, por exemplo, pode depender de um processo de aprovação de crédito.

A especialização (iii) permite definir processos específicos. Por exemplo, um processo de aquisição pode ser especializado num processo de aquisição de material de expediente.

Na dimensão parte (iv), o processo pode ser visto descomposto em atividades, isto é, em partes. A aquisição, por exemplo, pode ser dividida em definir requisitos, selecionar fornecedores etc.

No *Process Handbook* o usuário pode, para cada processo, pesquisar suas generalizações, especializações, partes e tudo mais que o processo possuir. O catálogo também suporta busca textual.

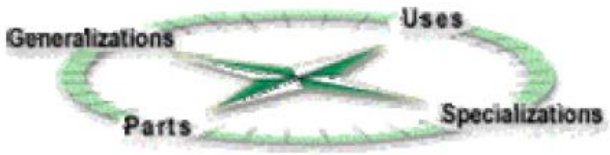


Figura 16 – MIT Process Handbook: Business Compass
Fonte: (MIT, 2005)

Os processos disponíveis no catálogo são agrupados em dez categorias: aquisição, logística, marketing, vendas, sistemas de informação, recursos humanos, planejamento estratégico, financeiro, manufatura e engenharia. (SHAHZAD, ANDERSSON, *et al.*, 2009).

O Process Handbook tem o mérito de ser uma das primeiras propostas de catálogo de processos de negócio. O fato de ser baseado em descrições textuais e não em uma linguagem formal constitui um grande entrave em relação ao contexto deste trabalho. Por mais que seja vasto na quantidade de processos, a forma como eles são representados não permite que sejam interpretados por mecanismos automatizados (CHOI *et al.*, 2007). Desta forma, a interoperabilidade entre processos e serviços acaba tendo que ser feita de forma manual, caso a caso, que, dentre outras desvantagens, não agiliza a integração BPM&SOA.

2.6.2 SBPR – Semantic Business Process Repository

O Semantic Business Process Repository (SBPR) (CHOI *et al.*, 2007) é um catálogo baseado em ontologias para o armazenamento de processos de negócio.

O SBPR não define um modelo de dados do processo específico, permitindo que o catálogo seja configurado para trabalhar com uma determinada ontologia de processo, por exemplo, BPMO (YAN *et al.*), sBPMN, sBPEL etc.

A busca é feita através do uso de WSML (*Web Service Modeling Language)* (DE BRUIJN *et al.*, 2005) utilizando o motor de inferência IRIS (*Integrated Rule Inference System*) (BISHOP *et al.*, 2008). Este permite fazer buscas semânticas com um poder de expressão maior do que linguagens como SQL.

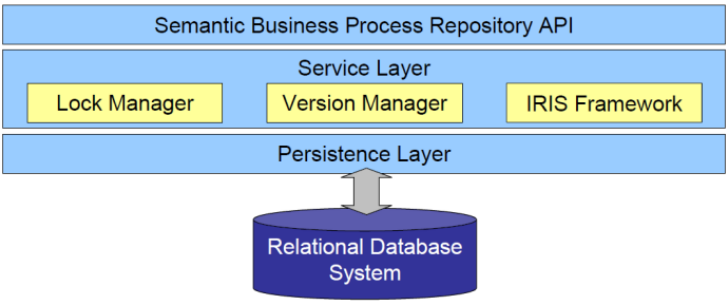


Figura 17 – Arquitetura: Semantic Business Process Repository
Fonte: (MA *et al.*, 2006)

Na Figura 17 mostra-se a arquitetura do SBPR, que é composta pelos componentes *Semantic Business Process Repository API*, *Service Layer* e *Persistence Layer*.

A *Semantic Business Process Repository API* provê uma camada de acesso às funções do catálogo. Estas incluem as funções básicas de obter, inserir, editar e remover processos, além de funções de busca e controle de versão.

A *Service Layer* implementa a API da camada superior e a lógica do catálogo e é dividida em três módulos: (i) Lock Manager, (ii) Version Manager e (iii) IRIS Framework. O Lock Manager (i) é responsável pelas operações de *lock* e *unlock* dos modelos de processos. Estas operações são efetuadas quando um usuário está editando um processo e não deseja que outros tenham acesso a ele. O *unlock* ocorre quando o trabalho é finalizado e o processo se torna disponível aos demais. O Version Manager (ii) lida com o controle de versionamento e de mudança. O IRIS Framework (iii) é responsável pelo mecanismo de busca no catálogo.

A *Persistence Layer* provê acesso e gerencia os dados armazenados no catálogo. Através dele é possível abstrair a forma como os dados são persistidos. Por padrão, o SBPR utiliza um banco de dados relacional para armazenar os dados.

O Semantic Business Process Repository é um catálogo que utiliza ontologias para o armazenamento dos processos. Dessa forma, ele consegue atribuir semântica e, por conseguinte, capacidade de inferência em cima dos processos. Isso é uma característica positiva e que vai ao encontro das necessidades deste trabalho. A arquitetura em camadas e módulos do catálogo merece destaque, pois permite separar

concisamente as responsabilidades de armazenagem de dados, versionamento, busca etc.

Por outro lado, SBPR não possui o código fonte disponível, de forma que não é possível obter maiores informações sobre a sua implementação. Outra desvantagem do catálogo é que o mesmo é uma aplicação isolada, não podendo ser integrado a um editor de processos de negócio, por exemplo. Essas desvantagens fazem com que o SBPR não seja um bom candidato de catálogo de processos de negócio num ambiente de integração BPM&SOA.

2.6.3 IBM BPEL Repository

O IBM BPEL Repository (VANHATALO *et al.*, 2006) é um *plug-in* desenvolvido para a plataforma Eclipse que tem o intuito de armazenar e recuperar processos de negócio, já em formato de execução, descritos na linguagem BPEL. Esta linguagem é usada para descrever processos passíveis de serem executados por máquinas, ao contrário do BPMN, que é usado para descrever processos em termos mais genéricos, independentemente da tecnologia de implementação.

Nesse catálogo, a armazenagem interna da informação é feita utilizando o padrão XML. Externamente, o modelo de dados pode ser exposto como objetos Java, o que permite que aplicações trabalhem diretamente com estes objetos, abstraindo a complexidade de se trabalhar com modelos XML. Dessa forma possibilita que o desenvolvedor enfoque no modelo de objetos da sua aplicação. O *Eclipse Modeling Framework* (EMF) (ECLIPSE, 2009) é utilizado para fazer a serialização e desserialização entre objetos Java e XML. Isto permite que qualquer arquivo XML possa ser armazenado no catálogo, trazendo flexibilidade no uso do mesmo.

A armazenagem dos dados é feita através de arquivos, ao contrário de outros catálogos que utilizam mecanismos baseados em banco de dados (BD). A vantagem dessa abordagem é permitir que novos tipos de documentos sejam persistidos no catálogo, sem ter que se preocupar em alterar os esquemas relacionais da base de dados. Por outro lado, soluções baseadas em BD tipicamente possuem um desempenho e escalabilidade superior (VANHATALO *et al.*, 2006).

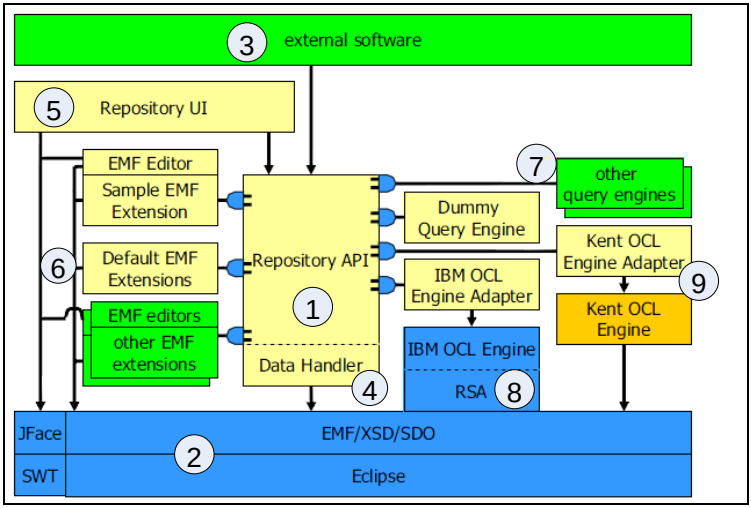


Figura 18 – Arquitetura: IBM BPEL Repository
(Fonte: VANHATALO et al., 2006)

Na Figura 18 mostra-se a arquitetura do catálogo. Todos os componentes são *plug-ins* da plataforma Eclipse. O *Repository API* (1) é o componente central, responsável por prover as funções do catálogo. Ele interage com os componentes da plataforma Eclipse (2) e com *softwares* externos (3), através do uso da API. O *Data Handler* (4) é responsável pelo acesso aos dados no sistema de arquivos. Ele abstrai a escolha do meio de armazenamento dos outros componentes. A *Repository UI* (5) é uma interface gráfica para manipulação dos processos do catálogo.

As extensões EMF (6) são responsáveis por prover os modelos orientados a objetos dos dados do repositório. Por padrão há modelos para o BPEL, WSDL e XML-Schemas. O catálogo pode ser estendido para outros formatos através do desenvolvimento de novos componentes desse tipo.

A busca no catálogo é feita através do uso de uma linguagem de busca orientada a objetos chamada de *Object Constraint Language* (OCL). Diferentes implementações de motores de busca (7) baseados em OCL podem ser utilizadas em conjunto com o catálogo, como, por exemplo, a implementação do IBM Rational Software Architect (8) ou da Universidade de Kent (9).

O BPEL Repository foi publicado no IBM alphaWorks (IBM, 2006), estando disponível através de uma licença especial para uso acadêmico.

O IBM BPEL Repository tem a característica de ser integrado a um editor de BPM, no caso baseado na plataforma Eclipse. Esta integração com um editor de processos de negócio é um requisito deste trabalho, portanto sendo um ponto positivo deste catálogo. O uso da linguagem BPEL permite que se atribua informações relativas aos serviços *web* implementadores do processo, de forma que esse catálogo possa fazer parte de um ambiente SOA, onde os processos são executados através do uso de serviços.

Por outro lado, o uso de BPEL como linguagem para manipulação de processos pelo usuário também é uma desvantagem em termos da facilidade de desenvolvimento de aplicações BPM&SOA. Isso ocorre porque BPEL é uma linguagem técnica, que demanda conhecimento especializado, não acessível de modo geral às pessoas responsáveis por modelar processos de negócio nas empresas.

Outro ponto negativo é que o catálogo não permite associar informações semânticas junto os processos. Isso inviabiliza o uso de mecanismos de descoberta de serviços, já que não é possível criar um expressão de busca que leve em conta as características do processo, em termos de semântica.

2.6.4 RepoX

RepoX (SONG *et al.*, 2001) é um catálogo de modelos de processos baseado no padrão XML. Ele faz parte do projeto METEOR (METEOR, 2001), que é um sistema de *workflow*. Ele foi projetado para padronizar a troca de dados entre as ferramentas de modelagem de processo e o sistema executor de *workflow*.

O catálogo proposto facilita o desenvolvimento de especificações de *workflow* e o gerenciamento de documentos XML associados a elas. A Figura 19 apresenta a arquitetura do RepoX:

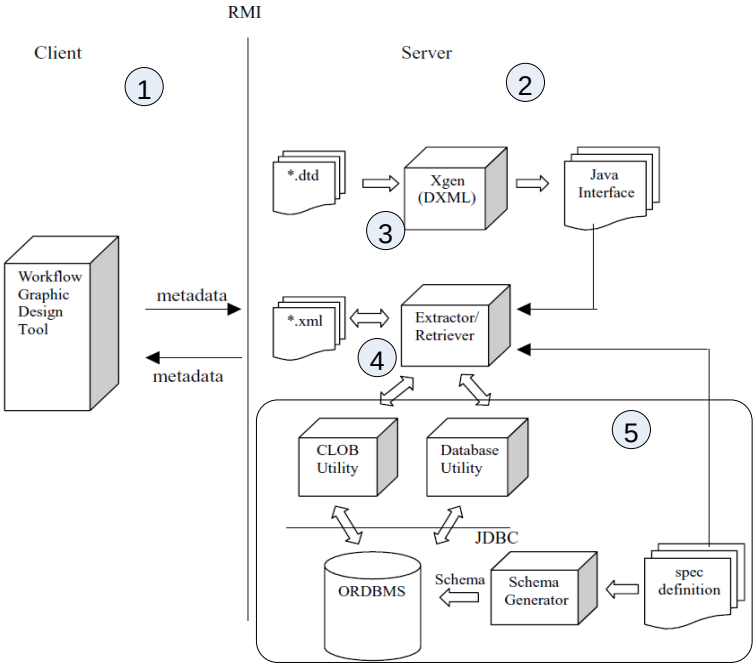


Figura 19 – Arquitetura RepoX
Fonte: (SONG *et al.*, 2001)

A arquitetura do catálogo é composta de um módulo cliente e servidor conectados através do uso de *Remote Method Invocation* (RMI) do Java.

O módulo cliente (1) é composto de uma interface gráfica que é integrada a uma ferramenta de *workflow* desenvolvida pelo Naval Research Center. O usuário desenvolvedor do workflow constrói o seu processo utilizando a ferramenta e em seguida salva o *workflow* (baseado em XML) no catálogo. A interface gráfica também possui uma ferramenta de pesquisa, que permite que o usuário faça buscas dentro do catálogo.

O módulo servidor (2) é o responsável pelo armazenamento e gerenciamento dos processos. Cada tipo de documento XML, antes de ser armazenado no banco, precisa ter seu DTD (*Document Type Definition*) analisado e processado pelo catálogo, que irá criar as tabelas no banco de dados referentes aos dados do documento. Isso ocorre em (3). Depois dessa primeira etapa, o documento pode ser armazenado ou

recuperado em (4). O mecanismo responsável pela persistência é mostrado em (5). O catálogo guarda aspectos do controle de fluxo, junto com os dados e as autorizações necessárias para se executar o processo. O catálogo foi projetado para guardar somente processos específicos de uma empresa, não sendo indicado para se guardar processos genéricos.

O RepoX também possui funcionalidades de gerência de versão e configuração. Diferentes versões de tarefas individuais, sub-processos ou processos inteiros podem ser armazenados. Além disso, é possível armazenar informações de configuração que dizem respeito a qual versão de tarefa ou sub-processo é parte de determinada versão do processo.

Assim como o IBM BPEL Repository, ele é integrado com uma ferramenta de edição de processos, dessa forma permitindo que o ambiente de modelagem de processos possa trabalhar em conjunto com o catálogo.

Uma das desvantagens do RepoX é o fato de não permitir a associação de semântica aos processos, que dificulta a integração BPM&SOA no sentido da descoberta e vinculação de serviços. Além disso, o código fonte do catálogo não está disponível, de forma que dificulta o estudo mais aprofundado dos aspectos de implementação do mesmo.

2.6.5 Critérios de Avaliação – Requisitos do Catálogo

Esta seção tem como objetivo elicitar os requisitos funcionais e não funcionais desejados em um catálogo de processos de negócio. Tais requisitos serão utilizados para avaliar os repositórios pesquisados e definir a aderência aos requisitos desejados.

A análise de requisitos significa decidir quais as funcionalidades o novo *software* deve ter. Deve-se ser claro na especificação quais as funcionalidades que o sistema deve ter e o que não deve ter. Isso evita um trabalho desnecessário e perda de foco na fase de implementação (O'DOCHERTY, 2005).

O trabalho de (SHAHZAD, ANDERSSON, *et al.*, 2009) faz um *survey* de alguns catálogos de processos existentes e extrai requisitos desejáveis. Estes dizem respeito ao modelo de armazenamento de dados e ao repositório em si. Na Tabela 3, mostram-se os requisitos:

Tabela 3 – Requisitos do Repositório
Fonte: (SHAHZAD, ANDERSSON, *et al.*, 2009)

Requisitos para o Modelo de Dados	
Reusável	É desejável que o conteúdo do repositório possa ser reutilizável com pouca ou nenhuma modificação.
Independente de Linguagem	O conteúdo armazenado no repositório deve ser independente de qualquer linguagem de modelagem de processos de negócio.
Independente de um domínio específico ou de uma organização	Significa que o repositório deve guardar processos que sejam interorganizacionais, isto é, que não sejam específicos de uma determinada organização.
Requisitos para o Repositório	
Extensível	Deve ser possível acomodar futuras expansões no conteúdo do repositório. Deve ser possível, por exemplo, adicionar novos modelos de processos ou alterar os já existentes.
Flexível	Poder armazenar múltiplas variantes de um mesmo processo. Essas variantes são fruto de customizações feitas a fim de permitir o reuso.
Sem Restrições	O repositório deve ser livre de qualquer restrição legal. Isso significa que seus usuários podem editar, remover, atualizar processos sem necessidade de nenhuma permissão legal por parte dos criadores do repositório.
Aceitação	A organização do conteúdo do repositório, de acordo com vários esquemas de classificação, é permitido e encorajado.
Usabilidade	O repositório deve prover acesso fácil e intuitivo ao seu conteúdo, preferencialmente utilizando uma interface gráfica.
Navegável	Deve-se prover suporte a navegação e busca para que seus usuários possam achar o conteúdo pretendido.

Baseado nos requisitos levantados em seu trabalho, utilizou-se os requisitos de reusabilidade, independência de linguagem e flexibilidade no modelo proposto nesta dissertação. Os demais requisitos não foram levados em conta, pois não são essenciais ao escopo do

trabalho, no sentido de contribuir para a agilidade na integração BPM&SOA. Em especial, o requisito sem restrições não parece ser muito realista, no sentido que geralmente existem normas que restringem a manipulação dos processos. Portanto, não é comum qualquer usuário ter a permissão de remover, editar e atualizar os processos. Somente alguns usuários, que exercem posições especiais dentro da empresa, costumam possuir permissão para tal.

Outros requisitos desejáveis dizem respeito ao trabalho de (SOUZA, 2009). São eles: i) a capacidade do catálogo de possuir semântica associada ao processo (uso de ontologias); ii) possuir processos padronizados (usando uma especificação como UBL, por exemplo); iii) ser integrado com SOA; iv) ser disponibilizado como um serviço (arquitetura SOA) e v) ser integrado com um editor de BPM.

A semântica associada ao processo permite capturar o contexto de negócios associado ao mesmo e facilita o trabalho da descoberta dos serviços implementadores do processo. Aliado a este, o catálogo deve utilizar uma especificação padronizada de processos, pois permite mitigar questões de interoperabilidade através do uso de ontologias que implementem estas especificações.

Outro requisito necessário ao catálogo é a integração com a arquitetura orientada a serviços (SOA). Isto significa que o catálogo deve ser capaz de armazenar e gerenciar processos de forma que eles possam ser implementados através do uso de serviços de *software*. Algumas linguagens, como BPEL, permitem vincular serviços *web* às atividades do processo. Dessa forma, o processo por ser executado através da invocação coordenada dos serviços que o implementam (DORN *et al.*, 2007). A vinculação pode ser estática ou dinâmica. Na vinculação estática os respectivos serviços já estão definidos à priori e invocados na execução do processo. Na vinculação dinâmica, informações semânticas são armazenadas junto ao processo, permitindo que se busque, no momento da execução, possíveis serviços candidatos a executar o processo.

O catálogo deve ser disponibilizado como um serviço *web*, pois fará parte de uma arquitetura SOA, caracterizada por ter seus componentes (serviços) distribuídos e interoperáveis. A comunicação com o serviço é feita através de protocolos padronizados, de forma que o catálogo, implementado como serviço, expõe suas funcionalidades através de uma *interface*. Qualquer sistema que tenha a capacidade de utilizar os protocolos usados nos serviços *web* tem condição de utilizar o

catálogo. Nesse contexto, portanto, a disponibilização do catálogo como um serviço corrobora com uma maior interoperabilidade.

A integração do catálogo com o ambiente de edição BPM permite que o usuário-projetista utilize o catálogo de maneira transparente, como se este fizesse parte do editor BPM.

Por último, deseja-se que o catálogo seja livre ou extensível, de forma que o mesmo possa ser modificado com a inclusão de novas funcionalidades.

Com base nos requisitos acima a Tabela 4 faz um resumo dos requisitos que serão utilizados para avaliar os catálogos estudados frente ao catálogo proposto.

Tabela 4 – Requisitos para a avaliação dos catálogos

Requisito	Descrição
Processos Reusáveis	Um catálogo de processos deve servir como um facilitador ao reuso de processos já existentes.
Independente de linguagem	Os processos são armazenados num formato independente de linguagem de modelagem.
Flexível	Flexível no sentido que os processos muitas vezes precisam ser customizados para atender certas necessidades e dessa forma suas variantes precisam ser armazenadas no catálogo.
Semântica Associada	Como forma de capturar o contexto semântico e facilitar o processo de descoberta de serviços.
Especificação padronizada	Os processos devem ser descritos de acordo com alguma especificação de processos de negócio.
Integração com SOA	O catálogo deve armazenar os processos de forma que possam ser integrados e executados num ambiente SOA.
Ser um serviço	O catálogo deve ser implementado na forma de um serviço, pois fará parte de um ambiente que utiliza a arquitetura SOA.
Integração com Editor BPM	Permitir que o usuário utilize o catálogo de maneira transparente, no mesmo ambiente que usa para modelar os processos.
Código Aberto	O código-fonte do catálogo deve estar disponível, de forma que ele possa ser extensível. Soluções proprietárias não permitem isso, de forma que não são desejadas.

2.6.6 Avaliação dos Catálogos

Usando os critérios expostos na seção anterior, avaliaram-se os catálogos com respeito à aderência aos critérios, bem como se identificou onde o catálogo a ser proposto pretende se diferenciar frente aos demais. O resultado dessa análise está exposto na Tabela 5.

Tabela 5 – Avaliação dos Requisitos dos Catálogos

Requisito	Process Handbook	SBPR	IBM BPEL	RepoX	Modelo Proposto
Processos Reusáveis	Sim	Sim	Sim	Sim	Sim
Independente de linguagem	Sim	Não	Não	Não	Sim
Flexível	Não	Sim	Sim	Sim	Sim
Semântica Associada	Não	Sim	Não	Não	Sim
Especificação padronizada	Não	Não	Não	Não	Sim
Integração com SOA	Não	Não	Sim	Não	Sim
Ser um serviço	Não	Não	Não	Não	Sim
Integração com editor BPM	Não	Não	Sim	Sim	Sim
Código Aberto	Não	Não	Não	Não	Sim

A reusabilidade dos processos é um ponto chave e uma motivação para o uso catálogos de processos de negócio. Dessa forma, é de se esperar que todos os catálogos estudados estivessem em conformidade com esse requisito.

A independência de linguagem está presente no Process Handbook. Isto é esperado, já que os seus processos são descritos de forma textual. O SBPR deixa ao encargo do usuário definir a ontologia que será usada para armazenagem, portanto ficando dependente desta. IBM BPEL utiliza a linguagem de modelagem BPEL e o RepoX estrutura a base de dados de acordo o DTD do arquivo XML, ficando desta forma dependente deste.

A flexibilidade permite estender o catálogo adicionando novos processos ou editando os existentes. Com exceção do Process

Handbook, que possui seus processos já descritos e imutáveis, os demais catálogos admitem extensões em seus processos.

A semântica associada diz respeito à definição ontológica que o modelo de processos deve ter. Isto significa que devem haver ontologias que descrevam de forma precisa os processos. Nesse requisito, somente o SBPR faz isso, pois utiliza ontologias para permitir o armazenamento de processos em seu catálogo.

A especificação padronizada diz respeito ao uso de algum padrão para a definição dos processos. Alguns exemplos de padrões são a UBL (OASIS, 2006b) e a RosettaNet (ROSETTANET, 2008). Nenhum dos catálogos pesquisados utiliza ou se baseia em alguma dessas padronizações na definição de seus processos.

No que diz respeito à integração com SOA, somente o IBM BPEL está preparado para tal. Isto porque ele armazena os processos no formato BPEL, considerado o padrão *de facto* para execução e orquestração de processos de negócio num ambiente SOA (KAMOUN, 2007). Apesar disso, BPEL não é uma linguagem acessível a pessoas sem conhecimento técnico, de forma que de modo geral não é utilizável pelos usuários-projetistas de processos. Isso é um entrave para a agilidade na concepção de aplicações BPM&SOA.

No requisito disponibilidade como um serviço, nenhum dos catálogos cumpre esta função. O *Process Handbook* é disponibilizado em um *site*, contendo somente informações na forma textual. O SBPR, IBM BPEL e RepoX são aplicações que utilizam *interfaces* específicas de comunicação e dessa forma não foram projetadas para atuarem como serviços *web*.

A integração com o editor BPM permite que o catálogo seja visto de forma transparente pelo usuário. Nesse sentido, os catálogo IBM BPEL Repository e RepoX possuem integração com editores de processos. IBM BPEL utiliza a arquitetura de *plug-ins* do Eclipse para se integrar com editores baseados nesta plataforma. O RepoX se integra com uma ferramenta de *workflow* desenvolvida pelo Research Center.

Nenhum dos catálogos estudados mostrou-se livre ou extensível, no sentido que não foi possível ter acesso ao código fonte deles. Dessa forma torna-se muito difícil aproveitá-los ou estendê-los.

2.7 Considerações

A Figura 20 resume a relação entre os temas pesquisados e o escopo deste trabalho:

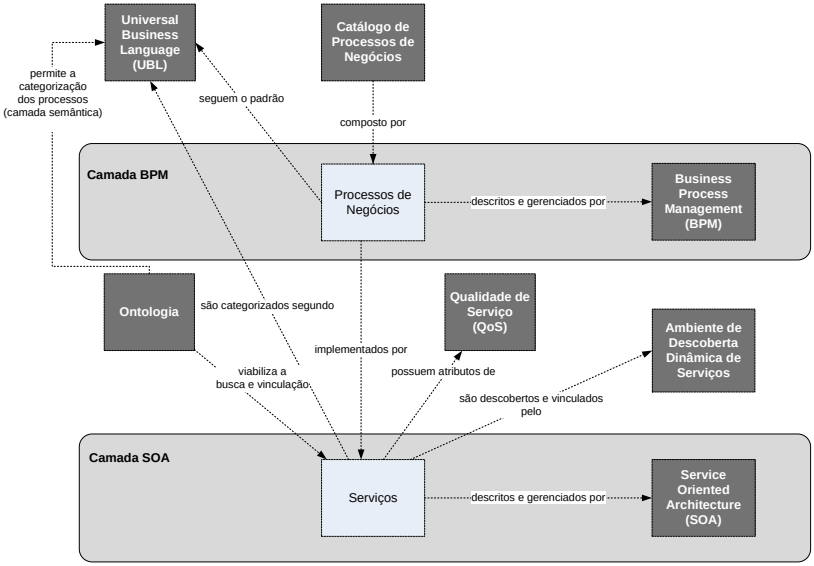


Figura 20 – Relação entre os temas pesquisados e o escopo do trabalho

O conceitos presentes em BPM são utilizados para descrever os aspectos do processo como, por exemplo, a forma como são modelados, compostos, descritos etc. O analista de negócios é a pessoa envolvida nesse nível, sendo responsável por modelar os processos da organização.

A especificação UBL permite o uso de processos padronizados, o que poupa tempo do projetista e permite incluir informações semânticas aos processos.

A ontologia permite modelar o conhecimento acerca dos processos de negócio contidos na especificação UBL, permitindo atribuir informações semânticas a estes. Essa informação permite que estes processos sejam categorizados e viabiliza a busca e vinculação dos serviços *web* a estes.

Optou-se pelo uso de UBL, em detrimento do RosettaNet e ebXML, porque este primeiro é uma especificação gratuita e voltada a definição de processos de forma horizontal, possuindo uma maior abrangência na quantidade de processos. RosettaNet está voltada em definir padrões verticais, dando ênfase à indústria de componentes eletrônicos. Além disso, é uma especificação paga, e com acesso restrito. No caso de ebXML, a especificação não está sendo muita usada

atualmente, não sendo atualizada, visto que o seu catálogo de processos de negócio possui sua última versão datada de 2001.

SOA é a arquitetura utilizada nas implementações dos processos e da arquitetura. Ela é caracterizada pelo uso de serviços, disponíveis na *web* e que possuem uma alta reusabilidade e interoperabilidade. Estes serviços, possuem atributos de qualidade de serviço (QoS), por exemplo tempo de resposta, confiabilidade, etc.

O ambiente de descoberta dinâmica de serviços permite fazer a busca e ligação entre os processos e serviços, levando em conta a qualidade de serviço desejada e a classificação ontológica UBL dos serviços.

O catálogo de processos de negócio é composto por processos que estão de acordo com a especificação UBL e descritos e gerenciados segundo os preceitos de BPM.

Os serviços fazem parte da camada SOA e implementam as atividades dos processos. A descoberta desses serviços e a sua vinculação aos processos é feita através do Ambiente de Descoberta Dinâmica de Serviços.

Capítulo 3

Modelo Conceitual do Catálogo

Detalhando o objetivo do trabalho, descrito na seção 1.5, juntamente com os requisitos levantados no capítulo 2, esse capítulo apresenta o modelo conceitual de uma abordagem para um catálogo de processos de negócios que visa agilizar a integração entre os níveis BPM e SOA.

Modelos conceituais permitem a construção de abstrações que podem ser usadas no desenvolvimento de arquiteturas. Dessa forma, um modelo conceitual consiste de um conjunto de relações e conceitos, dentro de um domínio em específico, independente de padrões, tecnologias, implementações e outros detalhes mais concretos (OASIS, 2006a).

A primeira parte do capítulo apresenta a proposta do trabalho. Em seguida, na segunda parte, os requisitos do catálogo, mostrados de maneira mais ampla no Capítulo 2, são refinados em requisitos funcionais e não funcionais. A terceira parte apresenta os atores e casos de uso envolvidos no sistema. Na quarta parte, mostra-se a arquitetura conceitual do sistema, finalizando com algumas conclusões e considerações.

3.1 Proposta

A proposta deste trabalho é essencialmente desenvolver um catálogo de processos baseado na especificação UBL que seja integrado a um editor BPM. No entanto, este catálogo é um dos elementos de um modelo mais amplo, sendo desenvolvido numa tese de doutorado também dentro do PPGEAS/Grupo GSIGMA, de descoberta de serviços em repositórios largamente distribuídos. Estes repositórios são logicamente agrupados numa estrutura chamada de Federação de Serviços, onde os mais diversos provedores de serviços de *software* publicam e disponibilizam serviços usando uma semântica comum (no caso, de acordo com as especificações da UBL) e dentro de certas especificações de QoS (SOUZA, 2009).

De maneira sucinta, o ambiente proposto por (SOUZA, 2009) divide o problema da descoberta de serviços em duas fases: fase de projeto (*off-line*) e fase de execução (*on-line*), conforme ilustrado na Figura 21:

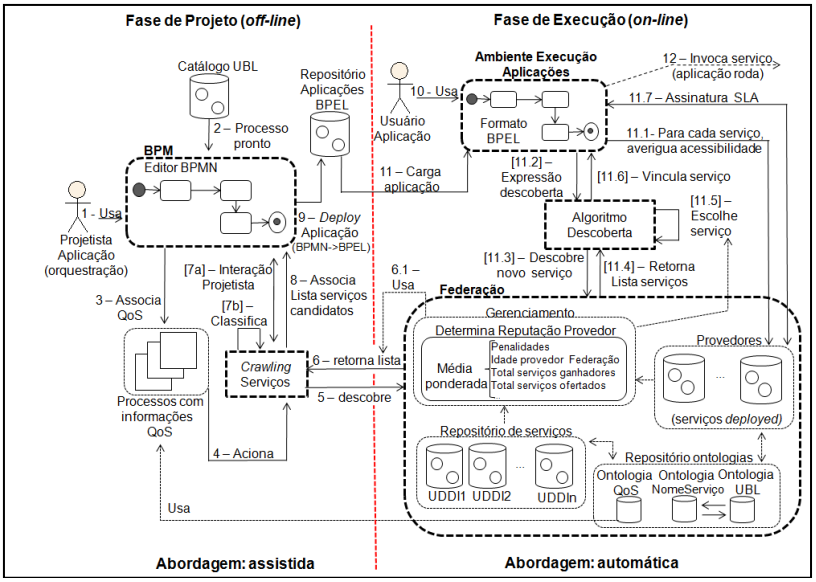


Figura 21 – Visão geral do modelo para descoberta de serviços web
Fonte: (SOUZA, 2009)

A fase *off-line* (lado esquerdo da Figura 21) se refere à especificação da aplicação SOA, onde são definidos o contexto de negócios e os parâmetros de QoS. Ambos são de extrema importância para o processo de descoberta de serviços. Enquanto faz a modelagem da aplicação, o usuário-projetista conta com o auxílio do ambiente, representado pelo editor BPM (passo 1). Esse ambiente provê acesso a um catálogo de processos de negócio, que está em conformidade com a especificação UBL (passo 2). Isso permite que as aplicações tomem como base processos já padronizados e as suas especializações. Dessa forma, o contexto de negócios da aplicação é definido pela utilização dos processos do catálogo. O uso dessa abordagem acelera o projeto de aplicação, além melhorar o processo de descoberta, já que os serviços são publicados levando em conta a semântica presente na especificação UBL.

A cada atividade do processo, o projetista atribui critérios de QoS (passo 3), que são definidos numa ontologia específica que também é utilizada pelos provedores que publicam os serviços. No passo 4, a requisição é enviada ao mecanismo de *crawling*, que busca serviços candidatos que correspondam aos requisitos de negócios, incluindo os critérios de QoS (passo 5). A lista dos serviços encontrados é retornada (passo 6) e pode conter nenhum, exatamente um ou mais serviços. No primeiro caso, o sistema sugere ao projetista que utilize critérios de QoS menos rigorosos a fim de tentar achar um candidato, ou manter os critérios originais (e o risco) e fazer a busca em tempo de execução. No segundo caso o sistema automaticamente vincula o candidato único à atividade do processo correspondente e no terceiro caso o projetista tem a opção de escolher qual deles vincular. Esta interação entre o sistema e o projetista é representada no passo 7. No passo 8, através da associação dos serviços às atividades, a aplicação SOA é construída e em seguida convertida para o formato BPEL e exportada para o Repositório de Aplicações BPEL, o que a torna apta para ser executada (passo 9).

Na fase *on-line* (lado direito da Figura 21) o usuário-executor (não mais o projetista) irá navegar dentro do "Repositório de Aplicações BPEL" e escolher qual aplicação deseja executar (passos 10 e 11). O sistema verifica se os serviços que foram previamente vinculados às atividades estão disponíveis (passo 12). Caso algum deles não esteja, o mecanismo de busca irá procurar serviços candidatos, de maneira semelhante a que ocorre na fase *off-line*. Finalmente, a aplicação projetada pode ser executada, através do motor de execução BPEL (passo 12).

O trabalho de (SOUZA, 2009) faz uso de algumas hipóteses, usadas para sustentar o seu modelo: i) diz que as atividades do processo são somente efetuadas por serviços *web*, isto é, sem intervenção humana; ii) se refere ao uso da especificação e de ontologias de QoS como forma de mitigar os problemas de interoperabilidade semântica; iii) os serviços estão registrados em repositórios distribuídos e autônomos chamados UDDI's e acessados automaticamente através do algoritmo de descoberta; iv) existe uma relação 1:1 entre a atividade e o serviço que a executa, isto é, somente 1 serviço executa determinada atividade; v) o usuário-projetista é uma pessoa experiente que conhece os processos da empresa, a especificação UBL e sabe quais requisitos de QoS determinada serviço deve ter.

A Figura 22 posiciona a contribuição desta dissertação nesse modelo, explicitando as três partes (1, 2 e 3) onde este trabalho se reflete, em particular a parte 1, que é o seu foco.

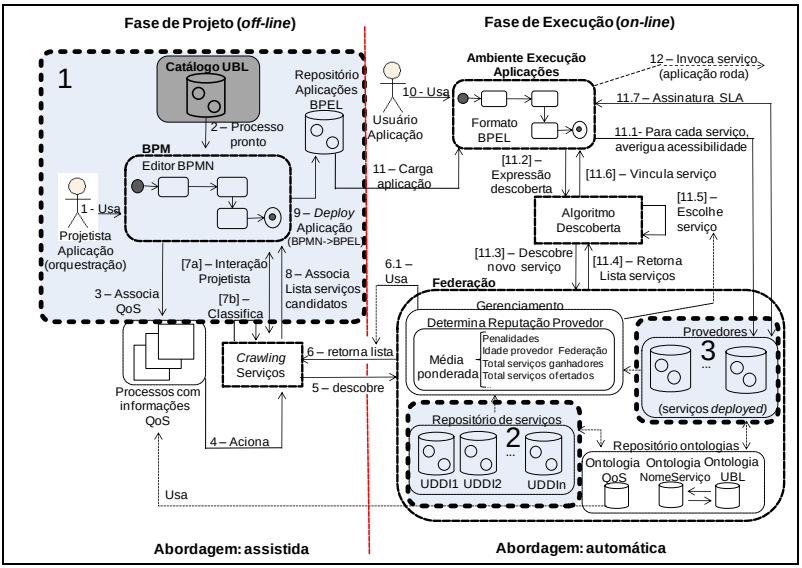


Figura 22 – Escopo do Trabalho (adaptado de: SOUZA, 2009)

Na fase de projeto, em 1, o catálogo UBL é integrado ao editor BPM e interage com o ambiente de descoberta, permitindo a associação de critérios de QoS e a busca semântica e associação de serviços.

O processo pronto e com seus serviços vinculados deve ser exportado para um formato de execução e implantado num repositório de aplicações. Esse repositório deve ter um motor de execução capaz de orquestrar os diversos serviços vinculados, bem como invocar o mecanismo de descoberta, caso alguns dos serviços que o processo requeira não estejam disponíveis, e seja necessário encontrar substitutos para eles.

Na fase de execução, em 2 e 3, implementam-se alguns serviços *web* vinculados à especificação UBL (em 3), definindo valores de QoS para os mesmos e em seguida publicando-os em repositórios (em 2). Esse procedimento tem como intuito avaliar a capacidade do ambiente em fazer a busca e a vinculação dos serviços às atividades do processo.

De acordo com a abordagem descrita na seção 2.1.2, onde foi apresentado o método tradicional de modelagem e implantação de processos BPM, este trabalho é diferente nos seguintes aspectos:

- Existe um catálogo de processos padronizados, no caso fornecidos pela especificação UBL, de forma que não é necessário que o processo seja totalmente construído do zero, como ocorre nas fases 1 e 2 da abordagem tradicional. Isso porque o catálogo permite o reuso de processos e a sua utilização como base para a concepção de novas aplicações BPM&SOA;
- A conversão do processo para a linguagem de execução, que na abordagem tradicional ocorre na fase 4, é transparente ao usuário, visto que o catálogo junto com o ambiente de descoberta de serviços se encarrega de fazer a vinculação dos serviços que irão compor o processo. Esses serviços seguem estritamente a especificação UBL, de forma a garantir a interoperabilidade. Não é necessário, portanto, que a equipe de TI precise fazer adaptações no código executável criado pelo trabalho proposto.

