

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

**IASWS – ABORDAGEM ITERATIVA PARA
DESENVOLVIMENTO DE *SOFTWARE* UTILIZANDO
*WEB SERVICES***

HIROMITI NAKAGAWA

ORIENTADORA: PROF^a. DR^a. ROSÂNGELA A. D. PENTEADO

Co-ORIENTADOR: PROF. DR. ANTONIO CARLOS DOS SANTOS

São Carlos - SP
Fevereiro/2012

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

**IASWS – ABORDAGEM ITERATIVA PARA
DESENVOLVIMENTO DE *SOFTWARE* UTILIZANDO
*WEB SERVICES***

HIROMITI NAKAGAWA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Engenharia de *Software*.

Orientadora: Dra. Rosângela A. D. Penteado.

Co-Orientador: Antonio Carlos dos Santos.

São Carlos - SP
Fevereiro/2012

**Ficha catalográfica elaborada pelo DePT da
Biblioteca Comunitária da UFSCar**

N163ia Nakagawa, Hiromiti.
IASWS – abordagem iterativa para desenvolvimento de
software utilizando *web services* / Hiromiti Nakagawa. -- São
Carlos : UFSCar, 2014.
164 f.

Dissertação (Mestrado) -- Universidade Federal de São
Carlos, 2012.

1. Software - desenvolvimento. 2. Arquitetura orientada a
serviços. 3. Modelagem de processos de negócios. I. Título.

CDD: 005.1 (20^a)

Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

**“IASWS - Abordagem Iterativa
para Desenvolvimento de Software
utilizando Web Services”**

Hiromiti Nakagawa

Dissertação de Mestrado apresentada ao
Programa de Pós-Graduação em Ciência da
Computação da Universidade Federal de São
Carlos, como parte dos requisitos para a
obtenção do título de Mestre em Ciência da
Computação

Membros da Banca:



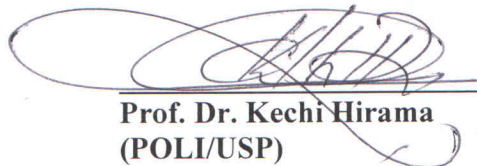
Prof. Dra. Rosângela Ap. Delosso Penteado
(Orientadora - DC/UFSCar)



Prof. Dr. Antonio Carlos dos Santos
(Co-Orientador - DC/UFSCar)



Prof. Dr. Antonio Francisco do Prado
(DC/UFSCar)



Prof. Dr. Kechi Hirama
(POLI/USP)

São Carlos
Março/2012

Dedico este trabalho aos meus pais, Hiromi e Mitie, pelo apoio incondicional, pela confiança e por todos os esforços e sacrifícios realizados para que eu pudesse ter a oportunidade de estudar; aos meus irmãos, Hiroshi, Heiji e Erika; à minha namorada Caroline Perlin pelo apoio, motivação, paciência e compreensão; e aos meus grandes amigos que sempre estiveram presentes nesta jornada.

AGRADECIMENTO

A todos que direta ou indiretamente me ajudaram a concluir este trabalho, gostaria de dizer que sou imensamente grato pelo apoio e ajuda sempre que precisei...

A Deus, por estar sempre presente, principalmente nas horas de dificuldade, por me dar forças para continuar e por cuidar de mim, de minha família e de todas as pessoas a quem quero bem....

Aos meus pais Hiromi e Mitie, pelo amor, pelo cuidado, pelo apoio e por terem batalhado e deixado muitas coisas de lado para me proporcionar a oportunidade de regressar ao Brasil e concluir meus estudos...

Ao meu irmão Hiroshi e sua esposa pelo apoio e compreensão nesta etapa da minha vida...

Aos meus orientadores Profa. Dra Rosângela Penteado e Prof. Dr. Antonio Carlos, pela oportunidade, confiança, paciência e por todo o conhecimento e experiência repassados a mim durante o desenvolvimento deste trabalho...

À Prof. Dr. Rosana Braga e ao Prof. Dr. Antonio Francisco Prado, por sua participação na banca de qualificação e pelos comentários e observações que muito contribuíram para a concretização deste trabalho...

Aos professores Hélio, Sandra, Politano e Takashi e Evélton pelas conversas e pelo apoio...

À minha namorada Caroline Perlin, que sempre esteve presente, apoiando, motivando, ouvindo, participando e ajudando, tanto nas horas boas quanto nas ruins...

Aos meus queridos amigos, Fe, Dani, Tati, Andy, Baby, Cleo, Andre DT, Pintcher, Perninha, Bob Dog, Du Cirilo, Guilherme Freire, Gilmar, Waltão, Vanessa, pela amizade, apoio e paciência...

RESUMO

O desenvolvimento de sistemas de *software* exige ferramentas, tecnologias e processos de desenvolvimento adequados para criar sistemas que atendem às necessidades de negócio do cliente e que sejam suficientemente flexíveis para acompanhar as constantes mudanças e evoluções dos negócios. A Computação Orientada a Serviços (COS) é um paradigma de TI que pode auxiliar nesse desenvolvimento, desde que haja um processo de desenvolvimento que considere as suas peculiaridades. Alguns desses processos foram analisados neste trabalho e observou-se que eles tem o enfoque no desenvolvimento de sistemas compostos exclusivamente por serviços, o que nem sempre é interessante: a) quando se deseja construir a interface gráfica do *software*, b) quando é preciso encapsular a lógica de negócio, que faz o diferencial competitivo entre os produtos, ou c) em empresas que estão iniciando a adoção da COS. Nessas situações a utilização conjunta de elementos da Orientação a Objetos (OO) com princípios, conceitos, tecnologias e técnicas da COS pode trazer benefícios. Dessa forma, nesta dissertação foi criada uma abordagem iterativa de desenvolvimento de *software* que utiliza OO e serviços, denominada IASWS (em inglês, *Iterative Approach for Software Development using Web Services*), tendo como base o modelo incremental; a modelagem de processos de negócio (PN), usando BPMN; XP e o perfil SoaML. A abordagem IASWS tem como diferencial a possibilidade da adoção gradual de serviços e da COS, sendo composta por nove fases: Identificar Requisitos, Contextualizar PN com Serviços, Projetar Serviços, Implementar Serviços, Testar Serviços, Projetar Solução, Implementar Solução, Testar Solução e Verificar Aceitação. No escopo desta dissertação são tratadas as quatro fases iniciais: Identificar Requisitos, Contextualizar PN com Serviços, Projetar Serviços e Projetar Solução as quais são as responsáveis pela: obtenção de requisitos do sistema, análise desses requisitos e elaboração de uma solução, projeto de serviços e projeto da solução. A cada iteração é gerado um incremento no *software*, que inclui a implementação de um ou mais processos de negócio, e que é entregue ao cliente para que esse possa visualizar o avanço no desenvolvimento, ao mesmo tempo em que fornece *feedbacks* sobre o desenvolvimento do sistema. A modelagem de processos de negócio utilizando BPMN, além de contribuir para o entendimento do negócio do cliente, favorece também a identificação dos requisitos. Serviços são representados e especificados utilizando o perfil SoaML que possibilita a geração automatizada do código do serviço na fase de implementação. Dois exemplos de aplicação da abordagem foram desenvolvidos para analisar a aplicabilidade da IASWS.

Palavras-chave: Processo, Desenvolvimento de *Software*, Serviço, Modelagem de Processos, Abordagem, SOA, BPMN, SoaML

ABSTRACT

Software system development demands appropriate tools, technologies and development processes in order to create systems that meet customer's business needs while being flexible enough to cope with business evolution and constant changes. Service-Oriented Computing (SOC) is an IT paradigm that might help on such development as long as a development process is established to address its peculiarities. As part of this work some of these processes were analyzed and it was observed that several of them focus on development of systems composed exclusively by services, which is not always interesting: first when building software's graphical user interface, second when the business logic that gives competitive advantage needs to be encapsulated, or third at enterprises starting the SOC adoption. In these cases a mix of Object-Oriented (OO) elements and SOC principles, concepts, technologies and techniques could be used to provide better results. This work presents an iterative approach to develop *software* that uses OO and services, named IASWS (Iterative Approach for *Software* Development using Web Services) based on the Incremental model. It incorporates business process modeling using BPMN; XP and SoaML profile. This approach differs from the others as it allows services and SOC gradual adoption and is comprised of nine phases: Requirements Identification, Business Process (BP) and Services Contextualization, Service Design, Service Implementation, Service Testing, Solution Design, Solution Implementation, Solution Testing and Acceptance Verification. This dissertation is focused on the four initial phases: Requirements Identification, BP and Services Contextualization, Service Design and Solution Design which are responsible for: system requirements gathering, requirements analysis and solution elaboration, service design and solution design. Software increments are delivered as iterations complete and includes the implementation of one or more business process. Delivering software increments to the customer allows for development progress visualization and provides feedback on what had been implemented. Business process modeling using BPMN contributes to understanding customer's business area and improves requirements identification. Services are modeled and specified using SoaML profile allowing automatic code generation at the implementation phase. Two examples applying approach to development were carried out to investigate the IASWS applicability.

Keywords: Process, Software Development, Service, Process Modeling, Approach, SOA, BPMN, SoaML

LISTA DE FIGURAS

Figura 2.1 Inter-relacionamento dos elementos da Computação Orientada a Serviços (ERL, 2007).....	24
Figura 2.2: Exemplo de arquitetura orientada a serviços (PAPAZOGLU, 2003). ...	25
Figura 2.3: Fases do método (PAPAZOGLU e van den HEUVEL, 2006).....	27
Figura 2.4: Fases do ciclo de vida (ERL, 2005).....	30
Figura 2.5: Atividades da Análise Orientada a Serviços (ERL, 2005).	31
Figura 2.6: Projeto Orientado a Serviços (ERL, 2005).	32
Figura 2.7: Fluxo de trabalho no <i>Rational SOMA 2.9</i> (IBM, 2010b).	33
Figura 2.8: Método de Análise e Projeto Orientado a Serviços (FUGITA, 2009).....	36
Figura 2.9: Elementos básicos do perfil SoaML (OMG, 2009b).	41
Figura 2.10: Exemplo de modelagem de participante e interface.....	41
Figura 2.11: Definição do elemento ServiceContract (OMG, 2009b).	42
Figura 2.12: Elementos <<Attachment>> e <<MessageType>> (OMG, 2009b).	42
Figura 2.13: Definição do elemento <<ServicesArchitecture>> (OMG, 2009b).....	43
Figura 2.14: Relacionamento dos principais elementos do perfil SoaML.	43
Figura 3.1: Objetos de fluxo do BPMN (adaptado de OMG, 2009a).	50
Figura 3.2: Objetos conectores do BPMN (adaptado de OMG, 2009a).	50
Figura 3.3: Exemplo de processo de negócio modelado usando BPMN.....	50
Figura 3.4: Objetos raia do BPMN (OMG, 2009a).....	51
Figura 3.5: Artefatos do BPMN (OMG, 2009a).....	51
Figura 3.6: Representação do subprocesso de devolução segundo a perspectiva de processo privado.	52
Figura 3.7: Representação do processo de devolução segundo a perspectiva de processo abstrato.	52
Figura 3.8: Representação do processo de devolução segundo a perspectiva de processo de colaboração.	52
Figura 3.9: Modelagem do processo de empréstimo de exemplares de biblioteca usando BPMN.	53
Figura 4.1 Ciclo de desenvolvimento da abordagem IASWS.....	58
Figura 4.2 Etapas da Abordagem IASWS.	61

Figura 4.3 Exemplo de Diagrama de Visão Geral de Processos construído com BPMN.....	67
Figura 4.4 Modelo <i>as-is</i> do processo PN1 usando BPMN.	67
Figura 4.5: Modelo <i>to-be</i> do processo PN1.....	69
Figura 4.6 Exemplo de um modelo de análise para o Serviço XY.....	81
Figura 4.7 Exemplo do uso do elemento <i>ServiceInterface</i>	81
Figura 4.8 Exemplo de representação da arquitetura do Serviço XY.....	87
Figura 4.9 Exemplo de especificação de um serviço usando elementos do perfil SoaML.....	89
Figura 4.10 Exemplo de representação da arquitetura de uma solução.	92
Figura 4.11 Exemplo de modelo de dados.	93
Figura 4.12: Exemplo de Diagrama de classes.	95
Figura 5.1 Diagrama de Visão Geral de Processos que serão implementados no SGBi.....	103
Figura 5.2 Modelo <i>as-is</i> do processo de negócio “Cadastrar usuário” do SGBi.....	104
Figura 5.3 Modelo <i>as-is</i> do processo de negócio “Realizar empréstimo de exemplar” do SGBi.....	104
Figura 5.4 Modelo <i>as-is</i> do processo de negócio “Devolver exemplar emprestado” do SGBi.....	104
Figura 5.5 Modelo <i>as-is</i> do processo de negócio “Reservar título não disponível para empréstimo no momento” do SGBi.	105
Figura 5.6 Modelo <i>as-is</i> do processo de negócio “Consultar acervo” do SGBi.	105
Figura 5.7 Modelo <i>as-is</i> do processo de negócio “Cadastrar título” do SGBi.....	105
Figura 5.8 Modelo <i>as-is</i> do processo de negócio “Excluir título” do SGBi.....	105
Figura 5.9 Modelo <i>as-is</i> do processo de negócio “Gerar relatório de empréstimos em com devolução em atraso” do SGBi.....	106
Figura 5.10 Modelo <i>as-is</i> do processo de negócio “Gerar relatório de empréstimos por data” do SGBi.....	106
Figura 5.11 Modelo <i>to-be</i> do processo de negócio “Cadastrar usuário” do SGBi. ..	108
Figura 5.12 Modelo <i>to-be</i> do processo de negócio “Realizar empréstimo de exemplar” do SGBi.....	109
Figura 5.13 Modelo <i>to-be</i> do processo de negócio “Devolver exemplar emprestado” do SGBi.....	109
Figura 5.14 Modelo <i>to-be</i> do processo de negócio “Reservar título não disponível para empréstimo no momento” do SGBi.....	110