No desenvolvimento desse trabalho, utilizou-se o maior número possível de padrões e especificações, a fim de permitir uma maior interoperabilidade com ambientes heterogêneos e garantir que o uso das boas práticas presentes fossem assimiladas neste trabalho.

3.2 Requisitos Funcionais e não Funcionais

Os requisitos funcionais e não funcionais do ambiente geral do catálogo proposto foram elaborados com base na seção 2.6.5 e no trabalho de (SOUZA, 2009) e são apresentados na Tabela 6.

Tabela 6 – Requisitos do Catálogo

Requisitos Funcionais	
Buscar Processos	Permitir ao usuário buscar processos presentes no catálogo
Importação de Processo do Catálogo	O processo deve poder ser importado para a ferramenta BPM, a fim de se fazer adaptações e edições eventualmente necessárias para reuso.
Exportação de Processo para o Catálogo	Deve ser possível exportar um processo para o ambiente do catálogo.
Definição de QoS	Deve permitir associar restrições de QoS aos serviços que se vincularão ao processo
Descoberta e Vinculação de Serviços	Permitir a descoberta e associação de serviços que atendam a classificação ontológica desejada para o processo e aos critérios de QoS definidos.
Exportação para Linguagem Executável	Deve ser possível exportar o processo para um formato que permita a execução dos respectivos serviços vinculados.
Executar o processo num ambiente de execução	O processo deve ser executado a partir da invocação orquestrada dos serviços que o compõem. O usuário deverá ter acesso aos processos disponíveis e escolher qual deseja executar.
Requisitos não Funcionais	
Reusável	Servir como um facilitador ao reuso de processos já existentes.
Independente de linguagem	Permitir que a representação dos dados do processo no catálogo não fique atrelada a uma determinada linguagem de modelagem de processos de negócio.
Flexível	Possibilitar que processos previamente armazenados possam ser customizados para atender certas necessidades, ou seja o desenvolvimento de aplicações SOA, e suas variantes serem armazenadas no ambiente do catálogo.

Usar o padrão UBL	Os processos descritos no catálogo devem ser dispostos e implementados de acordo com a Universal Business Language (UBL).
Ser uma composição de serviços	O catálogo deve ser visto com um conjunto de serviços, pois fará parte de um ambiente e arquitetura SOA. Essa composição implica que os módulos da arquitetura são modelados como serviços *web*, utilizando protocolos interoperáveis.
Usar ontologias UBL e QoS	Os processos armazenados no repositório devem utilizar ontologias para mitigar problemas de interoperabilidade semântica. No caso do catálogo, deve conter ontologias UBL e QoS, para permitir sua integração com um ambiente de descoberta dinâmica de serviços.
Permitir a execução do processo por humanos	Os processos executáveis gerados pelo sistema devem ser passíveis de serem executados por humanos que exerçam papéis no processo modelado.

3.3 Atores e Casos de Uso

Um caso de uso é uma descrição de como o sistema é usado por seus usuários ou por outros sistemas. O seu diagrama mostra como eles estão relacionados uns com os outros e como os usuários se relacionam com eles. Casos de uso são considerados a ferramenta mais eficiente para descrever requisitos funcionais. Um caso de uso bem especificado pode inclusive derivar muitos dos requisitos não funcionais (O'DOCHERTY, 2005).

O catálogo proposto possui dois atores: o **Usuário-Projetista de Aplicações** e o **Usuário-Executor de Aplicações**.

O usuário-projetista atua na fase de concepção da aplicação SOA, interagindo com o editor BPM. A Figura 23 mostra as operações que o usuário executa.

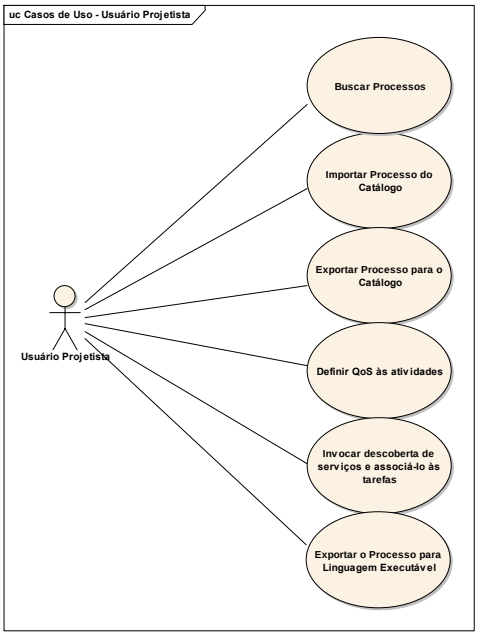


Figura 23 – Casos de Uso – Usuário-Projetista

O usuário-projetista tem acesso ao catálogo, onde pode buscar um processo e em seguida importá-lo. Através das funcionalidades do ambiente BPM, ele pode alterar a dinâmica do processo, por exemplo, alterando o seu fluxo de execução. Em seguida, poderá definir restrições de qualidade de serviço às tarefas do processo e invocar o mecanismo de descoberta a fim de fazer a vinculação dos serviços com essas tarefas. Em seguida, pode salvar o processo no catálogo e exportá-lo para uma linguagem de execução e fazer a implantação (*deployment)* no Ambiente de Execução.

Já o ator Usuário-Executor de Aplicações irá interagir com esse ambiente, conforme Figura 24.

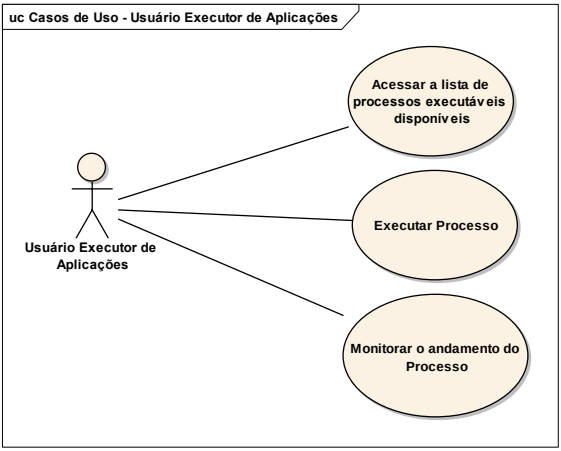


Figura 24 – Casos de Uso do Usuário-Executor de Aplicações

O usuário tem acesso à lista dos processos disponíveis, podendo executá-los e monitorá-los, por exemplo, vendo quais das suas instâncias foram completadas, quais estão paradas em determinada atividade, quais terminaram com erros etc.

3.4 Visão e Arquitetura Conceitual

Esta seção apresenta a arquitetura conceitual da abordagem proposta, sem se atrelar a nenhuma tecnologia de implementação, seja para os serviços seja para o catálogo. O próximo capítulo apresenta uma "instância" de implementação desta arquitetura na forma de um protótipo computacional. O modelo proposto pretende refletir uma visão de um avançado ambiente de integração entre os níveis BPM&SOA, potencializando um ambiente transparente, de fácil uso e fortemente baseado nos vários padrões utilizados atualmente, além de propiciar o reuso e adaptações de processos de negócio e sua praticamente imediata transformação numa aplicação SOA. Já do lado de provedores de serviços, prevê-se num futuro próximo uma grande quantidade de serviços sendo desenvolvidos e disponibilizados na "nuvem" (numa federação) segundo especificações padronizadas e encontrados através de mecanismos inteligentes, que consideram diversos modelos de negócios de acesso, QoS e ciência do contexto do processo e ambiente computacional existentes para a sua execução.

Este modelo conceitual do catálogo proposto assume os seguintes pressupostos:

(1) O usuário-projetista é uma pessoa experiente, que conhece muito bem a linguagem BPM e os processos de negócio da especificação UBL. Este pressuposto é visto como bastante realístico, pois os usuários-projetistas de BPM são naturalmente experientes na medida em que são eles os responsáveis por expressarem os processos de negócio da empresa numa linguagem formal, dentro de uma certa especificação. Já o uso da especificação UBL, isto ainda não é muito comum haja vista que se trata de um padrão relativamente recente, porém, é aberto e vem sendo apoiado e mantido por inúmeras organizações mundiais para fazer frente ao padrão proprietário e principalmente voltado ao setor de eletrônica, como é o padrão RosettaNet;

(2) A especificação de processos de negócio possui processos e documentos perfeitamente definidos e passíveis de serem interpretados por um computador. A categorização dos processos e de suas atividades é precisa de forma que se consegue estabelecer uma relação um para um entre uma atividade do processo e uma interface de serviço. O processo de modelagem tem uma parcela de certa subjetividade quando se trata de modelar um processo de negócio abstrato para um ambiente de modelagem tipo BPM. Por outro lado, advoga-se para qualquer empresa organizada que usa soluções BPM naturalmente precisa organizar seus processos, tanto para facilitar o acesso e edição, como para promover o reuso. Existem vários critérios para tal organização, sendo um deles – e o assumido neste trabalho – o de modelar um processo (e suas atividades) como um conjunto de serviços de *software* e vincular um serviço a cada atividade. Desta forma, o reuso e interoperabilidade são facilitados assim como a descoberta de serviços nos vários provedores.

(3) As interfaces dos serviços são descritas por uma linguagem formal, passível de processamento por computador. A descrição deve ser completa o suficiente para que com apenas ela todos os parâmetros de um serviço sejam conhecidos, incluindo seus tipos de dados e estrutura. Este pressuposto é visto como bastante natural, na medida em que o principal padrão de implementação de serviços de software (*web services*) requer que a *interface* de cada serviço seja definida e conhecida (no caso, através do padrão WSDL).

(4) Existem provedores de serviços que disponibilizam serviços que implementam as atividades dos processos UBL. Estes provedores publicam seus serviços em repositórios, acessíveis através de uma

interface padrão. Apesar do modelo do catálogo não ser dependente do padrão UBL, para fins desta dissertação e teste da hipótese sobre o uso de um catálogo como abordagem de agilização da integração BPM&SOA, assume-se que os diversos provedores de serviços implementam seus serviços de acordo com uma certa especificação, no caso, a da UBL.

(5) Existe um mecanismo de busca capaz de procurar serviços dentro desses repositórios. A área de descoberta de serviços é uma das mais pesquisadas atualmente em SOA e há inúmeros mecanismos/motores de busca. No caso deste trabalho, assume-se o uso do mecanismo de descoberta desenvolvido em (SOUZA, 2009).

Com base nessa visão e nesses pressupostos, concebeu-se a arquitetura conceitual do sistema, apresentada na Figura 25.

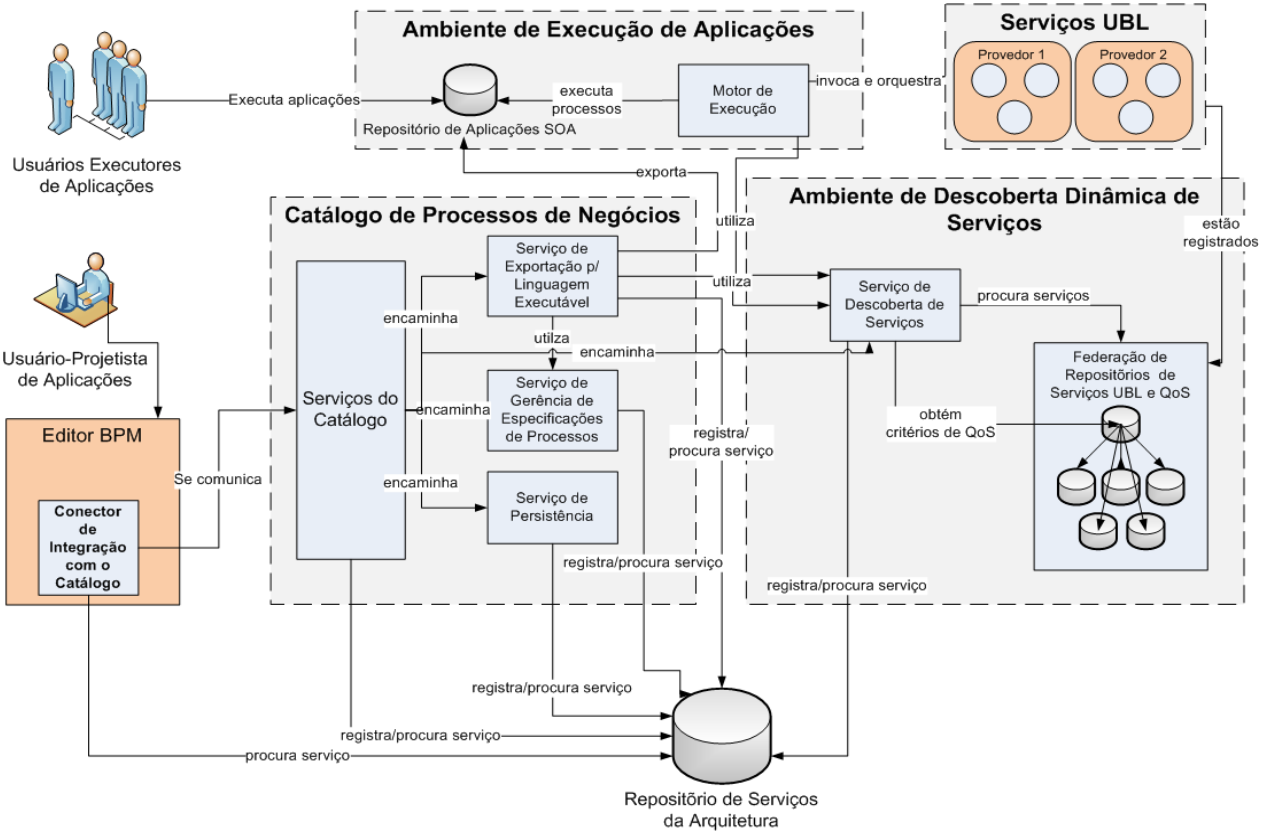


Figura 25 – Arquitetura Conceitual do Sistema

O Projetista de Aplicações interage com o editor BPM, que possui o Conector de Integração com o Catálogo dentro de sua estrutura. Este interage com os Serviços do Catálogo, que é uma composição de serviços responsável por todas as funcionalidades que o catálogo oferece.

O Ambiente de Execução de Aplicações permite que as aplicações concebidas pelo projetista sejam executadas e monitoradas.

O Catálogo de Processos de Negócio, assim como o Ambiente de Execução de Aplicações, utilizam os serviços providos pelo Serviço de Descoberta, presente no Ambiente de Descoberta Dinâmica de Serviços.

A Federação de Repositórios de Serviços UBL e QoS é o local onde os serviços UBL são procurados e também a onde está armazenada a ontologia de QoS, utilizada pela classificar os serviços em termos de qualidade de serviço.

Os Serviços UBL são implementados por diversos provedores de serviço. Cada serviço segue rigorosamente a especificação, que define a sua *interface*. Dessa forma, todos eles apresentam exatamente o mesmo comportamento em termos de tipos de dados trocados e de operações.

Todos os serviços que fazem parte da arquitetura possuem sua referência armazenada no Repositório de Serviços da Arquitetura. Dessa forma, a localização de cada um deles pode ser alterada, sem prejuízo aos demais serviços, visto que somente esse repositório guarda as referências de localização dos serviços. Isso também permite obter uma arquitetura que vá ao encontro dos preceitos da Arquitetura Orientada a Serviços, explicada com mais detalhes na seção 2.2.

As seções a seguir explicam em maiores detalhes cada um dos componentes da arquitetura, modelados no padrão UML (OMG, 2005).

3.4.1 Conector de Integração com Editor BPM

O Conector de Integração com o Editor BPM é responsável por fazer a ligação entre o editor BPM e o Catálogo de Processos de Negócio. Dessa forma, ele estende as funcionalidades do editor de forma a permitir sua integração com o catálogo. A Figura 26 mostra quais são os módulos que compõem esse componente:

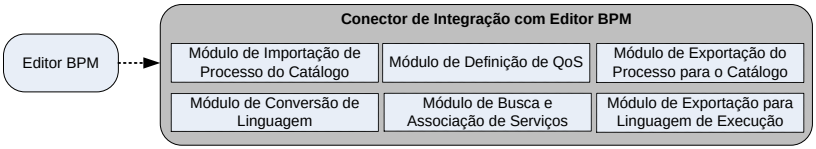


Figura 26 – Módulos do Conector de Integração com o Editor BPM

- *Módulo de Importação de Processos do Catálogo* – esse módulo é responsável por importar processos contidos no catálogo, escolhido pelo usuário, para dentro do editor BPM;
- *Módulo de Conversão de Linguagem* – faz a tradução entre a linguagem de modelagem de processos utilizada no editor BPM e a usada no catálogo. Esse módulo é usado nas operações de importação e exportação de processos;
- *Módulo de Definição de QoS* – permite que o usuário defina restrições de qualidade de serviço QoS associadas para cada serviço implementador da atividade dentro do processo;
- *Módulo de Busca e Associação de Serviços* – faz a busca dos serviços candidatos, dentro das restrições de QoS, e permite que eles sejam vinculados (*binding*) com as tarefas do processo;
- *Exportação do Processo para o Catálogo* – exporta para o catálogo o processo presente no editor BPM, juntamente com as informações dos serviços vinculados e restrições de QoS;
- *Módulo de Exportação para Linguagem de Execução* – exporta o processo, já vinculado com os serviços, para um formato executável. Dessa forma o processo pode ser executado dentro de um ambiente de execução de processos.

Os módulos do conector são implementados de acordo com os componentes descritos na Figura 27. O conversor de linguagem do processo é representado pelos componentes à esquerda da figura. As interfaces de interação com o usuário são representadas pelos componentes com prefixo “ui::” (*user interface*). Estes componentes se comunicam com o catálogo através de uma interface de serviço.

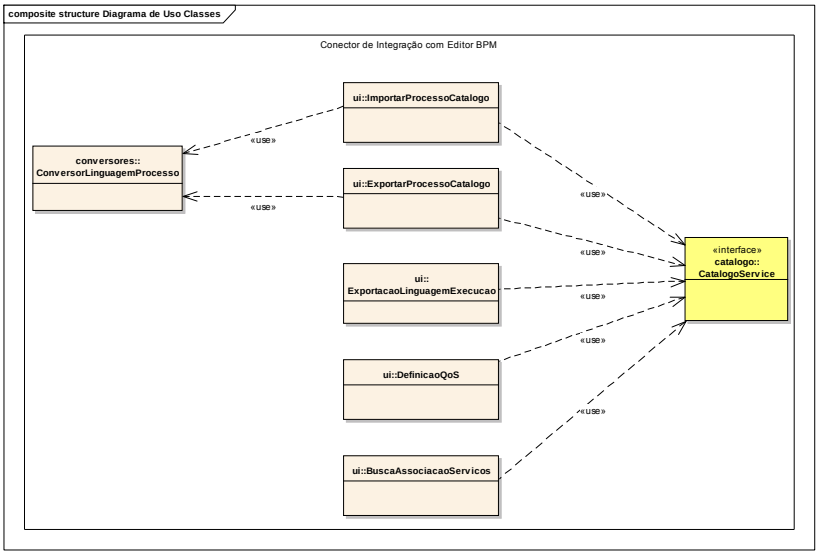


Figura 27 – Componentes do Conector de Integração com o Editor BPM

As interfaces de importação e exportação de processos para o catálogo fazem uso do "ConversorLinguagemProcesso", que permite que seja feita a conversão entre a linguagem utilizada pela ferramenta e a usada pelo catálogo. A exportação para a linguagem de execução utiliza os serviços do catálogo. A seção a seguir irá detalhar o funcionamento de cada um dos módulos.

3.4.1.1 Importação de Processos do Catálogo

O funcionamento deste módulo é descrito pelo diagrama de sequência exposto na Figura 28.

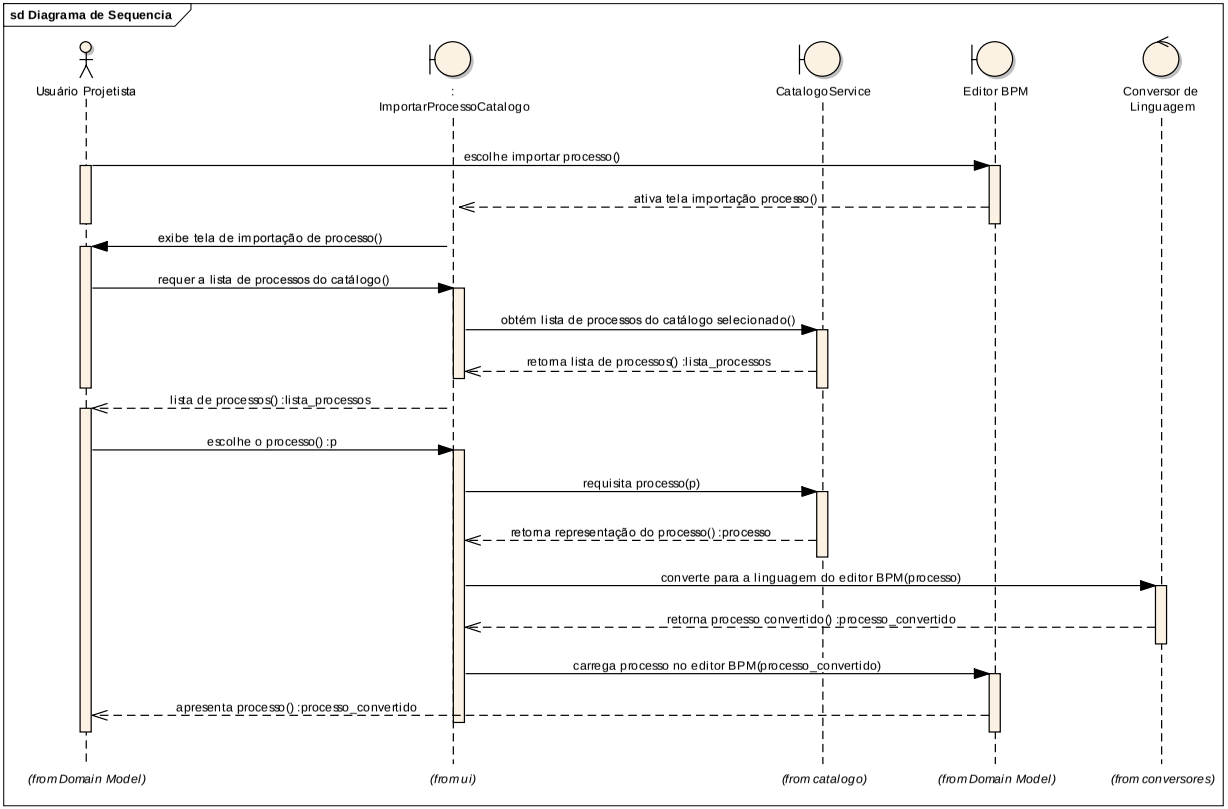


Figura 28 – Diagrama de Sequência – Importação de Processo

O usuário do sistema requisita ao editor BPM a importação de um processo do catálogo. Este ativa o componente "ImportarProcessoCatalogo", responsável pela interface gráfica (UI – *user interface*) de importação. Em seguida o usuário requer a lista de processos, fazendo com que a UI 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 a UI. Esta invoca o Conversor de Linguagem, que irá converter o processo recém obtido para a linguagem utilizada pelo editor BPM. Em seguida, o processo já convertido será carregado pelo editor e finalmente exibido ao usuário.

3.4.1.2 Definição de QoS

O funcionamento deste módulo é descrito pelo diagrama de sequência exposto na Figura 29.

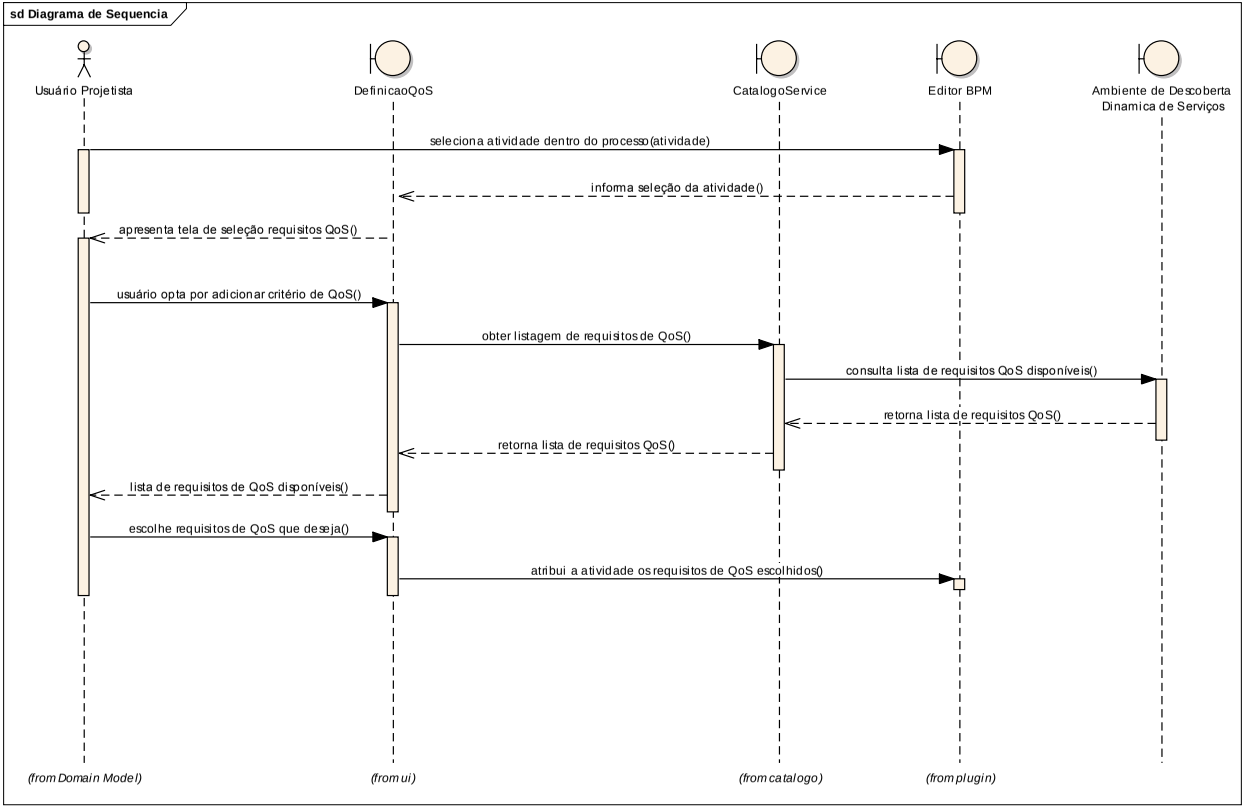


Figura 29 – Diagrama de Seqüência – Definição de QoS

Após o processo ter sido importado, o usuário irá selecionar alguma de suas atividades. O editor BPM irá informar a UI de Definição de QoS que uma atividade foi selecionada, fazendo com que a tela de seleção de requisitos fique disponível ao usuário. O usuário escolhe adicionar um critério de QoS à atividade. A UI manda uma requisição ao catálogo, que por sua vez entra em contato com o Ambiente de Descoberta Dinâmica de Serviços, que irá retornar todos os critérios de QoS disponíveis para serem utilizados, incluindo os valores que o mesmo pode ter. Estes critérios são encaminhados até o usuário, que irá escolher quais deles deseja utilizar. Quando o usuário termina de atribuir os requisitos que deseja, o editor BPM grava junto à atividade as informações de QoS estabelecidas. Estas informações serão utilizadas posteriormente, no momento da descoberta dos serviços implementadores das atividades do processo.

3.4.1.3 Busca e Associação de Serviços

O funcionamento deste módulo é descrito pelo diagrama de sequência exposto na Figura 30.

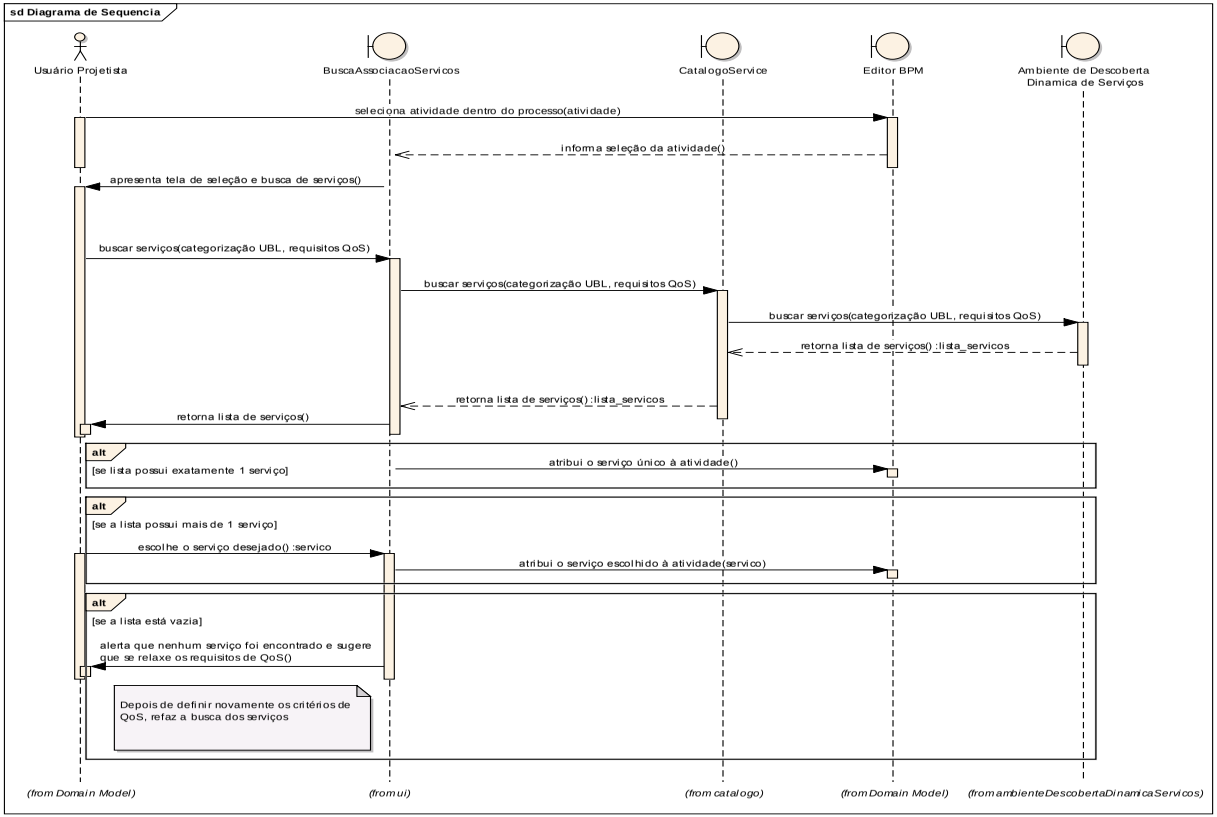


Figura 30 – Diagrama de Sequência – Módulo de Busca e Associação de Serviços

O usuário irá selecionar a atividade do processo que deseja associar um serviço. O editor BPM irá informar à UI de Busca e Associação de Serviços que uma atividade foi selecionada, fazendo com que a tela de busca de serviços fique disponível ao usuário. O usuário escolhe buscar serviços que implementem as atividades do processo. Esta busca é feita com base na categorização da atividade, segundo a especificação UBL e dos requisitos de QoS atrelados à atividade. A busca é encaminhada até o Ambiente de Descoberta Dinâmica de Serviços, que irá efetuá-la e retornar uma lista com os possíveis serviços candidatos ao usuário. Nesse ponto, podem ocorrer três situações possíveis:

- Se a lista possui somente um serviço, este é automaticamente atrelado à atividade;
- Se a lista possui mais de um serviço, o usuário tem a opção de escolher qual serviço deseja vincular à atividade;
- Se a lista não possui nenhum serviço, a UI de Busca e Associação de Serviços sugere que o usuário relaxe alguns dos critérios de QoS, a fim de refazer a busca de serviços e achar um possível candidato.
- Com relação a essa última situação, faculta-se ao usuário o relaxamento dos critérios de QoS. Caso não deseje fazê-lo, ainda é possível tentar encontrar um serviço candidato no momento da execução do processo, apesar de ser uma escolha arriscada.

3.4.1.4 Exportação do Processo para o Catálogo

O funcionamento deste módulo é descrito pelo diagrama de sequência exposto na Figura 31.

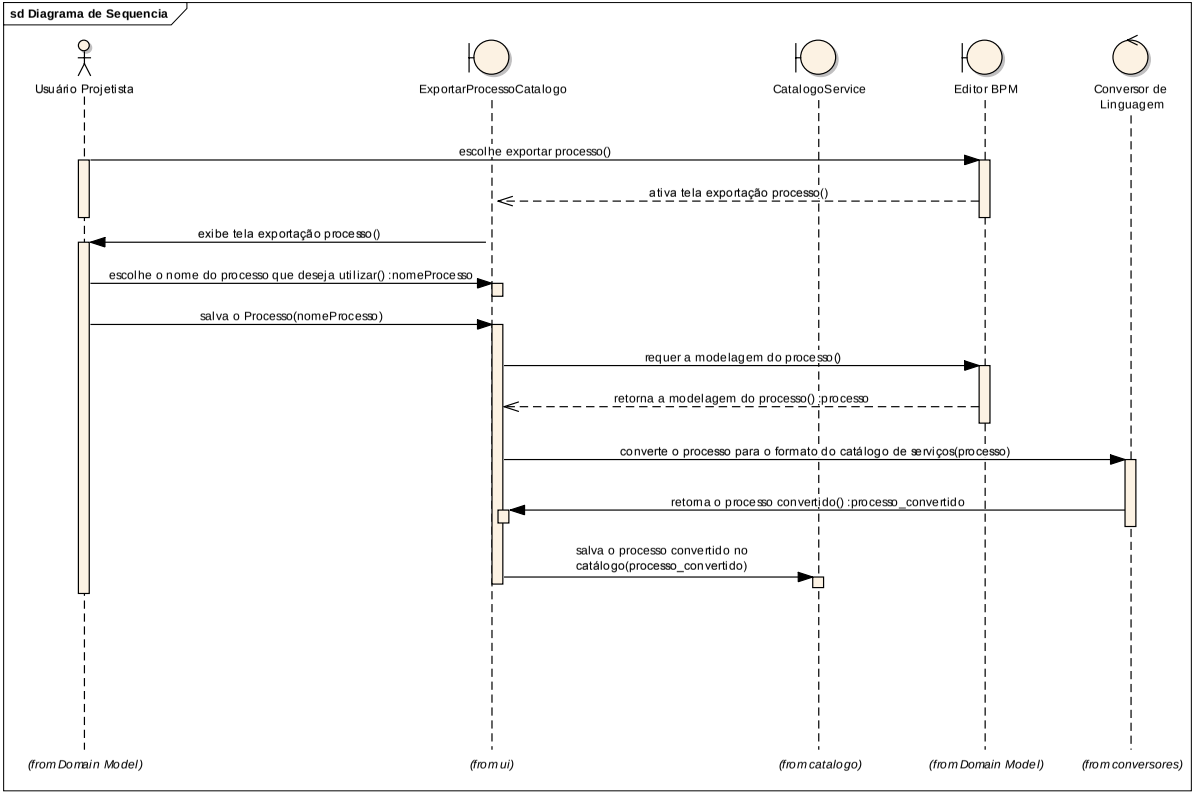


Figura 31 – Diagrama de Sequência – Módulo de Exportação do Processo para o Catálogo

O usuário, depois de ter o seu processo já definido, com as restrições de QoS e serviços vinculados, escolhe exportar o processo para o catálogo. O editor BPM ativa a tela de Exportação de Processo que é exibida ao usuário. Este escolhe qual o nome deseja atribuir ao processo e procede a exportação. Nesse momento, é requisitado ao editor BPM a modelagem do processo, no formato específico da ferramenta. A modelagem é em seguida enviada ao Conversor de Linguagem, que irá convertê-la para o formato usado pelo Catálogo de Processos. Em seguida, a função responsável por salvar o processo no catálogo é invocada, passando como parâmetro o processo convertido.

3.4.1.5 Exportação para Linguagem de Execução

O funcionamento deste módulo é descrito pelo diagrama de sequência exposto na Figura 32.

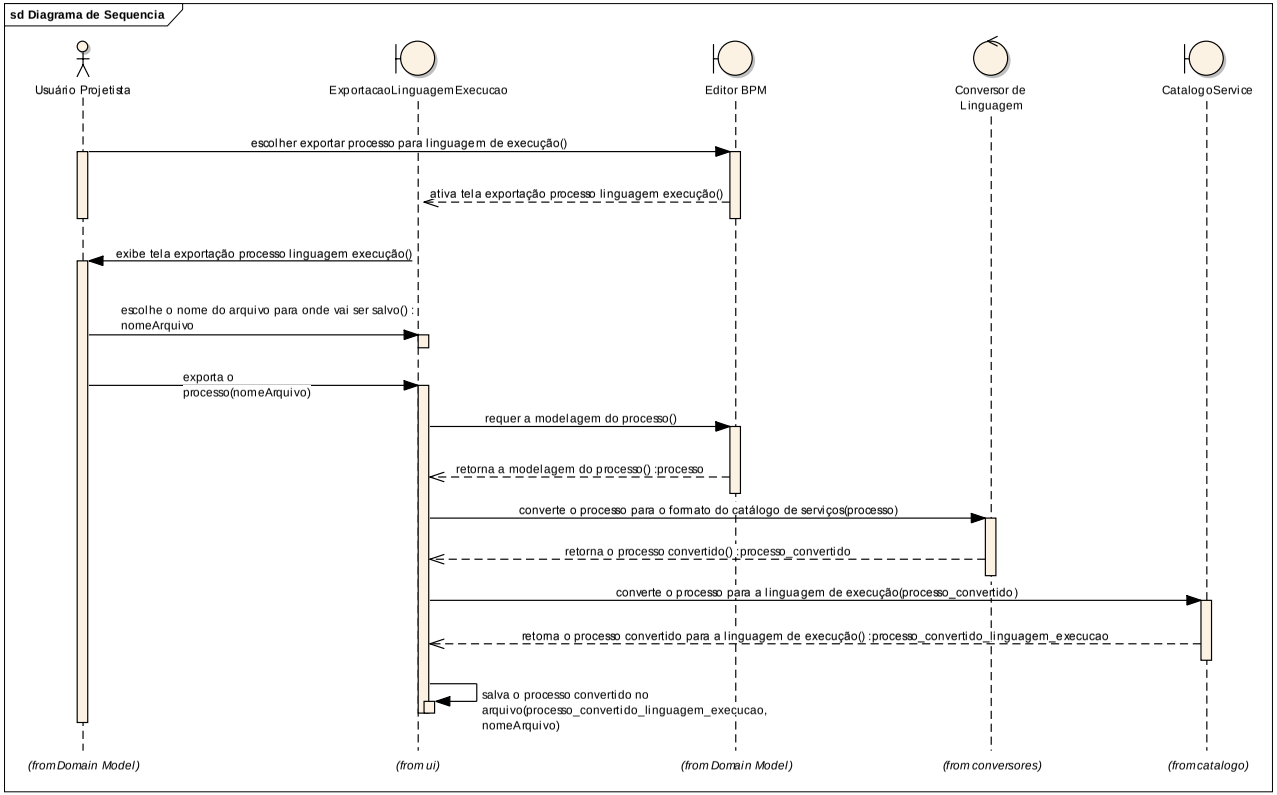


Figura 32 – Diagrama de Sequência – Módulo de Exportação para Linguagem de Execução

O usuário, depois de ter o seu processo já definido com as restrições de QoS e serviços vinculados, escolhe exportar o processo para a linguagem de execução a fim de poder executá-lo posteriormente. O editor BPM ativa a tela 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, é requisitado ao editor BPM a modelagem do processo, no formato específico da ferramenta. A modelagem é em seguida enviada ao Conversor de Linguagem, que irá fornecer a representação do processo no formato do catálogo. Em seguida, o catálogo é novamente requisitado para efetuar a conversão para o formato de execução. O retorno é então salvo no arquivo escolhido pelo usuário.

3.4.2 Catálogo de Processos de Negócio

O Catálogo de Processos de Negócio é, em termos conceituais, o principal elemento da arquitetura e de contribuição científica da dissertação. Ele é responsável pela estrutura de armazenamento e gerenciamento dos processos e das especificações de processos de negócio (por exemplo, a UBL). Cabe a ele também a exportação dos processos para uma linguagem executável e a integração com o mecanismo de busca de serviços provido pelo Ambiente de Descoberta de Serviços. O catálogo é constituído de cinco grandes módulos, conforme mostrado na Figura 33:

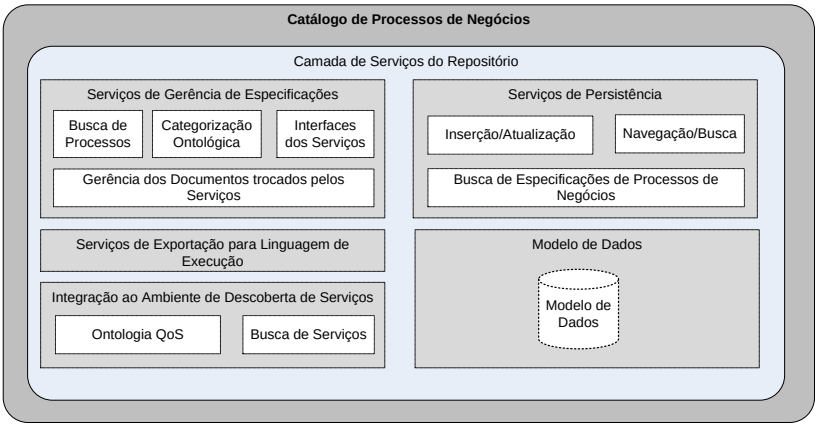


Figura 33 – Módulos do Catálogo de Processos de Negócio UBL

O *Serviço de Gerência de Especificações* permite que o catálogo armazene as especificações de processos de negócio. A especificação 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. Dessa forma, exerce o papel de ligação entre a modelagem do processo e sua implementação através de serviços.

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

O *Serviço de Exportação para Linguagem de Execução* faz a conversão do processo para uma linguagem executável. Essa conversão permite que o processo seja executado dentro de um servidor de aplicações.

A *Integração ao Ambiente de Descoberta de Serviços* permite que o catálogo tenha acesso à descoberta de serviços e a ontologia de qualidade de serviço (QoS), utilizada pelo catálogo.

O *modelo de dados* possui as entidades responsáveis por estruturar os processos, especificações e demais elementos contidos no

catálogo. O modelo é projetado para poder ser persistido e manipulado pelos módulos da arquitetura do catálogo.

Nas seções a seguir, mostra-se em maiores detalhes cada um desses módulos.

3.4.2.1 Gerência de Especificações

O serviço de Gerência de Especificações é responsável por armazenar as especificações de processos de negócio contidas no catálogo. A Figura 34 apresenta as operações que esse serviço fornece:

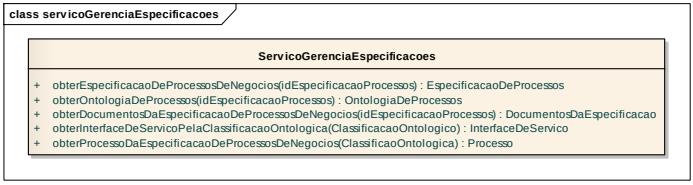


Figura 34 – Serviço de Gerência de Especificações

- *obterEspecificacaoDeProcessosDeNegocios* – retorna a especificação de processos através do seu identificador;
- *obterOntologiaDeProcessos* – retorna a ontologia de processos da especificação, isto é, a classificação taxonômica e a forma como os processos estão organizados dentro da especificação;
- *obterDocumentosDaEspecificacaoDeProcessosDeNegocios* – obtém a lista de documentos que estão associados a uma determinada especificação de processos de negócio. Tais documentos são trocados entre os serviços pertencentes à especificação;
- *obterInterfaceDeServicoPelaClassificacaoOntologica* – fornece a interface de serviço através da classificação ontológica. A interface é utilizada pelos serviços que implementam as atividades do processo;
- *obterProcessoDaEspecificacaoDeProcesosDeNegocios* – retorna um processo que faz parte da especificação. Utiliza a classificação ontológica para fazer a busca.

Mostra-se na Figura 35 as classes conceituais utilizadas para representar o modelo de dados utilizado pelo Serviço de Gerência de Especificações.

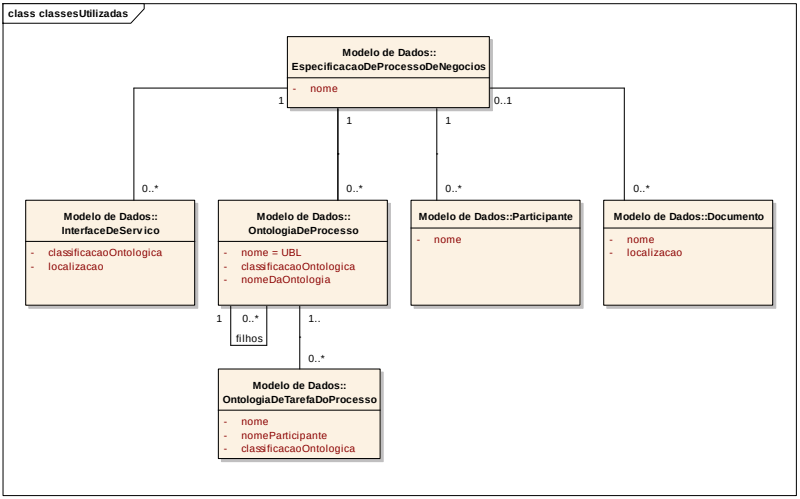


Figura 35 – Classes conceituais utilizadas no Serviço de Gerência de Especificações

A classe *EspecificacaoDeProcessoDeNegocios* serve como encapsuladora das informações da especificação. As classes *InterfaceDeServico* e *Documento* permitem associar, respectivamente, as *interfaces* dos serviços implementadores das tarefas do processo e os documentos que são trocados por estes serviços. O atributo *localizacao* indica qual é a localização física dos arquivos que descrevem estes artefatos (por exemplo, arquivos *WSDL* e *XMLSchema*).

A especificação é composta por participantes, que executam determinadas atividades dentro dos processos. Isso é modelado através da classe *Participante.*

A classe *OntologiaDeProcesso* é responsável pela categorização hierárquica dos processos. Trata-se de uma estrutura baseada em árvore, onde as folhas são classes do tipo OntologiaDeTarefaDoProcesso, que descrevem as tarefas pertencentes ao processo.

O atributo *classificacaoOntologica*, presente nessa classe permite obter a identificação única da atividade dentro da especificação. Esse mesmo atributo é usado na classe *InterfaceDeServico* para fazer a vinculação do serviço com a atividade. Dessa forma, o serviço de gerência de especificações, quando requisitado a fornecer a *interface* de

serviço através da classificação ontológica da tarefa, tem condição de efetuar essa operação.

A regra de formação da classificação ontológica é mostrada a seguir:

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

Figura 36 – Regra de formação da classificação ontológica

Conforme a Figura 36, por exemplo, a classificação ontológica "ubl/payment/paymentprocess/accountingCustomer/authorizePayment" identifica univocamente a tarefa "*Autorize Payment*", executada pelo participante "*Accouting Customer*", no processo "*Payment Process*", pertencente à categoria "*Payment*" da especificação UBL. Na Tabela 7 é apresentado um extrato completo da ontologia que descreve todas as atividades do processo de pagamento (*Payment Process*). A descrição completa da ontologia UBL está presente no Apêndice A.

Tabela 7 – Ontologia do Processo de Pagamento (Payment Process)

Categoria	Nome do Processo	Participante	Atividade
Payment	Payment Process	Accounting Customer	Authorize Payment
Ontologia: **ubl/payment/paymentprocess/accountingCustomer/authorizePayment**			
Payment	Payment Process	Accounting Customer	Notify of Payment
Ontologia: **ubl/payment/paymentprocess/accountingCustomer/notifyOfPayment**			
Payment	Payment Process	Accounting Supplier	Notify Payee
Ontologia: **ubl/payment/paymentprocess/accountingSupplier/notifyPayee**			
Payment	Payment Process	Accounting Supplier	Receive Advice
Ontologia: **ubl/payment/paymentprocess/accountingSupplier/receiveAdvice**			
Payment	Payment Process	Payee Party	Receive Advice
Ontologia: **ubl/payment/paymentprocess/payeeParty/receiveAdvice**			

O serviço de Gerência de Especificações está projetado para permitir que múltiplas especificações possam coexistir numa mesma instância do Catálogo de Processos de Negócio. Apesar de que o escopo desse trabalho abranger somente a especificação UBL, nada impediria que o catálogo tivesse alguma outra.

3.4.2.2 Persistência

O Serviço de Persistência é responsável por armazenar os dados dos processos que são desenvolvidos pelos usuários do catálogo. A Figura 37 apresenta as operações fornecidas por este serviço:

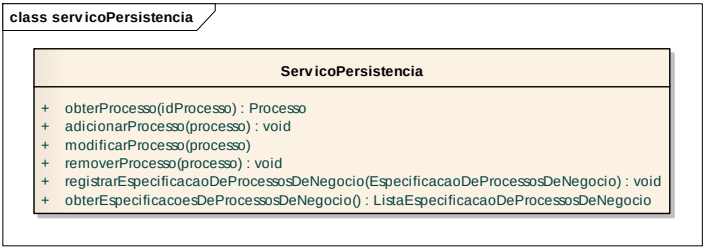


Figura 37 – Serviço de Persistência

- *obterProcesso* – permite obter o processo através do fornecimento do seu identificador (*idProcesso*);
- *adicionarProcesso* – adiciona um novo processo;
- *modificarProcesso* – modifica um processo já existente;
- *removerProcesso* – remove um processo;
- *registrarEspecificacaoDeProcessosDeNegocio* – registra uma especificação de processos de negócio dentro do catálogo. Isso insere um registro no repositório de dados, que faz um apontamento ao local físico onde os arquivos as especificação estão localizados;
- *obterEspecificacoesDeProcessosDeNegocio* – obtém a lista de especificações de processos de negócio que estão registradas no catálogo (vide item 3.4.2.1 - Gerência de Especificações).

O Serviço de Persistência faz a ligação entre o modelo de dados e o mecanismo que é usado para armazená-los. Isso pode ser visto na Figura 38.

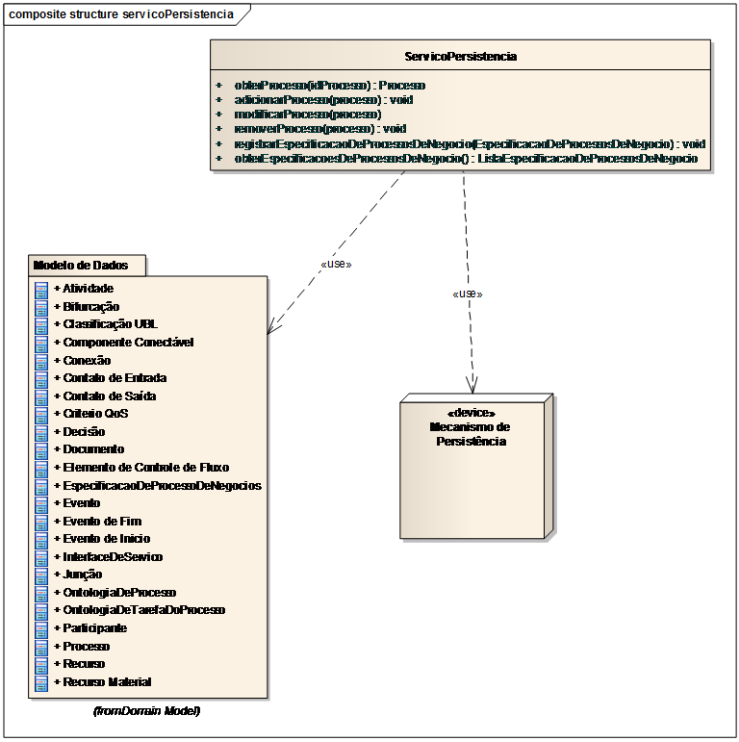


Figura 38 – Camada de Persistência

O modelo de dados utilizado pelo Serviço de Persistência será explicado com maiores detalhes na seção 3.4.3 - Modelo Conceitual de Dados.

3.4.2.3 Exportação para Linguagem de Execução

O serviço de Exportação para Linguagem de Execução permite que o processo modelado seja exportado para um formato executável, dessa forma podendo ser executado num servidor de aplicação. A Figura 39 apresenta a classe conceitual desse serviço.

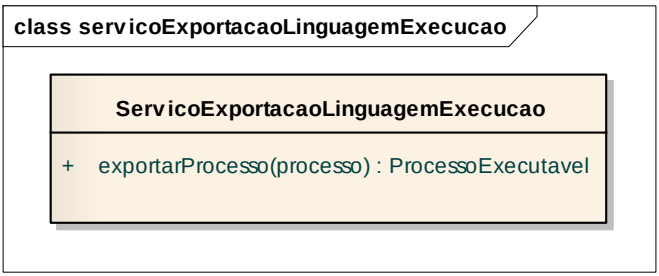


Figura 39 – Serviço de Exportação para Linguagem de Execução

A operação *exportarProcesso* tem como entrada um processo, no formato reconhecido pelo Catálogo de Processos de Negócio. A saída é um processo executável. A Figura 40 apresenta as etapas em que ocorre a exportação:

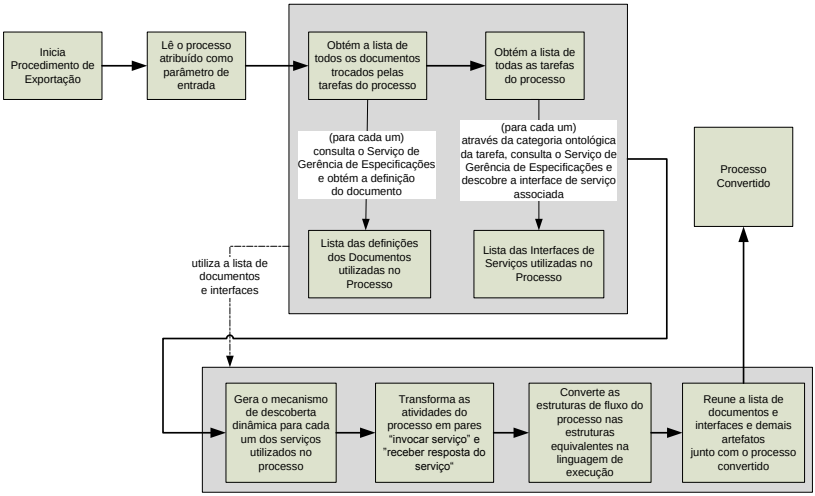


Figura 40 – Etapas da Exportação para Linguagem de Execução

Primeiramente, o processo de entrada é lido, obtendo-se a lista dos seus documentos trocados e de suas tarefas. O Serviço de Gerência de Especificações é requisitado a fim de fornecer as definições desses documentos e também dos serviços vinculados a cada tarefa (através do

uso da classificação ontológica). Em seguida, armazena-se em listas o resultado dessas consultas, para uso posterior.

No retângulo maior representado na parte inferior da Figura 40, a exportação continua. Primeiramente gera-se o mecanismo de descoberta de serviços para cada um dos serviços utilizados pelo processo. A Figura 41 mostra a estrutura de fluxo gerada no processo para dar suporte à descoberta.

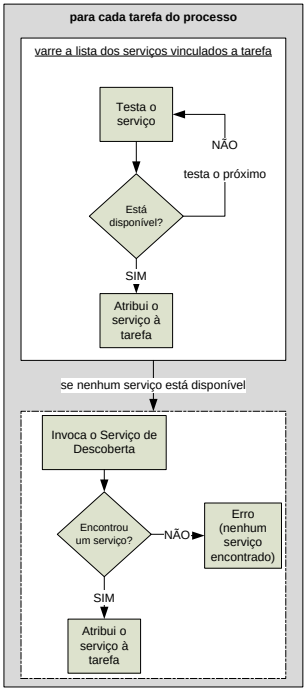


Figura 41 – Mecanismo de verificação e descoberta de serviços em tempo de execução

Para cada atividade do processo, os serviços candidatos a executá-la são verificados. A lista de serviços é varrida, até que se ache um serviço disponível. Caso positivo, a verificação dos serviços para aquela atividade é concluída e o serviço é atribuído a ela. Caso contrário, a lista de serviços continua a ser varrida, até que se ache um serviço disponível ou a lista acabe. Se até aí não se encontrou um

serviço disponível, o mecanismo de descoberta é invocado, recebendo como argumentos a classificação ontológica do serviço e os parâmetros de QoS. Depois de realizada a busca, o mecanismo irá retornar o endereço do serviço. Caso não ache nenhum, um erro é lançado e o processo é abortado, por não se ter conseguido fazer a vinculação de todos os serviços às atividades do processo.

Continuando o fluxo da Figura 40, logo após o mecanismo de descoberta ter sido construído, o serviço de exportação irá transformar as atividades do processo em pares "invocar serviço" e "receber resposta do serviço". Isso permite que o processo, logo após invocar o serviço, fique esperando sua resposta. Essa comunicação assíncrona, no sentido do mecanismo de comunicação com o serviço, e não na perspectiva do processo, permite que este processo aguarde indefinidamente por sua resposta, sem riscos de haver *timeout*, o que ocorreria caso o mecanismo de comunicação fosse síncrono.

O próximo passo consiste em fazer as conversões das estruturas de fluxo do processo para as equivalentes na linguagem de execução de processos escolhidas. Isso permite que ele seja executado obedecendo às mesmas regras de controle do processo original.

O último passo é agregar o processo convertido com os demais arquivos necessários ao *deployment* no servidor de aplicações. Esses arquivos são:

- A descrição do fluxo do processo e sua interação com os serviços;
- Arquivos com as definições das *interfaces* dos serviços invocados, juntamente com os endereços de localização dos mesmos;
- Arquivos que descrevem os documentos utilizados pelo processo (por exemplo: os documentos UBL);

Com o processo exportado, o mesmo pode ser executado no Ambiente de Execução de Processos, através de um servidor de aplicações. Maiores detalhes podem ser vistos no item 3.4.5 - Ambiente de Execução .

Vale ressaltar que o processo de exportação do processo em BPM para a linguagem de execução de processos é totalmente automatizado e transparente, de forma que o usuário-projetista não precisa ter conhecimento técnico para efetuar a exportação do processo modelado em BPM para o ambiente onde ele será executado. O ambiente integrado ao catálogo é responsável por fazer a conversão e a vinculação dos serviços que irão executar em conjunto com o processo

de negócio. Dessa forma, esse componente da arquitetura serve com um elemento que agiliza a concepção de aplicações na integração entre BPM e SOA.

3.4.3 Modelo Conceitual de Dados

Esta seção apresenta o modelo conceitual de dados utilizados para estruturar os processos e demais elementos dentro do catálogo de processos de negócio.

Modelos de processos representam informação sobre o que é feito, onde e quando é feito, quem faz, como, por que e do que depende para ser feito (SHAHZAD, ELIAS, *et al.*, 2009). Modelos de processos podem, portanto, serem vistos sobre diferentes perspectivas. O trabalho de (CURTIS *et al.*, 1992) organiza estas perspectivas em quatro: funcional, comportamental, organizacional e informacional. O uso destas permite separar e analisar as informações do processo sobre diferentes pontos de vista, permitindo assim organizar melhor as entidades que farão parte do modelo. Usar-se-á, portanto, o trabalho de (CURTIS *et al.*, 1992) no desenvolvimento do modelo proposto.

Uma quinta perspectiva, ontológica, é utilizada para tratar os aspectos referentes à captura da semântica que envolve os processos.

A Tabela 8 apresenta as perspectivas e quais componentes do modelo de dados se enquadram nelas. Em seguida, explica-se em maiores detalhes cada uma delas. Por último apresenta-se o modelo de dados.

Tabela 8 – Perspectivas e Componentes

Perspectiva	Componente
Funcional	Processo
	Atividade
	Evento
	Evento de Início
	Evento de Fim
Comportamental	Componente Conectável
	Conexão
	Contato de Entrada
	Contato de Saída
	Elemento de Controle de Fluxo
	Bifurcação
	Decisão

	Junção
Organizacional	Participante
Informacional	Documento
Ontológica	Especificação de Processo de negócio
	Ontologia de Processo
	Ontologia de Tarefa do Processo
	Interface Serviço
	Critério QoS

Perspectiva Funcional – representa quais os elementos do processo serão executáveis. Os elementos executáveis são as atividades e os sub-processos. Um processo inicia através de um evento de início e termina com um evento de fim.

Perspectiva Comportamental – representa "quando" e "como" as atividades do processo serão executadas. O controle de fluxo faz parte dessa perspectiva, definindo a ordem de execução das atividades, bem como relações de dependência, condições de desvio, escolha etc.

O controle de fluxo e a conexão entre os elementos foram modelados através de pontos de contato. Cada elemento que pode ser ligado a outro herda as características do Componente Conectável, que por sua vez possui um lista de contatos de entrada e de saída. Os seguintes componentes são conectáveis:

- Eventos (evento de início, evento de fim);
- Atividades (incluindo sub-processos);
- Elementos de Controle de Fluxo (bifurcação, junção, decisão);

Cada componente tem sua particularidade em relação aos pontos de contato. Um elemento de bifurcação, por exemplo, tem geralmente um ponto de contato de entrada e vários de saída. Um elemento de junção, ao contrário, tem vários pontos de contato de entrada e um de saída. O componente Conexão representa concretamente a conexão entre um ponto de contato de saída e um ponto de contato de entrada entre componentes que são conectáveis.

Perspectiva Organizacional – descreve as entidades organizacionais que podem executar a tarefa. No modelo proposto, uma atividade pode ser executada por um ou mais participantes. O participante possui uma determinada função, que lhe torna apto a executar a atividade, por exemplo: chefe de vendas.

Perspectiva Informacional – representa os documentos que são produzidos e/ou manipulados pelo processo. Por exemplo, os

documentos da especificação UBL que são trocados pelas atividades do processo são modelados através do componente Documento.

Perspectiva Ontológica - Os componentes dessa perspectiva são utilizados para atribuir informação semântica aos processos. Isso permite que eles sejam categorizados, por exemplo, pela especificação ao qual pertencem. Maiores detalhes podem ser vistos na seção 3.4.2.1 - Gerência de Especificações.

As restrições de QoS são modeladas através do componente Critério de QoS. Este possui quatro valores, que são o nome do item de QoS, o nome do atributo de QoS, o seu valor e a comparação desejada do valor. Por exemplo, o item de QoS pode ser "*Performance*", o atributo "*Latency*", o valor "10 segundos" e a comparação "menor que". Isto é, atribui-se um critério de QoS, que trata de performance, mais especificamente de latência, onde queremos que esta seja menor que 10 segundos. A listagem completa dos itens de QoS e os seus atributos podem ser vistos na Tabela 9 da seção 3.4.6.

Esses valores são armazenados junto ao modelo de dados do processo no momento em que o usuário atribui restrições de QoS às atividades e opta por salvar o processo no catálogo.

O modelo conceitual de dados é apresentando na Figura 42.

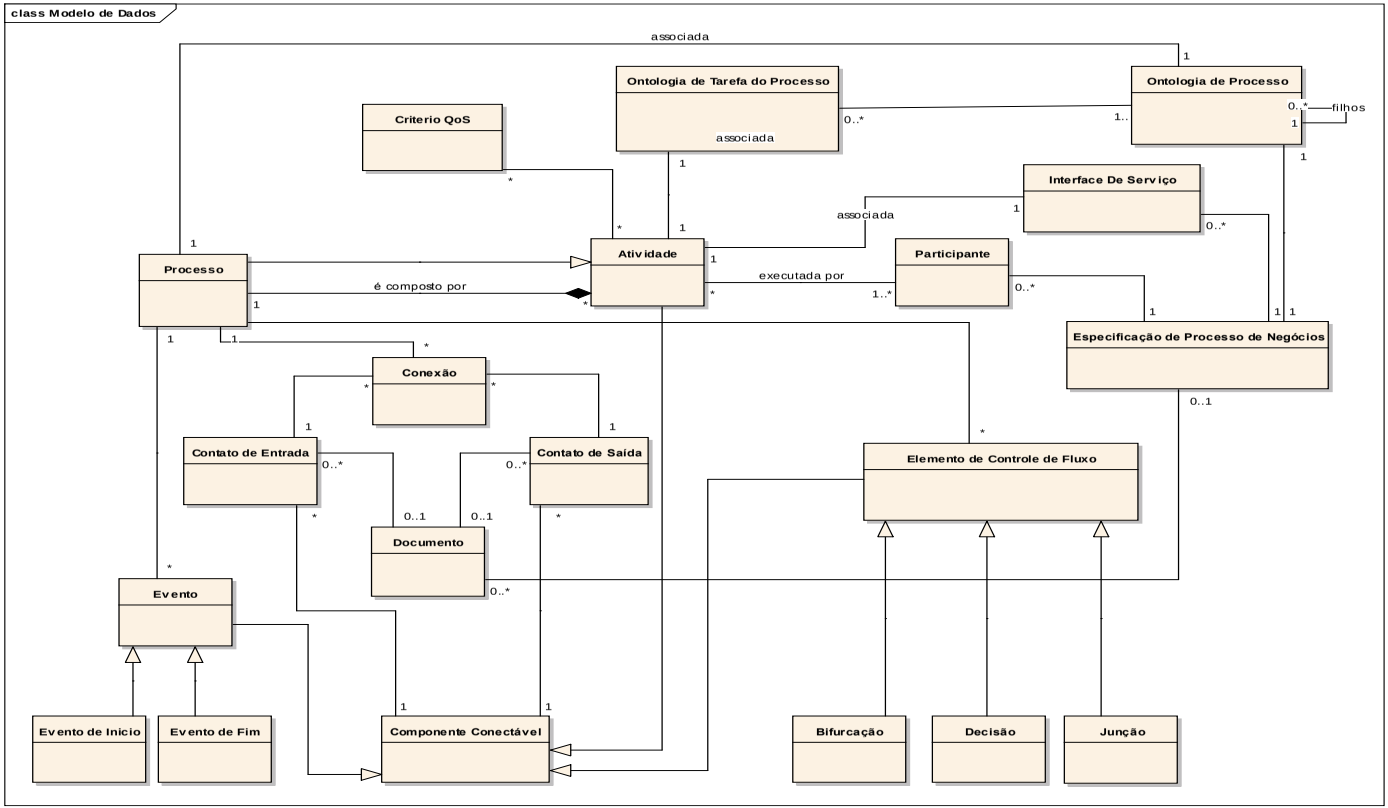


Figura 42 – Modelo Conceitual de Dados

3.4.4 Serviços UBL

Esse componente da arquitetura contém as implementações de referência de alguns dos processos da especificação UBL. Conforme a Figura 43, o comportamento de cada tarefa pertencente a um processo é modelado como um serviço, que implementa uma interface descrita pela especificação do qual o processo faz parte. Isso permite que o serviço utilize fielmente os tipos de dados requeridos pela tarefa em termos de entradas e saídas.

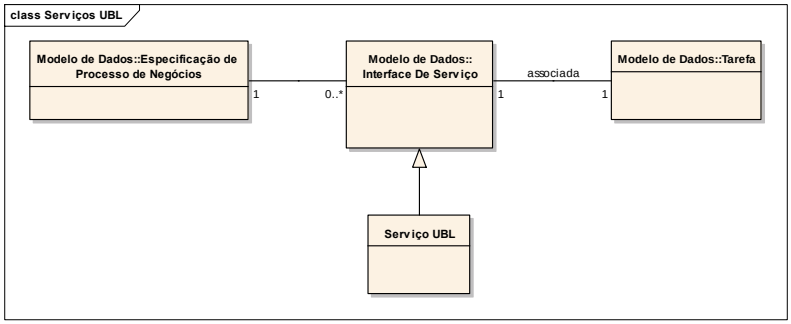


Figura 43 – Diagrama de Classes – Serviço UBL

O mecanismo de comunicação dos serviços com o processo deve ser do tipo assíncrono. Isso é necessário, pois frequentemente o tempo de resposta do serviço é longo, seja por envolver processamentos onerosos, seja por depender de intervenção humana. Dessa forma, qualquer mecanismo síncrono de comunicação, que envolva *timeouts* é pouco aconselhável, pois é impossível estipular o tempo de resposta da invocação do serviço.

O serviço UBL pode ter a participação de pessoas na sua execução. A característica assíncrona das invocações permite que as requisições sejam enfileiradas e enviadas as pessoas responsáveis por executá-las. A Figura 44 mostra a interação do processo com o serviço.

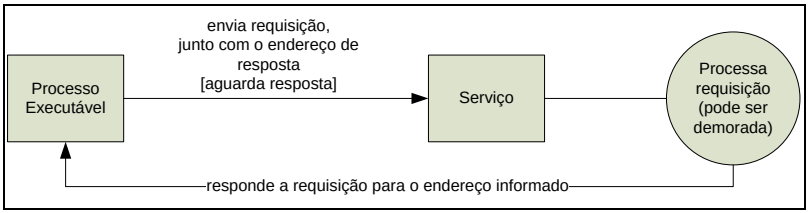


Figura 44 – Interação do Processo com o Serviço

O processo faz a requisição ao serviço, passando o seu endereço para resposta. O serviço recebe a requisição e começa a processá-la. Quando a mesma é finalizada, uma resposta é enviada ao processo, que continua sua execução.

3.4.5 Ambiente de Execução de Aplicações

O Ambiente de Execução de Aplicações é o local onde os processos exportados são armazenados e executados. Ele é composto de um Repositório de Aplicações, que armazena os processos num repositório de dados e de um Motor de Execução, que interpreta a linguagem de execução dos processos e os executa.

A fim de maximizar questões de interoperabilidade, o ambiente deve utilizar uma linguagem padronizada para a descrição dos processos executáveis.

É de responsabilidade do ambiente, também, o gerenciamento das instâncias dos processos que estão em execução. Estas devem ser persistidas de forma que possam ser recuperadas mesmo que o motor de execução seja interrompido.

3.4.6 Ambiente de Descoberta Dinâmica de Serviços

O Ambiente de Descoberta Dinâmica de Serviços é o componente responsável pela busca de serviços e gerenciamento da ontologia de QoS. Apesar de ser utilizado no modelo global proposto, o serviço de descoberta não faz parte do desenvolvimento dessa dissertação.

A busca é realizada dentro de uma federação, que é um elo responsável por coordenar e permitir a consulta a um conjunto de repositórios distribuídos.

O Serviço de Descoberta é o componente que faz a busca dos serviços UBL dentro da federação. Ele possui três operações, as duas primeiras utilizadas respectivamente nas fases de concepção e execução

do processo e a terceira, responsável por fornecer a ontologia de QoS. A Figura 45 apresenta essas operações:

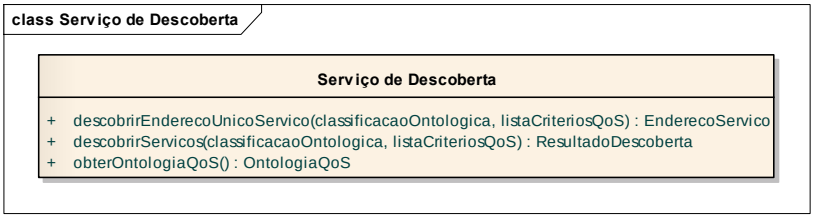


Figura 45 – Diagrama de Classes – Serviço de Descoberta

A operação "descobrirServicos" tem como parâmetros a classificação ontológica do serviço desejado e a lista de critérios de QoS desejados. O retorno é um "ResultadoDescoberta", que encapsula as seguintes informações:

- A lista de serviços em que houve correspondência perfeita das restrições de QoS impostas;
- A lista dos serviços em que houve correspondência parcial das restrições de QoS importas;

A operação "descobrirEnderecoUnicoServico" utiliza os mesmos parâmetros da primeira operação. A diferença está em seu resultado, onde apenas o endereço único do serviço escolhido é retornado. Os critérios utilizados na seleção desse único serviço não fazem parte do escopo desse trabalho.

A operação "obterOntologiaQoS" retorna a ontologia de QoS utilizada para descrever os critérios de qualidade de serviço fornecidos pelos serviços. Essa ontologia está armazenada na federação de serviços, sendo apresentada na Tabela 9.

Tabela 9 – Ontologia de QoS

Atributo	Item	Unidade
Accuracy		
	GeneratedError	ErrorNumber
Acessibility		
	Acessibility	Percentage
Availability		
	Availability	Percentage
Capacity		
	Capacity	RequestNumber
ExceptionHandling		
	Functionality	Percentage
Integrity		
	DataIntegrity	Percentage
	TransactionalIntegrity	Percentage
Interoperability		
	Interoperability	Percentage
Performance		
	Latency	Second
	Throughput	Second
	ExecutionTime	Second
	TransactionTime	Second
	ResponseTime	Second
Reliability		
	MessageWarranty	Percentage
	Month	FailureNumber
	Day	FailureNumber
	Year	FailureNumber
Robustness		
	Robustness	Percentage
Scalability		
	Latency	Second
	TransactionTime	Second
	ExecutionTime	Second
	Throughput	Second
	ResponseTime	Second

A ontologia de QoS está organizada em atributo, cada um deles podendo ter um ou mais itens associados. Por exemplo, o atributo *scalabilility*, possui os itens *latency*, *transactionTime*, *executionTime*, *throughput* e *responseTime*. Cada um desses itens pode ter um valor na unidade definida para aquele item.

O usuário-projetista, no momento em que está definindo os critérios de QoS da aplicação SOA, visualiza essa ontologia num formato de árvore, onde no primeiro nível se encontram os atributos e no segundo nível os itens pertencentes ao atributo. O usuário-projetista escolhe um conjunto de itens, pertencendo a um ou mais atributos de QoS e define os valores que esses itens devem ter.

A ontologia de QoS foi desenvolvida dentro do escopo do trabalho de (SOUZA, 2009), onde podem ser obtidos maiores detalhes sobre a mesma.

3.5 Considerações

Este capítulo apresentou o modelo conceitual utilizado na concepção da arquitetura proposta nesse trabalho.

A seção 3.1 apresentou a proposta do trabalho, explicando em linhas gerais o funcionamento do modelo proposto. As seções 3.2 e 3.3 apresentaram respectivamente os requisitos necessários e os atores e casos de uso envolvidos. A seção 3.4 dedicou-se a apresentar em detalhes a arquitetura proposta.

A contribuição deste capítulo foi a apresentação da visão geral do cenário trabalhado e de detalhes de todas as entidades envolvidas, além de expor detalhes dos módulos da abordagem proposta e de como estes podem ser utilizados, possibilitando um melhor enquadramento do trabalho.

Capítulo 4

Implementação do Protótipo

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

Este protótipo é, como descrito por (SCHACH, 2005), um protótipo de prova de conceito, que é construído para se estabelecer a viabilidade, limitações e direções de um projeto de software. São apresentados detalhes de sua arquitetura e funcionamento e escolhas tecnológicas feitas durante o desenvolvimento.

4.1 Tecnologias e Ferramentas utilizadas na Implementação

Esta seção apresenta as principais ferramentas e tecnologias utilizadas na concepção do protótipo computacional. Procurou-se utilizar, tanto quanto possível, ferramentas abertas e gratuitas, mas de boa qualidade.

4.1.1 Eclipse Plataform

A plataforma Eclipse é um ambiente *open source* para desenvolvimento de software. Devido à sua extensibilidade e interface gráfica de extrema qualidade, o Eclipse tem sido uma das IDE

(*Integrated Development Enviroment*) mais utilizadas profissionalmente e em trabalhos acadêmicos (LÜER, 2002).

A ferramenta suporta diversas linguagens de programação (por exemplo, Java, C/C++, PHP) e pode ser executada em muitas plataformas, como Windows, Linux, Solaris e MacOS. Oferece aos desenvolvedores as ferramentas necessárias para criar aplicativos profissionais de desktop, empresariais, Web etc.

O Eclipse começou como um projeto da IBM do Canadá. Ele foi desenvolvido para ser um substituto da IDE VisualAge. Em novembro de 2001 um consórcio foi formado para garantir o desenvolvimento do Eclipse como software livre. Em janeiro de 2004, a Fundação Eclipse foi criada (WIKIPEDIA, 2010).

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

A Figura 46 mostra a estrutura do Eclipse em termos de *plug-ins* e suas relações de dependência. Vê-se o ambiente com um grande conjunto de componentes provindo funcionalidades específicas.

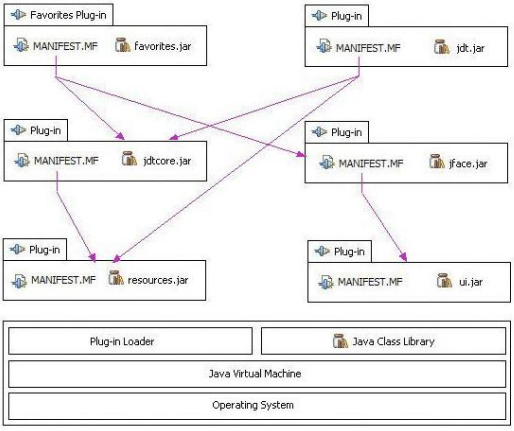


Figura 46 – Estrutura de plug-ins do Eclipse
Fonte: (CLAYBERG *et al.*, 2006)

Utilizou-se nesse trabalho a versão 3.5 (*Galileo*) do Eclipse. Essa escolha foi feito devido à experiência do autor no uso dessa ferramenta e por ser a última versão disponível no momento em que essa dissertação foi escrita.