Figura 5.15 Modelo <i>to-be</i> do processo de negócio “Consultar acervo” do SGBi.....	110
Figura 5.16 Modelo <i>to-be</i> do processo de negócio “Cadastrar título” do SGBi.	110
Figura 5.17 Modelo <i>to-be</i> do processo de negócio “Excluir título” do SGBi.	111
Figura 5.18 Modelo <i>to-be</i> do processo de negócio “Gerar relatório de empréstimos com devolução em atraso” do SGBi.....	111
Figura 5.19 Modelo <i>to-be</i> do processo de negócio “Gerar relatório de empréstimos por data” do SGBi.....	112
Figura 5.20 Modelo de análise de novos serviços.....	119
Figura 5.21 Arquitetura do serviço “Recuperar Dados Cadastrais” usado no SGBi.	122
Figura 5.22 Arquitetura do serviço “Gerar código de barras” usado no SGBi.	123
Figura 5.23 Especificação do serviço “Recuperar dados cadastrais” para o SGBi.	124
Figura 5.24 Arquitetura da solução para o PN “Cadastrar Usuário” do SGBi.....	125
Figura 5.25 Modelagem de dados para implementação do PN “Cadastrar Usuário” do SGBi.....	126
Figura 5.26 Tabelas que armazenam dados de alunos e funcionários do SGBi.....	126
Figura 5.27 Diagrama de classes para implementação do processo “Cadastrar usuário” do SGBi.....	129
Figura 5.28 Modelagem em BPMN do processo “Atendimento de pacientes” do Sistema de Atendimento da Unidade Saúde Escola.	134
Figura 5.29 Modelo <i>as-is</i> do subprocesso “Cadastrar usuário” do Sistema de Atendimento da Unidade Saúde Escola.....	135
Figura 5.30 Modelo <i>as-is</i> do subprocesso “Consultar cadastro de usuários” do Sistema de Atendimento da Unidade Saúde Escola.....	135
Figura 5.31 Modelo <i>to-be</i> do subprocesso “Cadastrar Usuário” do Sistema de Atendimento da Unidade Saúde Escola.....	136
Figura 5.32 Modelo <i>to-be</i> do subprocesso “Consultar cadastro de usuários” do Sistema de Atendimento da Unidade Saúde Escola.....	136
Figura 5.33 Arquitetura da solução para o PN “Cadastrar Usuário” do Sistema de Atendimento da Unidade Saúde Escola.....	145
Figura 5.34 Modelo de dados do PN “Cadastrar Usuário” do Sistema de Atendimento da Unidade Saúde Escola.....	146
Figura 5.35 Diagrama de classes da solução do Sistema de Atendimento da Unidade Saúde Escola	148

LISTA DE TABELAS

Tabela 2.1: Cenários de reúso possíveis (FUGITA, 2009).....	37
Tabela 2.2: Cenários e estratégias de realização de serviços (FUGITA, 2009).....	37
Tabela 2.3: Estratégias de criação de interfaces de serviço no MAPOS (FUGITA, 2009)	38
Tabela 2.4: Estereótipos do perfil SoaML (OMG, 2009b).....	44
Tabela 2.5: Resumo sobre os métodos descritos neste capítulo.	46
Tabela 4.1: Fases da IASWS versus estágios do ciclo de vida clássico para desenvolvimento de <i>software</i>	59
Tabela 4.2 Visão geral das fases e etapas da IASWS com as respectivas atividades.	62
Tabela 4.3 Artefatos de entrada e saída da etapa “Levantar PN”	66
Tabela 4.4 Artefatos de entrada e saída da etapa “Modelar PN”	68
Tabela 4.5 Artefatos de entrada e saída da etapa “Extrair Requisitos de TI”.....	70
Tabela 4.6 Artefatos de entrada e saída da etapa “Planejar”	71
Tabela 4.7 Artefatos de entrada e saída da etapa “Selecionar PN”	72
Tabela 4.8 Fontes para identificação de serviços.	73
Tabela 4.9 Documentação de Serviços candidatos identificados.....	75
Tabela 4.10 Artefatos de entrada e saída da etapa “Identificar e Buscar Serviços” ..	75
Tabela 4.11 Documentação para serviços.	77
Tabela 4.12 Artefatos de entrada e saída da etapa “Elaborar Proposta de Solução”.	77
Tabela 4.13 Modelo para documentar casos de testes de serviço.....	78
Tabela 4.14 Artefatos de entrada e saída da etapa “Avaliar Serviços Reusados”. ...	79
Tabela 4.15 Artefatos de entrada e saída da etapa “Criar Casos de Teste de Novos Serviços”	80
Tabela 4.16 Artefatos de entrada e saída da etapa “Modelar Novos Serviços”	82
Tabela 4.17 Modelo para documentar casos de testes da solução.....	83
Tabela 4.18 Artefatos de entrada e saída da etapa “Criar Casos de Teste da Solução”.	83

Tabela 4.19 Artefatos de entrada e saída da etapa “Definir API de Implementação de <i>Web Services</i> ”.....	84
Tabela 4.20 Artefatos de entrada e saída da etapa “Definir Estratégia de Desenvolvimento”.....	86
Tabela 4.21: Artefatos de entrada e saída da etapa “Definir Arquitetura do Serviço”.....	88
Tabela 4.22: Artefatos de entrada e saída da etapa “Especificar Serviço”.....	89
Tabela 4.23: Artefatos de entrada e saída da etapa “Verificar Serviço”.....	90
Tabela 4.24: Artefatos de entrada e saída da etapa “Definir Arquitetura da Solução”.....	92
Tabela 4.25: Artefatos de entrada e saída da etapa “Criar Modelo de Dados”.....	93
Tabela 4.26: Artefatos de entrada e saída da etapa “Especificar Componentes da Solução”.....	95
Tabela 4.27: Comparação da abordagem IASWS com outros métodos e processos.....	96
Tabela 5.1 Informações coletadas com o cliente.....	100
Tabela 5.2 Atividades dos processos de negócio do cliente.....	101
Tabela 5.3 Informações dos processos de negócio para o SGBi.....	102
Tabela 5.4 <i>Backlog</i> de Processos de Negócio do SGBi com a respectiva estimativa de tempo.....	112
Tabela 5.5 Serviços candidatos identificados para o SGBi.....	114
Tabela 5.6 Serviço que fornece a funcionalidade “Recuperar dados cadastrais” para o SGBi.....	115
Tabela 5.7 Serviço que fornece a funcionalidade “Gerar código de barras” para o SGBi.....	115
Tabela 5.8 Informações do serviço de validação de CPF “ <i>CPF Service</i> ” para o SGBi.....	116
Tabela 5.9 Informações do serviço de validação de endereços de email “ <i>Tower Data Email Validation Service</i> ” para o SGBi.....	116
Tabela 5.10 Casos de teste para o serviço de validação de CPF “ <i>CPF Service</i> ” do SGBi.....	117
Tabela 5.11 Casos de teste para o serviço de validação de endereços de email “ <i>Tower Data Email Validation Service</i> ” do SGBi.....	117
Tabela 5.12 Casos de teste para serviço “Recuperar dados cadastrais” do SGBi.....	118
Tabela 5.13 Casos de teste para serviço “Gerar código de barras” do SGBi.....	118
Tabela 5.14 Casos de teste para o PN “Cadastrar usuário” do SGBi.....	120

Tabela 5.15 Informações coletadas com o cliente sobre a Unidade Saúde Escola.	131
Tabela 5.16 Atividades da operação de negócio “Atendimento de pacientes” da USE.	132
Tabela 5.17 Informações dos processos de negócio que serão informatizados pelo Sistema de Atendimento da Unidade Saúde Escola.....	133
Tabela 5.18 <i>Backlog</i> de Subprocessos de Negócio e respectiva estimativa de tempo do Sistema de Atendimento da Unidade Saúde Escola.....	137
Tabela 5.19 Serviços candidatos identificados para o Sistema de Atendimento da Unidade Saúde Escola.....	139
Tabela 5.20 Informações do serviço “Torpedo SMS <i>Service</i> ” para o Sistema de Atendimento da Unidade Saúde Escola.....	139
Tabela 5.21 Informações do serviço de envio de emails “ <i>Amazon Simple Email Service</i> ” para o Sistema de Atendimento da Unidade Saúde Escola...	140
Tabela 5.22 Informações do serviço “ <i>CEP Service</i> ” para o Sistema de Atendimento da Unidade Saúde Escola.....	140
Tabela 5.23: Informações do serviço de validação de endereços de email “ <i>Tower Data Email Validation Service</i> ” para o Sistema de Atendimento da Unidade Saúde Escola.....	140
Tabela 5.24 Casos de teste para o serviço “Torpedo SMS <i>Service</i> ” para o Sistema de Atendimento da Unidade Saúde Escola.....	142
Tabela 5.25 Casos de teste para o serviço “ <i>CEP Service</i> ” para o Sistema de Atendimento da Unidade Saúde Escola.....	143
Tabela 5.26 Casos de teste para o serviço “ <i>Amazon Simple Email Service</i> ” para o Sistema de Atendimento da Unidade Saúde Escola.....	143
Tabela 5.27 Casos de teste para o serviço “ <i>Tower Data Email Validation Service</i> ” para o Sistema de Atendimento da Unidade Saúde Escola.....	143
Tabela 5.28 Casos de teste para o PN “Cadastro de Usuário” do Sistema de Atendimento da Unidade Saúde Escola.....	144

LISTA DE ABREVIATURAS E SIGLAS

- API** - *Application Programming Interface*
- AWS** - *Amazon Web Services*
- BPM** - *Business Process Management*
- BPMN** - *Business Process Model and Notation*
- CASE** - *Computer-Aided Software Engineering*
- CBD** - *Component-Based Development*
- CBSE** – *Component-Based Software Engineering*
- CEP** – *Código de Endereçamento Postal*
- COTS** - *Commercial Off-The-Shelf*
- GUI** - *Graphical User Interface*
- HTML** - *Hypertext Markup Language*
- HTTP** - *Hypertext Transfer Protocol*
- IASWS** – *Iterative Approach for Software Development using Web Services*
- IDE** - *Integrated Development Environment*
- ISBN** - *International Standard Book Number*
- JET** - *Java Emitter Templates*
- KPI** - *Key Performance Indicator*
- MAPOS** - *Método de Análise e Projeto Orientado a Serviços*
- OASIS** - *Organization for the Advancement of Structured Information Standards*
- OMG** - *Object Management Group*
- PN** – *Processo de Negócio*
- ROI** - *Return On Investment*
- RSA** – *Rational Software Architect*
- RUP** - *Rational Unified Process*
- SADT** - *Structured Analysis and Design Technique*
- SDK** - *Software Development Kit*
- SGBi** – *Sistema Gerenciador de Biblioteca*
- SLA** - *Service Level Agreement*
- SMS** – *Short Message Service*
- SOA** - *Service-Oriented Architecture*

SoaML – *Service-Oriented Architecture Modeling Language*
SOAP - *Simple Object Access Protocol*
SOMA - *Service-Oriented Modeling and Architecture*
TI - *Tecnologia da Informação*
UDDI – *Universal Description, Discovery and Integration*
UML - *Unified Modeling Language*
URI - *Uniform Resource Identifier*
USE – *Unidade Saúde Escola*
W3C - *World Wide Web Consortium*
WS-BPEL - *Web Services Business Process Execution Language*
WSDL - *Web Services Description Language*
XML - *eXtensible Markup Language*
XP – *eXtreme Programming*
XSD - *XML Schema Definition*
YDN – *Yahoo! Developer Network*

SUMÁRIO

CAPÍTULO 1 - INTRODUÇÃO.....	14
1.1 Contextualização.....	14
1.2 Motivação.....	15
1.3 Objetivos e Resultados esperados.....	17
1.4 Metodologia.....	17
1.5 Organização da Dissertação.....	18
CAPÍTULO 2 - DESENVOLVIMENTO DE SOFTWARE ORIENTADO A SERVIÇOS	19
2.1 Considerações Iniciais.....	19
2.2 Computação Orientada a Serviços.....	22
2.3 Processos de Desenvolvimento de <i>Software</i> Orientado a Serviços.....	26
2.3.1 Método proposto por Papazoglou e van den Heuvel.....	26
2.3.2 Método proposto por Erl.....	30
2.3.3 IBM Rational Service-Oriented Modeling and Architecture (SOMA).....	33
2.3.4 Método de Análise e Projeto Orientado a Serviços - MAPOS.....	35
2.4 Perfil SoaML para Modelagem de Serviços.....	39
2.5 Considerações Finais.....	45
CAPÍTULO 3 - MODELAGEM DE PROCESSOS DE NEGÓCIO.....	47
3.1 Considerações Iniciais.....	47
3.2 Business Process Model and Notation (BPMN).....	49
CAPÍTULO 4 - IASWS- ABORDAGEM ITERATIVA PARA DESENVOLVIMENTO DE SOFTWARE UTILIZANDO WEB SERVICES.....	56
4.1 Considerações Iniciais.....	56
4.2 Abordagem IASWS.....	57
4.3 Fases de Requisitos, Análise e Projeto da Abordagem Iterativa para Desenvolvimento de <i>Software</i> utilizando <i>Web Services</i> - IASWS.....	60
4.3.1 Fase 1: Identificar Requisitos.....	65
4.3.1.1 Etapa 1.1: Levantar PN.....	65

4.3.1.2 Etapa 1.2: Modelar PN	66
4.3.1.3 Etapa 1.3: Extrair Requisitos de TI	68
4.3.1.4 Etapa 1.4: Planejar	70
4.3.2 Fase 2: Contextualizar PN com Serviços	71
4.3.2.1 Etapa 2.1: Selecionar PN	72
4.3.2.2 Etapa 2.2: Identificar e Buscar Serviços	72
4.3.2.3 Etapa 2.3: Elaborar Proposta de Solução	75
4.3.2.4 Etapa 2.4: Avaliar Serviços Reusados	77
4.3.2.5 Etapa 2.5: Criar Casos de Teste de Novos Serviços	79
4.3.2.6 Etapa 2.6: Modelar Novos Serviços	80
4.3.2.7 Etapa 2.7: Criar Casos de Teste da Solução	82
4.3.3 Fase 3: Projetar Serviços	83
4.3.3.1 Etapa 3.1: Definir API de Implementação de <i>Web Services</i>	84
4.3.3.2 Etapa 3.2: Definir estratégia de desenvolvimento	85
4.3.3.3 Etapa 3.3: Definir Arquitetura dos Serviços	86
4.3.3.4 Etapa 3.4: Especificar Serviço.....	88
4.3.3.5 Etapa 3.5: Verificar Serviço	90
4.3.4 Fase 4: Projetar Solução	90
4.3.4.1 Etapa 4.1: Definir Arquitetura da Solução	91
4.3.4.2 Etapa 4.2: Criar Modelo de Dados	92
4.3.4.3 Etapa 4.3: Especificar Componentes da Solução	94
4.4 Considerações Finais	95
CAPÍTULO 5 - EXEMPLOS DE APLICAÇÃO DA IASWS.....	98
5.1 Considerações Iniciais.....	98
5.2 Exemplo 1: Sistema Gerenciador de Biblioteca	99
5.2.1 Primeira iteração	99
5.2.1.1 Fase 1: Identificar Requisitos	99
5.2.1.2 Fase 2: Contextualizar PN com Serviços	113
5.2.1.3 Fase 3: Projetar serviços.....	121
5.2.1.4 Fase 4: Projetar solução.....	124
5.3 Exemplo 2: Sistema de Atendimento da Unidade Saúde Escola	130
5.3.1 Primeira iteração	131
5.3.1.1 Fase 1: Identificar Requisitos	131

5.3.1.2 Fase 2: Contextualizar PN com Serviços	137
5.3.1.3 Fase 3: Projetar serviços.....	144
5.3.1.4 Fase 4: Projetar solução.....	145
5.4 Considerações Finais	149
CAPÍTULO 6 - CONCLUSÕES.....	152
6.1 Resultados Obtidos	153
6.2 Limitações do Trabalho	155
6.3 Contribuições	157
6.4 Trabalhos Futuros	158
REFERÊNCIAS.....	160

Capítulo 1

INTRODUÇÃO

1.1 Contextualização

A Tecnologia da Informação (TI) (LEAVITT e WHISLER, 1958), quando aplicada adequadamente aos negócios, visa prover agilidade na realização de negócios, sendo importante para obtenção de melhores resultados de negócios em empresas das mais diversas áreas de atuação (DAHMAN, CHAROY e GODART, 2011). Sua presença nos negócios pode ser notada desde a utilização de simples planilhas eletrônicas, passando por sistemas que automatizam o gerenciamento de recursos até os que permitem automatizar completamente o fornecimento de serviços. Muitas tarefas, antes realizadas manualmente, agora são realizadas com o auxílio de um ou mais elementos de TI como computadores, sistemas de *software*, dispositivos móveis, internet, comércio eletrônico, entre outros. Dessa forma, há a possibilidade de aumentar a produtividade, agilizar a execução de tarefas envolvidas nos negócios, reduzir custos, entre outros benefícios, o que tornam muitos desses elementos de TI fundamentais para a realização e continuidade dos negócios.

Nesse contexto, sistemas de *software* são os elementos da TI dotados de “inteligência” para execução da lógica envolvida na realização dos negócios, fazendo parte, portanto, da grande maioria das operações de negócio, tornando essencial a capacidade de desenvolver *software* ou adaptá-lo para aproveitar novas oportunidades e responder às pressões competitivas (SOMMERVILLE, 2007). Dessa forma, desenvolvimento rápido e entrega rápida tornam-se requisitos críticos para o desenvolvimento de sistemas de *software*, fazendo com que muitas empresas optem

por equilibrar qualidade e compromisso com requisitos de *software* com a entrega rápida (SOMMERVILLE, 2007).

A flexibilidade de um sistema de *software* é outro requisito importante para seu sucesso, pois permite que esses sistemas acompanhem as mudanças constantes nos negócios, sem tornar essa tarefa custosa ou problemática.

A Computação Orientada a Serviços (COS) (PAPAZOGLU, 2003; HUHNS e SINGH, 2005) é um paradigma de TI que utiliza serviços como elementos fundamentais no desenvolvimento de sistemas flexíveis e alinhados às necessidades de negócio. Serviços são definidos como entidades autônomas e independentes de plataforma que podem ser descritos, publicados, descobertos e fracamente acoplados das mais variadas maneiras para compor sistemas (PAPAZOGLU *et al.*, 2007). Tecnologias orientadas a serviço fornecem a base tecnológica para a implementação desses sistemas, enquanto técnicas estabelecem práticas para utilização dessas tecnologias em harmonia com os conceitos da orientação a serviços.

Embora existam recursos tecnológicos para tratar os desafios do desenvolvimento de *software*, é importante que o desenvolvimento siga um processo pré-estabelecido, de modo a possibilitar o uso adequado de tecnologias, técnicas e interações entre as pessoas envolvidas (SCACCHI, 2001), evitando que os recursos não sejam usados minimamente ou em demasia e minimizando as chances de ocorrer problemas durante o desenvolvimento.

1.2 Motivação

Os modelos de processos de desenvolvimento prescritivos (Cascata {Royce, 1970}, Incremental {BASILI e TURNER, 1975}, Espiral {BOEHM, 1988}, CBSE {BROWN e WALLNAU, 1996}, etc.) em decorrência da época em que foram criados, não se preocuparam com o desenvolvimento e reúso de serviços e com as peculiaridades da Computação Orientada a Serviços (PAPAZOGLU, 2003; HUHNS e SINGH, 2005). Papazoglou *et al.* (2007) afirmam que essas abordagens não podem ser aplicadas diretamente ao desenvolvimento orientado a serviços, pois não

tratam requisitos como serviços, fluxos de informações e componentes que realizam serviços.

Por outro lado, processos e métodos para desenvolvimento de sistemas orientados a serviço (FUGITA, 2009; RAMOLLARI, DRANIDIS e SIMONS, 2007) definem técnicas para o reúso e desenvolvimento de serviços. Esses processos tem o foco na construção de sistemas compostos exclusivamente por serviços, pois consideram que a funcionalidade de um sistema pode ser fornecida exclusivamente por meio deles. No entanto, alguns componentes de um sistema possuem características fortemente atreladas ao domínio e específicas do sistema, como é o caso de componentes de interface com o usuário. Existem ainda aqueles componentes que encapsulam a lógica de negócio, que constitui o diferencial competitivo do sistema. Nesses casos a utilização de serviços não traria vantagens, visto que esses serviços seriam específicos para o sistema ou não poderiam ser expostos (para evitar a perda da vantagem competitiva), ou seja, não se faz interessante compartilhar com outras empresas a lógica que provê o diferencial competitivo.

Em situações como essas, a utilização conjunta de elementos consagrados do desenvolvimento Orientado a Objetos (OO) (BOOCH, 1986) aliada a conceitos, tecnologias e técnicas da Computação Orientada a Serviços pode amenizar essas dificuldades enfrentadas no desenvolvimento de soluções totalmente orientadas a serviço. Além disso, Haines e Rothenberger (2010) afirmam que algumas organizações que adotaram a arquitetura orientada a serviços acabaram se decepcionando rapidamente, possivelmente pelo fato da adoção integral da orientação a serviços ser uma tarefa complexa que requer: mudanças nas técnicas de desenvolvimento, nas ferramentas de desenvolvimento, no conhecimento dos desenvolvedores e no comportamento das pessoas. Os autores ainda citam que essas organizações não se prepararam adequadamente em termos de processos e metodologias para a adoção da arquitetura orientada a serviços.

Dessa forma, diante do panorama exposto infere-se que estabelecer uma abordagem que permita o desenvolvimento de *software* que utilize OO e serviços, que favoreça tanto o reúso de serviços quanto ao desenvolvimento deles e que permita incorporação gradual da orientação a serviços é uma forma de lidar com algumas das considerações feitas anteriormente e é a motivação para este trabalho.

Além disso, não foram encontradas na literatura abordagens que cuidem do desenvolvimento de sistemas OO com o reúso de serviços de modo iterativo.

1.3 Objetivos e Resultados esperados

O principal objetivo desta dissertação é elaborar uma abordagem que viabilize o desenvolvimento de *software* OO com a utilização de serviços. Essa abordagem deverá ser flexível para favorecer o reúso e desenvolvimento de serviços e para viabilizar a adoção gradual de serviços e dos conceitos da COS.

Ao desenvolver essa abordagem, esperam-se obter os seguintes benefícios:

- Permitir incorporação gradual de serviços às soluções conforme a experiência da equipe de desenvolvimento é aprimorada;
- Incorporação gradual dos conceitos da COS, permitindo que equipe de desenvolvimento e cliente se adaptem e ganhem confiança na COS;
- Possibilitar o reúso de serviços em nível de análise;
- Identificar serviços reusáveis no contexto da funcionalidade do sistema;
- Identificar e especificar novos serviços;
- Utilizar a modelagem de processos de negócio (PN) para aproximar a área de negócios da área de TI; e
- Possibilitar entregas constantes de pequenos conjuntos de funções do sistema.

1.4 Metodologia

A metodologia aplicada para o desenvolvimento desta dissertação seguiu as seguintes etapas:

- Realização de estudo dos modelos tradicionais de desenvolvimento de *software*, para analisar as etapas existentes e avaliar a sua reutilização na construção da proposta desta dissertação;

- Realização de estudo de processos e métodos para desenvolvimento de sistemas orientados a serviço, para avaliar as atividades inerentes ao desenvolvimento e reúso de serviços e identificar suas melhores práticas;
- Proposição da abordagem com base em análise dos estudos realizados;
- Aplicação da abordagem no desenvolvimento de um sistema gerenciador de biblioteca com o intuito realizar observações sobre seu comportamento no desenvolvimento de um sistema;
- Refinamento da abordagem conforme observações e análises realizadas durante o desenvolvimento de sistemas hipotéticos; e
- Aplicação da abordagem, nas etapas de levantamento de requisitos, análise e projeto, em um sistema hipotético, de atendimento de uma Unidade de Saúde Escola em uma Universidade, a fim de verificar a sua aplicabilidade.

1.5 Organização da Dissertação

Esta dissertação está organizada em cinco outros capítulos, além das referências e deste, no qual foram contextualizados o problema, os objetivos e a motivação para realização do trabalho.

No Capítulo 2 apresenta-se parte do referencial teórico utilizado neste trabalho: a Computação Orientada a Serviços, o perfil SoaML para modelagem de sistemas orientados a serviços e quatro processos de desenvolvimento de *software* orientado a serviço.

O Capítulo 3 aborda outra parte teórica utilizada: os conceitos para modelagem de processos de negócio e uma breve descrição da linguagem BPMN para representação de processos, seus elementos e características.

No Capítulo 4 é descrita a abordagem iterativa para desenvolvimento de *software* utilizando *web services* (IASWS), com suas fases e atividades.

No Capítulo 5 são descritos dois exemplos hipotéticos de aplicação da abordagem IASWS, utilizados para auxiliar no refinamento e aplicação da abordagem.

Por fim, o Capítulo 6 apresenta os resultados obtidos, as contribuições, as limitações existentes e algumas sugestões para trabalhos futuros.

Capítulo 2

DESENVOLVIMENTO DE *SOFTWARE* ORIENTADO A SERVIÇOS

2.1 Considerações Iniciais

Nos dias atuais, sistemas de *software* desempenham um importante papel nos negócios, na ciência, na engenharia, nos transportes, na medicina, nas telecomunicações, nas indústrias, no entretenimento e em diversas outras áreas, tendo se tornado vital em muitas delas. Pressman (2006) afirma que o advento do *software*: tornou possível o surgimento de novas tecnologias (como engenharia genética); permitiu a extensão de tecnologias existentes (como telecomunicações); foi a força motriz por trás da revolução dos computadores pessoais; possibilitou o surgimento de uma vasta rede guiada por *software*, chamada internet, que modificou drasticamente a relação entre pessoas e computadores; e iniciou o declínio de tecnologias antigas (como tipografia). Com o aumento da dependência de sistemas de *software*, tanto o desenvolvimento quanto a manutenção desses sistemas se tornam tarefas críticas e, frequentemente, complexas. Assim, há um esforço contínuo da comunidade de *software* no desenvolvimento de tecnologias, técnicas e processos que facilitem o desenvolvimento e manutenção de sistemas de *software*, tornando essas tarefas mais rápidas e menos dispendiosas. A existência de um processo de desenvolvimento adequado e adaptado às tecnologias adotadas e às características da equipe de desenvolvimento (ou preferencialmente às práticas

definidas pela empresa) favorece a produção de *software* de qualidade, no prazo e custos estimados e que atende às expectativas do cliente.

Processo de desenvolvimento de *software*, ciclo de vida de *software* e processo de *software* são alguns dos termos utilizados para referenciar o fluxo de trabalho (atividades, ações e tarefas) necessários para o desenvolvimento de um *software* (LANE, BUCCHIARONE, RICHARDSON, 2011). Esse fluxo pode ser completamente *ad hoc* (ausência de processo) ou seguir um modelo predefinido com atividades que devem ser realizadas durante o desenvolvimento do *software*. Esse modelo é uma representação abstrata de um processo de *software*, que explica a abordagem (estratégia) utilizada para desenvolver o *software* e que pode ser considerado um *framework* de processo (SOMMERVILLE, 2007). O primeiro modelo de processo proposto foi o Cascata ou Linear (ROYCE, 1970) que prescreve uma abordagem sistemática e sequencial para o desenvolvimento de *software* (PRESSMAN, 2010). Como seu próprio nome sugere, o fluxo de trabalho proposto por esse modelo é estritamente linear, limitando seu uso em situações nas quais os requisitos estão bem compreendidos ou não sofrem alterações. Para resolver essa deficiência, o modelo Incremental (BASILI e TURNER, 1975) propõe uma estratégia de desenvolvimento iterativa, gerando incrementos no *software* (agregando novas funções ou recursos ao *software*) a cada iteração. Cada incremento é desenvolvido por meio da aplicação de sequências lineares de atividades a conjuntos de requisitos que estejam bem definidos ou compreendidos (PRESSMAN, 2010). Uma das principais vantagens desse modelo em relação ao Cascata é a possibilidade de receber *feedbacks* do cliente à medida que os incrementos vão sendo entregues, permitindo: corrigir defeitos no *software*; refinar o entendimento dos requisitos; e dar ao cliente a noção de progresso no desenvolvimento do *software*. Outra estratégia de desenvolvimento é proposta pelo modelo Evolucionário, tal como Espiral (BOEHM, 1988) e Prototipação Evolutiva (HEKMATPOUR, 1987), nos quais o desenvolvimento de *software* é abordado de maneira iterativa e gradual, evoluindo o *software* com o passar do tempo e permitindo que versões cada vez mais completas sejam desenvolvidas a cada iteração.

O modelo CBSE (em inglês, *Component-Based Software Engineering*) adota a filosofia “compre, não construa” (BROOKS JR, 1987), enfatizando o desenvolvimento de *software* por meio da composição de sistemas de *software*. A integração de componentes assume posição de destaque, antes ocupada pela

implementação de funções (CLEMENTS, 1995). Fazem parte desse modelo a Engenharia de Domínio, que tem por objetivo a identificação, o desenvolvimento, a catalogação e a disseminação de componentes de *software* que podem ser utilizados em *softwares* de um determinado domínio de aplicação (PRESSMAN, 2006), e o Desenvolvimento de Aplicações que criam sistemas a partir da integração de componentes.

Recentemente, processos ágeis para desenvolvimento de *software* tem se destacado cada vez mais, sendo o XP (em inglês *eXtreme Programming*) (BECK, 1999) e o Scrum (SCHWABER e BEEDLE, 2001) os principais processos utilizados no mercado, pois focam no desenvolvimento do *software* ao invés de projeto e documentação, permitindo que sejam adequados para o desenvolvimento de aplicações nas quais os requisitos mudam rapidamente. Beck (1999) estabelece que o XP baseia-se em quatro valores fundamentais: comunicação, simplicidade, *feedback* e coragem, traduzidos em cinco princípios básicos: *feedback* rápido, manter simplicidade, mudanças incrementais, aceitar mudanças e trabalho de qualidade. Algumas práticas são definidas, sendo as seguintes relevantes ao desenvolvimento desta dissertação: planejamento rápido de *releases* (entregas) combinando prioridades de negócio e estimativas técnicas; pequenas versões; estória simples utilizada como metáfora para guiar o desenvolvimento; e refatoração de código.

No processo Scrum os interessados (cliente, usuários, fornecedores) no sistema criam uma visão geral do sistema a ser desenvolvido. A partir daí, um planejamento é realizado, gerando o *Product Backlog* que contém os requisitos funcionais e não funcionais que irão satisfazer as funções vislumbradas/desejadas. O *Product Backlog* é priorizado com base no valor de negócio de cada função e dividido em propostas de versões a serem entregues. Essas versões são desenvolvidas em *sprints* que são iterações com duração máxima de trinta dias consecutivos. Durante um *sprint*, a própria equipe de desenvolvimento (denominada equipe Scrum) é responsável por gerenciar seu trabalho. Reuniões rápidas são realizadas diariamente com o objetivo de identificar dificuldades no desenvolvimento, propor correções de rumo e avaliar o comprometimento da equipe.