4.1.2 IBM Websphere Business Modeler

O Websphere Business Modeler (WBM) é uma ferramenta para modelagem de processos de negócio no formato BPMN desenvolvida pela IBM. Ela possui recursos de modelagem, simulação e análise, de forma a permitir que os analistas de negócios possam conceber, documentar e implantar processos de negócio (IBM, 2010).

O WBM foi projetado para permitir a modelagem de diversos artefatos, entre eles (NOEL, 2005):

- Métricas de progresso e desempenho do processo;
- Dados históricos da execução do processo;
- Funções organizacionais e suas inter-relações;
- Funções de negócios fornecidas por aplicações ou serviços;
- Regras de negócios e cooperações entre diferentes empresas;
- Dependências temporais complexas, seqüenciamento baseado em eventos e dependência de sub-processos.
- A ferramenta da IBM simplifica a modelagem do processo através o uso de um ambiente estruturado que prove uma boa interação com o usuário na fase de concepção do processo, permitindo facilmente incorporar mudanças em modelos já existentes, geralmente sem a necessidade de esforços de desenvolvimento (NOEL, 2005).
- Os processos modelados no WBM podem ser exportados para a linguagem BPEL, de forma a serem implementados no nível de TI.
- Outras ferramentas da família Websphere geralmente são usadas em conjunto com o WBM. Entre elas destaca-se o Websphere Integration Developer, que possibilita o desenvolvimento de soluções de integração junto aos processos modelados e o Websphere Process Server, que provê uma ambiente integrado de execução e avaliação dos processos de negócio.
- O Websphere Business Modeler foi escolhido como o editor de processos de negócio neste trabalho por ser uma ferramenta de mercado largamente conhecida, além de proporcionar uma boa experiência de uso aos seus usuários. Outro motivo foi o fato de ser baseada na plataforma Eclipse, que permite sua extensão através do uso de *plug-ins*.

4.1.3 Service Component Architecture (SCA)

A Service Component Architecture (SCA) é um padrão para a composição e implantação de aplicações que utilizam a Arquitetura Orientada a Serviços (SOA). Quando se utiliza SCA, é possível escrever código em qualquer linguagem de programação, se preocupando somente com a lógica de negócios. O SCA permite abstrair vários detalhes específicos de tecnologias de implementação, de forma que o ambiente SCA trata de ligar os componentes heterogêneos (MARGOLIS, 2007). A especificação SCA define, portanto, uma mecanismo comum de como combinar seus componentes para se criar uma aplicação (CHAPPELL, 2007).

Um componente em SCA é um pedaço de código escrito em alguma linguagem. Esses componentes podem estar distribuídos na mesma máquina 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. Este 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*. A forma como os componentes se comunicam dentro de um domínio é determinada pelo *middleware*. Por exemplo, caso os componentes estejam na mesma máquina, o mecanismo de comunicação escolhido pode ser o acesso direto a memória. Se estiverem em máquinas diferentes, o *middleware* pode escolher o uso de RMI por exemplo. Fica ao encargo da implementação SCA escolher o mecanismo de comunicação mais apropriado em cada situação.

A comunicação fora do domínio SCA é feita através de protocolos interoperáveis. Por exemplo, dois domínios gerenciados por *middlewares* de diferentes fornecedores podem se comunicar através do uso de serviços *web*. Também é possível atribuir explicitamente quais são os conectores (*bindings*) que um componente aceita. Pode-se, por exemplo, dizer que um componente é acessível via protocolo SOAP (serviços *web*), *Enterprise Java Beans* (EJB) e CORBA (CHAPPELL, 2007).

Toda essa configuração é feita de forma declarativa, com o uso da SCDL, o que evita que se mexa no código fonte dos componentes.

Dessa forma, SCA fornece uma estrutura para lidar com aplicações heterogêneas e que mudam constantemente, simplificando o gerenciamento das mesmas e tornando mais ágil o seu desenvolvimento.

Utilizou-se SCA nesse trabalho, pois este fornece uma metodologia para a concepção de sistemas SOA, arquitetura conceitual dos serviços *web*. Como este trabalho utiliza esse paradigma, optou-se por utilizar essa metodologia.

4.1.4 Apache ODE

O Apache ODE (*Orchestration Director Engine*) é um motor de execução de processos de negócio escritos na linguagem WS-BPEL. Ele é responsável por interagir com os serviços *web*, enviando e recebendo mensagens, lidando com a manipulação de dados e recuperação de erros. ODE suporta a execução de processos de longa e curta duração, orquestrando todos os serviços que fazem parte da aplicação (APACHE, 2010b).

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

4.1.5 Intalio BPMS

O BPMS é uma suíte de ferramentas de gerenciamento de processos de negócio desenvolvida pela empresa Intalio. Ela é construída em cima de diversas ferramentas open-source, como o *Eclipse BPMN Modeler*, *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 de aplicações baseado no *Apache ODE*. Uma interface WEB administrativa permite invocar e monitorar o estado dos processos, analisando quais estão em execução, quais foram completados etc.

O Intalio BPMS foi utilizado nesse trabalho por prover uma interface administrativa intuitiva, que permite acompanhar a situação dos processos. Ele, portanto, serve como um complemento ao Apache ODE, este primeiro executando de fato os processos e esse último permitindo acompanhá-los de uma forma amigável.

4.1.6 Apache jUDDI

O Apache jUDDI é uma implementação em Java do padrão *Universal Description Discovery and Integration* (UDDI) para armazenamento das referências dos serviços *web* publicados por provedores. Ele é um projeto *open-source* mantido pela fundação Apache. Alguns dos seus recursos são (APACHE, 2010a):

- Independência de plataforma;
- Poder ser usado com qualquer banco de dados relacional que suporte o padrão ANSI SQL (MySQL, Oracle, Sybase, Derby etc.);
- Compatível com qualquer servidor de aplicações Java que suporte a especificação Servlet 2.3 (por exemplo; Apache Tomcat);
- Suporte total à especificação UDDIv3 (OASIS, 2005);
- Fornece um cliente UDDI que pode ser usado para se conectar com qualquer implementação do padrão UDDI.

Escolheu-se o Apache jUDDI como implementação de repositórios de serviços. A motivação é o 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 em Java da especificação UDDI v3. Além disso, por ser uma implementação do padrão UDDIv3, ele suporta a busca em repositórios distribuídos, que é um dos requisitos do mecanismo de busca de serviços, no trabalho de (SOUZA, 2009).

4.1.7 Apache Tuscany

O Apache Tuscany é um projeto *open source* da fundação Apache que provê uma infraestrutura que implementa *Service Component Architecture* (SCA) (LAWS *et al.*, 2010).

A ferramenta provê uma integração transparente com diversas tecnologias, de forma a permitir um fácil desenvolvimento de aplicações baseadas na Arquitetura Orientada a Serviços. O módulo de execução do Tuscany foi projetado para ser minimalista e permitir a implantação em diversos ambientes (WIKIPEDIA, 2010).

Alguns dos recursos oferecidos pelo Apache Tuscany são (APACHE, 2010d):

- Um modelo para criação de composições de aplicações, através da definição de serviços e de seus relacionamentos.

Os serviços podem ser implementados com qualquer tecnologia;

- Permite que os desenvolvedores criem serviços que possuem somente a lógica de negócios. Os protocolos de comunicação são abstraídos e gerenciados pelo Tuscany;
- As aplicações podem facilmente se adaptar a mudanças de infraestrutura, já que aspectos de comunicação e de QoS (transações, segurança) são configurados de maneira declarativa;
- Aplicações existentes podem trabalhar em conjunto com composições SCA, de maneira a ir ao encontro de uma arquitetura mais flexível e reutilizável.

O Apache Tuscany é a implementação de referência da arquitetura SCA (LAWS *et al.*, 2010). Por ser gratuito, bem suportado e implementado em Java, optou-se por seu uso nesse trabalho.

4.1.8 Bibliotecas de Suporte

Esta seção apresenta algumas bibliotecas utilizadas na implementação do protótipo. Estas bibliotecas fornecem funcionalidades específicas, apresentadas na Tabela 10:

Tabela 10 – Bibliotecas de Suporte

Biblioteca	Versão	Finalidade	Site
Apache CXF	2.2.10	Suporte a serviços *web*	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

Essas bibliotecas, apesar de não desempenharem um papel principal na implementação desse protótipo, auxiliam na execução de algumas funções, de forma a poupar esforço de desenvolvimento.

4.2 Detalhes de Implementação dos Módulos da Arquitetura

Esta seção apresenta os detalhes da implementação do protótipo. O objetivo é dar uma visão geral dos módulos implementados e comentar sobre as tecnologias utilizadas para a concepção dos mesmos.

A implementação do protótipo computacional utilizou-se idéias da abordagem de desenvolvimento incremental. De acordo com (O'DOCHERTY, 2005) (2005), a metodologia incremental é caracterizada pela entrega de produtos intermediários, de forma que o sistema seja construído incrementalmente. Por exemplo, no desenvolvimento de um software, a versão 1.0 contém as funcionalidades básicas do sistema. Já a versão 1.1 contém um conjunto um pouco maior de funcionalidades e a versão 2.0 contém todas as funcionalidades do sistema. Nesse sentido a metodologia incremental permite que o interessado no *software* (*stakeholder*) possa acompanhar a evolução do desenvolvimento, podendo inclusive solicitar mudanças nos requisitos e funcionalidades (PRESSMAN, 2004).

No contexto desse trabalho, a abordagem incremental foi julgada adequada para servir como guia para implementação, pois não se tinha num primeiro momento o escopo de implementação completo e consolidado. Em vários momentos foi necessário fazer mudanças de funcionalidades e ajustes, o que inviabilizava ter uma especificação completa do sistema já na fase inicial (como ocorre em outras metodologias, como o modelo cascata, por exemplo). Dessa forma, dividindo a implementação em versões entregáveis e que iam ficando incrementalmente mais completas com o decorrer do tempo, permitiu-se que o protótipo fosse desenvolvido de uma maneira mais ágil e adequada.

Com relação à metodologia de desenvolvimento de software, ressalta-se que o enfoque desse capítulo não é descrever de que forma o protótipo foi implementado com o rigor necessário a um trabalho desta área. Portanto, as seções a seguir irão descrever a implementação dos módulos da arquitetura de forma mais geral, ressaltando as principais características e escolhas feitas.

4.2.1 Plug-in de Integração com o Catálogo

Este módulo é responsável por fornecer as funcionalidades do catálogo junto ao editor de processos de negócio. Ele foi implementado como um *plug-in* do *IBM Websphere Business Modeler*, o editor BPM escolhido nesse trabalho.

A implementação em *Java* é composta por seis pacotes, cada um responsável por uma funcionalidade específica do plug-in. A Figura 47 apresenta esses pacotes:

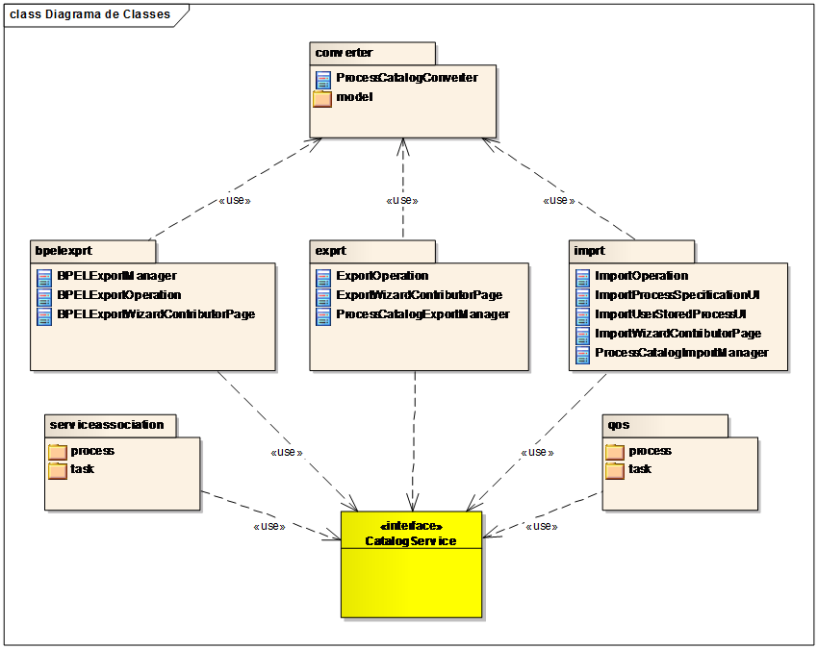


Figura 47 – Pacotes do plug-in de integração

- *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 (vide seção 3.4.3);
- *bpelexprt* – responsável pela exportação do processo para a linguagem de execução BPEL;
- *exprt* – faz a exportação do processo para o catálogo;

- *imprt* – responsável por importar o processo do catálogo para o editor;
- *serviceassociation* – faz a busca e associação dos serviços às atividades do processo. Possui dois subpacotes: *process* e *task*. O primeiro permite fazer a busca e associação de uma só vez a todas às atividades do processo. O segundo permite fazê-lo individualmente para cada atividade do processo;
- *qos* – permite a associação de critérios de QoS às atividades do processo. Assim como o pacote *serviceassociation*, possui os subpacotes *process* e *task*, que respectivamente permitem definir critérios de QoS para todas as atividades do processo de uma vez só, ou individualmente.

Os pacotes *bpelexprt*, *exprt*, *imprt*, *serviceassociation* e *qos* interagem com a interface de serviço *CatalogService*, permitindo que o *plug-in* execute as operações do Catálogo de Processos de Negócio.

O endereço do serviço é obtido através de um repositório UDDI. A Figura 48 apresenta a classe *WebServiceLocator* , que permite fazer a consulta ao repositório.

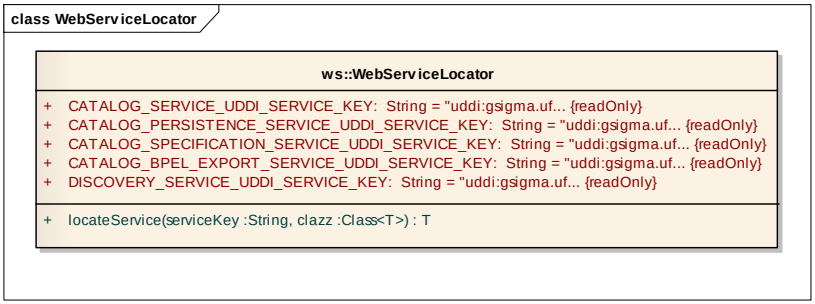


Figura 48 – Classe WebServiceLocator

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 p*roxy* que implementa a interface fornecida e contém a referência do serviço desejado.

O *Plug-in* de Integração com o Catálogo possui várias telas que permitem que o usuário interaja com o ambiente. A tecnologia escolhida para a implementação dessas telas foi o *SWT - Standard Widget Toolkit* (ECLIPSE, 2010). 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 *et al.*, 2006).

4.2.2 Serviços do Catálogo de Processos de Negócio

Este módulo provê as funcionalidades do Catálogo de Processos de Negócio. Ele está organizado internamente em serviços e estruturado através do uso da *Service Component Architecture* (SCA).

O diagrama da Figura 49 mostra os componentes do catálogo, o *Catálogo*, o *Exportador BPEL*, o *Gerenciador de Especificações* e o *Componente de Persistência*. Todos eles fazem parte do mesmo domínio SCA.

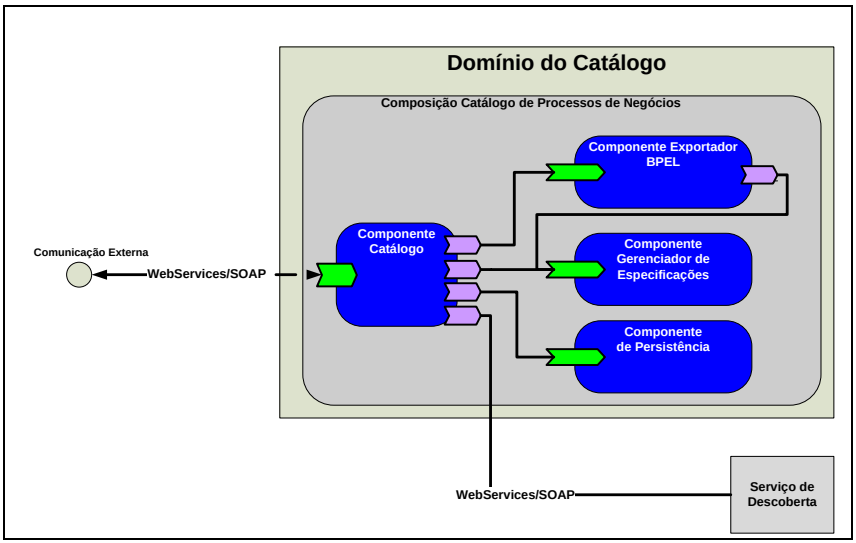


Figura 49 – Diagrama SCA do Catálogo de Processos de Negócio

O Serviço de Descoberta está representado fora do domínio do catálogo, pois o mesmo não faz do escopo de implementação do catálogo. Isso significa que ele é um serviço externo, que não está sujeito ao gerenciamento pelo *container* do domínio SCA. Dessa forma, o seu acesso é feito externamente, através de uma conexão SOAP (W3C, 2007a).

O canto esquerdo da Figura 49 mostra uma porta de comunicação externa, também utilizando o SOAP. Esta porta é usada para permitir a comunicação do Plug-in de Integração com o Catálogo (vide seção 4.2.1).

Cada um dos componentes é implementado por um pacote na linguagem Java, conforme a Tabela 11.

Tabela 11 – Pacotes Java do Catálogo de Processos de Negócio

Componente	Pacote
Exportador BPEL	br.ufsc.gsigma.catalog.services.bpelexport
Gerenciador de Especificações	br.ufsc.gsigma.catalog.services.specifications
Persistência	br.ufsc.gsigma.catalog.services.persistence
Serviço de Descoberta	br.ufsc.gsigma.servicediscovery

Cada um desses pacotes possui uma classe responsável por definir a interface de serviço e outra por implementar esta interface, possuindo a codificação das operações. A Tabela 12 apresenta essas classes:

Tabela 12 – Classes de Interface e Implementação

Componente	Interface	Implementação
Exportador BPEL	BPELExporterService	BPELExporterServiceImpl
Gerenciador de Especificações	CatalogSpecificationService	CatalogSpecificationServiceImpl
Persistência	CatalogPersistenceService	CatalogPersistenceServiceImpl
Serviço de Descoberta	DiscoveryService	-

Nota-se que o Serviço de Descoberta não possui uma classe de implementação definida. Isso ocorre, porque, como já dito, esse é um serviço externo ao catálogo, de forma que somente tem-se acesso à definição de operações através da interface. Mais detalhes de como esse serviço foi implementado podem ser vistos em (SOUZA, 2009).

O componente Catálogo age como uma “fachada”, que concentra as operações dos demais componentes. Dessa forma ele permite delegar as requisições para o componente mais adequado.

Na Figura 50 mostra-se como isso funciona. A interface CatalogService implementa as quatro interfaces apresentadas no canto

direito da figura. Isso permite que a CatalogService reúna todas as operações fornecidas por essas interfaces.

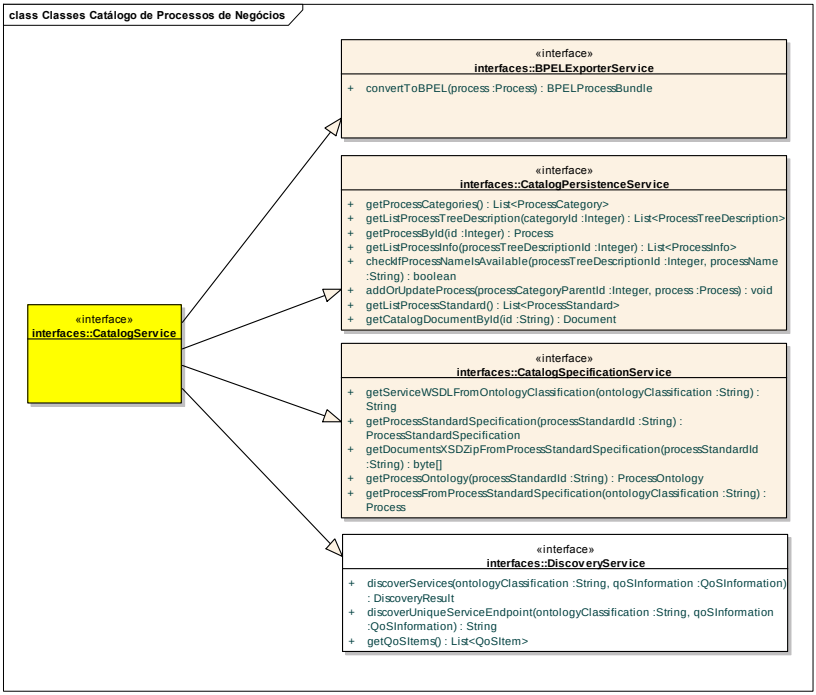


Figura 50 – Diagrama de Classes – Catálogo de Processos de Negócio

A implementação dessa interface, a *CatalogServiceImpl*, possui referências às instâncias dos componentes implementadores , de forma que quando uma requisição é feita ao catálogo, esta automaticamente é redirecionada à instância do componente apto a respondê-la (por exemplo, uma requisição de uma operação de persistência automaticamente é encaminhada ao componente de persistência).

O gerenciamento das instâncias dos componentes e a ligação entre eles é de responsabilidade do container SCA. O projetista somente precisa montar a composição, definido quais componentes fazem parte e de que forma eles se relacionam. Isso é feito através de um arquivo SCDL (*Service Component Definition Language*), apresentado na Figura 51. Este arquivo reflete a mesma configuração mostrada na Figura 49.

```
<csa:composite xmlns:csa="http://docs.oasis-
open.org/ns/opencsa/sca/200912" autowire="false" name="architecture"
targetNamespace="http://sca.infrastructure.gsigma.ufsc.br">

 <csa:component name="CatalogComponent">
  <csa:implementation.java
class="br.ufsc.gsigma.catalog.services.impl.CatalogServiceImpl" />
  <csa:service name="CatalogService">
   <csa:interface.java
interface="br.ufsc.gsigma.catalog.services.interfaces.CatalogService" />
   <csa:binding.ws uri="http://0.0.0.0:8081/CatalogService" />
  </csa:service>
  <csa:reference name="catalogSpecificationService"
target="CatalogSpecificationComponent/CatalogSpecificationService" />
  <csa:reference name="bpelExporterService"
target="BPELExporterComponent/BPELExporterService" />
  <csa:reference name="catalogPersistenceService"
target="CatalogPersistenceComponent/CatalogPersistenceService" />
 </csa:component>

 <csa:component name="CatalogSpecificationComponent">
  <csa:implementation.java
class="br.ufsc.gsigma.catalog.services.specifications.impl.CatalogSpecific
ationServiceImpl" />
  <csa:service name="CatalogSpecificationService">
   <csa:interface.java
interface="br.ufsc.gsigma.catalog.services.specifications.interfaces.Catal
ogSpecificationService" />
  </csa:service>
 </csa:component>

 <csa:component name="BPELExporterComponent">
  <csa:implementation.java
class="br.ufsc.gsigma.catalog.services.bpelexportAsync.impl.BPELExporterAs
yncServiceImpl" />
  <csa:service name="BPELExporterService">
   <csa:interface.java
interface="br.ufsc.gsigma.catalog.services.bpelexport.interfaces.BPELExpor
terService" />
  </csa:service>
  <csa:reference name="catalogSpecificationService"
target="CatalogSpecificationComponent/CatalogSpecificationService" />
 </csa:component>

 <csa:component name="CatalogPersistenceComponent">
  <csa:implementation.spring
location="br/ufsc/gsigma/catalog/services/persistence/conf/applicationCont
ext.xml" />
 </csa:component>

</csa:composite>
```

Figura 51 – Código fonte do arquivo de composição SCDL

O ambiente SCA escolhido para se fazer a implantação do catálogo foi o *Apache Tuscany 2.0* (APACHE, 2010d). Todos os componentes do catálogo foram configurados para executar na mesma

máquina, com exceção do Serviço de Descoberta, que roda numa máquina separada.

Os serviços que compõem o catálogo são registrados em um repositório UDDI. A implementação escolhida para este foi o Apache jUDDI. Este registro permite que outras aplicações, que não sabem a priori o endereço onde os serviços do catálogo se encontram (por exemplo o *plug-in* de integração), possam obter esses endereços. O único requisito é ter conhecimento do endereço do repositório, de forma a poder consultá-lo.

As seções a seguir apresentam, em maiores detalhes, os componentes *Gerenciador de Especificações*, *Persistência* e *Exportador BPEL*.

4.2.2.1 Gerenciador de Especificações

Este componente faz o Gerenciamento das Especificações de Processos de Negócio presentes no Catálogo.

A especificação é armazenada fisicamente através de uma pasta, que possui um conjunto de arquivos que descrevem a especificação.

A pasta "process_standards" contém uma pasta chamada "ubl", referente à especificação UBL, que possui a seguinte estrutura:

- *process* – pasta que possui os processos da especificação, descritos na linguagem XML;
- wsdl – possui as definições em WSDL dos serviços que implementam as atividades do processo;
- xsd – possui a definição dos documentos trocados pelas atividades do processo;
- specification.xml – arquivo que contém a descrição completa da especificação, em formato XML.

Os arquivos XML são serializados e desserializados em objetos Java através do *framework Simple XML*. Detalhes da forma como os objetos foram modelados podem ser vistos nas seções 3.4.2.1 e 3.4.3.

Os documentos trocados utilizam o formato XSD (*XML Schema Definition*) e foram obtidos da especificação oficial da UBL, disponível em (OASIS, 2006b).

Os seguintes processos da especificação UBL foram modelados:

- Criação de Catálogo (*Create Catalogue*);
- Processo de Ordem (*Ordering Process*);

- Processo de Pagamento (*Payment Process*).

4.2.2.2 Persistência do Modelo de Dados

O componente de persistência faz o armazenamento do modelo de dados (visto na seção 3.4.3) em um banco de dados relacional. Utilizou-se o framework *Hibernate* (JBOSS, 2010) para fazer o mapeamento da modelagem em objetos para o banco de dados relacional.

O banco escolhido foi o HSQLDB (HYPERSQL, 2010), que funciona integrado à aplicação, sem a necessidade de ser instalado isoladamente. Isso traz a facilidade de se executar o ambiente em qualquer máquina, sem precisar instalar o banco de dados separadamente.

O esquema relacional usado na persistência será explicado em maiores detalhes na seção 4.3.4.

4.2.2.3 Exportador BPEL

O Exportador BPEL faz a conversão dos processos descritos em BPMN armazenados no catálogo para um formato executável, no caso a *Business Process Execution Language* (BPEL).

Os processos exportados são compatíveis com a especificação BPEL 2.0 (OASIS, 2007). Apesar de ser um padrão, a forma como os arquivos são exportados e organizados varia de acordo com cada implementação da especificação. Neste trabalho, optou-se por gerar código compatível com o motor de execução Apache ODE (APACHE, 2010b). Nada impediria, com pequenas alterações, de usar algum outro motor de execução, desde que o mesmo implemente a especificação BPEL 2.0.

O algoritmo de conversão BPMN para BPEL foi construído utilizando-se os passos descritos a seguir.

Primeiramente avaliaram-se os processos da especificação UBL 2.0 (OASIS, 2006b) em termos das suas estruturas de controle de fluxo e de algumas características, como, por exemplo, a troca de documentos entre as atividades do processo. Esta análise permitiu adequar o conversor de linguagem no sentido de fazer dessas estruturas descritas em BPMN para as suas equivalentes no formato BPEL. Esta conversão permite que a lógica do processo original, descrita em BPMN, seja

mantida. Apresenta-se na Tabela 13 o resultado desta análise, e em seguida explica-se cada uma das estruturas e características comparadas.

Tabela 13 – Comparação das Estruturas dos Processos UBL

Processo	Troca de Documentos	União Inclusiva	União Exclusiva	Decisão	Bifurcação	Ciclo Simples
Sourcing – Catalogue Provision						
Create Catalogue Process	X	X		X	X	X
Update Item Specification Process	X	X		X	X	X
Update Catalogue Pricing Process	X	X		X	X	X
Delete Catalogue Process	X			X		X
Sourcing – Customer Initiated Sourcing						
Customer Initiated Sourcing Process	X					
Sourcing – Punchout Sourcing Process						
Punchout Sourcing Process	X					
Fulfilment						
Fulfilment with Despatch Advice Process	X		X	X	X	
Fulfilment with Receipt Advice Process	X		X	X		X
Billing – Traditional Billing						
Billing with Credit Note Process	X	X	X	X	X	X
Billing with Debit Note Process	X	X	X	X	X	X
Billing – Self Billing						
Self Billing with Credit Note Process	X	X	X	X	X	X
Self Billing with Self Billed Credit Note Process	X			X		X
Billing – Freight Billing						
Freight Billing Process	X					
Billing – Reminder for Payment						
Reminder for Payment Process	X					
Ordering						
Ordering Process	X		X	X	X	X
Payment						
Payment Process	X	X				
Statement Process	X					
Initiate Transport Services						
Initiate Transport Services Process	X	X			X	
Certification of Origin of Goods						
Certification of Origin of Goods Process	X		X	X		X
Report Status of Goods						
Report Status of Goods Process	X		X			

- Troca de Documentos – se há troca de documentos entre as atividades do processo. Por exemplo, se numa atividade ocorrer a criação de uma ordem de compra e na outra o recebimento dessa ordem, houve uma troca do documento "ordem de compra" entre as duas atividades;
- União Inclusiva – se existe um ponto no processo em que dois ou mais fluxos paralelos tenham que se unir para que o processo possa continuar;

- União Exclusiva – se existe um ponto no processo onde somente um dos fluxos paralelos precise terminar para que o processo possa continuar;
- Decisão – se a escolha do fluxo a ser seguido é feita com base num elemento de decisão, que utilize alguma expressão de escolha;
- Bifurcação – se num determinado ponto o processo é quebrado em dois ou mais fluxos paralelos;
- Ciclo Simples – se há no processo uma região que pode entrar em ciclo. A qualidade simples refere-se a não ter ciclos aninhados (um ciclo dentro do outro).

O próximo passo foi fazer a correspondência dessas estruturas do processo BPMN para as estruturas equivalentes utilizadas pela linguagem BPEL. Utilizou-se como base o trabalho de (FASBINDER, 2007), que detalha a correspondência das estruturas utilizadas pelo *IBM Websphere Business Modeler* (que utiliza a linguagem BPMN) e as estruturas correspondentes na linguagem BPEL. No Apêndice C é apresentada a conversão do processo de pagamento (*Payment Process)* da linguagem BPMN para BPEL.

O último passo, após fazer a conversão das estruturas BPMN para BPEL, é fazer a exportação para um BPEL. Essa exportação é feita utilizando o mecanismo de *templating FreeMarker* (FREEMARKER, 2010), que permite gerar a linguagem através de arquivos de modelo. O modelo possui a estrutura básica do arquivo BPEL que será exportado. O exportador gera um conjunto de dados, com base no processo de entrada, que é anexado a esse modelo, em seguida processado pelo *Freemarker*. Em seguida, um conjunto de arquivos é gerado pelo *Freemarker*, contendo todos as informações do processo BPEL exportado. Alguns exemplos dessas informações são os XML *Schema* (W3C, 2000) dos tipos de dados utilizados pelo processo e as definições das interfaces (em formato WSDL) dos serviços vinculados ao processo exportado.

4.2.3 Serviços UBL

Os serviços UBL 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 contidos no Serviço de Gerência de Especificações (vide seção 4.2.2.1). O resultado da execução da

ferramenta é um conjunto de arquivos Java que descrevem as operações e tipos de dados trocados pelo serviço. Um desses arquivos é a interface de serviço, que é referenciada pela classe Java que implementa o mesmo. No Apêndice D pode-se ver um extrato do arquivo WSDL utilizado para descrever as interfaces dos serviços UBL.

Um dos requisitos do trabalho, exposto na seção 3.2, é que os processos exportados permitam a interação humana, isto é, que certas atividades do processo dependam da decisão humana para retornar a resposta. Como forma de resolver esse problema, foi criada a especificação BPEL4People (AGRAWAL *et al.*, 2007), que lida com esses aspectos de como modelar a interação humana dentro de um processo BPEL.

Nesse trabalho, algumas ideias do BPEL4People foram utilizadas para modelar a interação humana. Uma delas é modelar a requisição e reposta dos serviços *web* de forma assíncrona. Isso faz sentido, pois, por dependerem de humanos para retornar uma resposta, 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 (o processo BPEL). Apesar de ser simples, esse artifício permite ao mecanismo de execução de processos 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 pelo processo. Essa lista de tarefas é organizada numa fila, de forma que o usuário possa estar envolvido com vários processos. O mecanismo de *workflow* cuida de encaminhar as respostas do usuário às instâncias corretas dos processos que as requereram.

No que tange à execução dos serviços implementados, utilizou-se o *Jetty* (CODEHAUS, 2010), que é um pequeno servidor HTTP que executa em conjunto com o *Apache CXF*. A vantagem do seu uso é a não necessidade de implantar os serviços em um *container web* completo, como o Apache Tomcat (APACHE, 2010c). Isso ajuda a poupar recursos de hardware e dar agilidade ao desenvolvimento do protótipo.

Para que se pudesse avaliar o protótipo computacional desenvolvido, tornou-se necessário publicar esses serviços em repositórios distribuídos, chamados de UDDI. A implementação escolhida para esses repositórios foi o Apache jUDDI e a classe em Java utilizada para fazer a publicação dos serviços UBL está descrita no Apêndice E.

Esses repositórios distribuídos fazem parte de um ente chamado federação, que é o local onde os serviços são buscados pelo algoritmo de busca utilizado pelo Ambiente de Descoberta Dinâmica de Serviços. Maiores detalhes a respeito desse ambiente e da federação podem ser vistos em (SOUZA, 2009).

4.2.4 Ambiente de Execução de Aplicações

A ferramenta *Intalio BPMS*, que utiliza o motor *Apache ODE* para execução dos processos, foi escolhida para rodar os processos. Não se utilizou somente o *Apache ODE*, pois apesar dele fornecer toda a estrutura necessária para a execução dos processos, não possui uma interface gráfica amigável, que permitisse monitorar o andamento dos processos. Por isso, utilizou-se o *Intalio BPMS*, que possui uma interface de gerenciamento completa, permitindo a monitoração eficiente dos processos.

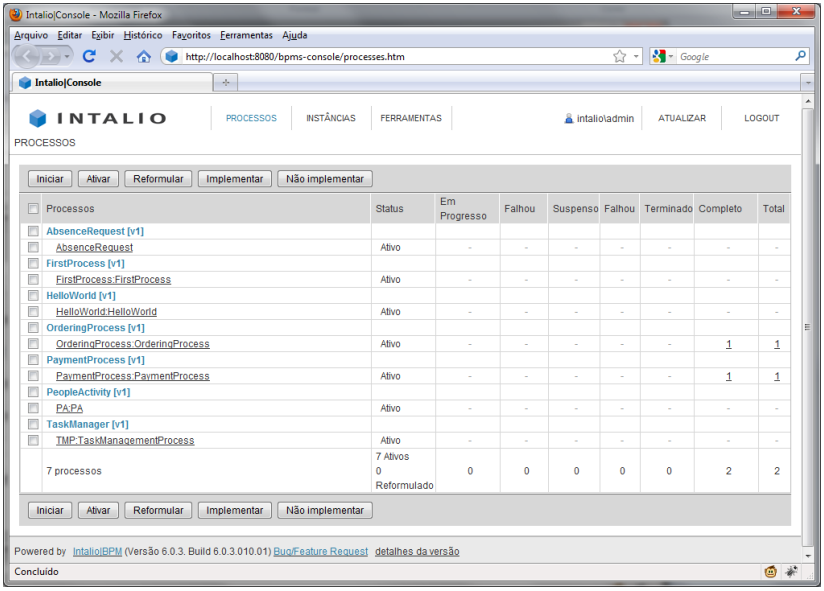


Figura 52 – Interface de Gerenciamento – Intalio BPMS

A Figura 52 apresenta a Interface de Gerenciamento do *Intalio BPMS*. Os processos são listados numa tabela, junto com algumas informações. O usuário pode iniciar um processo, ver em que situação ele se encontra etc.

4.2.5 Ambiente de Descoberta Dinâmica de Serviços

Como já foi falado, o Ambiente de Descoberta Dinâmica (SOUZA, 2009) é um componente da arquitetura do catálogo que não foi implementado pelo autor desta dissertação.

4.3 O Protótipo visto em Perspectivas

Uma arquitetura de software pode ser descrita através de diversas perspectivas, também chamadas de *views*. Elas permitem ver o sistema sobre diferentes óticas, como por exemplo na forma de unidades de códigos (*module view*), sobre a perspectiva de execução (*runtime view*), sobre a forma como sistema é implantado (*deployment view)* e como seus dados são modelados e armazenados (*data view*). O uso de

views consegue, na maioria dos casos, prover uma descrição completa do sistema (MERSON, 2005).

Esta seção apresenta o protótipo computacional visto através dessas perspectivas.

4.3.1 Module View

A *module view* é uma perspectiva que permite mostrar a estrutura do sistema em termos de módulos, que são as unidades de código utilizadas na implementação do sistema. O uso da *module view* possibilita representar a planta (*blueprint*) utilizada na construção do *software (MERSON, 2005)*.

A Figura 53 apresenta a *module view* do sistema proposto. Ela é formada por cinco camadas que são:

- (1) Camada do Plug-in de Integração Editor BPM;
- (2) Camada do Catálogo de Processos de Negócio;
- (3) Camada do Ambiente de Descoberta Dinâmica de Serviços;
- (4) Camada dos Serviços UBL;
- (5) Camada do Ambiente de Execução de Processos.