As características dos modelos e processos de desenvolvimento citados anteriormente são referências para a elaboração de novos processos de desenvolvimento de *software* específicos para um domínio de aplicação e adequado

às peculiaridades encontradas no desenvolvimento desse tipo de *software* (equipe, requisitos, documentação, etc.), ou ainda, para permitir a utilização de novas tecnologias e paradigmas de desenvolvimento como a Computação Orientada a Serviços (COS), que visa aumentar a flexibilidade dos sistemas.

Neste capítulo são apresentados os principais conceitos da Computação Orientada a Serviços, alguns métodos/processos de desenvolvimento e uma linguagem de modelagem para desenvolvimento de sistemas utilizando serviços. Na Seção 2.2 os conceitos relacionados à Computação Orientada a Serviços são descritos; na Seção 2.3 são descritos quatro métodos de desenvolvimento de sistemas orientados a serviço; na Seção 2.4 é apresentada a linguagem de modelagem utilizada para projetar sistemas de *software* orientados a serviço; e por fim na Seção 2.5 estão as considerações finais.

2.2 Computação Orientada a Serviços

A Computação Orientada a Serviços (COS) é um paradigma computacional que utiliza serviços para apoiar o desenvolvimento rápido, a custos reduzidos, de sistemas de *software* altamente distribuídos, interoperáveis e flexíveis (PAPAZOGLU *et al.*, 2007). Nesse contexto, serviços são elementos computacionais independentes de plataforma e autodescritivos, que constituem a unidade fundamental que fornece funcionalidade em um sistema de *software* (solução) orientado a serviços (PAPAZOGLU, 2003), sendo, portanto, a unidade de réuso. Serviços materializam uma estratégia de programação “orientada a serviços” baseada na ideia de compor sistemas de *software* que encontram e invocam serviços, via rede, para realizar tarefas ou fornecer as funções necessárias (PAPAZOGLU, 2007). Para tanto, esses serviços devem: ser tecnologicamente neutros e adotar padrões amplamente aceitos; possuir baixo acoplamento; e possuir transparência de localidade para possibilitar que possam ser invocados independentemente de sua localização.

Por meio da computação orientada a serviços, organizações e departamentos podem expor suas competências primárias (*core business*) utilizando a *internet* ou *intranet*, linguagens e protocolos padronizados e padrões abertos, possibilitando

interoperabilidade desses serviços em qualquer plataforma. Papazoglou (2003) afirma que serviços constituem a próxima evolução na computação distribuída, fornecendo uma infraestrutura computacional distribuída para integração e colaboração de sistemas de *software* tanto intra quanto extraempresa.

Erl (2007) cita que a computação orientada a serviços incorpora conceitos, tecnologias, *frameworks*, princípios e padrões de projeto, linguagens de padrões e um modelo arquitetural diferenciado, em seus componentes: Serviços, Paradigma Orientado a Serviços (Orientação a Serviços), Solução Orientada a Serviços, Composições de Serviços, Repositório de Serviços e Arquitetura Orientada a Serviços (SOA, em inglês *Service-Oriented Architecture*). A SOA estabelece um modelo arquitetural no qual Serviços constituem o principal elemento na elaboração de Soluções Orientadas a Serviço com foco na melhoria da eficiência, agilidade e produtividade de uma empresa (ERL, 2007), sendo projetada para apoiar a criação, execução e evolução das Soluções Orientadas a Serviço, compostas por Serviços e Composições de Serviços. Serviços são projetados, modelados e implementados seguindo os princípios da Orientação a Serviço, sendo publicados e disponibilizados para uso em um Repositório de Serviços. Na Figura 2.1 são exibidos os relacionamentos entre esses elementos.

Erl (2007) define Orientação a Serviços como um paradigma de projeto que estabelece uma abordagem para projetar a lógica da solução de um problema, de forma distribuída e baseada na teoria de separação de assuntos (em inglês *separation of concerns*), que preconiza que um problema deve ser atacado segundo uma estratégia de decomposição do problema em unidades menores, possibilitando a construção de uma lógica de solução particionada na qual pequenas unidades de lógica de solução, encapsuladas sob a forma de serviços, endereçam partes do problema. Os seguintes princípios constituem a Orientação a Serviços (ERL, 2007):

- Contrato de serviços padronizado: considerações a respeito do projeto da interface do serviço utilizada para: expor suas funções; definir os tipos e modelos de dados utilizados; estabelecer as políticas para utilização do serviço; avaliar a natureza e quantidade de informações expostas por meio da interface;
- Baixo acoplamento de serviços: redução da dependência entre o contrato de serviço, sua implementação e seus consumidores, possibilitando que a

lógica de negócio e sua implementação possam ser alterados sem afetar o contrato de serviço e consumidores desse serviço;

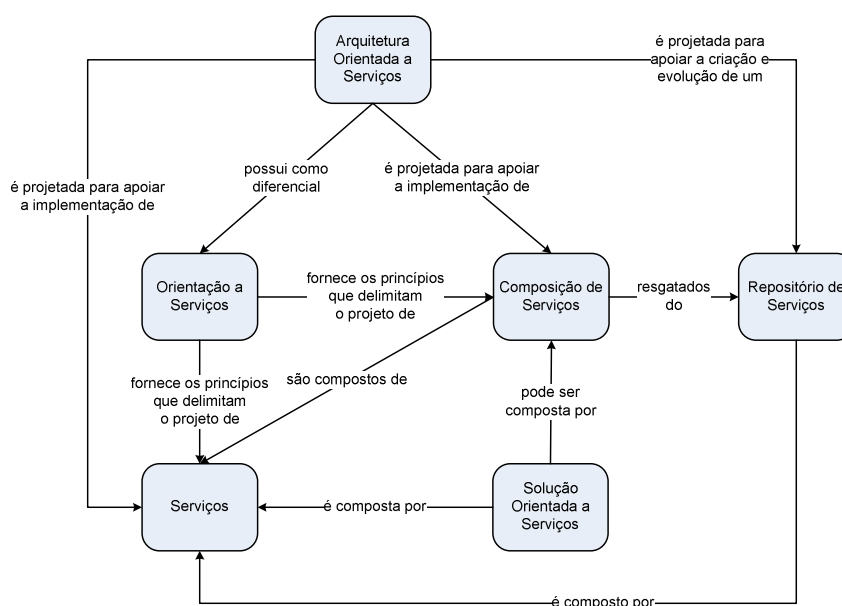


Figura 2.1 Inter-relacionamento dos elementos da Computação Orientada a Serviços (ERL, 2007).

- **Abstração de serviços:** detalhes inerentes à implementação dos serviços são ocultados por meio do uso adequado de contratos de serviço, favorecendo o baixo acoplamento de serviços;
- **Reusabilidade de serviços:** definir funções do serviço e projetá-lo de forma a maximizar seu reuso, nos mais diversos contextos de implementação de sistemas;
- **Autonomia de serviços:** garantir que a implementação do serviço possui controle adequado sobre seu ambiente computacional e recursos necessários, para a correta execução da lógica de solução que fornece a funcionalidade do serviço;
- **Service statelessness:** serviços devem ser projetados para evitar o armazenamento de informações de estado, sob risco de comprometer a disponibilidade, desempenho e escalabilidade do serviço;
- **Habilidade de descoberta de serviços:** facilitar a identificação e descoberta de serviços e o entendimento de suas funções, quando houver oportunidades de reuso, evitando a necessidade de implementar a funcionalidade e aumentando o ROI (em inglês, *Return on Investment*);

- Habilidade de composição de serviços: projetar serviços de forma a evitar a necessidade de adaptação desses serviços quando houver necessidade de reusá-lo em composições de serviços que fornecem soluções para problemas específicos.

O Repositório de Serviços é uma entidade padronizada, controlada de forma independente e responsável pelo catálogo de serviços disponíveis em uma empresa, desempenhando papel importante em uma Arquitetura Orientada a Serviços. Nesse repositório são documentadas informações sobre o serviço e como acessá-lo, constituindo o ponto de partida para a elaboração de Soluções Orientadas a Serviço.

Na Figura 2.2 é apresentado um exemplo de uma arquitetura orientada a serviços constituída por provedores de serviço, consumidores de serviços e um repositório de serviços, que se comunicam por meio de mensagens. Provedores de serviços fornecem as implementações de serviços que podem ser invocados. O repositório de serviços é responsável pela centralização de informações dos serviços disponíveis. Consumidores de serviços são sistemas de *software* que utilizam as funcionalidades fornecidas por serviços publicados no repositório de serviços. Provedores de serviços publicam as descrições dos serviços fornecidos por eles no repositório de serviços, que é consultado por consumidores de serviços em busca de implementações para funções necessárias ao sistema de *software*. Uma vez que o consumidor encontrou o serviço desejado e obteve sua descrição, é possível invocá-lo por meio do envio de mensagens, respeitando o contrato de serviços. A principal tecnologia utilizada para implementação de serviços atualmente é a tecnologia *Web Services* (W3C, 2004a), que engloba vários padrões adotados na indústria e apoiados por comunidades de desenvolvimento e fornecedores de *software* (ERL, 2007).

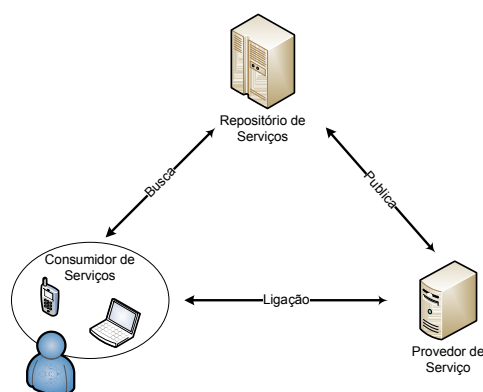


Figura 2.2: Exemplo de arquitetura orientada a serviços (PAPAZOGLU, 2003).

O sucesso na adoção da COS para o desenvolvimento de sistemas de *software* está intimamente relacionado à governança de SOA que fornece uma estrutura para priorizar e apoiar os objetivos de negócio da empresa em nível estratégico, funcional e operacional. O modelo de governança define “o que fazer”, “como fazer”, “quem deve fazer” e “como medir os resultados”, estabelecendo regras, processos, métricas e elementos organizacionais necessários para planejar, tomar decisões, ajustar a direção e controlar o comprometimento com a COS para satisfazer as necessidades do negócio (BIEBERSTEIN *et al.*, 2005). Os processos e métodos para desenvolvimento de *software* orientado a serviços descritos neste capítulo podem ser considerados parte de um modelo de governança, pois definem atividades, tarefas e ações que norteiam o desenvolvimento dessas soluções de *software*.

2.3 Processos de Desenvolvimento de Software Orientado a Serviços

Nas subseções seguintes são apresentados quatro métodos/processos de desenvolvimento de sistemas de *software* orientados a serviços.

2.3.1 Método proposto por Papazoglou e van den Heuvel

Este método para projeto e desenvolvimento de *software* orientado a serviços estabelece um processo iterativo e incremental, com base em modelos de desenvolvimento como o *Rational Unified Process* (RUP) (IBM, 2008), desenvolvimento baseado em componentes e modelagem de processos de negócios (PAPAZOUGLOU e van den HEUVEL, 2006).

O ciclo de desenvolvimento é composto por nove fases, sendo uma fase preparatória e oito fases iterativas, representadas por meio de seis blocos na Figura 2.3. As fases iterativas possibilitam uma abordagem de descoberta, criação e implementação a cada iteração, sempre tomando como referência os processos de negócio.

Os objetivos e requisitos de negócio devem direcionar o projeto, desenvolvimento e testes de sistemas de *software* que automatizam processos de negócio. A tecnologia *Web Services* (W3C, 2004a) é usada para implementação dos serviços (PAPAZOUGLOU e van den HEUVEL, 2006).

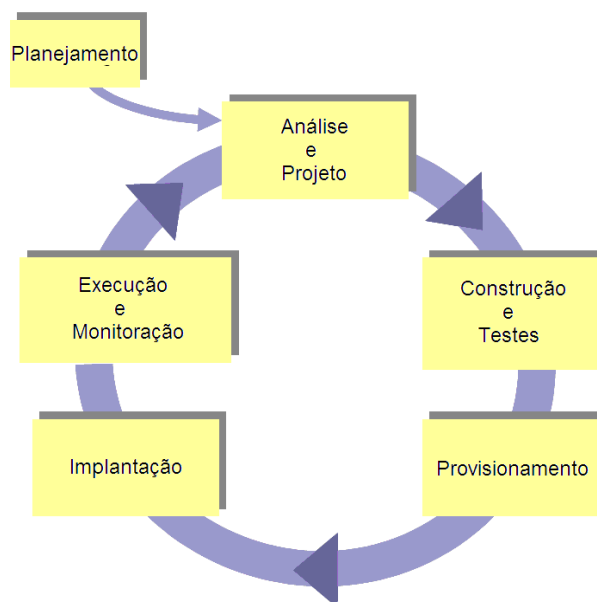


Figura 2.3: Fases do método (PAPAZOUGLOU e van den HEUVEL, 2006).

Na fase de planejamento são tratados a viabilidade do projeto, a natureza e escopo da solução baseada em serviços, os objetivos, as regras e os procedimentos. A compreensão do ambiente de negócios e a incorporação de mecanismos de controle no projeto da solução são os seus principais objetivos. Dentre as atividades dessa fase estão: análise das necessidades de negócio, transformando-as em objetivos que guiarão o desenvolvimento; revisão das tecnologias utilizadas atualmente no ambiente de negócios, elaboração de requisitos do novo ambiente e mapeamento de tecnologias para atender a esses requisitos; análise financeira dos custos envolvidos e benefícios, incluindo o orçamento disponível e um plano de desenvolvimento de *software* estabelecendo um cronograma listando tarefas, prazos e entregas.

A fase de análise trata da elicitación dos requisitos do sistema de *software* a ser desenvolvido. Um modelo *as-is* é construído para auxiliar os interessados no projeto a entender os processos de negócio existentes, como as atividades desses processos são realizadas e o contexto no qual o sistema de *software* será inserido. Análises, verificações e simulações realizadas usando o modelo *as-is* permitem

identificar pontos de melhoria nos processos de negócio para possibilitar um melhor retorno do investimento. O resultado dessas ações é representado no modelo *to-be*, que traduz os requisitos que serão implementados na solução a ser desenvolvida. O principal objetivo da fase de análise é o reúso de implementações de funções de negócio na solução. Quatro atividades principais são definidas para essa fase:

- Identificação de processos, a partir do agrupamento de funções de negócio;
- Definição do escopo de processos, com o intuito de garantir que um único processo não englobe todas as funções de negócio existentes, tornando o processo grande, complexo e difícil de manter;
- Análise de *gap*, com o objetivo de determinar quais serviços devem ser desenvolvidos, quais serão reusados e quais podem ser comprados de um provedor de serviços para atender as funções representadas no modelo *to-be* e requeridas para implementar o processo;
- Análise de realização de processos, que determina a estratégia de desenvolvimento a ser adotada na implementação dos processos de negócio, considerando fatores como riscos, benefícios, retorno de investimento, requisitos e prioridades de negócio.