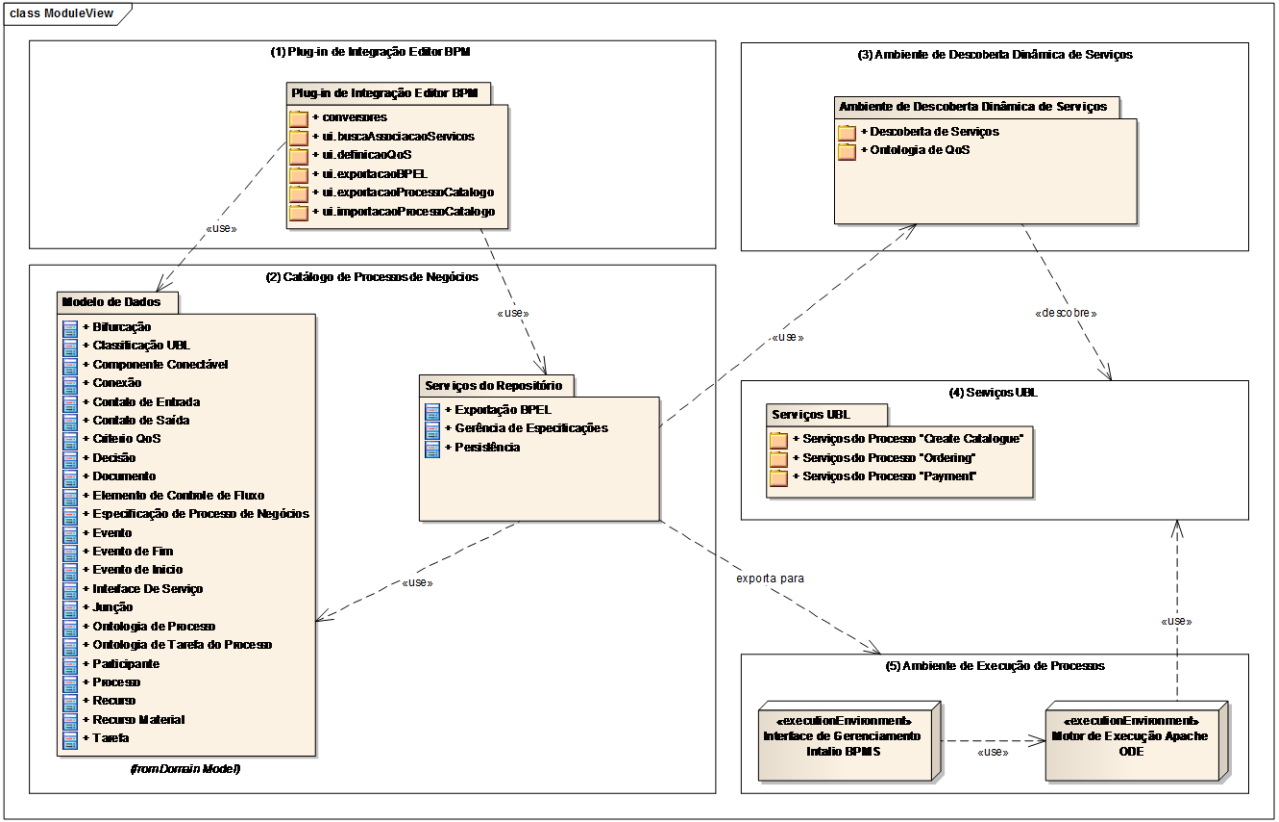


Figura 53 – *Module View*

Essas camadas são equivalentes aos módulos apresentados na seção 4.2. O objetivo é mostrar de que forma elas se relacionam umas com as outras, de forma a ter uma visão geral do sistema.

4.3.2 Runtime View

As *runtime views* permitem obter um "raio-X" do sistema em execução. Sua importância está em permitir entender o funcionamento do sistema e analisar as propriedades que se manifestam em tempo de execução, como, por exemplo, a performance (MERSON, 2005).

A Figura 54 mostra a *runtime view* do modelo proposto. Ela é composta por seis camadas:

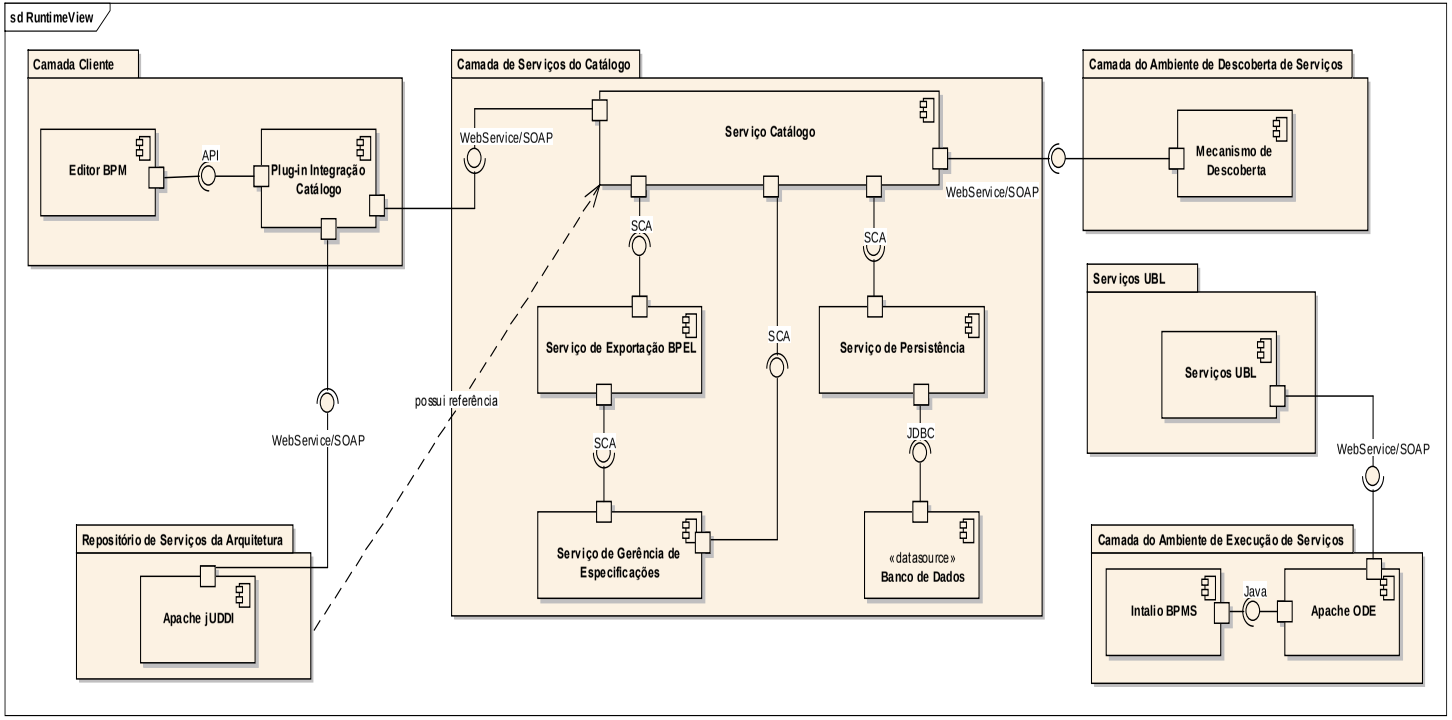


Figura 54 – *Runtime View*

A Camada do Cliente possui os componentes que interagem com o usuário. É composta pelo editor BPM e o *plug-in* de integração com o catálogo. O editor está integrado ao catálogo através do uso de uma API (da Plataforma Eclipse). A comunicação do *plug-in* com o catálogo é feita através do uso de serviços *web*. A referência ao serviço do catálogo é obtida através de uma consulta ao Repositório de Serviços da Arquitetura.

A Camada de Serviços do Catálogo possui o núcleo responsável pelas funcionalidades do catálogo. Seus componentes comunicam-se através do mecanismo de *binding* fornecido pelo *middleware* Apache Tuscany. Isso é possível porque todos os componentes do catálogo fazem parte do mesmo domínio SCA (vide seção 4.2.2).

O módulo de busca de serviços, contido no Ambiente de Descoberta Dinâmica de Serviços, utiliza serviços *web* como mecanismo de comunicação.

A Camada dos Serviços UBL contém todas as implementações dos serviços atrelados aos processos. Eles são acessados através do protocolo SOAP.

A última camada, responsável pela execução dos processos, é composta pelo *Intalio BPMS* e pelo *Apache ODE*. Ambos rodam integrados entre si, comunicando-se através dos mecanismos providos pelo Java. Neste caso, como estes dois elementos estão executando na mesma máquina, através de compartilhamento de memória

4.3.3 Deployment View

A *view* de *deployment* (ou de implantação) tem o objetivo de mostrar como o sistema será implantado dentro dos limites da infraestrutura física computacional. Esse diagrama pode incluir vários tipos de artefatos, tais como máquinas, processos, arquivos e dependências (O'DOCHERTY, 2005). Através dessa *view*, é possível saber qual a estrutura de hardware necessária para rodar o *software* (MERSON, 2005).

A Figura 55 apresenta o diagrama UML de implantação do modelo proposto. Nesse exemplo ele é implantando em sete máquinas. Como se trata de uma arquitetura SOA, cada um dos serviços que a compõe pode ser executado em máquinas distintas. Portanto, o número escolhido para ser demonstrado nesse diagrama reflete somente uma possível disposição de implantação do modelo proposto.

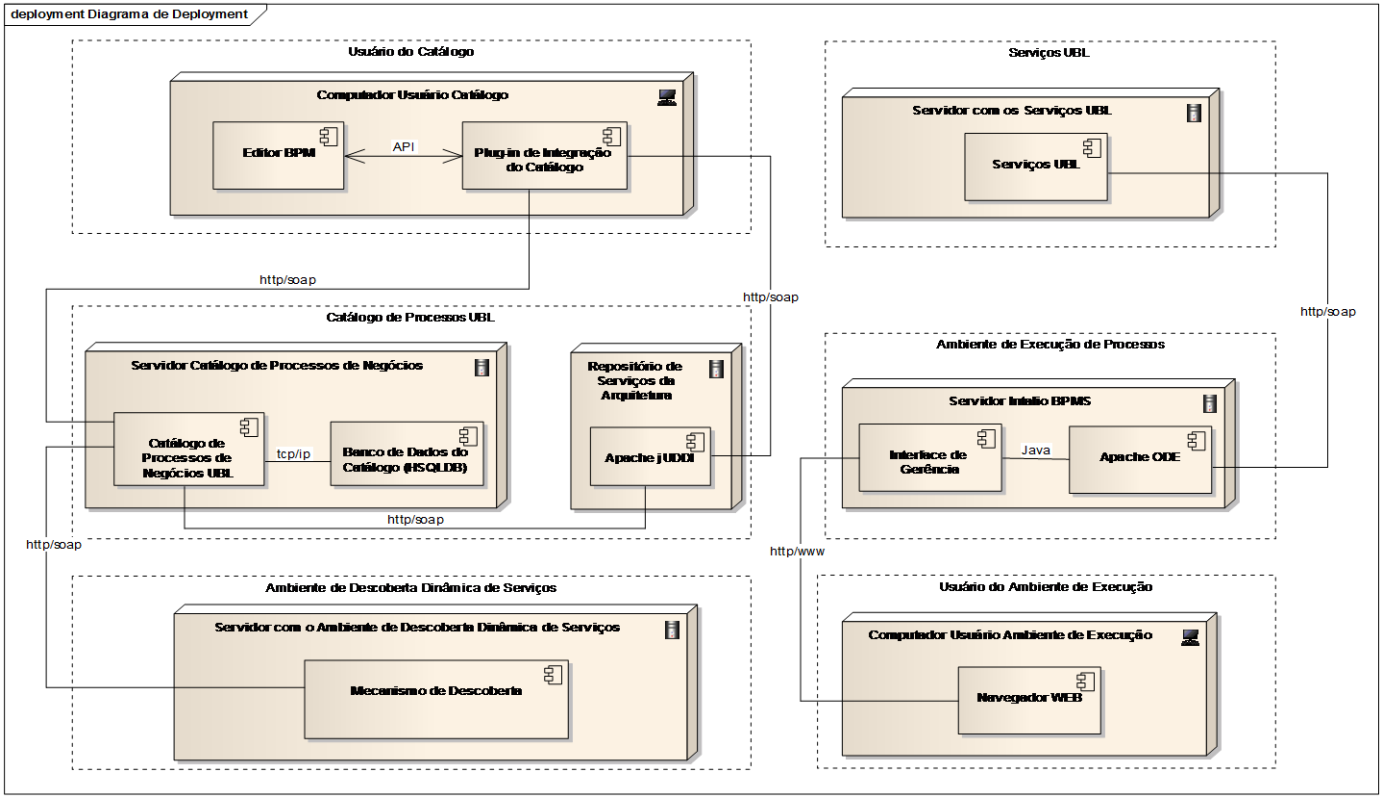


Figura 55 – Diagrama de Implantação do Modelo Proposto

A parte superior esquerda do diagrama mostra o usuário do catálogo, que tem o editor de processos executando em sua máquina (nodo Computador Usuário). O editor BPM está integrado ao *plug-in* de integração do catálogo. O *plug-in* se comunica com o repositório de serviços da arquitetura e com o catálogo através do protocolo SOAP.

O Catálogo de Processos de Negócio UBL está representado na parte central esquerda figura. Nesse exemplo ele está sendo implantado em duas máquinas. Na primeira delas está localizado o Servidor Catálogo de Processos de Negócio, que possui o catálogo propriamente dito e o banco de dados (BD) responsável por armazenar os dados do catálogo. A comunicação dele com o catálogo é feita através do protocolo TCP/IP, no dialeto específico da implementação do banco de dados. Na segunda máquina está localizado o servidor com o Repositório de Serviços da Aplicação. A implementação dele é o *Apache jUDDI*, acessado pelo protocolo SOAP.

O Ambiente de Descoberta Dinâmica de Serviços está representado na parte inferior esquerda da figura. Somente o mecanismo de busca está representado, que é o que interage com o catálogo. O acesso é feito através do protocolo SOAP.

Na parte superior direita da figura está localizado o nodo com os serviços UBL. Apesar de em um ambiente real eles estariam distribuídos em inúmeras máquinas, nesse protótipo, por questões de facilidade, eles estão todos juntos. Ressalta-se que no escopo desse trabalho, os serviços UBL são implantados somente com o intuito de avaliar o modelo proposto nessa dissertação. Num ambiente real, estes serviços fazem parte de uma federação, estando largamente distribuídos, em inúmeros repositórios. O trabalho de (SOUZA, 2009) contém maiores detalhes sobre essa federação de serviços.

A parte central direita da figura apresenta o Ambiente de Execução de Processos, implementado pelo *Intalio BPMS* e utilizando o motor de execução *Apache ODE*. Este ambiente acessa via SOAP os serviços UBL, orquestrando-os de acordo com as regras definidas nos processos BPEL.

Por último, o canto inferior direito apresenta o usuário que utiliza o ambiente de execução, que o acessa utilizando um navegador WEB, que se conecta na interface de gerenciamento de processos através de uma conexão HTTP/WEB.

4.3.4 Modelo de Dados

A perspectiva de modelo de dados é usada normalmente quando o sistema possui uma base de dados que precisa ser modelada (MERSON, 2005). No caso deste trabalho, o modelo de dados reflete a necessidade de se armazenar os processos do catálogo fisicamente, o qual será feito através de um banco de dados.

Na Figura 56, apresenta-se o diagrama entidade relacional (ER) do banco de dados. Utilizou-se o banco de dados HSQLDB para o gerenciamento das tabelas.

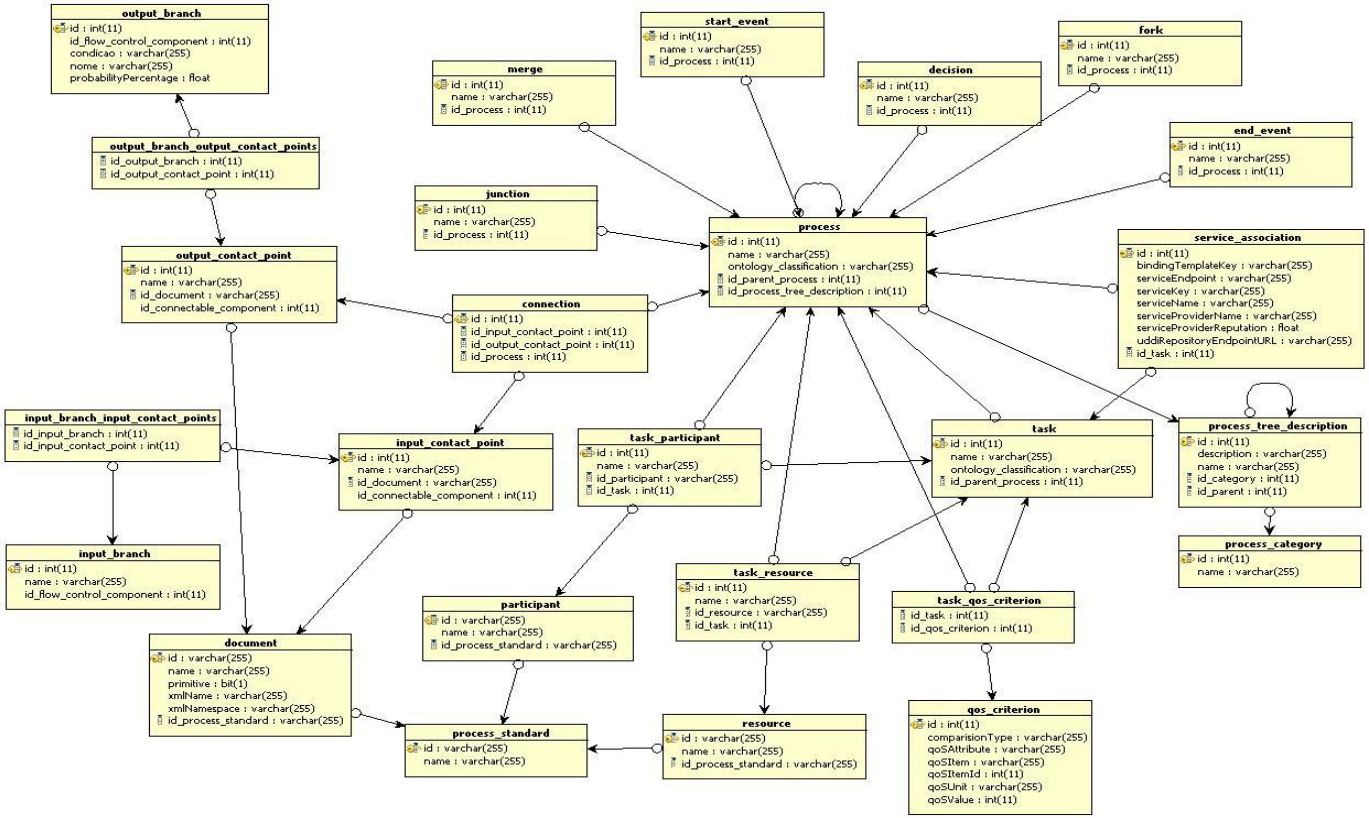


Figura 56 – Diagrama Entidade Relacional do Banco de Dados

4.3.5 Resumos das Ferramentas, Tecnologias e Padrões Utilizados e Integrados

Esta seção apresenta um resumo das ferramentas, tecnologias e padrões utilizados na implementação de cada um dos elementos da arquitetura conceitual proposta.

A implementação desses elementos demandou estudo e conhecimento de cada uma das tecnologias, ferramentas e padrões utilizados. Procurou-se utilizar o máximo possível de padrões e tecnologias abertas de forma a garantir que a solução proposta fosse interoperável. Isso pode ser visto na Tabela 14.

Tabela 14 – Ferramentas, Tecnologias e Padrões utilizados

Elemento da Arquitetura	**Serviços do Catálogo de Processos de Negócio**
Ação ou Funcionalidade	Armazenamento e gerenciamento de processos de negócio, persistência, gerência da especificação UBL e exportação para a linguagem BPEL.
Tecnologias	- BPEL; - XML; - Web-services; - UDDI; - SCA; - Java; - Banco de Dados SQL.
Ferramentas	- Eclipse Plataform; - Eclipse BPEL Designer; - Apache CXF; - Apache jUDDI; - Apache Tuscany; - JBOSS Hibernate; - HSQLDB; - Freemarker; - SimpleXML.
Padrões	- Universal Business Language (UBL); - Protocolo SOAP; - Web Service Definition Language (WSDL);

	- Universal Description Discovery and Integration (UDDI); - Business Process Execution Language (BPEL); - Service Component Architecture (SCA).
Protocolos de Comunicação	- Hipertext Transfer Protocol (HTTP); - Protocolo SOAP; - Java Database Connectivity (JDBC).
Elemento da Arquitetura	**Ontologia UBL**
Ação ou Funcionalidade	Categorização dos processos da especificação UBL e dos serviços *web* implementadores dos processos.
Tecnologias	- Web Ontology Language (OWL)
Ferramentas	- Stanford Prótegé
Padrões	- Web Ontology Language (OWL)
Elemento da Arquitetura	**Plug-in de Integração com o Editor BPM**
Ação ou Funcionalidade	Integração do ambiente de edição BPM com o Catálogo de Processos de Negócio
Tecnologias	- Eclipse Plug-in Framework; - XML; - Web-services; - UDDI; - Java.
Ferramentas	- Websphere Business Modeler; - Eclipse Plataform; - Apache CXF; - Apache jUDDI.
Padrões	- Business Process Modeling Notation (BPMN); - Protocolo SOAP; - Web Service Definition Language (WSDL); - Universal Description Discovery and Integration (UDDI).
Protocolos de Comunicação	- Hipertext Transfer Protocol (HTTP); - Protocolo SOAP.
Elemento da Arquitetura	**Ambiente de Execução de Processos**
Ação ou Funcionalidade	Execução e monitoramento dos processos de negócio descritos em BPEL
Tecnologias	- BPEL - XML

	- Web-services
Ferramentas	- Apache ODE; - Intalio BPMS
Padrões	- Protocolo SOAP; - Web Service Definition Language (WSDL); - Business Process Execution Language (BPEL).
Protocolos de Comunicação	- Hipertext Transfer Protocol (HTTP); - Protocolo SOAP.
Elemento da Arquitetura	**Serviços UBL**
Ação ou Funcionalidade	Implementações de serviços da especificação UBL utilizados para a avaliação do comportamento do modelo conceitual proposto.
Tecnologias	- XML; - Web-services.
Ferramentas	- Eclipse Platafom; - Apache CXF.
Padrões	- Universal Business Language (UBL); - Protocolo SOAP; - Web Service Definition Language (WSDL).
Protocolos de Comunicação	- Hipertext Transfer Protocol (HTTP); - Protocolo SOAP.

4.4 Exemplo de Funcionamento

Esta seção apresenta um exemplo de funcionamento do protótipo computacional implementado, mostrando um passo a passo resumido do sistema.

O fluxo de trabalho começa com o usuário-projetista iniciando a ferramenta de edição de processos, o *IBM Websphere Business Modeler*. Em seguida, o usuário escolhe a opção de importar um processo do catálogo de processos de negócio para o editor BPM. A Figura 57 apresenta a tela de importação do processo.

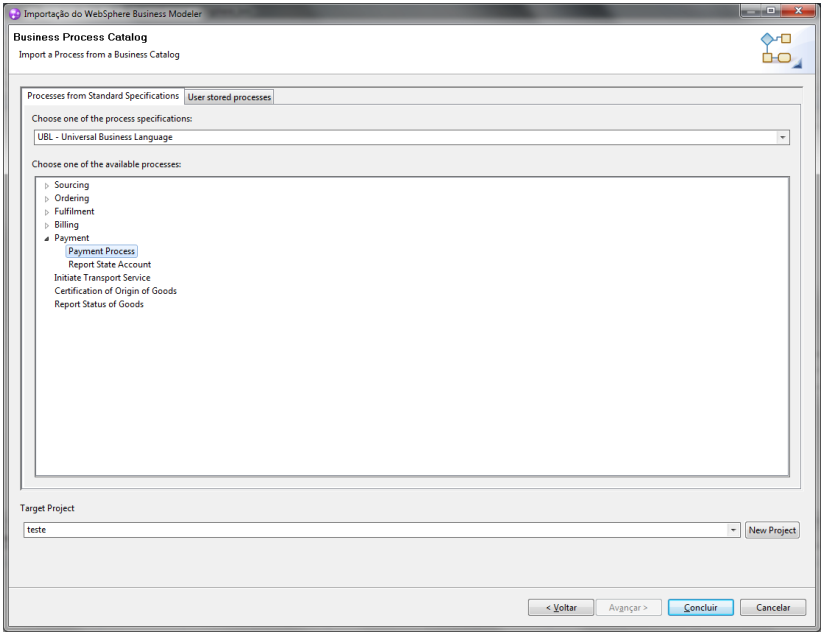


Figura 57 – Importação de Processo do Catálogo

O *plug-in* se conecta ao catálogo e lista todos os processos que estão disponíveis. O projetista escolhe um deles e solicita a importação, como pode ser visto na Figura 58. As duas abas presentes no canto superior da figura permitem escolher entre a importação de um processo da especificação (*Processes from Standard Specifications*) ou processos previamente armazenados no catálogo (*User stored process*). Neste exemplo considera-se que o usuário está importando um processo da especificação UBL.

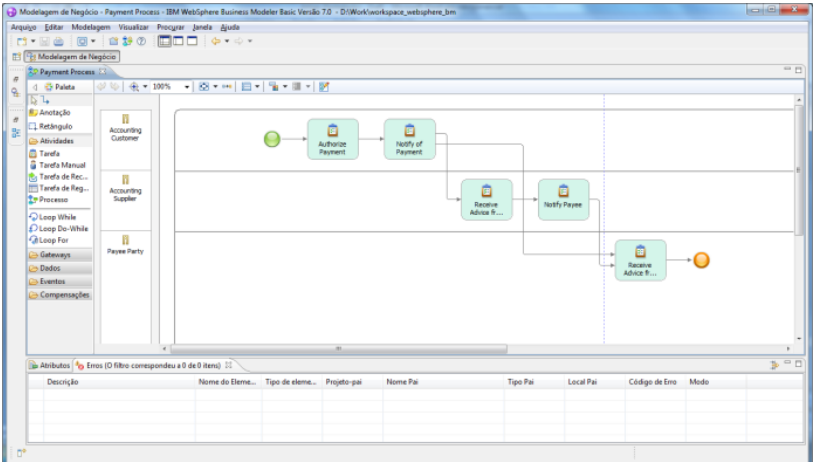


Figura 58 – Processo importado para o editor

Após a importação ser realizada, o processo modelado em BPMN é apresentado na tela. O processo importado é fiel à especificação UBL, contendo todas as atividades e estruturas de fluxo descritas na especificação. As classificações ontológicas do processo e das atividades que o compõe já vêm pré-atribuídas no momento da importação. Assim, o usuário-projetista já tem um processo padronizado que pode ser usado como base para a concepção de suas aplicações SOA.

A fim de poder encontrar serviços adequados para serem vinculados ao seu processo, o projetista define as restrições de qualidade de serviço a cada uma das atividades do processo. Selecionando a atividade, a aba “QoS Constraints” fica disponível. Através do botão “Add QoS Constraint” pode-se adicionar um critério de QoS, conforme apresentado na Figura 59.

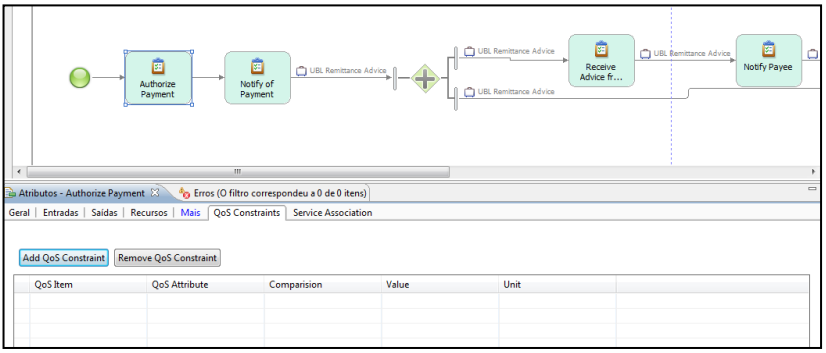


Figura 59 – Tela de Atribuição de QoS

Após selecionar a opção "*Add QoS Constraint*", uma tela contendo a lista de critérios de QoS disponíveis é apresentada (Figura 60).

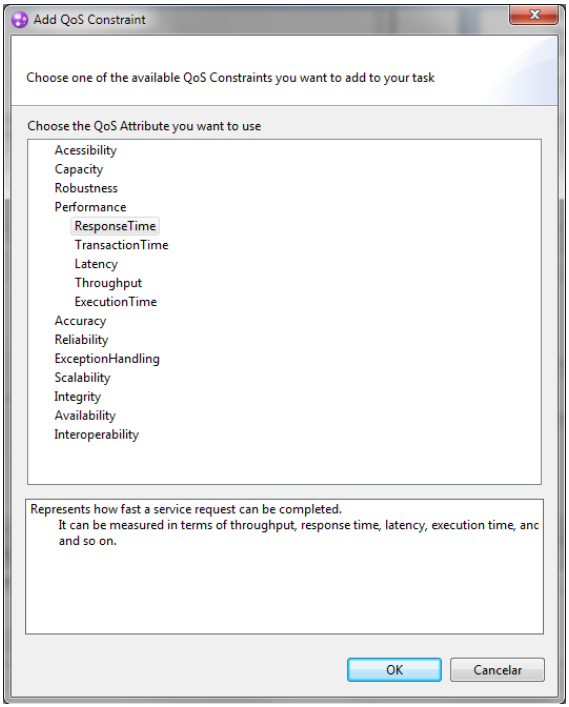


Figura 60 – Tela com a Lista de Critérios de QoS

Essa lista de critérios de QoS é a mesma apresentada na Tabela 9 da seção 3.4.6. Ela é dividida em itens, cada um deles podendo ter um ou mais atributos. Por exemplo, o item *Performace* tem os atributos *ResponseTime*, *TransactionTime*, *Latency*, *Throughput* e *ExecutionTime*. O usuário escolhe um desses itens e define qual valor o mesmo deve ter, como pode ser visto na Figura 61. Além do valor, o usuário também pode definir qual a comparação que deve ser feita em relação a esse, por exemplo, menor, maior, menor ou igual, maior ou igual ou igual. O tipo de comparação depende do item de QoS escolhido. Podem haver casos em que valores menores sejam melhores, como, por exemplo, o tempo de resposta, latência etc. Por outro lado, pode-se desejar valores maiores como por exemplo no item robustez ou disponibilidade.

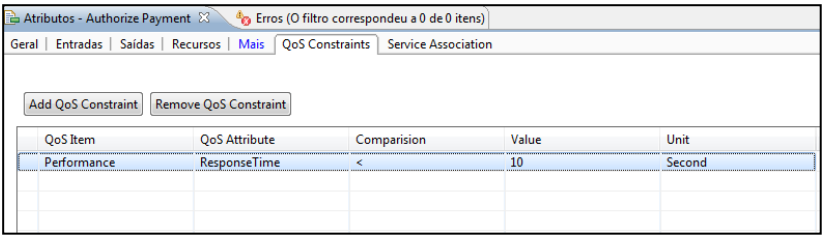


Figura 61 – Definindo o valor do atributo QoS

Depois de ter-se escolhido as restrições de qualidade de serviço, o próximo passo é selecionar a aba "*Service Association*", que permite fazer a busca e a vinculação dos serviços às atividades do processo.

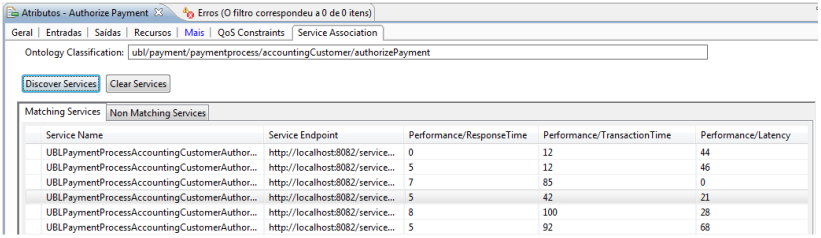


Figura 62 – Tela de Associação de Serviços

O usuário seleciona a opção "*Discover Service*" (Figura 62), que irá invocar o mecanismo de busca, que utiliza como parâmetros a classificação ontológica da tarefa e a lista de critérios de QoS. O retorno

da operação é um conjunto de serviços, divididos entre os que atendem totalmente os critérios de QoS e os que atendem parcialmente. Com base nesse o resultado, o projetista pode decidir se as restrições de qualidade de serviço escolhidas são adequadas. Caso nenhum serviço tenha atendido-as totalmente a essas restrições, a vinculação não será feita. Nesse caso ele tem a opção de relaxar os critérios de QoS a fim de encontrar algum serviço ou correr o risco e tentar fazer essa descoberta em tempo de execução, quando o processo já estiver no Ambiente de Execução.

O procedimento de definição de QoS e associação de serviços deve ser feito para cada atividade do processo. Comumente, o mesmo conjunto de restrições de qualidade de serviço aplica-se a todas as atividades do processo. Pensando nisso, o *plug-in* de integração possui uma opção que permite definir de uma só vez um conjunto de critérios de QoS a todas as atividades do processo e também fazer a busca de serviços. Dessa forma, poupa-se tempo do usuário-projetista.

A Figura 63 apresenta essa funcionalidade. A lista das tarefas do processo é mostrada, juntamente com o a classificação ontológica de cada uma delas, o número de restrições de QoS de cada uma e a quantidade de serviços que possuem o *matching* total e parcial.

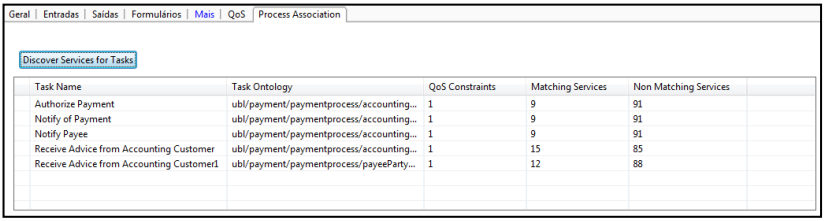


Figura 63 – Descoberta de Serviços para Todas as Atividades

Depois de ter o processo com todos os serviços vinculados e com as restrições do QoS definidas, o usuário-projetista irá exportá-lo para a linguagem de execução BPEL.

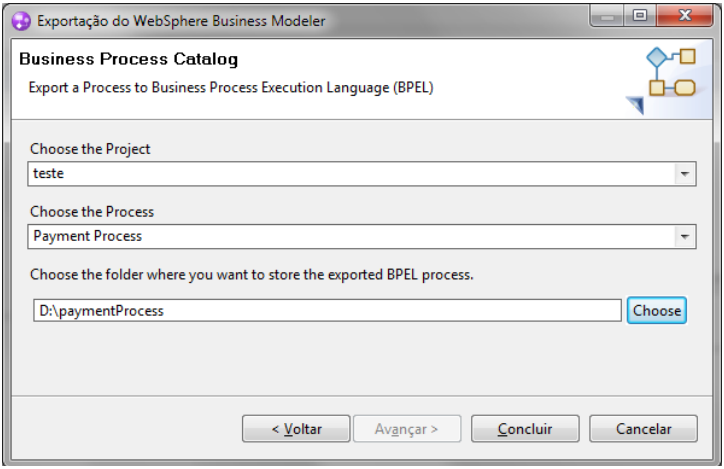


Figura 64 – Exportação BPEL

A Figura 64 apresenta a tela de exportação BPEL. O usuário escolhe qual processo deseja exportar e para qual pasta os arquivos serão gravados. Essa pasta é então copiada para o Ambiente de Execução de Processos, onde o processo ficará disponível para ser executado. Nesse sentido, nota-se que a exportação do processo é totalmente transparente ao usuário, pois o processo exportado já está pronto para ser executado. Não há necessidade de conhecimento técnico por parte do usuário-projetista para se fazer esse procedimento. O uso de padrões, como por exemplo, a especificação UBL, permite que a ligação entre o processo descrito num nível mais abstrato, em BPMN, e a implementação concreta do mesmo, em BPEL, seja feita de maneira automatizada. A classificação ontológica precisa do processo permite que a ligação com os serviços, presentes na federação e encontrados através do ambiente de descoberta dinâmica de serviços, ocorra de maneira natural.

Nesse sentido, a abordagem mostrada no exemplo é mais ágil do que as soluções tradicionais, onde por exemplo é necessário uma equipe técnica para se implementar ou vincular manualmente os serviços junto ao processo ou mesmo para se traduzir o processo em BPMN para BPEL.

Após o processo ter sido exportado , o mesmo fica disponível para execução. Nota-se na Figura 65 o processo "*Payment Process*", pronto para se executado. O Usuário-Executor de Aplicações irá ter

acesso a *interface* do ambiente de execução e, através dela, solicitar a execução do processo. O motor de execução Apache ODE é então acionado, começando a fazer a orquestração dos serviços. Nesse momento, cada um dos serviços que foram previamente vinculados pelo usuário-projetista, na fase de concepção do processo, é invocado.

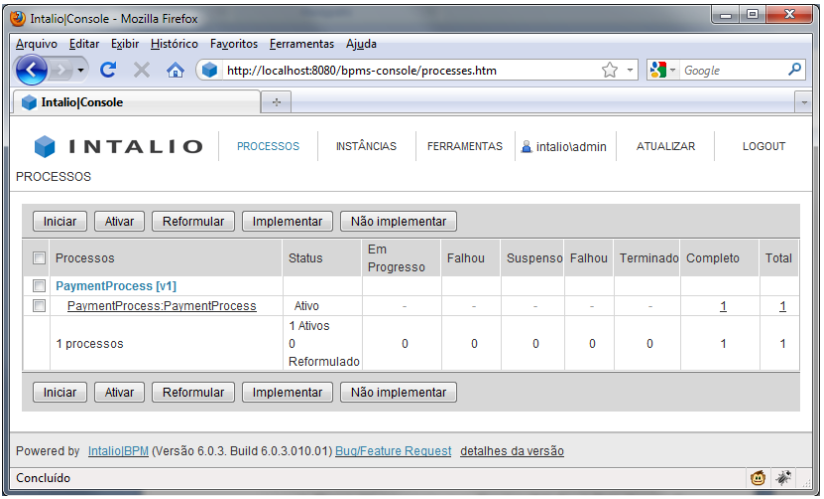


Figura 65 – Ambiente de Execução – Intalio BPMS

Na seção 4.2.3 foi apresentado o mecanismo utilizado para se simular a interação humana, no momento da execução do processo. Portanto, cada serviço no momento da invocação, apresenta uma mensagem de confirmação, que faz o serviço responder ao processo que o invocou. A Figura 66 apresenta um exemplo dessa mensagem de confirmação. No momento em que o usuário seleciona a opção "OK", a resposta do serviço é retornada ao processo.