A fase de projeto transforma processos e serviços conceituais, identificados na fase de análise, em um conjunto de interfaces inter-relacionadas e independentes de plataforma. Como a tecnologia *Web Services* é usada para a implementação dos serviços, esta fase deve conter as especificações de serviços em linguagem *Web Services Description Language (WSDL)* (W3C, 2007). As atividades que compõem a fase de projeto são:

- Especificação de serviços: trata da especificação dos *web services* que fornecem a implementação de funções de negócio identificadas na fase de análise. Essas especificações consideram os conceitos de acoplamento, coesão, granularidade de serviços, reusabilidade de serviços, composição de serviços e requisitos não funcionais (por exemplo: segurança, confidencialidade e integridade de informações e controle de acesso).
- Especificação de processos de negócio: os processos de negócio identificados na fase de análise são especificados por meio da descrição da estrutura do processo, associação dos papéis envolvidos no processo de negócio (PN) e especificação de requisitos não funcionais. A estrutura

de um processo representa o fluxo lógico que interconecta *web services*, definindo subprocessos, estruturas condicionais e dependências entre etapas de processo. Utiliza-se a linguagem de orquestração de serviços *Web Services Business Process Execution Language (WS-BPEL)* (OASIS, 2007) para descrever a estrutura de um processo. Em seguida são identificadas as responsabilidades associadas às atividades de processo de negócio e seus respectivos responsáveis. Acordos de nível de serviço (em inglês, *Service Level Agreement - SLA*) são estabelecidos entre um provedor de serviço e consumidores de serviço no qual os requisitos não funcionais são especificados. Os principais requisitos levados em consideração são: desempenho, modelo de pagamento, modelo de segurança, comportamento transacional, escalabilidade e disponibilidade.

Na fase de construção utiliza-se a tecnologia *Web Services* para desenvolvimento das implementações dos serviços especificados durante a fase de projeto. A estratégia de implementação dos *web services* segue a diretriz de realização de processos definida na fase de análise, na qual novos *web services* podem ser criados, aplicações existentes podem ter suas funções expostas através de *web services* ou composições de *web services* podem ser criadas. Nessa fase também são definidas as descrições das interfaces dos serviços e as descrições das implementações dos serviços.

Na fase de testes é avaliado se os requisitos de negócio são atendidos pelos processos e serviços implementados. Vários tipos de testes podem ser realizados com o intuito de assegurar que o sistema orientado a serviços desempenha satisfatoriamente as funções solicitadas. É importante que os aspectos não funcionais envolvidos em um sistema de *software*, como por exemplo, privacidade, integridade de mensagens, autenticação, autorização e não repúdio, sejam testados.

A fase de provisionamento de serviços trata dos aspectos organizacionais dos serviços. Alguns desses aspectos são governança de serviços, certificação de serviços, auditoria de serviços, medições e cobrança pela utilização de um serviço.

Na fase de implantação disponibiliza-se a implementação de um processo de negócio para todos os usuários, incluindo outras aplicações e processos. Para tanto, as interfaces dos serviços que compõem o processo de negócio são publicadas no repositório de serviços e as implementações são implantadas no provedor de serviços.

Na fase de execução um consumidor de serviços pode utilizar o repositório de serviços para localizar um serviço que atenda às suas necessidades e invocar as operações definidas na interface de serviço.

Na fase de monitoração os serviços são monitorados e problemas são identificados e relatados com o intuito de melhorar a qualidade dos serviços utilizados em uma solução orientada a serviços. O foco dessa fase está na avaliação contínua dos níveis de serviço e do desempenho por meio da extração de medidas de qualidade de serviços a partir dos acordos de nível de serviço (em inglês *Service Level Agreement* – SLA). Também é necessário preocupar-se com o constante acompanhamento das cargas de trabalho de cada um dos serviços fornecidos, permitindo que ajustes sejam feitos pelo provedor de serviços a fim de cumprir os SLAs.

2.3.2 Método proposto por Erl

As fases do ciclo de vida do método proposto por Erl (2005) são exibidas na Figura 2.4. A fase Análise Orientada a Serviços e Projeto Orientado a Serviços são responsáveis por incorporar as características e princípios da orientação a serviços à solução a ser desenvolvida. As demais fases implementam, verificam e administram os serviços projetados.

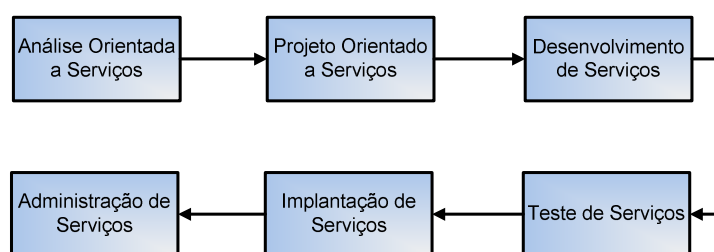


Figura 2.4: Fases do ciclo de vida (ERL, 2005).

Na Análise Orientada a Serviços (Figura 2.5) é determinado como os requisitos de negócio são mapeados em uma solução orientada a serviços. Identificam-se os serviços candidatos que atendem aos requisitos e as respectivas lógicas de negócio encapsuladas em cada um desses serviços. Nesse contexto, serviços candidatos são definições abstratas de serviços em nível de análise que podem vir a ser realizados durante a fase de projeto orientado a serviços.

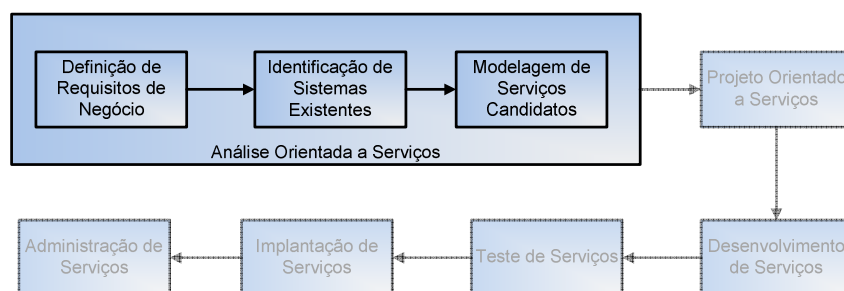


Figura 2.5: Atividades da Análise Orientada a Serviços (ERL, 2005).

A Definição dos Requisitos de Negócio identifica e documenta os requisitos para a elaboração e a documentação de um processo de negócio que será a base para o desenvolvimento da solução orientada a serviços. Sistemas computacionais existentes tais como *web services*, componentes, aplicações ou sistemas de *software* que fornecem funções que suprem total ou parcialmente um ou mais requisitos de negócio devem ser identificados. Esses recursos identificados são serviços candidatos.

Na Modelagem de Serviços Candidatos, processos de negócio são decompostos com o intuito de identificar e criar os serviços, norteados por princípios da COS como reusabilidade e autonomia de serviços. As etapas do processo de negócio que podem ser implementadas por uma composição de serviços são identificadas nessa atividade.

Na fase Projeto Orientado a Serviços, com base nos serviços candidatos identificados elabora-se o projeto dos serviços que define como os serviços devem ser implementados. Na Figura 2.6 são apresentadas as atividades realizadas no Projeto Orientado a Serviços:

- Composição da Arquitetura Orientada a Serviços: definem-se quais tecnologias e componentes arquiteturais são necessários para implementar a solução orientada a serviços e como esses elementos devem ser utilizados, incluindo as camadas de serviços que serão utilizadas na arquitetura, padrões e tecnologias que fornecem as características da orientação a serviços;
- Projeto de serviços baseados em entidades: especifica os serviços correspondentes às entidades de dados de negócio permitindo acessar e utilizar essas entidades por meio de troca de mensagens e invocação de operações pertencentes aos serviços. Erl (2005) propõe que essa

especificação utilize a linguagem de descrição de serviços WSDL (W3C, 2007) e que esquemas XSD (em inglês *XML Schema Definition*) (W3C, 2004b) sejam utilizados para especificar os tipos de mensagem;

- Projeto de serviços de aplicação: especifica os serviços responsáveis pela execução de funções utilitárias (por exemplo: conversão de um texto em formato pdf);
- Projeto de serviços baseados em tarefas: especifica os serviços que executam conjuntos de etapas de processo de negócio. Esses serviços são específicos para esse contexto e assim possuem baixa reusabilidade em outros contextos;
- Projeto de processo de negócio: define as orquestrações de serviços necessárias para atender aos requisitos de negócio. Uma orquestração organiza os serviços, mensagens e interações envolvidas no fluxo de trabalho de um processo de negócio, considerando possíveis erros durante a execução dos serviços e como o processo deve reagir nesses casos. Erl (2005) propõe que a definição dos processos de negócio e respectivas orquestrações sejam representadas por meio da linguagem WS-BPEL.

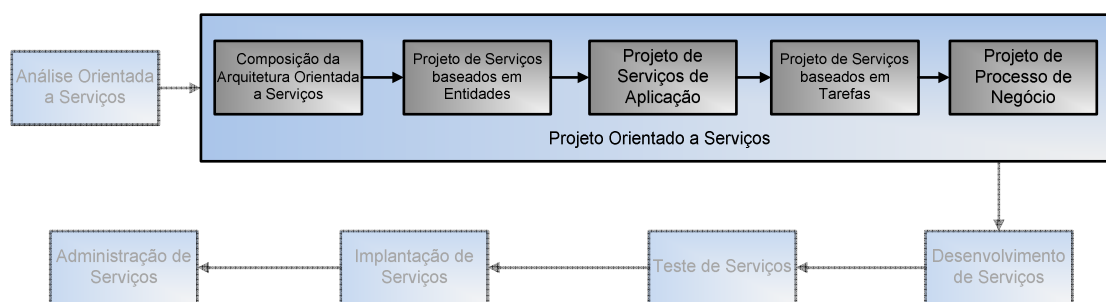


Figura 2.6: Projeto Orientado a Serviços (ERL, 2005).

Na fase Desenvolvimento de **Serviços** ocorre a construção dos serviços projetados na fase anterior. Decisões em relação aos aspectos relacionados à plataforma de implementação são tomadas no início dessa fase e determinam a forma de implementação dos serviços e das orquestrações de processos de negócio.

Em virtude da natureza genérica de um serviço e do seu potencial de reúso, a fase Testes de Serviços deve ser rigorosa e abordar aspectos funcionais do serviço, suas características, tecnologias de implementação, uso do serviço, entre outros.

Durante a fase Implantação de Serviços realizam-se a instalação e a configuração dos componentes, das interfaces de serviço e do *middleware* necessários para o funcionamento dos serviços e seus componentes. As tarefas envolvidas nessa fase são dependentes da tecnologia de implementação dos serviços.

A fase Administração de Serviços tem como principais objetivos: monitorar a utilização de serviços implantados; estabelecer a política de controle de versões utilizada para gerenciar as descrições de serviços; gerenciamento e localização de mensagens; e detecção de gargalos.

2.3.3 IBM Rational Service-Oriented Modeling and Architecture (SOMA)

O *Rational SOMA*, versão 2.9, possui um conjunto de práticas para identificação e especificação de serviços e soluções orientadas a serviços, utilizadas na fase de modelagem do ciclo de vida de serviços proposto pela IBM (2010b). Além dessa fase, esse modelo possui outras três: Montagem, Implantação e Gerenciamento.

O fluxo de atividades no *Rational SOMA 2.9* é constituído de quatro práticas, mostradas na Figura 2.7 por meio de um diagrama de atividades da UML. Esse fluxo pode ser executado tanto no contexto do modelo de ciclo de vida incremental e/ou iterativo quanto no contexto do modelo de ciclo de vida sequencial (IBM, 2010b).

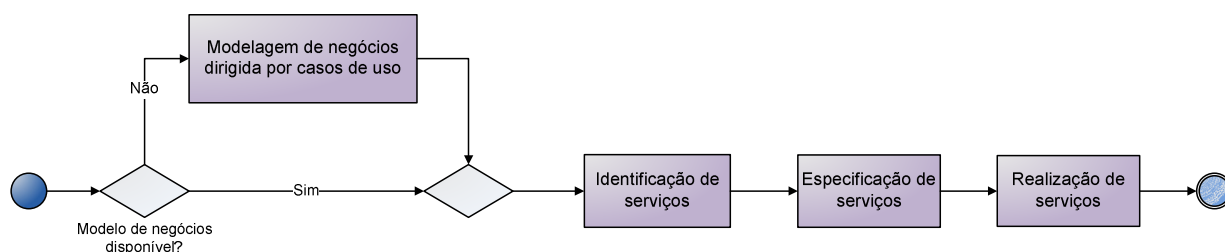


Figura 2.7: Fluxo de trabalho no *Rational SOMA 2.9* (IBM, 2010b).

A prática Modelagem de negócios dirigida por casos de uso tem a finalidade de descrever os processos de negócio e fornecer informações a respeito desses para fomentar o projeto. Casos de uso de negócio, como definidos na UML, são usados para descrever processos de negócio, identificando os limites e processos do negócio, assim como as interações entre esses processos.

Na prática Identificação de serviços, são identificados, descritos e classificados os serviços para compor a solução orientada a serviços. É importante que haja um alinhamento entre serviços candidatos e negócios, de forma que sejam criados serviços apenas para funções com valor de negócio (IBM, 2010b). Serviços candidatos são identificados de várias fontes, sendo que cada fonte requer a execução de um conjunto de passos. As principais fontes adotadas pelo *Rational SOMA* são:

- Modelos funcionais de negócio: representam o domínio do negócio, seus subdomínios, as áreas funcionais (grupos de funções) associadas a cada subdomínio e as principais funções de negócio em si;
- Objetivos de negócio: serviços candidatos são identificados a partir da correspondência entre os objetivos de negócio e os serviços necessários para alcançar esses objetivos. Também são identificados indicadores de desempenho (*Key Performance Indicators – KPI*) e métricas que serão utilizados para avaliar, monitorar e verificar o nível em que os objetivos foram alcançados;
- Processos de negócio: representados por meio de linguagens de notação de processos, são decompostos até o nível em as que etapas dos processos possam ser mapeadas para casos de uso de sistema;
- Regras de negócio: são consideradas quando há necessidade de uniformização das regras de negócio utilizadas nos sistemas de *software* de uma empresa. As que são específicas de um sistema ou que sofrem constantes mudanças não devem ser mapeadas para serviços candidatos;
- Modelos de caso de uso de negócio: os casos de uso devem ser decompostos em elementos de realização até um nível adequado que permita mapeá-los para casos de uso de sistema;
- Modelos de domínio: definem os tipos de dados pertencentes a um domínio específico. Operações de criação, leitura, atualização e deleção de informações para cada um destes tipos de dados podem ser agrupadas em serviços candidatos;
- Recursos existentes: funções de negócio fornecidas por sistemas legados são identificadas e avaliadas quanto à viabilidade de sua utilização.

À medida que diversas fontes são utilizadas, é comum que haja sobreposição de serviços candidatos, assim é necessário realizar uma refatoração, removendo

redundâncias e melhorando a compreensão, consistência e flexibilidade do modelo. Todos os serviços candidatos identificados nessa fase são documentados em um portfólio de serviços candidatos.

Na prática Especificação de serviços, os serviços candidatos são especificados juntamente com os elementos relacionados e suas interações. A especificação de um serviço enfoca a definição da interface de serviços que fornece todas as informações que um consumidor necessita para decidir pelo uso do serviço bem como informações sobre como usá-lo. Essas interfaces de serviço são representadas utilizando o elemento *ServiceInterface* do perfil SoaML.

Os serviços presentes no portfólio de serviços candidatos são avaliados para determinar se devem ou não ser expostos como serviços. Um conjunto de testes simples e objetivos é aplicado a cada serviço, avaliando aspectos técnicos e também a viabilidade de sua implementação. Serviços que não passarem nos testes podem ser implementados, mas não terão suas interfaces expostas.

Uma vez definidos os serviços que serão expostos iniciam-se a identificação e a especificação de soluções orientadas a serviço. Para isso, são criadas especificações de serviço e colaborações UML, documentadas no artefato modelo de serviços. Esse artefato também armazena documentação de requisitos não funcionais avaliados, ameaças e vulnerabilidades identificadas e a especificação dos componentes dos serviços.

Durante a prática Realização de serviços, decisões são tomadas para garantir que a implementação e a arquitetura da solução orientada a serviços atendam aos requisitos funcionais e não funcionais. Essas informações são armazenadas no artefato Modelo de serviços ou modelo de componentes. Com base nas decisões tomadas, é desenvolvida uma prova de conceito da arquitetura para avaliar a sua viabilidade técnica.

2.3.4 Método de Análise e Projeto Orientado a Serviços - MAPOS

MAPOS (FUGITA e HIRAMA, 2008a) (FUGITA e HIRAMA, 2008b) é uma proposta de método para análise e projeto fundamentada nos princípios da orientação a serviços e nas melhores práticas existentes em vários métodos de desenvolvimento de sistemas orientados a serviço, como: Papazoglou e van den Heuvel (2006), RUP *plugin for SOMA* (IBM, 2008), Erl (2005) e Marks e Bell (2006).

Na Figura 2.8 são mostradas as fases do ciclo de desenvolvimento no qual MAPOS se insere e suas respectivas atividades.

Na fase Modelagem de Negócios especificam-se os requisitos do sistema de *software*, com a realização das atividades: Modelagem de processos *as-is* e Modelagem de processos *to-be*. A modelagem de processos *as-is* cria uma representação dos processos de negócio da forma como eles são executados na empresa. Esse modelo é útil para que clientes e interessados no sistema de *software* entendam como seus negócios são conduzidos. Esse entendimento permite maior precisão sobre requisitos do sistema de *software* e maior retorno do investimento. Na modelagem de processos *to-be* o analista de negócios avalia o modelo *as-is* buscando identificar possíveis melhorias, originando o modelo de processos *to-be*. Os requisitos para desenvolvimento do sistema de *software* derivam deste modelo. Esses modelos podem ser representados usando-se casos de uso da UML 2.0 ou por meio da linguagem *Business Process Model and Notation* (BPMN).

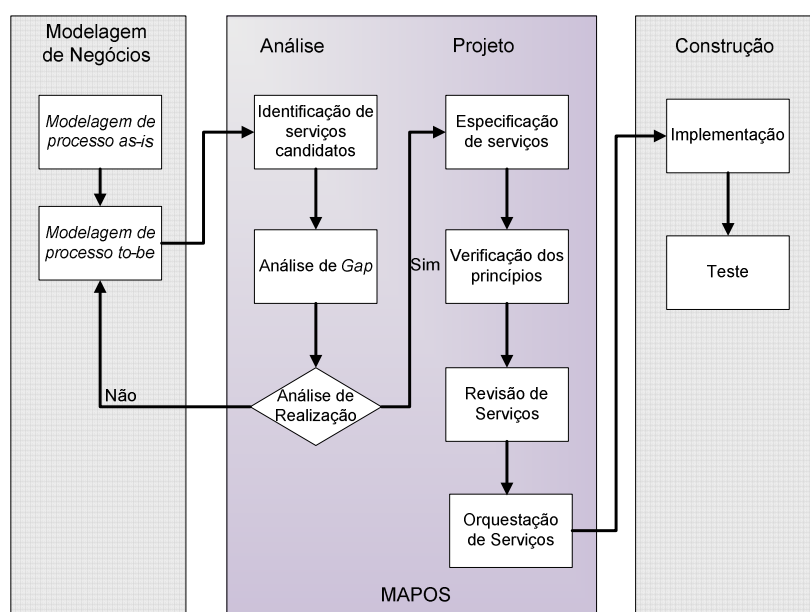


Figura 2.8: Método de Análise e Projeto Orientado a Serviços (FUGITA, 2009).

Na fase Análise, três atividades são conduzidas para identificar os serviços que irão compor a solução SOA:

- Identificação de serviços candidatos: utiliza como artefato de entrada o modelo de processos *to-be*. Os processos são decompostos em uma série de passos, que são mapeados como operações candidatas. As operações com um mesmo contexto lógico e semântico são agrupadas, gerando os

serviços candidatos, representados no modelo de serviços candidatos que constitui o artefato de saída dessa atividade;

- **Análise de *gap*:** avalia quais operações candidatas possuem sua funcionalidade fornecida por um serviço já implementado ou por um sistema legado. Os cenários de reuso que podem resultar dessa análise são mostrados na Tabela 2.1. Após a realização da análise de *gap* o modelo de serviços candidatos deve ser revisado, visto que algumas operações de serviços mencionados podem já ter sido implementadas;

Tabela 2.1: Cenários de reuso possíveis (FUGITA, 2009).

Cenário	Descrição
1	Funcionalidade fornecida por um serviço já implementado
2	Funcionalidade parcialmente fornecida por um serviço
3	Funcionalidade fornecida por um sistema legado
4	Funcionalidade parcialmente fornecida por um sistema legado
5	Funcionalidade não implementada

- **Análise de realização:** define a estratégia de realização para cada um dos cenários de reuso identificados na análise de *gap*, como mostrado na Tabela 2.2. Custos, viabilidade técnica, riscos técnicos e de projeto, esforço e retorno do investimento são alguns dos aspectos que influenciam na decisão da estratégia de realização de um serviço candidato.

Tabela 2.2: Cenários e estratégias de realização de serviços (FUGITA, 2009).

Cenário	Estratégia de realização
1	Reuso do serviço, envolve apenas custo de utilização do serviço.
2	Alterar o serviço existente (risco inerente à alteração de um serviço) ou implementar uma lógica de mediação, possibilitando o reuso do serviço da forma como está implementado atualmente.
3	Criar um serviço encapsulador que acessa a funcionalidade através de APIs ou adaptadores; caso não existam tais mecanismos deve-se avaliar a viabilidade, custos e riscos envolvidos na criação de tais mecanismos.
4	Criar um serviço encapsulador que acessa apenas a parte da funcionalidade reusável ou realizar alterações no sistema legado de forma a fornecer toda a funcionalidade necessária.
5	Especificar e implementar o serviço, incorrendo em custos de desenvolvimento.

O analista de serviços, com apoio do gerente de projetos e do arquiteto corporativo, decide a viabilidade da realização dos serviços conforme as estratégias e cenários de reuso. Caso alguma estratégia de realização não seja viável, retorna-

se à atividade de modelagem de processos *to-be* para que em seguida as atividades de análise sejam executadas novamente.

A fase Projeto é constituída das seguintes atividades:

- Especificação de serviços: utiliza como artefato de entrada o modelo de serviços candidatos para criar as interfaces dos serviços na qual constam as operações fornecidas por cada serviço e as mensagens de entrada e saída relacionadas a cada operação. Podem ser utilizados esquemas XSD para especificar o formato das mensagens e a linguagem WSDL para descrever as interfaces de serviço. Na Tabela 2.3 estão apresentadas as estratégias para criação das interfaces de serviço. A lógica envolvida na funcionalidade fornecida por uma operação é especificada por meio de diagramas de atividades da UML. As interações com outros serviços e sistemas legados são especificadas por meio de diagramas de sequência ou de colaboração. Políticas de serviços representam os requisitos não funcionais relacionados aos serviços e são especificadas por meio de SLAs. Serviços devem ser classificados de acordo com sua granularidade e nível de funcionalidade.

Tabela 2.3: Estratégias de criação de interfaces de serviço no MAPOS (FUGITA, 2009)

Estratégia	Descrição	Cenários abrangidos
<i>Top-down</i>	A interface do serviço é criada a partir do serviço candidato.	5 e 2
<i>Bottom-up</i>	Especifica as operações e mensagens baseadas nas características da funcionalidade existente	3
<i>Meet-in-the-middle</i>	A interface do serviço é parcialmente baseada na funcionalidade existente ao mesmo tempo em que se faz necessário utilizar uma abordagem <i>top-down</i> para especificar o que deve ser adaptado para que a funcionalidade da operação candidata seja completamente atendida.	4
Importar interface	Como a funcionalidade já existe e atende aos requisitos do serviço, importa-se a interface do serviço.	1

- Verificação dos princípios: as especificações dos serviços são avaliadas quanto à conformidade com os princípios da orientação a serviços;
- Revisão dos serviços: tem a finalidade de corrigir problemas identificados na atividade de verificação dos princípios. As mesmas considerações realizadas na atividade de especificação de serviços são aplicadas a essa atividade.

- Orquestração de serviços: especifica-se a lógica de orquestração de serviços, utilizando a linguagem BPEL, que determina a implementação do processo de negócio e engloba as chamadas aos serviços através de suas interfaces, a manipulação de variáveis de processo, o gerenciamento de informações de estado, o tratamento de exceções e a compensação.

A fase Construção, composta por duas atividades, é descrita brevemente por Fugita (2009), pois o método tem enfoque nas fases Análise e Projeto:

- Implementação: conduzida por desenvolvedores que recebem as especificações de serviços e implementam os *web services* e outros componentes necessários.
- Teste: conduzida por testadores que utilizam as especificações de serviço para elaborar os casos de teste que serão aplicados aos *web services* desenvolvidos.

Fugita (2009) afirma que a partir desta fase o ciclo tradicional de desenvolvimento de *software* orientado a objetos ou baseado em componentes pode ser utilizado.

2.4 Perfil SoaML para Modelagem de Serviços

O SoaML (em inglês *Service Oriented Architecture Modeling Language*) (OMG, 2009b) é um perfil da UML (OMG, 2005) para modelagem de serviços e arquiteturas orientadas a serviço. Esse perfil possui um padrão independente de tecnologia para criação, comunicação, manutenção e aperfeiçoamento de uma arquitetura orientada a serviços, sendo esse um dos seus pontos fortes juntamente com a capacidade de modelagem de arquiteturas orientadas a serviços em diferentes níveis, desde o de negócios, passando pelo de sistemas e até o de subsistemas.

Serviço é definido pelo SoaML como fornecido ou consumido por uma entidade denominada participante e é especificado por meio de interfaces de serviço que definem protocolos de interação e comunicação (OMG, 2009b).

O meta-modelo do perfil SoaML estende o meta-modelo da UML 2.0 com o objetivo de apoiar a modelagem de serviços sob perspectivas como descrição de

serviço, modelagem de arquitetura orientada a serviços e definição de contrato de serviço. Quatro novas áreas são adicionadas: participantes, interface de serviços, contrato de serviço e dados de serviço. Participantes permitem definir provedores e consumidores de serviço, representando componentes, sistemas, pessoas ou empresas. Interfaces de serviço permitem especificar a forma de interação com um serviço, por meio da definição da interface, e estabelecer responsabilidades de um participante ao fornecer ou consumir um serviço. Contratos de serviço especificam o acordo entre provedores e consumidores de um serviço em relação aos termos, condições, interfaces e interações para que o serviço possa ser fornecido ou utilizado. Dados de serviços permitem modelar mensagens e anexos de mensagens envolvidas na interação entre provedor e consumidor de serviços.

Na Figura 2.9 estão representados os elementos do perfil SoaML utilizados para modelagem de participantes e interfaces de serviço. Um elemento `<<Participant>>` pode representar um provedor de serviços, um consumidor de serviços ou ambos. Um participante atuando como provedor de serviços possui um elemento *port* da UML 2.0 com o estereótipo `<<ServicePoint>>` que define o ponto de interação pelo qual um serviço é fornecido. Se o participante é um consumidor de serviços, possui um elemento *port* com o estereótipo `<<RequestPoint>>` para representar a necessidade de “consumir” um serviço fornecido por outro participante. Os estereótipos `<<ServicePoint>>` e `<<RequestPoint>>` associam uma interface UML (`<<Interface>>`) ou uma interface de serviço (`<<ServiceInterface>>`) que define todas as informações necessárias para interagir com o serviço.

Um exemplo de modelagem de um serviço de empréstimos de exemplares em uma biblioteca é apresentado na Figura 2.10. Nesse exemplo, as funções do serviço são fornecidas pelo participante “Setor de empréstimos” por meio da interface de serviço “EmprestimoServices”. Um elemento *port* com o estereótipo `<<ServicePoint>>` ligado ao participante especifica a interface pela qual as funcionalidades devem ser invocadas.

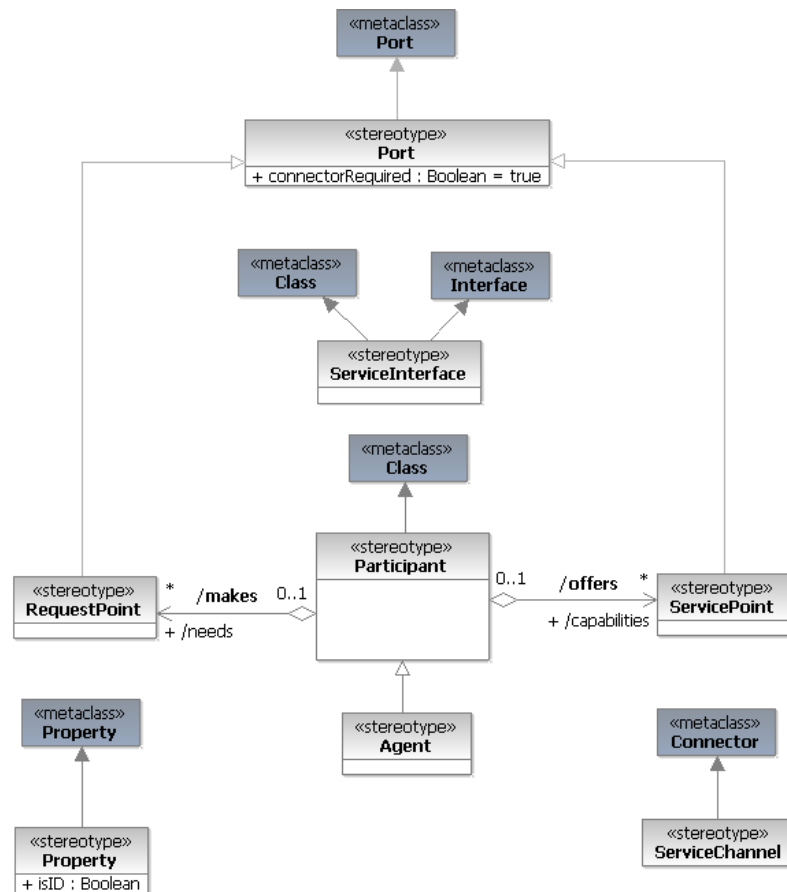


Figura 2.9: Elementos básicos do perfil SoaML (OMG, 2009b).