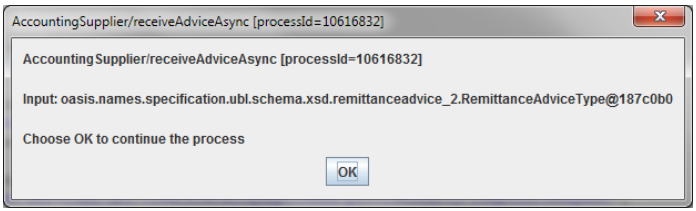


Figura 66 – Mensagem de Confirmação – Serviço UBL

Depois de todos os serviços envolvidos no processo retornarem a resposta, o processo é finalizado. Através da *interface* administrativa do *Intalio BPMS* (Figura 65), pode-se ver quais instâncias do processo foram executadas com sucesso, em quais ocorreram erros etc. A partir desse ponto, o exemplo de funcionamento da arquitetura conceitual proposta é finalizado.

Com base no exemplo apresentado, pode-se ter uma noção dos módulos da arquitetura funcionando conjuntamente, com o intuito de agilizar a integração de BPM com SOA desde o momento em que processo é importado e ajustado pelo usuário-projetista, passando pela sua exportação em BPEL, até quando o mesmo é executado no ambiente de execução de processos.

4.5 Considerações

O objetivo desse capítulo foi apresentar o protótipo computacional, que foi implementado para por em prática as funcionalidades do modelo proposto.

Um dos destaques é o uso intensivo de padrões, tais como *web-services*, BPMN, BPEL, UDDI, SCA etc. O seu uso permite desenvolver aplicações interoperáveis, utilizando as melhores práticas estabelecidas pelo uso desses padrões.

A implementação do *plug-in* de integração com o editor BPM mostrou que é possível tornar transparente para o usuário o uso do catálogo de processos de negócio.

Em respeito ao catálogo, a sua implementação em módulos permitiu separar eficientemente às responsabilidades de cada um deles. A tecnologia SCA permite modelá-los como componentes, de forma que a questão de comunicação entre eles fica abstraída, ficando ao encargo do *middleware*, no caso o *Apache Tuscany*.

O Ambiente de Execução de Processos foi desenvolvido utilizando ferramentas de mercado e consolidadas, como o *Intalio BPMS* e *o Apache ODE*. Isso permitiu verificar a execução dos processos e a eficácia da conversão do processo para BPEL num ambiente realista, passível de ser usado em produção. As implementações dos serviços UBL permitiram testar o mecanismo de invocação e orquestração utilizado pelo ambiente de execução.

O exemplo de funcionamento serviu como uma demonstração sucinta das funcionalidades da arquitetura proposta, abrangendo desde a

fase de concepção da aplicação (onde o usuário-projetista utiliza o ambiente pra facilitar a modelagem do seu processo) até a fase de execução (onde o processo com os serviços já vinculados é executado).

Por último, pode-se concluir que o protótipo desenvolvido serviu como base para avaliação do modelo, que será detalhada no próximo capítulo.

Capítulo 5

Verificação, Avaliação e Análise dos Resultados

O objetivo desse capítulo é apresentar os procedimentos metodológicos que foram efetuados para fazer a verificação e avaliação, bem como a análise geral dos resultados do modelo conceitual proposto no capítulo 3, juntamente como o protótipo computacional implementando. O objetivo é ter as bases suficientes para se responder a pergunta de pesquisa desta dissertação.

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, especializado em testes unitários, e o *SoapUI*, que permite gerar relatórios de conformidade WS-I, 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 conceitual proposto consegue responder a pergunta de pesquisa apresentada no capítulo 1. Por ser um trabalho qualitativo, a estratégia foi a elaboração de um questionário, respondido por especialistas na área de processos de negócio e serviços *web*, de forma a avaliar o trabalho e seus resultados.

Por ser um trabalho exploratório, compreendendo um cenário considerado de certa forma bastante avançado (em termos conceituais e

tecnológicos) não foi possível validar o modelo junto a empresas e seus processos no dia-a-dia.

Por último, uma análise geral a verificação e avaliação é apresentada, com uma discussão crítica a respeito.

5.1 Verificação

A verificação destina-se a mostrar que um software atende a sua especificação (SOMMERVILLE, 2006). 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 de checagem e análise são classicamente empregadas: as inspeções e os testes de software.

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, 2006).

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.

Esta seção apresenta, portanto, 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 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.

5.1.1 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, 2004).

Os testes de unidade foram efetuados nos seguintes módulos do sistema:

- *Plug-in* de Integração com o Catálogo;
- Catálogo de Processos de Negócio;
- Serviços UBL;
- Ambiente de Execução de Aplicações;
- Ambiente de Descoberta Dinâmica de Serviços.

O detalhamento dos testes de unidade está descrito no Apêndice B.

5.1.2 Teste de Integração

Esta seção apresenta os testes de integração realizados no protótipo computacional implementado.

O processo de integração envolve construir um sistema a partir de seus componentes. Portanto, torna-se necessário testar o resultado dessa integração, como forma de garantir que não haja problemas na interação entre eles (SOMMERVILLE, 2006).

Assim, (PRESSMAN, 2004) define testes de integração como uma técnica sistemática para a construção de arquiteturas de software, que ocorre junto com a execução de testes a fim de garantir a perfeita coesão entre os módulos. O objetivo é, após os testes de unidade, garantir que essas unidades possam trabalhar de maneira conjunta, ou seja, que o sistema como um todo funcione corretamente.

Para se efetuar o teste de integração, primeiramente implantou-se os módulos da arquitetura num 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 por instalá-los numa única máquina. Escolheu-se essa abordagem pois facilita a execução dos testes, visto que não é necessário uma infraestrutura tão grande (várias máquinas) para sua execução. Como os componentes se comunicam utilizando serviços *web*, pode-se inferir que o comportamento dos testes será muito parecido, tanto com os componentes implantados localmente, como de maneira distribuída. É importante salientar que essa afirmação se dá 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 da latência e

velocidade da rede. A Tabela 15 apresenta o ambiente utilizado na execução dos testes.

Tabela 15 – Ambiente de Testes de Integração

Local/Ambiente	Laboratório de Tecnologia de Informação e Comunicação do Departamento de Automação e Sistemas da UFSC
Computador	• Processador: Intel Core i3 330M • Microsoft Windows 7 Professional • Memória RAM de 4GB • Conexão com a Internet
Módulos do Protótipo Computacional	
Plug-in de Integração	Executando dentro do **Websphere Business Modeler versão 6.0**
Catálogo de Processos de Negócio	O endereço do *web-service* do catálogo é **http://localhost:8081/CatalogService**
Ambiente de Execução de Processos	Executando o **Intalio BPMS versão 6.0** com o motor de execução **Apache ODE**
UDDI de Serviços da Arquitetura	O endereço do *web-service* do repositório é **http://localhost:8000/juddiv3/services/ inquiry**
Ambiente de Execução de Processos	
Serviço de Descoberta	O endereço do *web-service* da descoberta é **http://localhost:8070/DiscoveryService**
UDDI de Serviços 1	O endereço da UDDI de Serviços 1 é **http://localhost:8079/juddiv3/services/ inquiry**
UDDI de Serviços 2	O endereço da UDDI de Serviços 2 é **http://localhost:8078/juddiv3/services/ inquiry**
UDDI de Serviços 3	O endereço da UDDI de Serviços 3 é **http://localhost:8077/juddiv3/services/ inquiry**
UDDI de Serviços 4	O endereço da UDDI de Serviços 4 é **http://localhost:8076/juddiv3/services/ inquiry**
UDDI de Serviços 5	O endereço da UDDI de Serviços 5 é **http://localhost:8075/juddiv3/services/ inquiry**

O ambiente é composto pelos módulos do protótipo computacional, no caso o *plug-in* de integração, o catálogo de processos de negócio e o ambiente de execução de processos. A UDDI de serviços tem o papel de guardar os endereços dos serviços desses módulos e

permitir a sua localização. O ambiente de descoberta de serviços está instalado localmente, efetuando a busca de serviços em cinco UDDI's. Optou-se por alocar os serviços em cinco UDDI's, pois é um número considerado adequado para a execução em uma única máquina, visto que um número maior poderia sobrecarregá-la. Como o objetivo do teste de integração não é avaliar a eficiência da busca de serviços, o número de repositórios UDDI's mostra-se adequado para o escopo do teste.

O teste de integração deu-se através da utilização do ambiente, sob a perspectiva do usuário. O objetivo foi abranger as fases da concepção da aplicação, onde o usuário-projetista interage com o editor BPM, passando pela busca e associação de serviços, até a exportação do processo para o ambiente de execução, onde este processo pode ser executado pelo usuário-executor de aplicações. Utilizou-se o exemplo de funcionamento apresentado na seção 4.4 para se fazer o teste de integração. Com base na sua execução, pode-se concluir que todos os módulos pertencentes à arquitetura se comportam corretamente, quando trabalhando em conjunto.

5.1.1 Considerações

Esta seção apresentou os testes de unidade e de integração do protótipo computacional implementado.

O teste de unidade foi executado em cada módulo da arquitetura. Seu objetivo foi verificar, de maneira individualizada, se cada um deles atende a especificação e se estão implementados corretamente.

Como não se pode garantir que módulos individualmente testados irão trabalhar corretamente em conjunto (PRESSMAN, 2004), efetuou-se também um teste de integração. Este foi feito sobre a perspectiva do usuário, de forma a reproduzir o seu fluxo de trabalho. Dessa forma conseguiu-se avaliar o funcionamento do sistema como um todo através da interação entre seus módulos. Isso permitiu comprovar que ele está correto também em termos de integração.

5.2 Avaliação

Esta seção apresenta a avaliação do modelo conceitual e do protótipo computacional.

A literatura apresenta vários métodos que podem ser utilizados para a avaliação e validação de trabalhos relacionados com tecnologia

(ZELKOWITZ *et al.*, 2002). Nesse trabalho utilizou-se o método de avaliação *expert panel*, que segundo (ZELKOWITZ *et al.*, 2002) é um método que 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, que segundo (MENEZES *et al.*, 2005) é um conjunto ordenado de perguntas que devem ser respondidas pelos entrevistados. 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 *et al.*, 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, 2002). 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, 2003).

O questionário foi aplicado a um grupo de 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), sendo que no caso de uma delas, três pessoas responderam conjuntamente o mesmo questionário. As seis pessoas restantes são do meio acadêmico, cinco pertencentes ao grupo de pesquisa GSIGMA (o meu do autor deste trabalho) e um ex-integrante deste grupo. O objetivo da aplicação do questionário é usar o seu resultado como forma comprovar que os objetivos gerais e específicos do trabalho foram atingidos e que a hipótese do trabalho (ver seção 1.4) é factível.

A cada um dos entrevistados foi mostrada uma pequena apresentação em slides sobre o trabalho desenvolvido e em seguida apresentou-se o protótipo computacional. Após isto, cada entrevistado utilizou o sistema. O autor deste trabalho ficou ao lado de cada entrevistado, para sanar eventuais dúvidas. Após a utilização do sistema, cada entrevistado respondeu a um questionário (apresentado na Tabela 16).

Tabela 16 – Perguntas utilizadas na avaliação do trabalho

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 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 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?

A seção a seguir apresenta o resultado da aplicação deste questionário, juntamente com a análise dos resultados.

5.2.1 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 *et al.*, 2005). Nesse sentido, o resultado do questionário aplicado é apresentado e em seguida apresenta-se a vinculação deste com o problema, a hipótese e os objetivos da dissertação.

1ª pergunta - Na sua opinião o problema da limitada agilidade na integração entre BPM e SOA é relevante de ser melhorado ou resolvido?

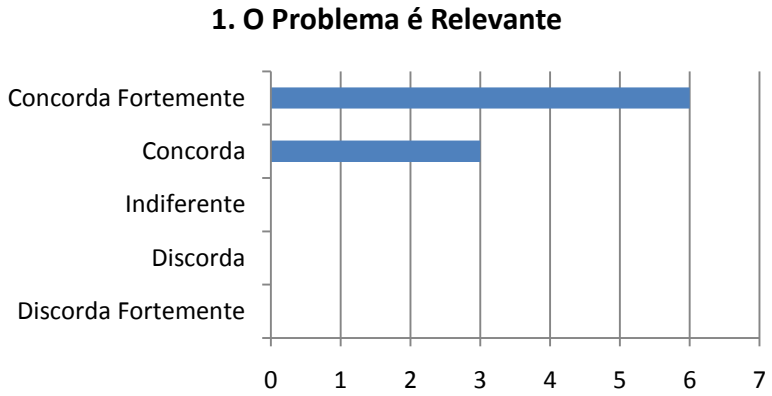


Figura 67 – Gráfico sobre a relevância do problema

Com um nível de concordância variando de concorda fortemente e 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 dissertação, visto que fornece uma fundamentação consistente no que diz respeito à justificativa do problema tratado.

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".

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 nas figuras 68, 69 e 70.

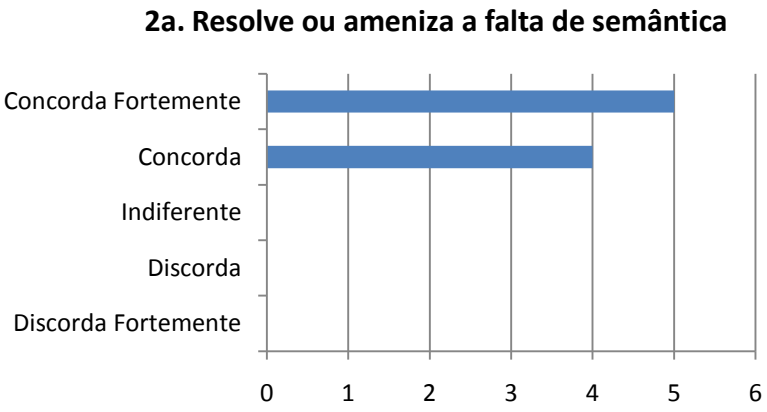


Figura 68 – Gráfico sobre a resolução ou amenização da falta de semântica

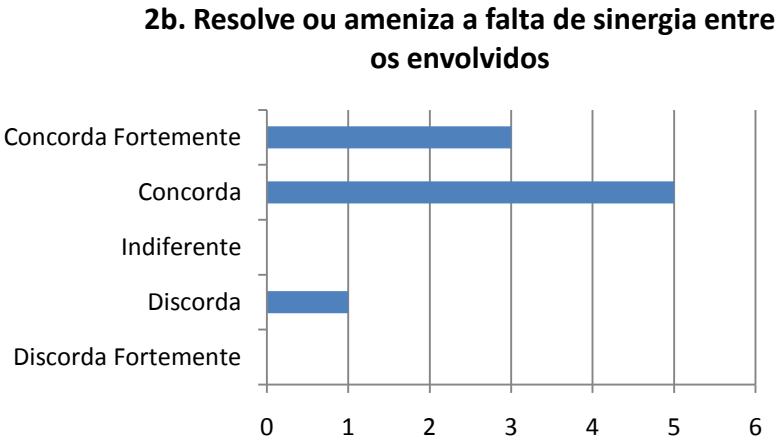


Figura 69 – Gráfico sobre a resolução ou amenização da falta de sinergia entre os envolvidos

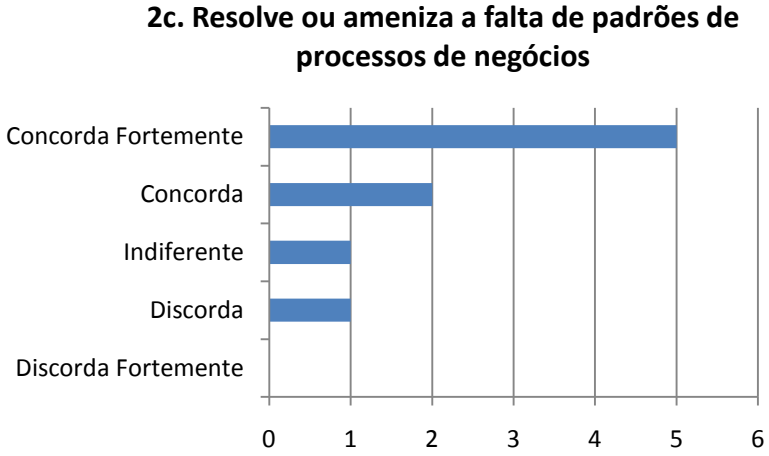


Figura 70 – Gráfico sobre a resolução ou amenização da falta de padrões de processos de negócio

No que diz respeito à questão de 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 catálogo 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 nivel de *software*, a maioria dos entrevistados concorda em maior ou menor grau que a abordagem proposta ajuda a minimizar o problema. A única opinião expressa pelo entrevistado (de uma empresa privada) discordando da questão é baseada na sua opinião de que essa 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 e 2) ressaltam o problema tecnológico entre os níveis BPM e SOA. E segundo, porque o catálogo fornece um ambiente em que a modelagem do processo de negócio e a sua posterior implementação, em SOA, ocorre de uma forma

automatizada, no sentido de tornar 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, visto que 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 empresa, porém elas caminham nesta direção. E isso acontece com a mobilização dos diversos setores empresarias, o que de fato está ocorrendo atualmente.

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".

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?

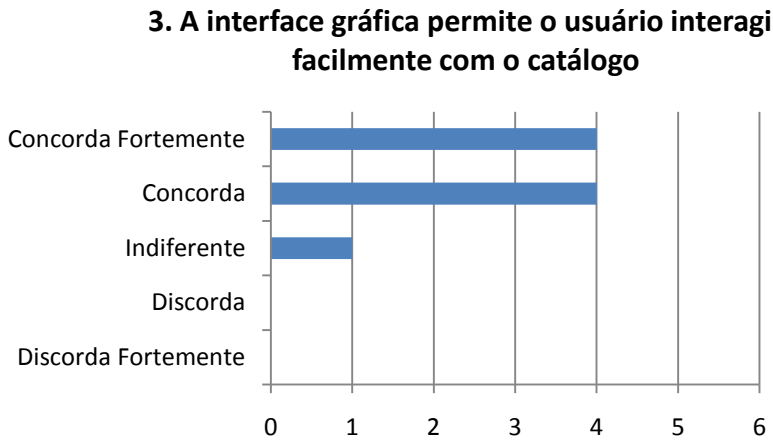


Figura 71 – Gráfico sobre a interface gráfica permitir o usuário interagir facilmente com o catálogo

No que diz respeito ao fato de a interface que o usuário-projetista interage permitir a ele utilizar as funcionalidades do catálogo de uma maneira fácil, 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 consegue interagir com o catálogo de processos de negócio utilizando a *interface* gráfica sem ter dificuldades. Como o questionário foi aplicado a nove pessoas, e boa parte delas concordou que a *interface* permite interagir facilmente com o catálogo, considera-se, de fato, que tal quesito foi cumprido satisfatoriamente.

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".

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?

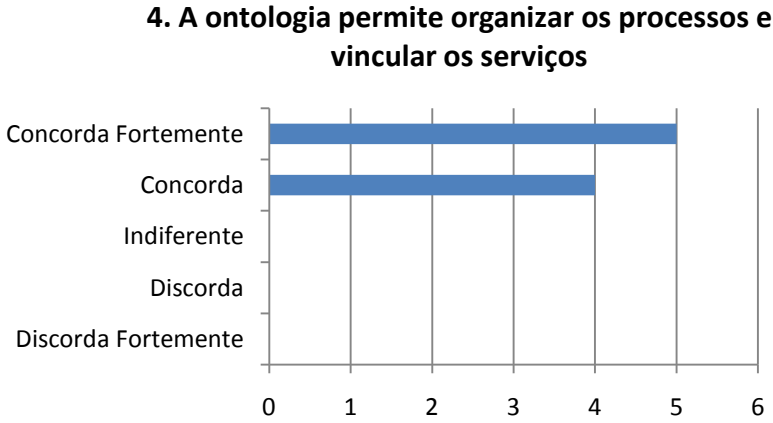


Figura 72 – Gráfico sobre a ontologia permitir organizar os processos e vincular os serviços

No que se refere à ontologia, os entrevistados responderam com unanimidade, em maior ou menor grau, que ela permite organizar os processos de negócio de forma a transparentemente permitir 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.

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".

5ª pergunta - Você concorda que o mecanismo de exportação dos processos permite fazer a conversão da linguagem BPMN para BPEL, simplificando a posterior execução no ambiente de execução de processos?

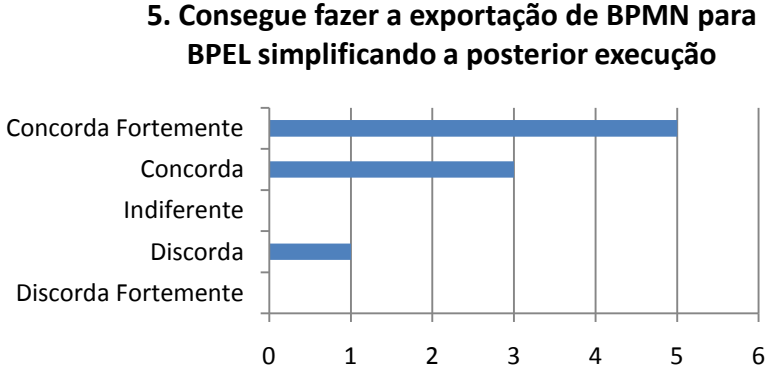


Figura 73 – Gráfico sobre o exportador conseguir fazer a exportação BPMN para BPEL simplificando a execução do processo

O resultado do questionário, no que diz respeito ao mecanismo de exportação de BPMN para 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 BPEL serem proprietárias, dessa forma fazendo parte da propriedade intelectual das empresas que as desenvolveram. Levando em conta que o conversor desenvolvido é baseado em padrões abertos (como BPMN e BPEL) e que o mesmo é disponibilizado gratuitamente, acredita-se que ele contribui para mudar um pouco esse cenário de soluções proprietárias. Além do mais, existem trabalhos acadêmicos, como por exemplo o de (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.

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 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".

6ª pergunta - Você concorda que o Ambiente de Execução de Processos permite ao usuário executar e acompanhar os processos em BPEL?

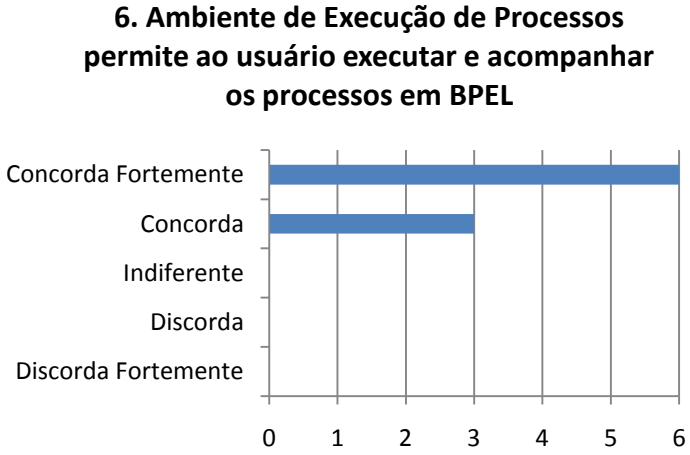


Figura 74 – Gráfico sobre o ambiente de execução de processos permitir executar e acompanha os procesos BPEL

No que diz respeito ao fato de o Ambiente de Execução de Processos permitir ao usuário executar e acompanhar os processos em 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 avaliar em que estado o processo se encontra, por exemplo, em qual atividade o mesmo está parado, quais estão em andamento, quais foram concluídos etc.

Algumas impressões dos avaliadores:

- "O ambiente de execução apresenta módulos de visualização do estado de execução das atividades, permitindo que o gestor de execução do processo possa identificar o estágio de execução do processo 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";

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?

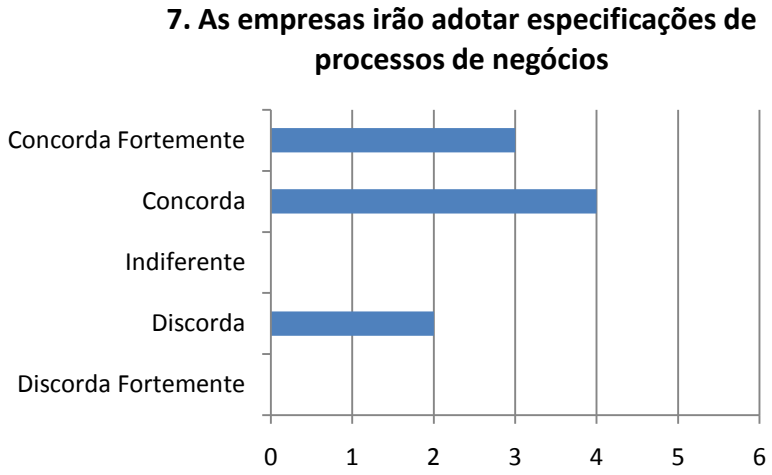


Figura 75 – Gráfico sobre se as empresas irão adotar especificações de processos de negócios

A última pergunta diz respeito ao fato de se as empresas irão adotar especificações de processos de negócio em suas transações internas e com outras empresas. No contexto desse trabalho, essa pergunta é importante, pois serve como um indicador da viabilidade da abordagem proposta neste trabalho. 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 servindo 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 hoje, 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 devidamente. 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 dissertação.

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";

Com base no resultado da aplicação do resultado, pode-se a fazer a ligação deste com o problema, a hipótese de pesquisa e os objetivos geral e específicos do trabalho. A Tabela 17 resgata esses conceitos, presentes do Capítulo 1 do trabalho.

Tabela 17 – Problema, hipótese e objetivos do trabalho

Problema	A integração entre os níveis de BPM e SOA não é tão ágil como poderia ser. Ágil no sentido de fazer uma integração mais transparente, interoperável e eficiente entre os processos de negócio e os serviços *web* que os implementam.
Hipótese de Pesquisa:	A não existência de algo como um catálogo padronizado de processos tira muito das vantagens potenciais de uma ágil integração BPM-SOA. Portanto, um catálogo com processos padronizados e de fácil acesso pelo usuário-projetista BPM pode dar maior qualidade (em termos de aderência ao processo e confiabilidade) e rapidez (em termos temporais) no processo de composição de aplicações SOA e integração BPM-SOA
Objetivo Geral:	Desenvolver um catálogo de processos de negócio, baseado numa instância de Modelo de Referência de Processos de Negócio – a especificação UBL – que auxilie na concepção de aplicações SOA, tornando o processo mais ágil.
Objetivos Específicos	• Concepção de um catálogo de processos de negócio, que será utilizado para a armazenagem de processos no padrão UBL; • Criação de uma ontologia de processos no padrão UBL, de forma a permitir que os processos contidos no catálogo possam ser categorizados segundo esta e permitir a vinculação destes processos com os serviços

	web correspondentes; • Integração do catálogo a um ambiente de edição de processos de negócio; • Criação de 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; • Construção de um ambiente de execução de aplicações, de forma a permitir que os processos exportados possam ser executados e acompanhados; • Integração do catálogo a um ambiente de descoberta dinâmica de serviços;

O problema de pesquisa pode ser comprovado como sendo 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 relevante de ser resolvido ou amenizado.

O objetivo geral da pesquisa pode ser comprovado pelo 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, resolvendo ou amenizando os problemas de:*". Esta pergunta é 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. Os entrevistados concordaram que o catálogo consegue resolver ou amenizar tais problemas, de forma a corroborar com o objetivo geral deste trabalho.

No que se refere aos objetivos específicos, estes são comprovados através da descrição da arquitetura conceitual e da implementação, respectivamente os capítulos 3 e 4, e pelos resultados da aplicação do questionário.

O objetivo específico "c*oncepção de um catálogo de processos de negócio, que será utilizado para a armazenagem de processos no padrão UBL*" é atingido com base no resultado da pergunta 2 e pela oportunidade que os entrevistados tiveram de interagir com o protótipo

computacional em execução, incluindo o catálogo de processos de negócio. Maiores detalhes podem ser vistos nas seções 3.4.2 e 4.2.2.

Já o objetivo específico *"Integração do catálogo a um ambiente de edição de processos de negócio*" é 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 concordam que a *interface* que permite a integração do editor de processos de negócio com o catálogo foi de fato feita, sendo considerada ainda de fácil uso pelos entrevistados. Maiores detalhes são vistos nas seções 3.4.1 e 4.2.1.

O objetivo específico "*Criação de uma ontologia de processos no padrão UBL, de forma a permitir que os processos contidos no catálogo possam ser categorizados segundo esta e permitir a vinculação destes processos com os serviços web correspondentes*" 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 ontologia UBL foi desenvolvida e apresentada em mais detalhes na seção 3.4.2.1

No que diz respeito ao objetivo específico "*Criação de 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 BPEL, simplificando a posterior execução no ambiente de execução de processos?*" permite avaliar se de fato isso foi cumprido. Demonstrando em consenso concordância com essa afirmação, os entrevistados afirmam 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 BPEL. Mais detalhes como isso foi feito são vistos nas seções 3.4.2.3 e 4.2.2.3.

O objetivo específico "*Construção de 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 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. Maiores detalhes de como isso foi feito podem ser vistos nas seções 3.4.5 e 4.2.4.

Por último, no que se refere ao objetivo específico "*Integração do catálogo a um ambiente de descoberta dinâmica de serviços*", este não pode ser diretamente avaliado pelos entrevistados em si. Isso ocorre pois esta integração não é vista através da utilização do protótipo. Quando o usuário seleciona a opção "buscar serviços", o catálogo se conecta transparentemente ao ambiente de descoberta, sem o usuário saber, de fato, que este ambiente existe. Nesse sentido, como forma de comprovar que esse objetivo de fato foi cumprido, recorrem-se as seções 3.4.6 e 4.2.5, que mostram em detalhes de que forma essa integração foi feita.

No capítulo 2 foram apresentados os requisitos utilizados para fazer a avaliação das propostas de catálogos de processos de negócio presentes na literatura. Estes requisitos são resgatados na Tabela 18.

Tabela 18 – Requisitos para a avaliação dos catálogos presentes na literatura

Requisito	Descrição
Processos Reusáveis	Um catálogo de processos deve servir como um facilitador ao reuso de processos já existentes.
Independente de linguagem	Os processos são armazenados num formato independente de linguagem de modelagem.
Flexível	Flexível no sentido que os processos muitas vezes precisam ser customizados para atender certas necessidades e dessa forma suas variantes precisam ser armazenadas no catálogo.
Semântica Associada	Como forma de capturar o contexto semântico e facilitar o processo de descoberta de serviços.
Especificação padronizada	Os processos devem ser descritos de acordo com alguma especificação de processos de negócio.
Integração com SOA	O catálogo deve armazenar os processos de forma que possam ser integrados e executados num ambiente SOA.
Ser um serviço	O catálogo deve ser implementado na forma de um

	serviço, pois fará parte de um ambiente que utiliza a arquitetura SOA.
Integração com Editor BPM	Permitir que o usuário utilize o catálogo de maneira transparente, no mesmo ambiente que usa para modelar os processos.
Código Aberto	O código-fonte do catálogo deve estar disponível, de forma que ele possa ser extensível. Soluções proprietárias não permitem isso, de forma que não são desejadas.

O catálogo de processos de negócio desenvolvido nesta dissertação cumpre todos estes requisitos, pois:

- Os processos que compõem o catálogo são reusáveis, visto que estes fazem parte da especificação UBL, que tem o intuito de promover este reuso. Além disto, o uso da ontologia UBL permite categorizá-los precisamente, o que também facilita a questão do reuso;
- Como foi apresentado na seção 3.4.3, o modelo de dados utilizado pelo catálogo é independente, por exemplo, da linguagem de modelagem utilizada pelo editor BPM *Websphere*. Portanto, esse desacoplamento permite que os processos sejam armazenados no catálogo independentemente da linguagem que foi utilizada para modelá-los;
- O catálogo é flexível, pois permite que além dos processos da especificação UBL, também sejam armazenados instâncias de processos desenvolvidas pelos usuários;
- A semântica associada está representada através do uso da ontologia UBL, descrita na seção 3.4.2.1;
- O catálogo utiliza uma especificação padronizada de processos, a *Universal Business Language* (UBL);
- Na arquitetura SOA desenvolvida, o catálogo é um serviço, que interage com outros, por exemplo, o Ambiente de Descoberta (ver seção 3.4);
- O catálogo é integrado com o editor BPM *Websphere Business Modeler*. Detalhes como isso foi feito podem ser vistos conceitualmente na seção 3.4.1 e maneira concreta na seção 4.2.1.

5.3 Considerações

Este capítulo apresentou a verificação, avaliação e análise dos resultados do modelo conceitual e da implementação do protótipo computacional.

A verificação permitiu comprovar que os módulos da arquitetura estão corretos, isto é, que cumprem as funcionalidades para os quais foram projetados. A corretute é medida tanto em termos individuais, isto é, testes de unidade que permitem verificar isoladamente cada módulo, como em termos de integração, que permitiu verificar o funcionamento conjunto dos módulos. Os testes foram baseados no uso de técnicas conhecidas de engenharia de *software*, como por exemplo, os testes baseados em casos de uso. Em especial, a verificação dos serviços *web* que implementam as funcionalidades dos processos UBL foi feita utilizado o teste WS-I *Basic Profile*, que verifica a interoperabilidade no mecanismo de comunicação destes serviços, questão fundamental quando lidamos com arquiteturas distribuídas que utilizam protocolos interoperáveis.

A avaliação teve como objetivo analisar se o modelo conceitual proposto e o protótipo computacional implementado de fato permitiram atingir os objetivos da pesquisa, comprovando dessa forma a hipótese do trabalho. A técnica utilizada na avaliação foi o *expert panel*, que é baseada no consenso de especialistas, que deram suas opiniões a respeito do trabalho apresentado. Elaborou-se um questionário contendo questões que lidavam diretamente com o problema e os objetivos geral e específicos almejados pelo trabalho. Com a concordância geral dos entrevistados, pode-se, portanto, inferir que de fato a hipótese de que um catálogo de processos de negócio utilizando especificações padronizadas pode agilizar a concepção de aplicações SOA para melhor se integrarem ao nível de BPM.

No que diz respeito à arquitetura conceitual e ao protótipo proposto, a avaliação mostrou que estes são aderentes aos requisitos utilizados para fazer a avaliação dos catálogos, que ocorreu no capítulo 2. A arquitetura é bastante modular e pouco acoplada, visto que seus componentes utilizam padrões interoperáveis, e estando acessíveis através de *interfaces* de serviço bem definidas. Por exemplo, caso fosse necessário mudar o editor BPM (o *Websphere Business Modeler*) para outro, seria necessário somente reescrever o Conector de Integração com Catálogo (ver seção 3.4.1). Os demais módulos da arquitetura são independentes do editor BPM escolhido.

Os resultados apresentados neste capítulo reforçam a importância da proposta, possibilitando proferir conclusões finais sobre o trabalho realizado. Ressalta-se, porém, que a grande complexidade envolvida nos temas abordados pelo trabalho faz com que esta dissertação possua algumas limitações, podendo ser melhorada através de trabalhos futuros. O capítulo 6 descreve estes elementos finais e conclusivos sobre a proposta apresentada.

Capítulo 6

Conclusões

Este trabalho apresentou uma proposta de um catálogo de processos de negócio, baseado na especificação UBL, para a concepção de aplicações BPM&SOA. O principal objetivo foi criar uma arquitetura que permitisse uma integração mais ágil entre os níveis BPM e SOA, isto é, entre o nível em que o processo é modelado pelo analista de negócio até o nível em que o processo é implementado computacionalmente. Nesse contexto, buscou-se utilizar o maior número possível de padrões e especificações de forma a buscar uma maior interoperabilidade e aproveitar as melhores práticas oriundas destes padrões e especificações.

Como forma de reunir subsídios para a elaboração do trabalho, efetuou-se uma revisão bibliográfica dos temas relacionados ao trabalho e, em especial, o estudo do estado da arte relativo aos catálogos de processos de negócio.

No que diz respeito à avaliação destes catálogos, concluiu-se que nenhum deles atende totalmente aos requisitos considerados adequados para uma maior agilidade na integração entre os níveis BPM e SOA. Como apresentado na seção 2.6.6, do que se pode perceber, somente o modelo proposto atende aos requisitos desejados, assim corroborando com a necessidade de se desenvolver um catálogo de processos de negócio. Além disso, o fato de nenhum dos catálogos

pesquisados possuir o código fonte aberto impossibilitou usá-los como base para o catálogo proposto. Por último, a bibliografia pesquisada mostrou que as propostas de catálogos existentes estão focadas principalmente no uso do catálogo como uma forma de reusar processos já existentes, e não como forma de integração entre BPM e SOA. Isto nos leva a inferir sobre algum grau de ineditismo do trabalho, visto que ele reúne o gerenciamento de processos de negócio (BPM) à arquitetura orientada a serviços (SOA), através de um ambiente de descoberta dinâmica de serviços, tendo o catálogo como elo integrador entre eles.

O ambiente do catálogo desenvolvido é composto por quatro grandes módulos: (i) o conector de integração do editor BPM com o catálogo; (ii) o catálogo de processos de negócio; (iii) o ambiente de execução de aplicações e (iv) o ambiente de descoberta dinâmica de serviços. Os três primeiros foram implementados no escopo desta dissertação. O último deles foi desenvolvido por (SOUZA, 2009) e utilizado neste trabalho. O ambiente do catálogo é também uma aplicação SOA, onde seus vários módulos foram desenvolvidos como serviços e podendo ser implantados de forma distribuída.

Dada a abrangência do sistema, inúmeras ferramentas de *software* e padrões tiveram que ser estudados, avaliados e posteriormente integrados. Tal integração, contudo, devido à heterogeneidade das ferramentas, é relativamente complexa, exigindo um bom conhecimento computacional.

Buscou-se trabalhar com ferramentas que de fato são utilizadas num contexto real. Como exemplo, o editor *Websphere* utilizado para modelar os processos e o ambiente de execução de processos *Intalio BPMS*. O principal objetivo disso foi aproximar o modelo desenvolvido e protótipo computacional de um ambiente que de fato seja realista.

Uma das principais contribuições científicas é a ontologia de processos de negócio, inserida num ambiente flexível para se adequar a padrões de processos. Na revisão bibliográfica feita não se encontrou a especificação UBL em forma computacionalmente tratável e semanticamente entendível, o que tornou a ontologia UBL desenvolvida de grande utilidade, pois pode ser reusada em outras aplicações.

A verificação e avaliação consistiram, respectivamente, no uso de técnicas de engenharia de *software* como forma de comprovar que os módulos da arquitetura estão funcionando adequadamente, inclusive em termos de integração e no uso da técnica *expert panel*, que permitiu que o trabalho fosse avaliado através do consenso de especialistas em áreas relacionadas com este trabalho. Estes procedimentos foram capazes de

responder à pergunta de pesquisa: *que o uso de um catálogo com processos de negócio padronizados pode agilizar o projeto de soluções SOA para melhor se integrarem ao nível BPM.* A avaliação também permitiu comprovar que o catálogo, de fato, cumpre os requisitos necessário a uma ágil integração BPM&SOA, apresentados no capítulo 2.

Por ser um trabalho qualitativo e exploratório, a avalição e validação é complicada. Isso ocorre, por exemplo, pela dificuldade de se aplicar o modelo proposto num ambiente real, onde o mesmo possa ser usado em sua plenitude. Alia-se a isso a dificuldade em se obter um número suficientemente grande de pessoas, principalmente de empresas, que teriam como avaliar o catálogo neste contexto real. Estas próprias pessoas são muitas vezes demasiadamente conservadores, visto que elas estão constantemente preocupadas com o dia-a-dia da empresa, não acompanhando dessa forma as últimas tendências e os trabalhos mais recentes da literatura. Nesse sentido, e como mostrado na avaliação, estas pessoas de empresas se mostram receosas quando em contato com um ambiente como o proposto neste trabalho.

Nesse sentido, a abordagem proposta traz uma série de impactos, caso fosse implementada num cenário real, em plenitude. Sabe-se que a ação de uma empresa adotar padrões para melhor integrar os níveis de negócio com o de TI significa grandes mudanças. Portanto, uma adoção em larga escala disso, seguramente não é uma tarefa trivial e rápida.

O foco desta dissertação, todavia, não é o de explorar essa complexidade toda, mas sim avaliar em que medida um catálogo pode agilizar a integração dos níveis BPM e SOA.

Ressalta-se que a história da tecnologia da informação vem sempre lidando com quebras constantes de paradigmas. Há vinte anos, por exemplo, não se imaginava que a programação orientada a objetos teria a popularidade que tem hoje. Acredita-se, portanto, que esse ambiente, apesar de no momento parecer distante da realiada, tem o potencial de se tornar algo concreto num futuro próximo.

6.1 Contribuições do Trabalho

Dentro dos objetivos específicos propostos na seção 1.6 e das atividades executadas para atingi-los, essa dissertação contribui nos seguintes aspectos:

- A principal contribuição deste trabalho foi um modelo de arquitetura que dá suporte à concepção de aplicações, desde a fase em que esta aplicação é modelada (através de processos, numa ferramenta de modelagem) até a fase em que ela é executada (no ambiente de execução). Esta arquitetura é fortemente baseada em padrões, tais como serviços *web*, BPMN, BPEL etc., e permite que o usuário conceba suas aplicações SOA sem se preocupar com questões técnicas relativas, por exemplo, a forma como os serviços são implementados, onde eles estão localizados etc.. A abordagem proposta é ágil no sentido de fazer uma integração mais transparente, interoperável e eficiente entre os níveis de BPM e SOA, seja por automatizar o máximo possível o processo de concepção da aplicação, seja por usar padrões reconhecidos e aceitos mundialmente;
- Criação de uma ontologia de processos da especificação UBL, que permite categorizar os processos de negócio de uma maneira suficientemente precisa de forma a permitir fazer a vinculação desses processos com os serviços *web* que os implementam (que estão presentes num ente chamado federação, que agrega diversos provedores de serviços presentes na "nuvem");
- Desenvolvimento de um protótipo computacional que implementa a abordagem proposta e que utiliza ferramentas de mercado, como o *Websphere Business Modeler* e o *Intalio BPMS*. O protótipo comprova a viabilidade técnica da proposta e pode ser usado como base para futuras extensões e melhorias;

Em relação às abordagens tradicionais, a desse é diferente, visto que não é necessário fazer a vinculação manual dos serviços, isto é, não se precisa saber a localização física dos serviços, nem implementá-los. Isso é viável, pois a ontologia UBL permite que a especificação dos serviços seja claramente definida e, dessa forma, permite que diferentes provedores de serviços possam fornecer implementações funcionalmente equivalentes desses serviços.

Ressalta-se que, embora esse trabalho tenha utilizado a especificação UBL, nada impediria que se usassem outras especificações de processos de negócio. O catálogo é preparado para permitir o armazenamento de várias especificações. Se, por exemplo,

uma empresa desejar ter sua própria especificação de processos, esta poderia ser armazenada no catálogo proposto. Nesse sentido, por exemplo, poderiam existir diversos catálogos, cada um deles com determinadas especificações padronizadas, como por exemplo, relativas ao setor eletrônico, automobilístico, de saúde etc. A empresa poderia fazer uso destes catálogos, que conteriam processos adequados a determinados contextos exigidos pela empresa.

Num contexto mais amplo, esse trabalho contribui para um maior potencial de colaboração entre as empresas, uma vez que elas podem se utilizar de serviços umas das outras, nesse caso atuando como provedores de serviços, independentemente de quais plataformas e tecnologias foram utilizadas nas suas implementações.

Em uma escala maior, este trabalho colabora para uma maior integração e interoperabilidade entre sistemas e empresas, dentro de um cenário onde se prevê a criação e existência de federações de serviços (RABELO *et al.*, 2007). Nesta, serviços de vários provedores/empresas são logicamente agrupados de forma a que parceiros possam utilizar serviços uns dos outros sem problemas maiores de interoperabilidade, dando aos consumidores/aplicações clientes melhores condições de agilidade, flexibilidade para suportar mudanças nos processos e, por fim, economia.

Várias sugestões e pontos de vista puderam ser assimilados através da concepção desse trabalho e dos comentários feitos pelos entrevistados. Nesse sentido, a realização do trabalho foi capaz de incrementar substancialmente, no autor, a capacidade de discernimento e análise crítica de circunstâncias diversas, dando-lhe um aumento intelectual significativo, que lhe oferece condições de seguir os caminhos trilhados por pesquisadores de notória sabedoria, ampliando e criando novos conhecimentos científicos.

6.2 Limitações da Proposta

Este seção apresenta algumas das limitações da abordagem proposta, tanto em termos conceituais como em termos técnicos (relativos à implementação do protótipo).

A abordagem proposta se baseia no pressuposto de existirem especificações de processos de negócio que possuam processos e documentos perfeitamente definidos e passíveis de serem executados por computadores. Nesse sentido, utilizou-se a especificação UBL. As

empresas, portanto, devem ter processos alinhados com essa especificação, o que nem sempre acontece na prática. Nesse sentido, a aplicabilidade do catálogo fica restrita a esse grupo de empresas, a não ser, que a especificação seja customizada ou refeita para atender as necessidades específicas de cada empresa. Além das empresas, pressupõem-se que existem provedores de serviços que implementam serviços UBL (seguindo esta especificação). Isso também só é possível se a especificação for reconhecida e aceita por um número abrangente de empresas e provedores de serviços.

Com relação aos processos descritos pela especificação, eles são, na sua grande maioria, operacionais. Fica a dúvida se chegará um ponto em que processos considerados estratégicos pelas empresas poderão fazer parte de uma especificação padronizada e inter-organizacional. Caso isso não ocorra, provavelmente, a empresa terá um catálogo próprio de processos privados, tendo sua própria federação de serviços que serão vinculados a esses processos estratégicos. Nesse caso, a abordagem proposta perde o foco inter-organizacional, visto que a especificação fica limitada ao escopo de uma única empresa.

No que diz respeito à exportação do processo, o protótipo desenvolvido não consegue lidar com todas as estruturas presentes nas linguagens BPMN e BPEL. No primeiro caso, não se mapeou todas as estruturas utilizadas pelo editor BPM *Websphere Business Modeler*, de forma que alguns processos não podem ser armazenados e recuperados do catálogo de maneira fiel ao original. No segundo caso, o conversor para a linguagem de execução BPEL não consegue fazer a conversão perfeita de qualquer processo descrito em BPMN. Nesse sentido, o conversor desenvolvido neste trabalho é limitado em comparação com outras soluções presentes no mercado. A experiência mostra que é muito difícil fazer um conversor totalmente eficaz. A linguagem BPMN (utilizada na notação dos processos) e a BPEL (utilizada para descrever sua execução) não são equivalentes a ponto de se poder converter qualquer estrutura de uma linguagem em outra. Dessa forma, portanto, sendo necessário fazer algumas transformações em estruturas equivalentes. Alguns trabalhos como os de (WEIDLICH *et al.*, 2008), (OUYANG *et al.*, 2006) e (WHITE, 2005) tratam sobre esse tema e relatam a dificuldade dessa conversão. Essa limitação, porém, não representa um entrave aos objetivos desse trabalho, visto que não é o seu foco criar um conversor perfeitamente compatível com a especificação BPMN e BPEL, e sim demonstrar que uma conversão entre esses formatos é possível, mesmo que com algumas limitações. Assim, o

conversor consegue lidar somente com as estruturas suficientes para exportar os processos descritos na seção 4.2.2.3 do trabalho. Apesar de que tais estruturas contemplam processos bastante complexos e assim, congregando a maior parte da especificação, esse é um ponto que pode ser melhorado, no sentido de fazer um conversor mais expressivo, que consiga lidar com toda a especificação.

Por último, questões de desempenho e escalabilidade não são tratadas por esse trabalho, e por consequência, pelo protótipo implementado. Nesse sentido, não houve uma preocupação com questões como, por exemplo: (i) tempo de resposta, no sentido do tempo de processamento requerido para efetuar as operações no catálogo; (ii) latência, o tempo necessário para a mensagem trafegar na infraestrutura de rede; (iii) escalabilidade, como a solução se comporta com um número elevado de serviços na federação, ou com um uso intenso por um grande número de usuários etc.

6.3 Trabalhos Futuros Sugeridos

Dada a abrangência relativa ao tema de pesquisa desse trabalho e também a diversidade de disciplinas envolvidas, pode-se sugerir algumas propostas futuras, tais como:

- Aprimorar o mecanismo de exportação do processo para BPEL, incorporando alguma ferramenta especializada a esta função, ou mesmo melhorando o algoritmo de exportação;
- Avaliar em que medida a arquitetura proposta é vista nos diferentes níveis organizacionais da empresa: gerencial, tático e operacional. E também de que forma o trabalho desenvolvido pode ajudar o CEO da empresa, o gerente de TI, o gerente de processos etc. ou seja, diferentes tipos de atores;
- Realizar a retroalimentação do sistema, no sentido de saber se a execução do processo foi realizada da maneira esperada, por exemplo, identificando gargalos em suas atividades ou mesmo verificando se os serviços web estão cumprindo com seus requisitos funcionais e não funcionais. Tal fase diz respeito ao diagnóstico no ciclo de vida do BPM. Isso é de fundamental importância no sentido de buscar a melhoria contínua no gerenciamento de processos de negócio;

- Estabelecer outros critérios para a busca de serviços web que implementem as atividades do processos. Nesse sentido, pode-se levar em conta o modelo de negócio oferecido pelo serviço (por exemplo, SaaS – *Software as a Service*) e a afinidade que o mesmo possui com a empresa que irá utilizá-lo, por exemplo em níveis de reputação, ou o fato de o provedor de serviços já ser parceiro da empresa etc.

Apêndices

Apêndice A

Ontologia UBL

Esta seção apresenta a ontologia UBL utilizada para classificar os processos de negócio que foram modelados nesse trabalho. A ontologia abrange os seguintes processos: Processo de Criação de Catálogo (*Create Catalogue Process*), Processo de Ordem (*Ordering Process*) e Processo de Pagamento (*Payment Process*).

Categoria	Nome do Processo	Participante Executor	Atividade
Sourcing/Catalogue Provision	Catalogue Provision	Receiver Party	Request Catalogue
Ontologia: **ubl/sourcing/catalogueprovision/createcatalogueprocess/receiverParty/requestCatalogue**			
Sourcing/Catalogue Provision	Catalogue Provision	Receiver Party	Receive Rejection
Ontologia: **ubl/sourcing/catalogueprovision/createcatalogueprocess/receiverParty/receiveRejection**			
Sourcing/Catalogue Provision	Catalogue Provision	Receiver Party	Receive Catalogue
Ontologia: **ubl/sourcing/catalogueprovision/createcatalogueprocess/receiverParty/receiveCatalogue**			
Sourcing/Catalogue Provision	Catalogue Provision	Receiver Party	Review Catalogue Content
Ontologia: **ubl/sourcing/catalogueprovision/createcatalogueprocess/receiverParty/reviewCatalogueContent**			
Sourcing/Catalogue Provision	Catalogue Provision	Receiver Party	Acknowledge Acceptance
Ontologia: **ubl/sourcing/catalogueprovision/createcatalogueprocess/receiverParty/acknowledgeAcceptance**			
Sourcing/Catalogue Provision	Catalogue Provision	Receiver Party	Accept Catalogue
Ontologia: **ubl/sourcing/catalogueprovision/createcatalogueprocess/receiverParty/acceptCatalogue**			
Sourcing/Catalogue Provision	Catalogue Provision	Receiver Party	Query Catalogue Content
Ontologia: **ubl/sourcing/catalogueprovision/createcatalogueprocess/receiverParty/queryCatalogueContent**			

Sourcing/Catalogue Provision	Catalogue Provision	Provider Party	Respond to Request
Ontologia: **ubl/sourcing/catalogueprovision/createcatalogueprocess/providerParty/respondToRequest**			
Sourcing/Catalogue Provision	Catalogue Provision	Provider Party	Process Catalogue Request
Ontologia: **ubl/sourcing/catalogueprovision/createcatalogueprocess/providerParty/processCatalogueRequest**			

Categoria	Nome do Processo	Participante Executor	Atividade
Sourcing/Catalogue Provision	Catalogue Provision	Provider Party	Send Rejection
Ontologia: **ubl/sourcing/catalogueprovision/createcatalogueprocess/providerParty/sendRejection**			
Sourcing/Catalogue Provision	Catalogue Provision	Provider Party	Send Acceptance Response
Ontologia: **ubl/sourcing/catalogueprovision/createcatalogueprocess/providerParty/sendAcceptanceResponse**			
Sourcing/Catalogue Provision	Catalogue Provision	Provider Party	Prepare Catalogue Information
Ontologia: **ubl/sourcing/catalogueprovision/createcatalogueprocess/providerParty/prepareCatalogueInformation**			
Sourcing/Catalogue Provision	Catalogue Provision	Provider Party	Produce Catalogue
Ontologia: **ubl/sourcing/catalogueprovision/createcatalogueprocess/providerParty/produceCatalogue**			
Sourcing/Catalogue Provision	Catalogue Provision	Provider Party	Distribute Catalogue
Ontologia: **ubl/sourcing/catalogueprovision/createcatalogueprocess/providerParty/distributeCatalogue**			
Sourcing/Catalogue Provision	Catalogue Provision	Provider Party	Receive Acknowledge Acceptance
Ontologia: **ubl/sourcing/catalogueprovision/createcatalogueprocess/providerParty/receiveAcknowledgeAcceptance**			
Sourcing/Catalogue Provision	Catalogue Provision	Provider Party	Decide on Action
Ontologia: **ubl/sourcing/catalogueprovision/createcatalogueprocess/providerParty/decideOnAction**			
Sourcing/Catalogue Provision	Catalogue Provision	Provider Party	Cancel Transaction
Ontologia: **ubl/sourcing/catalogueprovision/createcatalogueprocess/providerParty/cancelTransaction**			
Sourcing/Catalogue Provision	Catalogue Provision	Provider Party	Revise Content
Ontologia: **ubl/sourcing/catalogueprovision/createcatalogueprocess/providerParty/reviseContent**			
Ordering	Ordering Process	Seller Party	Receive Order
Ontologia: **ubl/ordering/orderingprocess/sellerParty/receiveOrder**			
Ordering	Ordering Process	Seller Party	Process Order
Ontologia: **ubl/ordering/orderingprocess/sellerParty/processOrder**			
Ordering	Ordering Process	Seller Party	Accept Order
Ontologia: **ubl/ordering/orderingprocess/sellerParty/acceptOrder**			
Ordering	Ordering Process	Seller Party	Reject Order
Ontologia: **ubl/ordering/orderingprocess/sellerParty/rejectOrder**			
Ordering	Ordering Process	Seller Party	Add Detail
Ontologia: **ubl/ordering/orderingprocess/sellerParty/addDetail**			
Ordering	Ordering Process	Seller Party	Cancel Order
Ontologia: **ubl/ordering/orderingprocess/sellerParty/cancelOrder**			
Ordering	Ordering Process	Seller Party	Change Order
Ontologia: **ubl/ordering/orderingprocess/sellerParty/changeOrder**			
Ordering	Ordering Process	Seller Party	Process Order Change
Ontologia: **ubl/ordering/orderingprocess/sellerParty/processOrderChange**			
Ordering	Ordering Process	Buyer Party	Place Order

Categoria	Nome do Processo	Participante Executor	Atividade
Ontologia: **ubl/ordering/orderingprocess/buyerParty/placeOrder**			
Ordering	Ordering Process	Buyer Party	Receive Response
Ontologia: **ubl/ordering/orderingprocess/buyerParty/receiveResponse**			
Ordering	Ordering Process	Buyer Party	Accept Order
Ontologia: **ubl/ordering/orderingprocess/buyerParty/acceptOrder**			
Ordering	Ordering Process	Buyer Party	Change Order
Ontologia: **ubl/ordering/orderingprocess/buyerParty/changeOrder**			
Ordering	Ordering Process	Buyer Party	Cancel Order
Ontologia: **ubl/ordering/orderingprocess/buyerParty/cancelOrder**			
Payment	Payment Process	Accounting Customer	Authorize Payment
Ontologia: **ubl/payment/paymentprocess/accountingCustomer/authorizePayment**			
Payment	Payment Process	Accounting Customer	Notify of Payment
Ontologia: **ubl/payment/paymentprocess/accountingCustomer/notifyOfPayment**			
Payment	Payment Process	Accounting Supplier	Notify Payee
Ontologia: **ubl/payment/paymentprocess/accountingSupplier/notifyPayee**			
Payment	Payment Process	Accounting Supplier	Receive Advice
Ontologia: **ubl/payment/paymentprocess/accountingSupplier/receiveAdvice**			
Payment	Payment Process	Payee Party	Receive Advice
Ontologia: **ubl/payment/paymentprocess/payeeParty/receiveAdvice**			

Apêndice B

Testes de Unidade

Esta seção apresenta o detalhamento dos testes de unidade efetuados em cada módulo da arquitetura implementado pelo protótipo computacional.

Plug-in de Integração com o Catálogo

O teste do Plug-In de Integração com o Catálogo foi feito tomando como base os casos de uso relacionados com esse módulo. O objetivo é avaliar se esse módulo, implementado pelo protótipo, atende a especificação.

A abordagem utilizada para esses testes é a *baseada em cenários*, que segundo (PRESSMAN, 2004) 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 *Plug-in* 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, 2008), 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 3.3, que descreve os casos de uso do modelo proposto, criou-se os seguintes casos de teste, com o intuito de testar o *Plug-in* de Integração com o Catálogo:

- 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 BPEL.

O ambiente em que os testes foram executados está descrito na Tabela 19.

Tabela 19 – Ambiente de Testes – Plug-In de Integração

Local/Ambiente	Laboratório de Tecnologia de Informação e Comunicação do Departamento de Automação e Sistemas da UFSC
Computador	• Processador: Intel Core i3 330M • Microsoft Windows 7 Professional • Memória RAM de 4GB • Conexão com a Internet
Observações	O Catálogo de Processos de Negócio, acessado pelo *plug-in*, encontra-se na mesma máquina deste.

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. A configuração do computador utilizado está descrita na Tabela 19.

Como o *plug-in* utiliza os serviços do catálogo para poder funcionar, é necessário que uma instância deste esteja em funcionamento, conforme salientado no item “Observações” da Tabela 19. Ressalta-se que o objetivo desse teste é garantir que o *plug-in* de

integração esteja funcionando. Apesar deste *plug-in* requerer o Catálogo de Processos de Negócio para poder funcionar, este não é o enfoque do teste. Na próxima seção mostra-se o teste específico para o Catálogo de Processos de Negócio.

A seguir, apresenta-se em detalhes os casos de teste desenvolvidos, juntamente com os resultados obtidos.

Tabela 20 – Caso de Teste – Importação do Processo do Catálogo

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 *plug-in*.
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.

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 processos BPM 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.

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.

Tabela 21 – Caso de Teste – Exportação do Processo para o Catálogo

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.

A análise do segundo acaso de teste aponta que o resultado da exportação é compatível com o esperado.

O resultado do teste mostrou que somente as principais estruturas da BPMN foram reconhecidas, de forma que algumas estruturas que o editor *Websphere* utiliza não são suportadas na exportação. As seguintes estruturas são suportadas na exportação:

- 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.

Tabela 22 – Caso de Teste – Definição de QoS às atividades do processo

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 do Processo** – 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).

A análise do terceiro caso de teste 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 do processo.

Tabela 23 – Caso de Teste – Invocar a Descoberta de Serviços e vinculá-los às tarefas

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 do Processo** – 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.

A análise do quarto caso de teste 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.

Tabela 24 – Caso de Teste – Exportar o Processo para a Linguagem BPEL

Caso de Teste 5 – Exportar o Processo para a Linguagem BPEL	
Descrição	Esse caso de teste verifica se o procedimento de exportação do processo para a linguagem 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 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 BPEL da exportação serão gravados.

Resultado Esperado	• Após a invocação pelo usuário, a tela de exportação de processos para 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 BPEL será salvo; • O sistema deve converter o processo escolhido para o formato BPEL e armazená-lo na pasta escolhida pelo usuário; • O processo deve poder ser executado no Ambiente de Execução de Processos.

A análise do quinto caso de teste aponta que o resultado da exportação do processo para BPEL é compatível, na medida do possível, com o esperado. Os seguintes processos puderam ser exportados e executados:

- Processo de Criação de Catálogo (*Create Catalogue Process*);
- Processo de Ordem (*Ordering Process*)
- Processo de Pagamento (*Payment Process*)

No que tange a compatibilidade com as estruturas BPMN, o exportador consegue lidar aquelas descritas na seção 4.2.2.3. Processos que não estão em conformidade com essas estruturas não podem ser exportados para BPEL, pois geram arquivos inválidos.

Com base na execução dos casos de testes acima, pode-se concluir que o *Plug-in* de Integração com o Catálogo, implementado pelo protótipo, atende aos requisitos estabelecidos no modelo conceitual.

Catálogo de Processos de Negócio

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, 2006).

O teste desenvolvido é do tipo caixa preta, que segundo (O'DOCHERTY, 2005) 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. Dessa forma, atesta-se a corretude somente analisando as entradas da operação e as saídas desejadas.

De acordo com essa definição, utilizou-se a *interface* CatalogService como base para os testes. O código completo do teste desenvolvido está disponível no Apêndice F.

O cenário utilizado na execução desse teste está descrito na Tabela 25.

Tabela 25 – Ambiente de Testes - Catálogo

Local/Ambiente	Laboratório de Tecnologia de Informação e Comunicação do Departamento de Automação e Sistemas da UFSC
Computador	• Processador: Intel Core i3 330M • Microsoft Windows 7 Professional • Memória RAM de 4GB • Conexão com a Internet
Observações	Os seguintes componentes do catálogo estão executando no mesmo computador: • Serviços de Persistência; • Serviços de Gerência de Especificações; • Serviços de Exportação BPEL. O Ambiente de Descoberta Dinâmica de Serviços está executando na mesma máquina, com cinco repositórios locais de serviços

O cenário é muito semelhante ao apresentado no teste do *Plug-in* de Integração com o Catálogo. Todos os componentes que compõe o Catálogo de Processos de Negócio estão executando na mesma máquina.

O teste foi executado dentro da IDE *Eclipse*, utilizando o *plug-in* de integração com o *JUnit*, que permite a execução desses testes de unidade. Todas as operações disponíveis na interface *CatalogService* foram testadas.

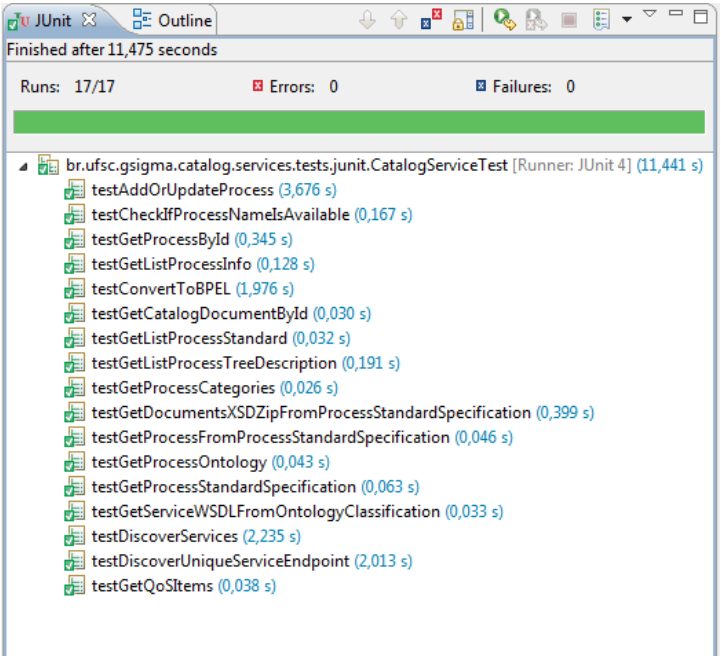


Figura 76 – Resultado do teste JUnit – Catálogo de Processos de Negócio

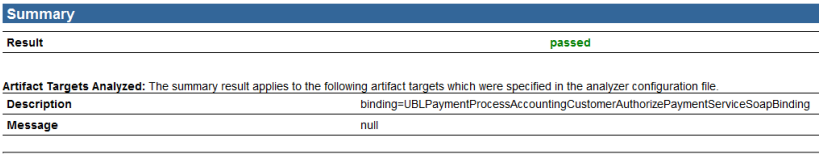
A Figura 76 apresenta o resultado do conjunto de testes. Pode-se ver que todos os testes realizados nas operações do serviço foram concluídos com êxito. O código completo do teste se encontra disponível no Apêndice A do trabalho.

A execução do teste mostra que o serviço Catálogo de Processos de Negócio está funcionando da maneira adequada. Como descrito na seção 4.2.2, o catálogo reúne as operações dos componentes que fazem parte dele. A nível de implementação, significa que o catálogo implementa a *interface* de todos os seus componentes, encaminhando as requisições para o componente mais adequado. Dessa forma, o teste executado serviu também para testar os componentes do catálogo, visto que eles acabam sendo acessados quando o teste é executado. Assim, conclui-se que o Catálogo de Processos de Negócio e os seus componentes estão verificados e corretos.

Serviços UBL

Para se testar os serviços UBL implementados, 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.



Summary	
Result	**passed**

Artifact Targets Analyzed: The summary result applies to the following artifact targets which were specified in the analyzer configuration file.

Description	binding=UBLPaymentProcessAccountingCustomerAuthorizePaymentServiceSoapBinding
Message	null

Figura 77 – Resultado do teste WS-I Basic Profile

A Figura 77 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. A Tabela 26 apresenta todos os serviços UBL que foram testados:

Tabela 26 – Testes Realizados nos Serviços UBL

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
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

Nome do Serviço	Resultado
AccountingCustomer/NotifyOfPayment	Aprovado
AccountingSupplier/NotifyPayee	Aprovado
AccountingSupplier/ReceiveAdvice	Aprovado
PayeeParty/ReceiveAdvice	Aprovado

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.

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 *plug-in* de integração. Essa escolha deve-se ao fato de, assim como o *plug-in*, 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.

De acordo com a seção 3.3, onde se descreve os casos de uso do sistema, os seguintes serão utilizados como base para os casos de teste:

- Acessar a Lista dos Processos Executáveis Disponíveis;
- Executar Processos;
- Monitorar o Andamento dos Processos.

O ambiente onde os testes foram executados está descrito na Tabela 27:

Tabela 27 – Ambiente de Testes – Execução de Aplicações

Local/Ambiente	Laboratório de Tecnologia de Informação e Comunicação do Departamento de Automação e Sistemas da UFSC
Computador	• Processador: Intel Core i3 330M • Microsoft Windows 7 Professional • Memória RAM de 4GB • 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 Dinâmica de Serviços está executando na mesma máquina, com cinco repositórios locais de serviços

O cenário de testes é executado num único computador. A ferramenta Intalio BPMS é responsável pelo gerenciamento e execução dos processos. O Ambiente de Descoberta Dinâmica de Serviços pode ser invocado no momento da execução do processo, caso não existam serviços previamente atribuídos às atividades desse processo.

A seguir, apresenta-se os casos de teste desenvolvidos, juntamente com os resultados de suas execuções.

Tabela 28 – Caso de Teste – Acessar a Lista de Processos Executáveis Disponíveis

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

A análise do primeiro caso de teste aponta que o resultado da listagem dos processos executáveis disponíveis é compatível com o esperado.

Tabela 29 – Caso de Teste – Acessar a Lista de Processos Executáveis Disponíveis

Caso de Teste 2 – Executar Processos	
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".

A análise do segundo caso de teste aponta que o resultado da execução dos processos é compatível com o esperado.

Tabela 30 – Caso de Teste – Acessar a Lista de Processos Executáveis Disponíveis

Caso de Teste 3 – Monitorar o Andamento dos Processos	
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

A análise do terceiro caso de teste 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, pode-se concluir que o Ambiente de Execução de Processos atende aos requisitos estabelecidos no modelo conceitual.

Ambiente de Descoberta de Serviços

A verificação do Ambiente de Descoberta Dinâmica de Serviços está fora do escopo desse trabalho. Mais detalhes de como esse ambiente foi verificado e avaliado podem ser vistos em (SOUZA, 2009).

Apêndice C

Conversão BPMN para BPEL

Esta seção apresenta um exemplo de conversão do processo de pagamento descrito em BPMN para o formato BPEL, utilizado na execução do processo. O processo, originalmente em BPMN, é apresentado na Figura 78. O processo convertido em BPEL é apresentado na Figura 79.