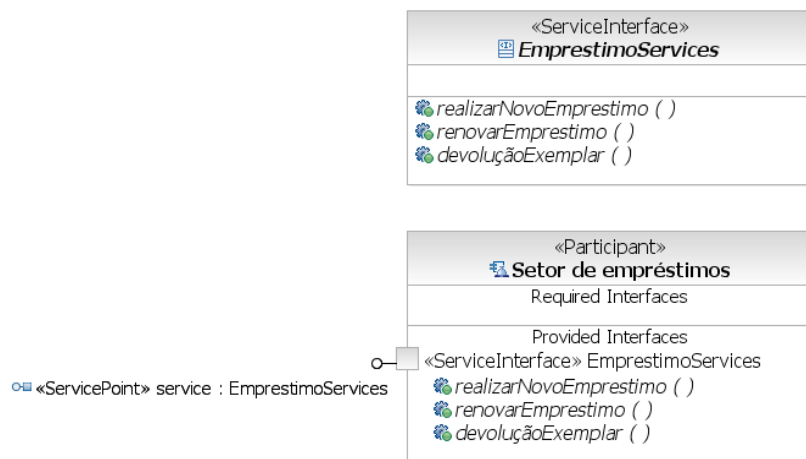


Figura 2.10: Exemplo de modelagem de participante e interface.

No perfil SoaML, um contrato de serviço é modelado utilizando-se o elemento `<<ServiceContract>>`, que posteriormente pode ser realizado por elementos como participantes, interfaces de serviço, `<<RequestPoint>>` e `<<ServicePoint>>`. A definição do meta-modelo desse elemento é apresentada na Figura 2.11.

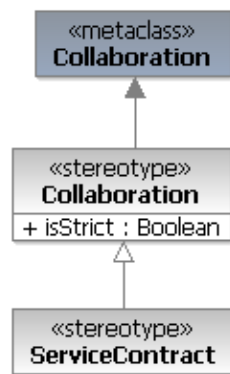


Figura 2.11: Definição do elemento ServiceContract (OMG, 2009b).

SoaML possui dois estereótipos para modelagem de dados: <<Attachment>> e <<MessageType>>, mostrados na Figura 2.12. O estereótipo <<MessageType>> é usado para identificar dados (mensagens) que trafegam em interações com um serviço, enquanto o estereótipo <<Attachment>> é usado para representar elementos de dados que possuem significado fora do contexto da interação com um serviço (por exemplo, documentos de texto, imagens).

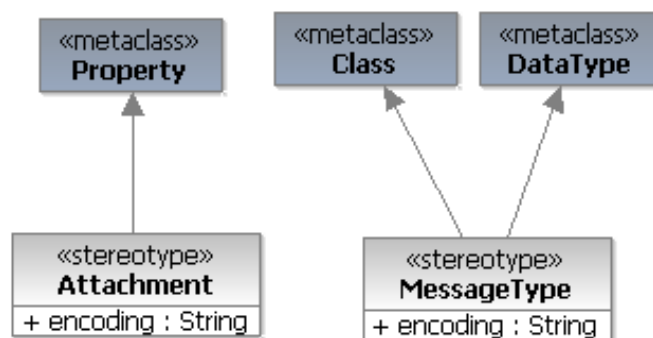


Figura 2.12: Elementos <<Attachment>> e <<MessageType>> (OMG, 2009b).

O meta-modelo do perfil SoaML também contempla modelagem em nível de negócios, por meio da utilização do elemento <<ServicesArchitecture>> para representar arquiteturas orientadas a serviços compostas por vários participantes que fornecem e utilizam serviços para atingir os objetivos de negócio ou executar processos. A definição do elemento <<ServicesArchitecture>> é mostrada na Figura 2.13.

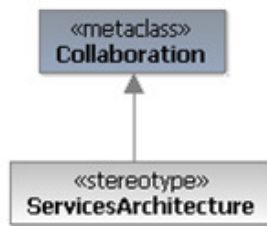


Figura 2.13: Definição do elemento <<ServicesArchitecture>> (OMG, 2009b).

Um serviço pode ser modelado com SoaML por meio da representação de suas funções utilizando o elemento <<Capability>>. Uma rede de funções com suas respectivas dependências pode ser construída para representar as necessidades e interações requeridas para o fornecimento da funcionalidade do serviço. Na Figura 2.14 observa-se a *capability* “Realizar novo empréstimo” que utiliza outras quatro *capabilities*: “Cadastro de usuários”, “Reservas”, “Políticas de empréstimos” e “Empréstimos”. Nesse exemplo o participante “Setor de empréstimos” realiza a *capability* “Realizar novo empréstimo”, indicando que as funções dessa *capability* serão realizadas por este participante.

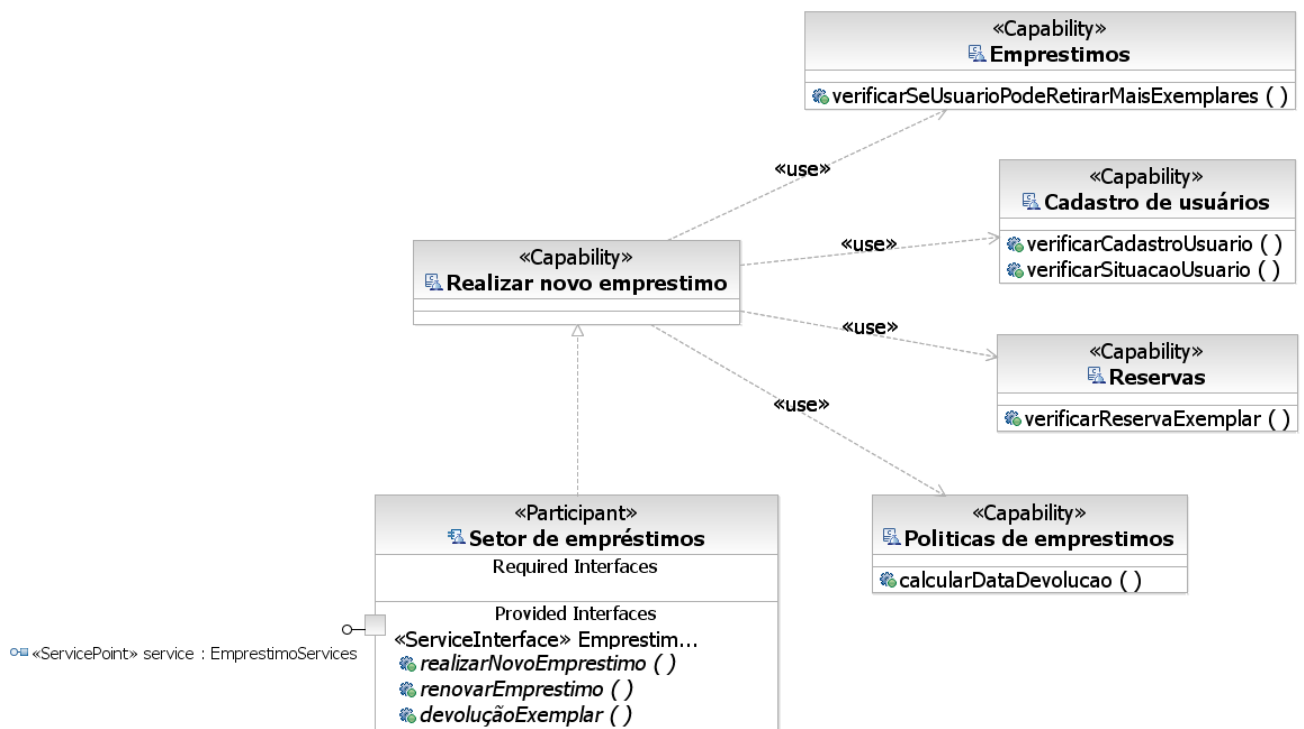


Figura 2.14: Relacionamento dos principais elementos do perfil SoaML.

Na Tabela 2.4 são listados os estereótipos pertencentes ao perfil SoaML juntamente com uma breve descrição para cada um deles.

Tabela 2.4: Estereótipos do perfil SoaML (OMG, 2009b).

Estereótipo	Descrição
<<Participant>>	Representa uma entidade ou componente que fornece e/ou consome serviços.
<<Agent>>	Representa entidades autônomas que possuem necessidades e <i>capability</i> que podem mudar ao longo do tempo.
<<MessageType>>	Especifica informações trocadas entre consumidores e provedores de serviço.
<<Attachment>>	Representa um elemento de dados anexado a uma mensagem (um documento de texto, arquivo de imagem).
<<Capability>>	Representa um conjunto coeso de funções fornecidas por um serviço.
<<Port>>	Representa pontos de interação com o serviço podendo indicar fornecimento ou consumo de funções.
<<Milestone>>	Informações complementares que permitem detalhar o comportamento ou fluxo de atividades.
<<ParticipantArchitecture>>	Descreve as interações dos participantes envolvidos no fornecimento de um serviço.
<<Property>>	Permite identificar instâncias de um tipo.
<<RequestPoint>>	Representa a utilização de um serviço por um participante. Define o ponto de conexão entre consumidor e provedor de serviços.
<<ServicePoint>>	Representa o ponto de conexão pelo qual um provedor de serviços oferece suas funções.
<<ServiceChannel>>	Utilizado para conectar requisições de um consumidor ao serviço fornecido por um provedor.
<<ServiceContract>>	Especifica o acordo de serviço entre provedores e consumidores de serviço quanto ao fluxo de informações, produtos, recursos, custos e obrigações.
<<ServiceInterface>>	Define a interface e responsabilidades de um participante ao consumir ou fornecer um serviço.
<<ServicesArchitecture>>	Permite representar uma visão geral da arquitetura orientada a serviços, com a descrição dos participantes que colaboram para fornecer uma funcionalidade de negócio.

2.5 Considerações Finais

Neste capítulo foram apresentados os principais conceitos da Computação Orientada a Serviços e quatro processos/métodos para desenvolvimento de sistemas de *software* orientados a serviço. A aplicação da Computação Orientada a Serviços na construção de sistemas de *software* traz benefícios como flexibilidade do sistema, redução do esforço de desenvolvimento devido ao reúso de serviços, alinhamento entre processos de negócio e componentes de *software* e integração de sistemas em diversas plataformas e linguagens de desenvolvimento. Para obter tais benefícios é preciso que os serviços e a solução orientada a serviços sigam os princípios da orientação a serviços. Cada um dos métodos descritos neste capítulo procura abordar o desenvolvimento de serviços de forma que esses incorporem tais princípios, apresentando todas as características necessárias para obtenção dos benefícios prometidos pela Computação Orientada a Serviços. Os métodos/processos analisados possuem diferenças entre si quanto à adoção de conceitos novos (por exemplo o conceito de serviço candidato no *Rational SOMA 2.9*) e quanto às atividades envolvidas na análise e projeto de sistemas orientados a serviço. Alguns métodos englobam o ciclo de vida completo de sistemas orientados a serviço enquanto outros, como o MAPOS, abordam apenas as fases iniciais de desenvolvimento (análise, projeto).

De uma maneira geral, todos os métodos descritos neste capítulo se assemelham quanto às atividades de análise e projeto para desenvolvimento de sistemas orientados a serviço, consistindo da modelagem dos processos de negócio, identificação de serviços e especificação de serviços.

A Tabela 2.5 apresenta um quadro comparativo com descrições dos métodos abordados neste capítulo.

Tabela 2.5: Resumo sobre os métodos descritos neste capítulo.

	PAPAZOGLOU e van den Heuvel	MAPOS	SOMA 2.9	Erl
Iterativo	Sim	Sim (pouco)	Sim*	Sim
Incremental	Sim	Não	Sim*	Não
Agilidade	Não	Não	Sim	Não
Prescreve ferramentas	Sim	Não	Sim	Não
Prescreve diagramas / artefatos	Sim	Sim	Sim	Não
Possui exemplos	Usa modelo de referência	Não	Poucos e não relacionados	Sim
Estabelece parâmetros para definir o tamanho de um incremento	Não	Não	Não	Não
Estabelece tecnologias de implementação	Sim (WS, BPEL, WSDL)	Sim (BPMN, UML, BPEL, WSDL)	Não	Sim (WS, WSDL, BPEL, XSD)
Facilita incorporação gradual de serviços ao sistema	Não	Não	Não	Não
Utiliza modelagem de processos de negócio	Sim	Sim	Sim	Sim

* Dependente do modelo de ciclo de vida utilizado

Capítulo 3

MODELAGEM DE PROCESSOS DE NEGÓCIO

3.1 Considerações Iniciais

A intensificação da globalização, o crescimento do comércio eletrônico e a expansão de sistemas de *software* por meio da internet expandiram os horizontes dos negócios e dos serviços, permitindo que empresas ou organizações públicas ou privadas atuem em domínios antes inexplorados, produzindo e fornecendo bens de consumo, prestando serviços, ou de qualquer outra forma que gere lucros ou benefícios. Nesse ambiente altamente dinâmico, um produto ou serviço com melhor relação custo-benefício possui vantagens competitivas no mercado, resultando em mais negócios ou serviços realizados. Além disso, Ko, Lee e Lee (2009) afirmam que a lucratividade e a sobrevivência de empresas são constantemente desafiadas por fatores como: o aumento na frequência de pedidos de bens ou serviços; a necessidade de transferência rápida de informações; tomada de decisão rápida; a necessidade de adaptar-se às mudanças *on-demand*; a competição internacional; e a necessidade de ciclos de desenvolvimento de sistemas de *software* mais curtos.

Uma forma de lidar com esses desafios baseia-se no gerenciamento dos processos de negócio de uma empresa. Nesse contexto, processo de negócio é uma sequência de atividades que agrega valor a uma entrada para produzir uma saída com um determinado objetivo de negócio (Yan *et al.*, 2007), podendo ser visto como um roteiro de atividades para se obter o produto ou entregar um serviço. O

gerenciamento de processos de negócios (BPM, em inglês *Business Process Management*) consiste em apoiar os processos de negócio utilizando métodos, técnicas e *software*, para projetar, representar, controlar e analisar processos (em qualquer nível) envolvendo pessoas, organizações, sistemas de *software*, documentos e outras fontes de informação (van der AALST, ter Hofstede, WESKE, 2003). O BPM é uma forma estruturada coerente e consistente para entender, modelar, analisar, simular, executar e modificar constantemente processos de negócio de ponta a ponta e todos os recursos envolvidos, para contribuir com a melhoria do desempenho do processo (RECHER *et al.*, 2006), reduzindo a ocorrência de problemas, reduzindo custos e aumentando a produtividade.