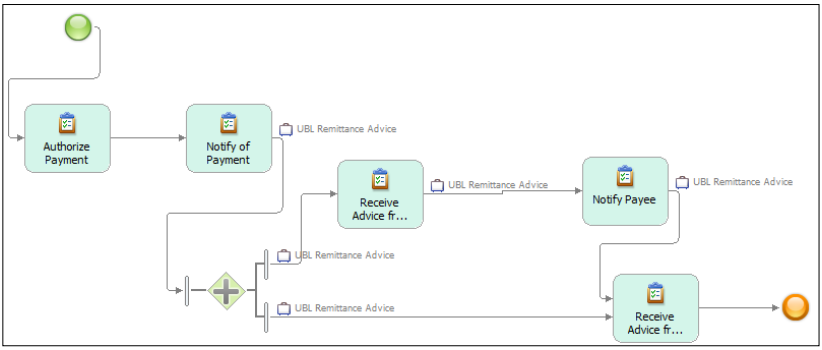


Figura 78 – Processo de Pagamento em Linguagem BPMN

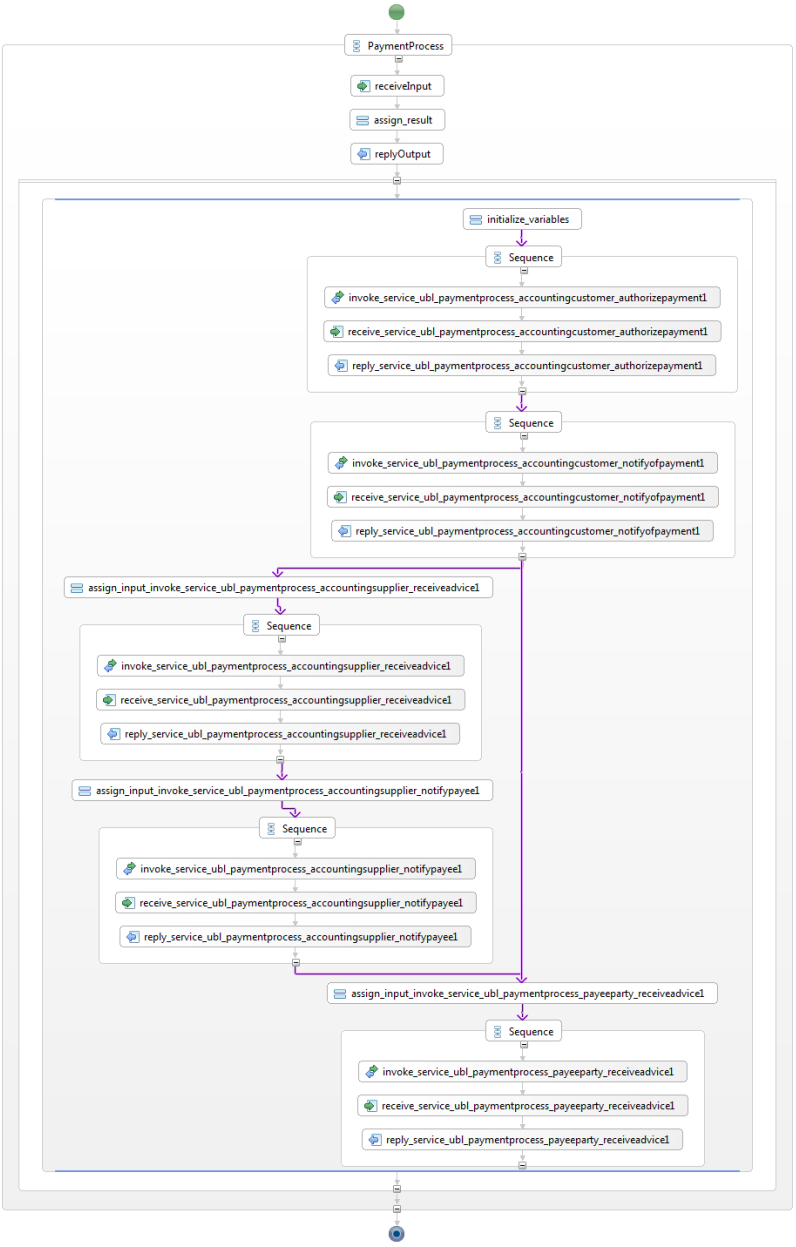


Figura 79 – Processo de Pagamento em Linguagem BPEL

Apêndice D

Interface WSDL do Serviço UBL

Esta seção 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 E

Publicação dos Serviços UBL na Federação

Esta seção apresenta o código fonte em Java utilizado para se fazer a publicação dos serviços UBL na federação de serviços, composta com repositórios UDDI.

```
package br.ufsc.gsigma.uddi;

import java.util.Random;

import org.apache.juddi.api_v3.*;
import org.apache.juddi.v3.client.*
import org.uddi.api_v3.*
import org.uddi.v3_service.UDDIPublicationPortType;
import org.uddi.v3_service.UDDISecurityPortType;

import br.ufsc.gsigma.infrastructure.util.UDDIKeys;

public class RegisterUDDIServices {
 private static Transport getUDDITransport(String nodeName) {
  try {
   String clazz =
UDDIClientContainer.getUDDIClerkManager(null).getClientConfig().getUDDINode(nod
eName).getProxyTransport();
   Class<?> transportClass = ClassUtil.forName(clazz, Transport.class);
   if (transportClass != null)
    return (Transport)
transportClass.getConstructor(String.class).newInstance(nodeName);
  } catch (Exception e) {
   e.printStackTrace();
  }
```

```
 return null;
 }

 public static void publishService(Transport transport, String authLogin,
String authPassword, String businessServiceKey,
   String serviceNameStr, String serviceDescriptionStr, String endpointStr,
TModelInstanceInfo[] tModelInstanceInfos,
   KeyedReference[] qoSConstraintsKeyedReferences) throws Exception {

  UDDISecurityPortType security = transport.getUDDISecurityService();
  UDDIPublicationPortType publish = transport.getUDDIPublishService();

  GetAuthToken getAuthTokenRoot = new GetAuthToken();
  getAuthTokenRoot.setUserID(authLogin);
  getAuthTokenRoot.setCred(authPassword);

  AuthToken authToken = security.getAuthToken(getAuthTokenRoot);

  BusinessService service = new BusinessService();

  service.setBusinessKey(businessServiceKey);

  Name serviceName = new Name();
  serviceName.setValue(serviceNameStr);
  service.getName().add(serviceName);

  Description serviceDescription = new Description();
  serviceDescription.setValue(serviceDescriptionStr);
  service.getDescription().add(serviceDescription);

  BindingTemplates bts = new BindingTemplates();
  service.setBindingTemplates(bts);

  BindingTemplate bt = new BindingTemplate();

  Description bindingDescription = new Description();
  bindingDescription.setValue("Binding for " + serviceName.getValue());
  bt.getDescription().add(bindingDescription);

  AccessPoint ap = new AccessPoint();
  ap.setValue(endpointStr);
  ap.setUseType("endpoint");
  bt.setAccessPoint(ap);
  bts.getBindingTemplate().add(bt);

  TModelInstanceDetails tModelInstanceDetails = new TModelInstanceDetails();
  bt.setTModelInstanceDetails(tModelInstanceDetails);

  TModelInstanceInfo tModelInstanceInfo = new TModelInstanceInfo();
  tModelInstanceInfo.setTModelKey("uddi:uddi.org:transport:http");
  tModelInstanceDetails.getTModelInstanceInfo().add(tModelInstanceInfo);

  if (tModelInstanceInfos != null) {
   for (TModelInstanceInfo t : tModelInstanceInfos)
    tModelInstanceDetails.getTModelInstanceInfo().add(t);
  }

  CategoryBag categoryBag = new CategoryBag();
  bt.setCategoryBag(categoryBag);
```

```
 KeyedReferenceGroup qoSkeyedReferenceGroup = new KeyedReferenceGroup();
 qoSkeyedReferenceGroup.setTModelKey("uddi:gsigma.ufsc.br:qos:serviceqoslist");
 categoryBag.getKeyedReferenceGroup().add(qoSkeyedReferenceGroup);

 if (qoSConstraintsKeyedReferences != null) {
  for (KeyedReference k : qoSConstraintsKeyedReferences)
   qoSkeyedReferenceGroup.getKeyedReference().add(k);
 }

 SaveService ss = new SaveService();
 ss.getBusinessService().add(service);
 ss.setAuthInfo(authToken.getAuthInfo());

 ServiceDetail sd = publish.saveService(ss);
 String myServKey = sd.getBusinessService().get(0).getServiceKey();

 System.out.println("myService key:  " + myServKey);

}

public static TModelInstanceInfo getTModelInstanceInfo(String tModelKey) {
 TModelInstanceInfo tModelInstanceInfo = new TModelInstanceInfo();
 tModelInstanceInfo.setTModelKey(tModelKey);
 return tModelInstanceInfo;
}

public static KeyedReference getKeyedReference(String tModelKey, String
keyName, Object keyValue) {
 KeyedReference keyedReference = new KeyedReference();
 keyedReference.setTModelKey(tModelKey);
 keyedReference.setKeyName(keyName);
 keyedReference.setKeyValue(keyValue.toString());
 return keyedReference;
}

public static KeyedReference getKeyedReference(String tModelKey, Object
keyValue) {
 KeyedReference keyedReference = new KeyedReference();
 keyedReference.setTModelKey(tModelKey);
 keyedReference.setKeyValue(keyValue.toString());
 return keyedReference;
}

public static BusinessDetail createBusinessEntity(Transport transport, String
authLogin, String authPassword, String name)
  throws Exception {

 UDDISecurityPortType security = transport.getUDDISecurityService();
 UDDIPublicationPortType publish = transport.getUDDIPublishService();

 GetAuthToken getAuthTokenRoot = new GetAuthToken();
 getAuthTokenRoot.setUserID(authLogin);
 getAuthTokenRoot.setCred(authPassword);

 AuthToken authToken = security.getAuthToken(getAuthTokenRoot);

 BusinessEntity businessEntity = new BusinessEntity();
 Name businessEntityName = new Name();
```

```
 businessEntityName.setValue(name);
 businessEntity.getName().add(businessEntityName);

 SaveBusiness sb = new SaveBusiness();
 sb.getBusinessEntity().add(businessEntity);

 sb.setAuthInfo(authToken.getAuthInfo());

 return publish.saveBusiness(sb);
}

 private static void populateServices(Transport transport, String authLogin,
String authPassword, String businessKey,
  String businessEntityName, String serviceBaseHost) throws Exception {

 if (businessKey == null) {
  BusinessDetail businessDetail = createBusinessEntity(transport, authLogin,
authPassword, businessEntityName);
  if (businessDetail.getBusinessEntity() != null &&
businessDetail.getBusinessEntity().size() > 0) {
   businessKey = businessDetail.getBusinessEntity().get(0).getBusinessKey();
  }

  (…)

  // Ordering Process
  publishService(transport, authLogin, authPassword, businessKey, //
    "UBLOrderingProcessBuyerPartyPlaceOrder", //
    "UBL Payment -> Ordering Process -> Buyer Party -> Place Order", //
    "http://" + serviceBaseHost +
"/services/ubl/orderingprocess/buyerParty/placeOrder", //
    new TModelInstanceInfo[] {
getTModelInstanceInfo("uddi:ubl:services:ordering_orderingprocess_buyerparty_pl
aceorder")
}, //
    buildRandomQoS());

  publishService(transport, authLogin, authPassword,
    businessKey, //
    "UBLOrderingProcessSellerPartyReceiveOrder", //
    "UBL Payment -> Ordering Process -> Seller Party -> Receive Order", //
    "http://" + serviceBaseHost +
"/services/ubl/orderingprocess/sellerParty/receiveOrder", //
    new TModelInstanceInfo[] {
getTModelInstanceInfo("uddi:ubl:services:ordering_orderingprocess_sellerparty_r
eceiveorder")
}, //
    buildRandomQoS());

  publishService(transport, authLogin, authPassword,
    businessKey, //
    "UBLOrderingProcessSellerPartyProcessOrder", //
    "UBL Payment -> Ordering Process -> Seller Party -> Process Order", //
    "http://" + serviceBaseHost +
"/services/ubl/orderingprocess/sellerParty/processOrder", //
    new TModelInstanceInfo[] {
getTModelInstanceInfo("uddi:ubl:services:ordering_orderingprocess_sellerparty_p
rocessorder")
}, //
    buildRandomQoS());
```

```
  publishService(transport, authLogin, authPassword,
    businessKey, //
    "UBLOrderingProcessSellerPartyAcceptOrder", //
    "UBL Payment -> Ordering Process -> Seller Party -> Accept Order", //
    "http://" + serviceBaseHost +
"/services/ubl/orderingprocess/sellerParty/acceptOrder", //
    new TModelInstanceInfo[] {
getTModelInstanceInfo("uddi:ubl:services:ordering_orderingprocess_sellerparty_a
cceptorder")
}, //
    buildRandomQoS());

  publishService(transport, authLogin, authPassword,
    businessKey, //
    "UBLOrderingProcessSellerPartyRejectOrder", //
    "UBL Payment -> Ordering Process -> Seller Party -> Reject Order", //
    "http://" + serviceBaseHost +
"/services/ubl/orderingprocess/sellerParty/rejectOrder", //
    new TModelInstanceInfo[] {
getTModelInstanceInfo("uddi:ubl:services:ordering_orderingprocess_sellerparty_r
ejectorder")
}, //
    buildRandomQoS());
  publishService(transport, authLogin, authPassword, businessKey, //
    "UBLOrderingProcessSellerPartyAddDetail", //
    "UBL Payment -> Ordering Process -> Seller Party -> Add Detail", //
    "http://" + serviceBaseHost +
"/services/ubl/orderingprocess/sellerParty/addDetail", //
    new TModelInstanceInfo[] {
getTModelInstanceInfo("uddi:ubl:services:ordering_orderingprocess_sellerparty_a
dddetail")
}, //
    buildRandomQoS());

  publishService(transport, authLogin, authPassword,
    businessKey, //
    "UBLOrderingProcessBuyerPartyReceiveResponse", //
    "UBL Payment -> Ordering Process -> Buyer Party -> Receive Response", //
    "http://" + serviceBaseHost +
"/services/ubl/orderingprocess/buyerParty/receiveResponse", //
    new TModelInstanceInfo[] {
getTModelInstanceInfo("uddi:ubl:services:ordering_orderingprocess_buyerparty_re
ceiveresponse")
}, //
    buildRandomQoS());

  publishService(transport, authLogin, authPassword,
    businessKey, //
    "UBLOrderingProcessBuyerPartyAcceptOrder", //
    "UBL Payment -> Ordering Process -> Buyer Party -> Accept Order", //
    "http://" + serviceBaseHost +
"/services/ubl/orderingprocess/buyerParty/acceptOrder", //
    new TModelInstanceInfo[] {
getTModelInstanceInfo("uddi:ubl:services:ordering_orderingprocess_buyerparty_ac
ceptorder")
}, //
    buildRandomQoS());
```

```
  publishService(transport, authLogin, authPassword,
    businessKey, //
    "UBLOrderingProcessBuyerPartyChangeOrder", //
    "UBL Payment -> Ordering Process -> Buyer Party -> Change Order", //
    "http://" + serviceBaseHost +
"/services/ubl/orderingprocess/buyerParty/changeOrder", //
    new TModelInstanceInfo[] {
getTModelInstanceInfo("uddi:ubl:services:ordering_orderingprocess_buyerparty_ch
angeorder")
}, //
    buildRandomQoS());

  publishService(transport, authLogin, authPassword,
    businessKey, //
    "UBLOrderingProcessBuyerPartyCancelOrder", //
    "UBL Payment -> Ordering Process -> Buyer Party -> Cancel Order", //
    "http://" + serviceBaseHost +
"/services/ubl/orderingprocess/buyerParty/cancelOrder", //
    new TModelInstanceInfo[] {
getTModelInstanceInfo("uddi:ubl:services:ordering_orderingprocess_buyerparty_ca
ncelorder")
}, //
    buildRandomQoS());

  publishService(transport, authLogin, authPassword,
    businessKey, //
    "UBLOrderingProcessSellerPartyCancelOrder", //
    "UBL Payment -> Ordering Process -> Seller Party -> Cancel Order", //
    "http://" + serviceBaseHost +
"/services/ubl/orderingprocess/sellerParty/cancelOrder", //
    new TModelInstanceInfo[] {
getTModelInstanceInfo("uddi:ubl:services:ordering_orderingprocess_sellerparty_c
ancelorder")
}, //
    buildRandomQoS());

  publishService(transport, authLogin, authPassword,
    businessKey, //
    "UBLOrderingProcessSellerPartyChangeOrder", //
    "UBL Payment -> Ordering Process -> Seller Party -> Change Order", //
    "http://" + serviceBaseHost +
"/services/ubl/orderingprocess/sellerParty/changeOrder", //
    new TModelInstanceInfo[] {
getTModelInstanceInfo("uddi:ubl:services:ordering_orderingprocess_sellerparty_c
hangeorder")
}, //
    buildRandomQoS());

  // Payment Process
  publishService(transport, authLogin, authPassword,
    businessKey, //
    "UBLPaymentProcessAccountingCustomerAuthorizePayment", //
    "UBL Payment -> Payment Process -> Accounting Customer -> Authorize
Payment", //
    "http://" + serviceBaseHost +
"/services/ubl/paymentprocess/accountingCustomer/authorizePayment", //
    new TModelInstanceInfo[] {
getTModelInstanceInfo("uddi:ubl:services:payment_paymentprocess_accountingcusto
mer_authorizepayment")
}, //
```

```
    buildRandomQoS());

  publishService(transport, authLogin, authPassword,
    businessKey, //
    "UBLPaymentProcessAccountingCustomerNotifyOfPayment", //
    "UBL Payment -> Payment Process -> Accounting Customer -> Notify of
Payment", //
    "http://" + serviceBaseHost +
"/services/ubl/paymentprocess/accountingCustomer/notifyOfPayment", //
    new TModelInstanceInfo[] {
getTModelInstanceInfo("uddi:ubl:services:payment_paymentprocess_accountingcusto
mer_notifyofpayment")
}, //
    buildRandomQoS());

  publishService(transport, authLogin, authPassword,
    businessKey, //
    "UBLPaymentProcessAccountingSupplierNotifyPayee", //
    "UBL Payment -> Payment Process -> Accounting Supplier -> Notify Payee", //
    "http://" + serviceBaseHost +
"/services/ubl/paymentprocess/accountingSupplier/notifyPayee", //
    new TModelInstanceInfo[] {
getTModelInstanceInfo("uddi:ubl:services:payment_paymentprocess_accountingsuppl
ier_notifypayee")
}, //
    buildRandomQoS());

  publishService(transport, authLogin, authPassword,
    businessKey, //
    "UBLPaymentProcessAccountingSupplierReceiveAdvice", //
    "UBL Payment -> Payment Process -> Accounting Supplier -> Receive Advice",
//
    "http://" + serviceBaseHost +
"/services/ubl/paymentprocess/accountingSupplier/receiveAdvice", //
    new TModelInstanceInfo[] {
getTModelInstanceInfo("uddi:ubl:services:payment_paymentprocess_accountingsuppl
ier_receiveadvice")
}, //
    buildRandomQoS());

  publishService(transport, authLogin, authPassword,
    businessKey, //
    "UBLPaymentProcessPayeePartyReceiveAdvice", //
    "UBL Payment -> Payment Process -> Payee Party -> Receive Advice", //
    "http://" + serviceBaseHost +
"/services/ubl/paymentprocess/payeeParty/receiveAdvice", //
    new TModelInstanceInfo[] {
getTModelInstanceInfo("uddi:ubl:services:payment_paymentprocess_payeeparty_rece
iveadvice")
}, //
    buildRandomQoS());
  }
 }

 private static KeyedReference[] buildRandomQoS() {

  Random random = new Random();

  return new KeyedReference[] { //
```

```
 //
   getKeyedReference(UDDIKeys.UDDI_QOS_ACCURACY_GENERATEDERROR,
"GeneratedError", random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_ACESSIBILITY_ACESSIBILITY,
"Acessibility", random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_AVAILABILITY_AVAILABILITY,
"Availability", random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_CAPACITY_CAPACITY, " apacity",
random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_EXCEPTIONHANDLING_FUNCTIONALITY,
"Functionality", random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_INTEGRITY_DATAINTEGRITY,
"DataIntegrity", random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_INTEGRITY_TRANSACTIONALINTEGRITY,
"TransactionalIntegrity", random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_INTEROPERABILITY_INTEROPERABILITY,
"Interoperability", random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_PERFORMANCE_LATENCY, "Latency",
random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_PERFORMANCE_THROUGHPUT, "Throughput",
random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_PERFORMANCE_EXECUTIONTIME,
"ExecutionTime", random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_PERFORMANCE_TRANSACTIONTIME,
"TransactionTime", random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_PERFORMANCE_RESPONSETIME,
"ResponseTime", random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_RELIABILITY_MESSAGEWARRANTY,
"MessageWarranty", random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_RELIABILITY_MONTH, "Month",
random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_RELIABILITY_DAY, "Day",
random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_RELIABILITY_YEAR, "Year",
random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_ROBUSTNESS_ROBUSTNESS, "Robustness",
random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_SCALABILITY_LATENCY, "latency",
random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_SCALABILITY_TRANSACTIONTIME,
"TransactionTime", random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_SCALABILITY_EXECUTIONTIME,
"ExecutionTime", random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_SCALABILITY_THROUGHPUT, "Throughput",
random.nextInt(101)), //
   getKeyedReference(UDDIKeys.UDDI_QOS_SCALABILITY_RESPONSETIME,
"ResponseTime", random.nextInt(101)), //
 };

}

 private static void createPublishers(Transport transport, int k) throws
Exception {

 UDDISecurityPortType security = transport.getUDDISecurityService();

 GetAuthToken getAuthTokenRoot = new GetAuthToken();
 getAuthTokenRoot.setUserID("root");
 getAuthTokenRoot.setCred("");
```

```
 AuthToken authToken = security.getAuthToken(getAuthTokenRoot);

 SavePublisher savePublisher = new SavePublisher();
 savePublisher.setAuthInfo(authToken.getAuthInfo());

 for (int i = 1; i <= k; i++) {

  Publisher publisher = new Publisher();
  publisher.setAuthorizedName("ubl" + i);
  publisher.setPublisherName("Company " + i);
  savePublisher.getPublisher().add(publisher);
 }

 transport.getJUDDIApiService().savePublisher(savePublisher);
}

//Publica os serviços em cinco repositórios UDDI
public static void main(String[] args) throws Exception {

 String host = "localhost:8082";
   createPublishers(getUDDITransport("ublRepository1"), 10);
  createPublishers(getUDDITransport("ublRepository2"), 10);
  createPublishers(getUDDITransport("ublRepository3"), 10);
  createPublishers(getUDDITransport("ublRepository4"), 10);
  createPublishers(getUDDITransport("ublRepository5"), 10);

  for (int i = 0; i < 20; i++) {
   populateServices(getUDDITransport("ublRepository1"), "ubl" + (new
Random().nextInt(10) + 1), "", null, "UBL Services 1", host);
   populateServices(getUDDITransport("ublRepository2"), "ubl" + (new
Random().nextInt(10) + 1), "", null, "UBL Services 2", host);
   populateServices(getUDDITransport("ublRepository3"), "ubl" + (new
Random().nextInt(10) + 1), "", null, "UBL Services 3", host);
   populateServices(getUDDITransport("ublRepository4"), "ubl" + (new
Random().nextInt(10) + 1), "", null, "UBL Services 4", host);
   populateServices(getUDDITransport("ublRepository5"), "ubl" + (new
Random().nextInt(10) + 1), "", null, "UBL Services 5", host);
  }
 }
}
```

Apêndice F

Código Fonte do Teste JUnit

Esta seção apresenta o código fonte do teste JUnit utilizado no teste de unidade do módulo catálogo de processos de negócio.

```
package br.ufsc.gsigma.catalog.services.tests.junit;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertNotNull;
import static org.junit.Assert.assertTrue;

import java.util.List;

import org.junit.Before;
import org.junit.Test;

import br.ufsc.gsigma.catalog.services.bpelexport.output.BPELProcessBundle;
import br.ufsc.gsigma.catalog.services.interfaces.CatalogService;
import br.ufsc.gsigma.catalog.services.model.Document;
import br.ufsc.gsigma.catalog.services.model.ProcessCategory;
import br.ufsc.gsigma.catalog.services.model.ProcessStandard;
import br.ufsc.gsigma.catalog.services.model.ProcessTreeDescription;
import br.ufsc.gsigma.catalog.services.output.QoSItem;
import br.ufsc.gsigma.catalog.services.specifications.output.ProcessOntology;
import
br.ufsc.gsigma.catalog.services.specifications.output.ProcessStandardSpecificat
ion;
import br.ufsc.gsigma.infrastructure.ws.WebServiceLocator;
import br.ufsc.gsigma.servicediscovery.model.DiscoveryResult;
import br.ufsc.gsigma.servicediscovery.model.QoSInformation;

public class CatalogServiceTest {

 // Locate the Catalog Service instance
```

```
 private CatalogService service =
WebServiceLocator.locateService(WebServiceLocator.CATALOG_SERVICE_UDDI_SERVICE_
KEY,
   CatalogService.class);

 @Before
 public void setUp() throws Exception {
  // Clear the processes from database. This is necessary to run the tests
  service.removeAllProcesses();

 }

 @Test
 public void testAddOrUpdateProcess() {
  br.ufsc.gsigma.catalog.services.model.Process process = new
br.ufsc.gsigma.catalog.services.model.Process("Teste");
  Integer processId = service.addOrUpdateProcess(1, process);
  br.ufsc.gsigma.catalog.services.model.Process retrievedProcess =
service.getProcessById(processId);
  assertNotNull(retrievedProcess);
  assertEquals("Teste", retrievedProcess.getName());

 }

 @Test
 public void testCheckIfProcessNameIsAvailable() {
  br.ufsc.gsigma.catalog.services.model.Process process = new
br.ufsc.gsigma.catalog.services.model.Process("Teste");
  service.addOrUpdateProcess(1, process);
  assertFalse(!service.checkIfProcessNameIsAvailable(1, "Teste"));
 }

 @Test
 public void testGetProcessById() {
  br.ufsc.gsigma.catalog.services.model.Process process = new
br.ufsc.gsigma.catalog.services.model.Process("Teste");
  Integer processId = service.addOrUpdateProcess(1, process);
  br.ufsc.gsigma.catalog.services.model.Process retrievedProcess =
service.getProcessById(processId);
  assertNotNull(retrievedProcess);
  assertEquals("Teste", retrievedProcess.getName());
 }

 @Test
 public void testGetListProcessInfo() {
 }

 @Test
 public void testConvertToBPEL() throws Exception {
  br.ufsc.gsigma.catalog.services.model.Process process = service
    .getProcessFromProcessStandardSpecification("ubl/payment/paymentprocess");
  BPELProcessBundle result = service.convertToBPEL(process);
  assertNotNull(result);

 }

 @Test
 public void testGetCatalogDocumentById() {
  Document result = service.getCatalogDocumentById("UBL_ORDER");
  assertNotNull(result);
```

```
 }

 @Test
 public void testGetListProcessStandard() {
  List<ProcessStandard> result = service.getListProcessStandard();
  assertNotNull(result);
  assertTrue(result.size() > 0);
 }

 @Test
 public void testGetListProcessTreeDescription() {
  List<ProcessTreeDescription> result =
service.getListProcessTreeDescription(1);
  assertNotNull(result);
  assertTrue(result.size() > 0);
 }

 @Test
 public void testGetProcessCategories() {
  List<ProcessCategory> result = service.getProcessCategories();
  assertNotNull(result);
  assertTrue(result.size() > 0);
 }

 @Test
 public void testGetDocumentsXSDZipFromProcessStandardSpecification() {
  byte[] result =
service.getDocumentsXSDZipFromProcessStandardSpecification("UBL");
  assertNotNull(result);
 }

 @Test
 public void testGetProcessFromProcessStandardSpecification() {
  br.ufsc.gsigma.catalog.services.model.Process result = service
    .getProcessFromProcessStandardSpecification("ubl/payment/paymentprocess");
  assertNotNull(result);
 }

 @Test
 public void testGetProcessOntology() {
  ProcessOntology result = service.getProcessOntology("UBL");
  assertNotNull(result);
 }

 @Test
 public void testGetProcessStandardSpecification() {
  ProcessStandardSpecification result =
service.getProcessStandardSpecification("UBL");
  assertNotNull(result);
 }

 @Test
 public void testGetServiceWSDLFromOntologyClassification() {
  String result =
service.getServiceWSDLFromOntologyClassification("ubl/payment/paymentprocess/ac
countingCustomer/authorizePayment");
  assertNotNull(result);
 }
```

```
 @Test
 public void testDiscoverServices() {
  DiscoveryResult result =
service.discoverServices("ubl/payment/paymentprocess/accountingCustomer/authori
zePayment",
    new QoSInformation());
  assertNotNull(result);
  assertTrue(result.getMatchingServices().size() > 0);
 }

 @Test
 public void testDiscoverUniqueServiceEndpoint() {
  String result =
service.discoverUniqueServiceEndpoint("ubl/payment/paymentprocess/accountingCus
tomer/authorizePayment",
    new QoSInformation());
  assertNotNull(result);
 }

 @Test
 public void testGetQoSItems() {
  List<QoSItem> list = service.getQoSItems();
  assertNotNull(list);
  assertTrue(list.size() > 0);
 }

}
```

Referências Bibliográficas

ADAM, S.; DOERR, J. **How to better align BPM & SOA - Ideas on improving the transition between process design and deployment**. 9th Workshop on Business Process Modeling, Development and Support, 2008.

AGRAWAL, A. et al. **WS-BPEL extension for people (BPEL4People), version 1.0**. Active Endpoints Inc., Adobe Systems Inc., BEA Systems Inc., International Business Machines Corporation, Oracle Inc., and SAP AG, 2007.

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**. Trabalho apresentado no XXIII Encontro Nacional de Engenharia de Produção. Ouro Preto, MG, Brasil, 2003.

APACHE. **jUDDI v3**. 2010a. Disponível em: < http://juddi.apache.org >. Acesso em: 17 jun 2010.

_____. **ODE (Orchestration Director Engine)**. 2010b. Disponível em: < http://ode.apache.org/ >. Acesso em: 14 jun 2010.

_____. **Tomcat**. 2010c. Disponível em: < http://tomcat.apache.org/ >. Acesso em: 08 jun 2010.

_____. **Tuscany**. 2010d. Disponível em: < http://tuscany.apache.org >. Acesso em: 18 jul 2010.

BAILEY, J. **Fast Discovery of Interesting Collections of Web Services**. Web Intelligence, 2006. WI 2006. IEEE/WIC/ACM International Conference on, 2006. 2007. p.152-160.

BALDAM, R. et al. **Gerenciamento de Processos de Negócios: BPM–Business Process Management**. 2007.

BISHOP, B.; FISCHER, F. **IRIS-Integrated Rule Inference System**. International Workshop on Advancing Reasoning on the Web: Scalability and Commonsense, 2008.

BOOTH, D. et al. **Web services architecture**. W3C Working Group Note**,** v. 11, p. 2005-1, 2004.

BPMINSTITUTE. **State of Business Process Management**. 2006. Disponível em: < http://www.bpminstitute.org/uploads/media/2006_State_of_BPM_-_EOY.pdf >. Acesso em: 23 ago 2010.

BREIVOLD, H. P.; LARSSON, M. **Component-based and service-oriented software engineering: Key concepts and principles**. Software Engineering and Advanced Applications, 2007. 33rd EUROMICRO Conference, 2007. p.13-20.

CANCIAN, M. H. **Uma Proposta de Guia de Referência para Provedores de Software como um Serviço**. 2009. (Dissertação de Mestrado). Programa de Pós-Graduação em Engenharia Elétrica. UFSC

CHAPPELL, D. **Introducing SCA**. A White Paper by David Chappell, 2007.

CHOI, I.; KIM, K.; JANG, M. **An XML-based process repository and process query language for integrated process management**. Business Change and Re-engineering**,** v. 14, n. 4, p. 303-316, 2007.

CLAYBERG, E.; RUBEL, D. **Eclipse: Building Commercial-Quality Plug-ins (Eclipse)**. Addison-Wesley Professional, 2006.

CODEHAUS. **Jetty Web Server**. 2010. Disponível em: < http://jetty.codehaus.org/jetty/ >. Acesso em: 07 set 2010.

CURTIS, B.; KELLNER, M. I.; OVER, J. **Process modeling**. Commun. ACM**,** v. 35, n. 9, p. 75-90, 1992.

DE BRUIJN, J. et al. **The web service modeling language WSML**. WSML Final Draft D**,** v. 16, 2005.

DIJKMAN, R. M.; DUMAS, M.; OUYANG, C. **Semantics and analysis of business process models in BPMN**. Information and Software Technology**,** v. 50, n. 12, p. 1281-1294, 2008.

DORN, J. et al. **From business to software: a B2B survey**. Information Systems and E-Business Management**,** v. 7, n. 2, p. 123-142, 2009.

_____. **A survey of B2B Methodologies and Technologies: From Business Models towards Deployment Artifacts**. v. 40, n. 5, 2007.

DUDLEY, C. **WebSphere Service Registry and Repository Handbook**. IBM, International Technical Support Organization, 2007.

ECLIPSE. **Eclipse Modeling Framework**. 2009. Disponível em: < http://www.eclipse.org/modeling/emf/ >. Acesso em: 21 set 2010.

_____. **SWT: The Standard Widget Toolkit**. 2010. Disponível em: < http://www.eclipse.org/swt/ >. Acesso em: 14 out 2010.

EVIWARE. **SoapUI**. 2010. Disponível em: < http://www.soapui.org >. Acesso em: 21 set 2010.

FALBO, R. A.; GUIZZARDI, G.; DUARTE, K. C. **An ontological approach to domain engineering**. Proceedings of the 14th international conference on Software engineering and knowledge engineering, 2002. p.358.

FASBINDER, M. **Transforming Business Models to BPEL with WebSphere Business Modeler 6.02**: Technical Article 2007.

FOURNIER, G. **Essential Software Testing: A Use-Case Approach**. Auerbach Publications, 2008.

FREEMARKER. **Freemarker Templating Engine**. 2010. Disponível em: < http://freemarker.sourceforge.net/ >. Acesso em: 21 mai 2010.

GARIMELLA, K.; LESS, M.; WILLIAMS, B. **BPM Basic for Dummies**. Indiana: Wiley Publishing, 2008.

GAROFALAKIS, J. et al. **Contemporary web service discovery mechanisms**. Journal of Web Engineering**,** v. 5, n. 3, 2006.

GIL, A. C. **Como elaborar projetos de pesquisa**. Atlas S. Paulo, 2002.

GOLDKUHL, G.; LIND, M. **Coordination and transformation in business processes: towards an integrated view**. Business Process Management Journal**,** v. 14, n. 6, p. 761-777, 2008.

GUIZZARDI, G. **Uma abordagem metodológica de desenvolvimento para e com reuso, baseada em ontologias formais de domínio**. 2000. Programa de Mestrado em Informática – Universidade Federal do Espírito Santo

GÜNTHER, H. **Como elaborar um questionário**. 2003.

HEPP, M. et al. **Semantic business process management: A vision towards using semantic web services for business process management**. ICEBE**,** v. 5, p. 535-540, 2005.

HILL, J. B.; PEZZINI, M.; NATIS, Y. V. **Findings: confusion remains regarding BPM terminologies**. Gartner Research, Stamford, CT**,** v. 501, n. G00155817, 2008.

HURBEAN, L. **The business of process integration**. 2007.

HYPERSQL. **HSQLDB - 100% Java Database**. 2010. Disponível em: < http://hsqldb.org >. Acesso em: 05 set 2010.

IBM. **BPEL Repository - IBM alphaWorks**. 2006. Disponível em: < http://www.alphaworks.ibm.com/tech/bpelrepository >. Acesso em: 03 ago 2010.

_____. **WebSphere Business Modeler**. 2010. Disponível em: < http://www-

01.ibm.com/software/integration/wbimodeler/advanced/features/ >. Acesso em: 25 set 2010.

INTALIO. **BPMS**. 2010. Disponível em: < http://www.intalio.com/bpms >. Acesso em: 23 out 2010.

ISO. **Electronic Business Extensible Markup Language (ebXML) -- Part 5: ebXML Core Components Technical Specification**. 2005. Disponível em: < http://www.iso.org/iso/catalogue_detail.htm?csnumber=41022 >. Acesso em: 23 set 2010.

JBOSS. **Hibernate - Relational Persistence for Java and .NET**. 2010. Disponível em: < http://www.hibernate.org/ >. Acesso em: 07 out 2010.

JUNIT. **Resources for Test Driven Development**. 2010. Disponível em: < http://www.junit.org/ >. Acesso em: 23 nov 2010.

JURIC, M.; PANT, K. **Business Process Driven SOA Using BPMN and BPEL**. Packt publishing, 2007.

JURIC, M. B.; SASA, A.; ROZMAN, I. **WS-BPEL extensions for versioning**. Information and Software Technology**,** v. 51, n. 8, p. 1261-1274, 2009.

KAMOUN, F. **A roadmap towards the convergence of business process management and service oriented architecture**. Ubiquity**,** v. 2007, n. April, p. 1-1, 2007.

KO, R. K. L. **A computer scientist's introductory guide to business process management (BPM)**. Crossroads**,** v. 15, n. 4, p. 11-18, 2009.

KO, R. K. L.; LEE, S. S. G.; LEE, E. W. **Business process management (BPM) standards: a survey**. Business Process Management Journal**,** v. 15, n. 5, 2009.

KOTINURMI, P.; HALLER, A.; OREN, E. **Ontologically enhanced RosettaNet B2B Integration**. 2008.

KOURTESIS, D. et al. **Web Service Discovery in a Semantically Extended UDDI Registry: the Case of FUSION**. International Federation for Information Processing-Publications-IFIP**,** v. 243, 2007.

LAMPATHAKI, F. et al. **Business to business interoperability: A current review of XML data integration standards**. Computer Standards & Interfaces**,** v. 31, n. 6, p. 1045-1055, 2009.

LAWS, S. et al. **Tuscany SCA in Action**. Manning Publications, 2010.

LEE, K. C. et al. **Qos for web services: Requirements and possible approaches**. W3C Working Group Note**,** v. 25, 2003.

LEYMANN, F.; ROLLER, D.; SCHMIDT, M. T. **Web services and business process management**. IBM Systems Journal**,** v. 41, n. 2, p. 198-211, 2002.

LIKERT, R. **A technique for the measurement of attitudes**. 1932.

LÜER, C. **Evaluating the Eclipse platform as a composition environment**. 3rd International Workshop on Adoption-Centric Software Engineering, Portland, Oregon, 2002. 2002.

MA, Z. et al. **Semantic business process repository**. Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management (SBPM), 2007. p.92–100.

MANI, A.; NAGARAJAN, A. **Understanding quality of service for Web services**. 2002.

_____. **Understanding quality of service for Web services**. 2005.

MARGOLIS, B. **SOA for the Business Developer: Concepts, BPEL, and SCA (Business Developers series)**. Mc Press, 2007.

MEDJAHED, B. et al. **Business-to-business interactions: issues and enabling technologies**. The VLDB Journal**,** v. 12, n. 1, p. 59-85, 2003.

MENDLING, J.; NEUMANN, G.; N\ÜTTGENS, M. **A Comparison of XML Interchange Formats for Business Process Modelling**. EMISA 2004, p. 129, 2004.

MENEZES, E. M.; SILVA, E. L. **Metodologia da Pesquisa e Elaboração de Dissertação**. Florianópolis: Ed. da UFSC, 2005.

MERSON, P. **Como Documentar Arquitetura de Software**. Minicurso realizado no Simpósio Brasileiro de Banco de Dados e Simpósio Brasileiro de Engenharia de Software, promoção SBC, 2005. Uberlândia. Brasil., 2005.

METEOR. **METEOR-S: Semantic Web Services and Processes** 2001.

MIT. **The MIT Process Handbook Project**. 2005. Disponível em: < http://ccs.mit.edu/ph >. Acesso em: 22 ago 2010.

NATALYA, F. N.; DEBORAH, L. M. **Ontology development 101: A guide to creating your first ontology**. McGuinnes. Stanford University, 2002.

NOEL, J. **BPM and SOA: Better together**. Whitepaper, IBM Corporation, 2005.

NURMILAAKSO, J. M. **EDI, XML and e-business frameworks: A survey**. Computers in Industry, 2007.

O'DOCHERTY, M. **Object-oriented analysis and design: understanding system development with UML 2.0**. John Wiley & Sons Inc, 2005.

OASIS. **ebXML Catalog of Common Business Processes v1.0**. 2001. Disponível em: < http://www.ebxml.org/specs/bpPROC.pdf >. Acesso em: 27 nov 2010.

_____. **Universal Description, Discovery and Integration v3.0.2 (UDDI)**. 2005. Disponível em: < http://uddi.xml.org/ >. Acesso em: 04 dez 2010.

_____. **Reference Model for Service Oriented Architecture**. 2006a. Disponível em: < http://www.oasisopen.org/committees/download.php/19679/soa-rm-cs.pdf >. Acesso em: 24 mar 2010.

_____. **Universal Business Language V2.0**. 2006b. Disponível em: < http://docs.oasisopen.org/ubl/os-UBL-2.0/UBL-2.0.html >. Acesso em: 05 jan 2010.

_____. **Web Services Business Process Execution Language Version 2.0**. 2007. Disponível em: < http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html >. Acesso em: 23 jun 2010.

_____. **Organization for the Advancement of Structured Information Standards**. 2010. Disponível em: < http://www.oasis-open.org >. Acesso em: 21 fev 2010.

OMG. **UML 2.0 Specification**. 2005. Disponível em: < http://www.omg.org/technology/documents/formal/uml.htm >. Acesso em: 13 mai 2010.

_____. **Business Process Modeling Notation (BPMN)**. 2006. Disponível em: < http://www.bpmn.org >. Acesso em: 25 abr 2010.

OUYANG, C. et al. **Translating BPMN to BPEL**. BPM Center Report BPM-06-02, BPMcenter. org, 2006.

PIAZZA, A. P. **Uma Abordagem para interoperabilidade entre plataformas heterogêneas de serviços web para Redes Colaborativas de Organizações**. 2007. (Dissertação de Mestrado). Programa de Pós-Graduação em Engenharia Elétrica. UFSC

PRESSMAN, R. **Software Engineering: A Practitioner's Approach**. 5 ed. McGraw-Hill Science/Engineering/Math, 2004.

RABELO, R. J.; GUSMEROLI, S. **A Service-oriented platform for collaborative networked organizations**. 8th IFAC Symposium on Cost Oriented Automation Affordable Automation Systems (Cuba). Proceedings, 2007.

ROSETTANET. **Página que contém especificações sobre Processos de Negócios (PIP)**. 2008. Disponível em: < http://www.rosettanet.org/cms/export/sites/default/RosettaNet/Downloads/RStandards/ClustersSegmentsPIPsOverview_10Oct2008.pdf >. Acesso em: 17 fev 2010.

SABATA, B. et al. **Taxonomy for QoS specifications**. 1997. p.100-107.

SCHACH, S. R. **Object-oriented and classical software engineering**. McGraw-Hill Science/Engineering/Math, 2005.

SCHEITHAUER, G.; WIRTZ, G.; TOKLU, C. **Bridging the semantic gap between process documentation and process execution**. The 2008 International Conference on Software Engineering and Knowledge Engineering (SEKE'08), Redwood City, California, USA, 2008.

SHAHZAD, K. et al. **Elicitation of Requirements for a Business Process Model Repository**. Business Process Management Workshops: Bpm 2008 International Workshops, Milano, Italy, September 1-4, 2008, Revised Papers, 2009.

SHAHZAD, K.; ELIAS, M.; JOHANNESSON, P. **Towards Cross Language Process Model Reuse - A Language Independent Representation of Process Models**. The Practice of Enterprise Modeling, p. 176-190, 2009.

SILVA, J. B.; PEZZIN, J. **Usando ontologias na construção de modelos MDA (model-driven architecture)**. IX Fórum de Tecnologia e XVI Seminário Regional de Informática, 2006.

SIMCHI-LEVI, E.; KAMINSKY, P. **Designing and managing the supply chain: concepts, strategies, and case studies**. Irwin/McGraw-Hill, 2003.

SOMMERVILLE, I. **Software Engineering**. 8 ed. Addison Wesley, 2006.

SONG, M.; MILLER, J. A.; ARPINAR, I. B. **REPOX: an XML repository for workflow designs and specification**. Ismailcem Arpinar August, 2001.

SOUZA, A. P. **Um Modelo de Descoberta Dinâmica de Serviços de Software Baseado no Contexto de Processos de Negócios Empresariais e em Qualidade de Serviço**. 2009. (Qualificação de Doutorado). Programa de Pós-Graduação em Engenharia Elétrica. UFSC

TAKAHASHI, T. **Sociedade da Informação no Brasil: Livro Verde**. Ministério de Ciência e Tecnologia, 2000.

UN/CEFACT. **UNECE - United Nations Economic Commission for Europe**. 2010. Disponível em: < http://www.unece.org/cefact >. Acesso em: 15 mai 2010.

VANHATALO, J.; KOEHLER, J.; LEYMANN, F. **Repository for business processes and arbitrary associated metadata**. Proc of BPM, 2006.

W3C. **XML Schema**. 2000. Disponível em: < http://www.w3.org/XML/Schema >. Acesso em: 21 out 2010.

_____. **Extensible Markup Language (XML)**. 2003. Disponível em: < http://www.w3.org/XML >. Acesso em: 13 abr 2010.

_____. **Web Services Architecture (WSA)**. 2004. Disponível em: < http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/ >. Acesso em: 15 ago 2010.

_____. **Simple Object Access Protocol (SOAP) v1.2**. 2007a. Disponível em: < http://www.w3.org/TR/soap12/ >. Acesso em: 13 out 2010.

_____. **Web Services Description Language (WSDL) Specification V2.0**. 2007b. Disponível em: < http://www.w3.org/TR/wsdl20/ >. Acesso em: 21 set 2010.

_____. **World Wide Web Consortium**. 2010. Disponível em: < World Wide Web Consortium >. Acesso em: 12 out 2010.

WANG, W. et al. **A comparison of business process modeling methods**. Service Operations and Logistics, and Informatics, 2006. SOLI'06. IEEE International Conference on, 2007. 2007. p.1136-1141.

WEIDLICH, M. et al. **BPEL to BPMN: the myth of a straight-forward mapping**. On the Move to Meaningful Internet Systems: OTM 2008, p. 265-282, 2008.

WESKE, M. **Business Process Management: Concepts, Languages, Architectures**. Springer-Verlag New York Inc, 2007.

WHITE, S. **Using BPMN to model a BPEL process**. BPTrends**,** v. 3, n. 3, p. 1-18, 2005.

WIKIPEDIA. **The free Encyclopedia**. 2010. Disponível em: < http://en.wikipedia.org >. Acesso em: 05 set 2010.

WS-I. **The Web Services Interoperability Organization**. 2010. Disponível em: < http://www.ws-i.org >. Acesso em: 14 nov 2010.

YAN, Z. et al. **BPMO: Semantic Business Process Modeling and WSMO Extension**. IEEE International Conference on Web Services, 2007. ICWS 2007, 2007. p.1185-1186.

YAN, Z.; DIJKMAN, R.; GREFEN, P. **Business Process Model Repositories - Framework and Survey**. 2009.

YAWL. **YAWL: Yet Another Workflow Language**. 2009. Disponível em: < http://www.yawlfoundation.org >.

ZELKOWITZ, M. V.; WALLACE, D. R. **Experimental models for validating technology**. Computer**,** v. 31, n. 5, p. 23-31, 2002.