Nesse contexto, a modelagem de processos de negócio, que faz parte do BPM (FERNÁNDEZ *et al.*, 2010), ganhou popularidade e tem por objetivo a identificação e especificação de processos de negócio, incluindo a representação de suas atividades e seus respectivos relacionamentos causais e temporais, assim como regras de negócio aplicadas aos processos de negócio (van der AALST, ter Hofstede, WESKE, 2003). Essa especificação geralmente combina elementos gráficos e anotações textuais para representar graficamente o processo de negócio, de maneira desacoplada de sua arquitetura técnica e de *software* (DAHMAN, CHAROY e GODART, 2011), facilitando o entendimento do processo, de suas atividades e das pessoas nele envolvidas; possibilitando realizar simulações; possibilitando a identificação de pontos de melhoria ou gargalos no processo; e permitindo o aprimoramento do processo para que se obtenha melhor resultado de acordo com objetivos de negócio. O artefato resultante da modelagem de processos de negócio pode ser utilizado como o principal meio de comunicação entre pessoas do meio de negócios e equipe técnica, conectando a área de negócios com a área técnica.

RECHER *et al.* (2006) comenta que diversas abordagens para modelagem de processos foram propostas ao longo dos anos, variando desde simples fluxogramas até variações de redes de Petri com alta expressividade, sendo o *Business Process Model and Notation* (BPMN) (OMG, 2009a) uma das mais recentes propostas de linguagem de modelagem de processos. O BPMN se popularizou rapidamente devido à sua notação intuitiva, à conformidade com o padrão WebService, e à promessa de se tornar o padrão para modelagem de processos de negócio (RECHER *et al.*, 2006).

Este capítulo aborda o BPMN, utilizado nesta dissertação para modelagem de processos de negócio. Na Seção 3.3 são apresentadas as considerações finais.

3.2 Business Process Model and Notation (BPMN)

O *Business Process Model and Notation* (BPMN) (OMG, 2009a) é uma técnica para modelagem de processos de negócios (RECHER *et al.*, 2006) que fornece uma notação compreensível por usuários, pessoas de negócios, analistas de negócios, desenvolvedores e mantenedores do processo.

O BPMN dispõe de um diagrama de processos de negócio (*Business Process Diagram*), no qual os processos de negócio podem ser modelados como uma rede de atividades interligadas por um fluxo de controle indicando a ordem de execução dessas atividades. Os elementos do BPMN utilizados no diagrama de processos de negócio são agrupados em quatro categorias: objetos de fluxo, objetos conectores, objetos *swimlane* e artefatos (OMG, 2009a).

Objetos de fluxo permitem representar o comportamento do processo de negócio, podendo ser de três tipos: eventos, atividades ou *gateways* (OMG, 2009a). Evento é um acontecimento durante o decorrer de um processo possuindo uma causa e um resultado. Na Figura 3.1 (A) estão os símbolos que representam eventos iniciais, intermediários e finais. Uma atividade representa um trabalho executado dentro de um processo, podendo ser classificado em: processo, subprocesso e tarefa, sendo os dois últimos representados na Figura 3.1 (B). *Gateways* são utilizados para controlar o fluxo sequencial por meio de ramificações, decisões, uniões e junções e sua representação está mostrada na Figura 3.1 (C).

Objetos conectores interligam objetos de fluxo, criando a sequência básica de um processo de negócio. Três tipos de conectores são definidos no BPMN: fluxo de sequência, fluxo de mensagem, e associação (OMG, 2009a). Um fluxo de sequência é usado para indicar a ordem na qual as atividades são executadas em um processo. Fluxos de mensagem indicam a troca de mensagens entre dois processos distintos. As associações são usadas para vincular dados, texto e outros artefatos a objetos de fluxo. A Figura 3.2 mostra os símbolos utilizados para representação de objetos conectores.

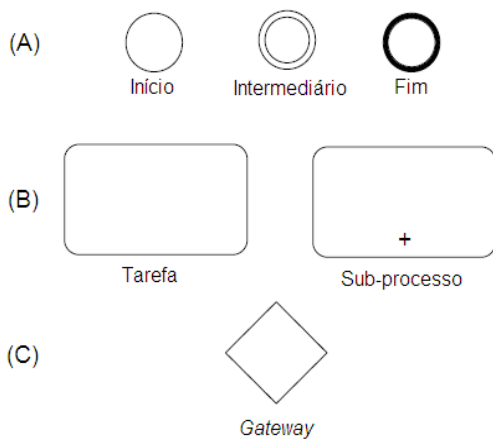


Figura 3.1: Objetos de fluxo do BPMN (adaptado de OMG, 2009a).

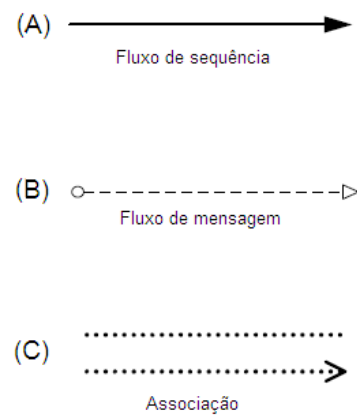


Figura 3.2: Objetos conectores do BPMN (adaptado de OMG, 2009a).

Na Figura 3.3 é exemplificada a representação de um processo de negócios adotado pelo setor de empréstimos de exemplares de uma biblioteca. Este exemplo de processo será utilizado ao longo deste capítulo para exemplificar os elementos de modelagem gráfica do BPMN. O processo de empréstimo da biblioteca tem início na identificação do tipo de requisição solicitada pelo usuário (realizar um novo empréstimo, renovar um empréstimo, ou devolver um exemplar). Dependendo do tipo da requisição do usuário, o fluxo segue por caminhos distintos, sendo que cada caminho passa pelas etapas pertinentes ao processo (subprocesso) que atende à requisição do usuário. Em outras seções deste capítulo serão abordadas etapas deste macro-processo.

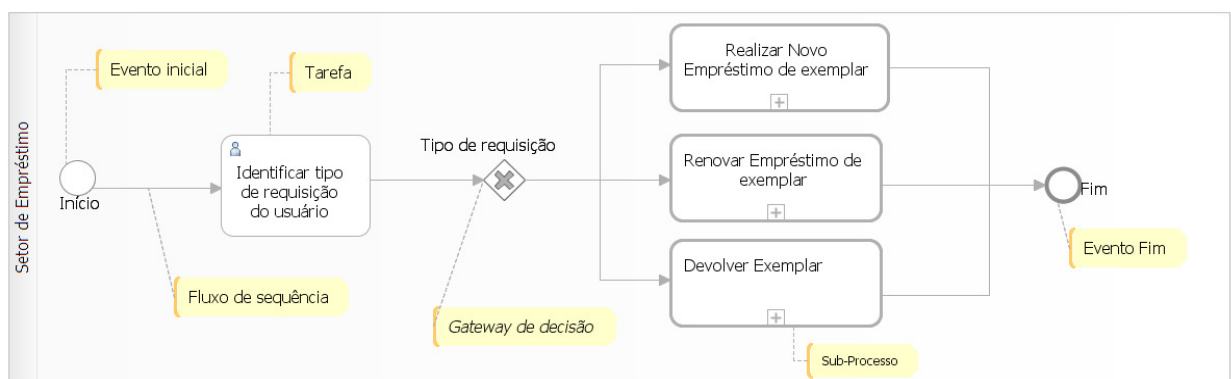


Figura 3.3: Exemplo de processo de negócio modelado usando BPMN.

O conceito de raia (*swimlane*) em modelos para representação de processos refere-se ao mecanismo para organização lógica dos elementos representacionais em categorias visuais distintas, com o objetivo de representar diferentes funcionalidades ou responsabilidades (WHITE, 2004). No BPMN esse mecanismo é

representado por objetos raia que podem ser de dois tipos: *pool* - utilizados para representar processos distintos em um diagrama de processos, e *lane* - subpartições de um *pool* utilizadas para separar atividades associadas a departamentos ou papéis distintos dentro de uma organização. A forma de representação de um *pool* (A) e duas *lanes* (B) está mostrada na Figura 3.4.

Artefatos são utilizados para fornecer informações adicionais sobre o processo de negócios (OMG, 2009a), podendo ser usados para estender a notação básica provida pelo BPMN para modelar situações especiais (WHITE, 2004). Três tipos de artefatos estão disponíveis: objeto de dados, grupo e anotação. Objetos de dados permitem exibir dados de entrada ou saída de uma atividade (Figura 3.5 A). Grupos reúnem elementos pertencentes a uma mesma categoria e são utilizados para fins de documentação ou análise e não interferem no fluxo sequencial (Figura 3.5 B). As anotações permitem adicionar informações textuais ao diagrama (Figura 3.5 C).

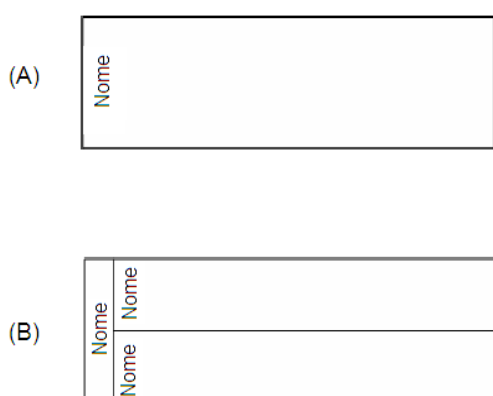


Figura 3.4: Objetos raia do BPMN (OMG, 2009a).

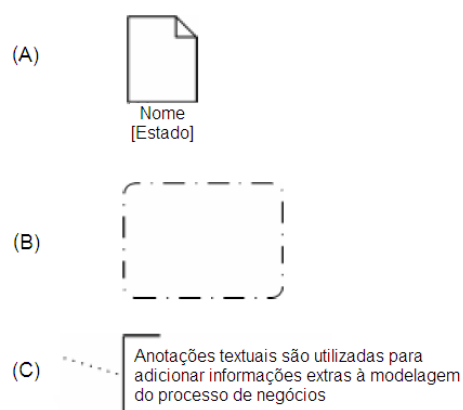


Figura 3.5: Artefatos do BPMN (OMG, 2009a).

Um processo de negócio pode ser modelado sob perspectivas distintas, cada uma representando diferentes tipos de informação, para públicos específicos. O BPMN fornece três perspectivas para modelagem de processos: processos privados, processos abstratos e processos colaborativos. Processos privados são usados para representar a organização interna das etapas de processos de negócio específicos de uma organização, como por exemplo a representação das atividades internas do subprocesso de devolução de exemplares emprestados da biblioteca, mostrada na Figura 3.6. Processos abstratos permitem expor as atividades, o fluxo de controle e a sequência de mensagens envolvidas na interação entre processos privados ou entre um processo privado e um participante envolvido no processo,

como mostrado no exemplo da Figura 3.7, na qual as interações entre a entidade Usuário e o subprocesso de devolução de exemplares emprestados da biblioteca são representadas. Processos colaborativos permitem representar os padrões de troca de mensagens nas interações entre dois ou mais processos de negócio (modelados como processos privados). Na Figura 3.8 é mostrada a representação do subprocesso de devolução de exemplares segundo a perspectiva de processos de colaboração.



Figura 3.6: Representação do subprocesso de devolução segundo a perspectiva de processo privado.

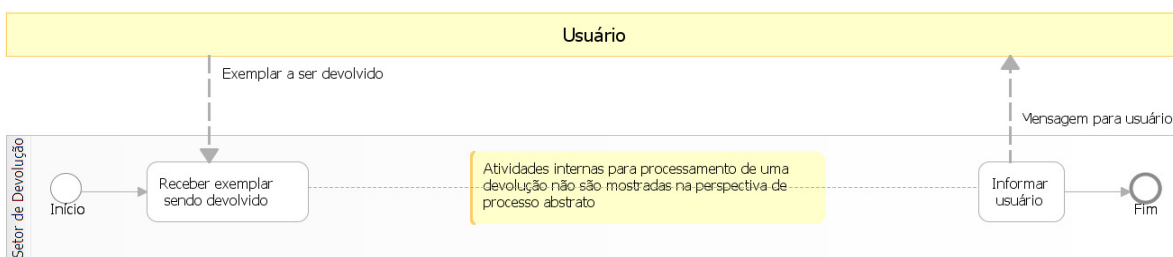


Figura 3.7: Representação do processo de devolução segundo a perspectiva de processo abstrato.

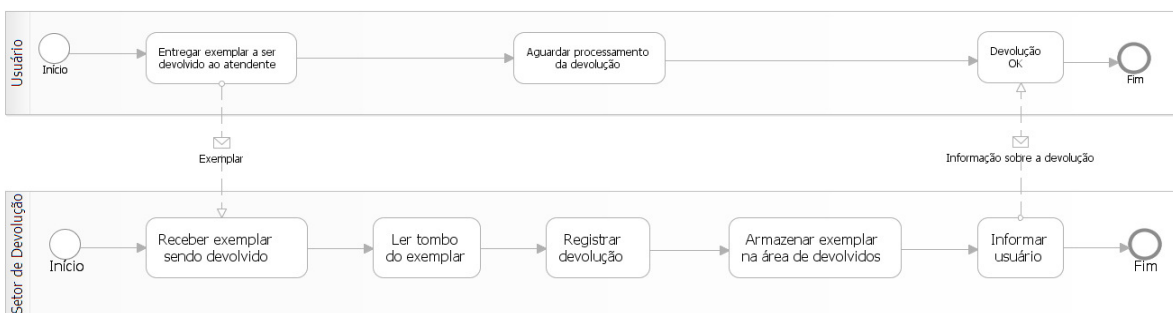


Figura 3.8: Representação do processo de devolução segundo a perspectiva de processo de colaboração.

Na Figura 3.9 é exemplificada a modelagem do subprocesso de empréstimo de exemplares de uma biblioteca com BPMN. Esse subprocesso envolve a interação do usuário com o atendente do setor de empréstimos, cada qual seguindo um processo, para efetivação do empréstimo. Os processos de negócio adotados pelo usuário e pelo setor de empréstimos são representados por meio de *pools* no diagrama de processos. Nesse exemplo, o usuário busca por informações sobre o

exemplar desejado, localiza-o nas prateleiras da biblioteca e leva-o até o balcão de atendimento para realizar o empréstimo. O funcionário do setor de empréstimos recebe o exemplar juntamente com a identificação do usuário e dá início ao processo de empréstimo. Para que um exemplar possa ser emprestado, o usuário deve estar cadastrado, sua situação deve ser regular e o exemplar sendo emprestado não pode ter reserva. Além disso, deve-se calcular a data de devolução e fazer o registro do empréstimo do exemplar ao usuário. Caso algum problema seja encontrado, o usuário é informado e toma as devidas ações. O fluxo de atividades representado para esse processo ilustra apenas o caminho principal, não representando caminhos alternativos que podem ser executados conforme o resultado (decisões) de algumas atividades.

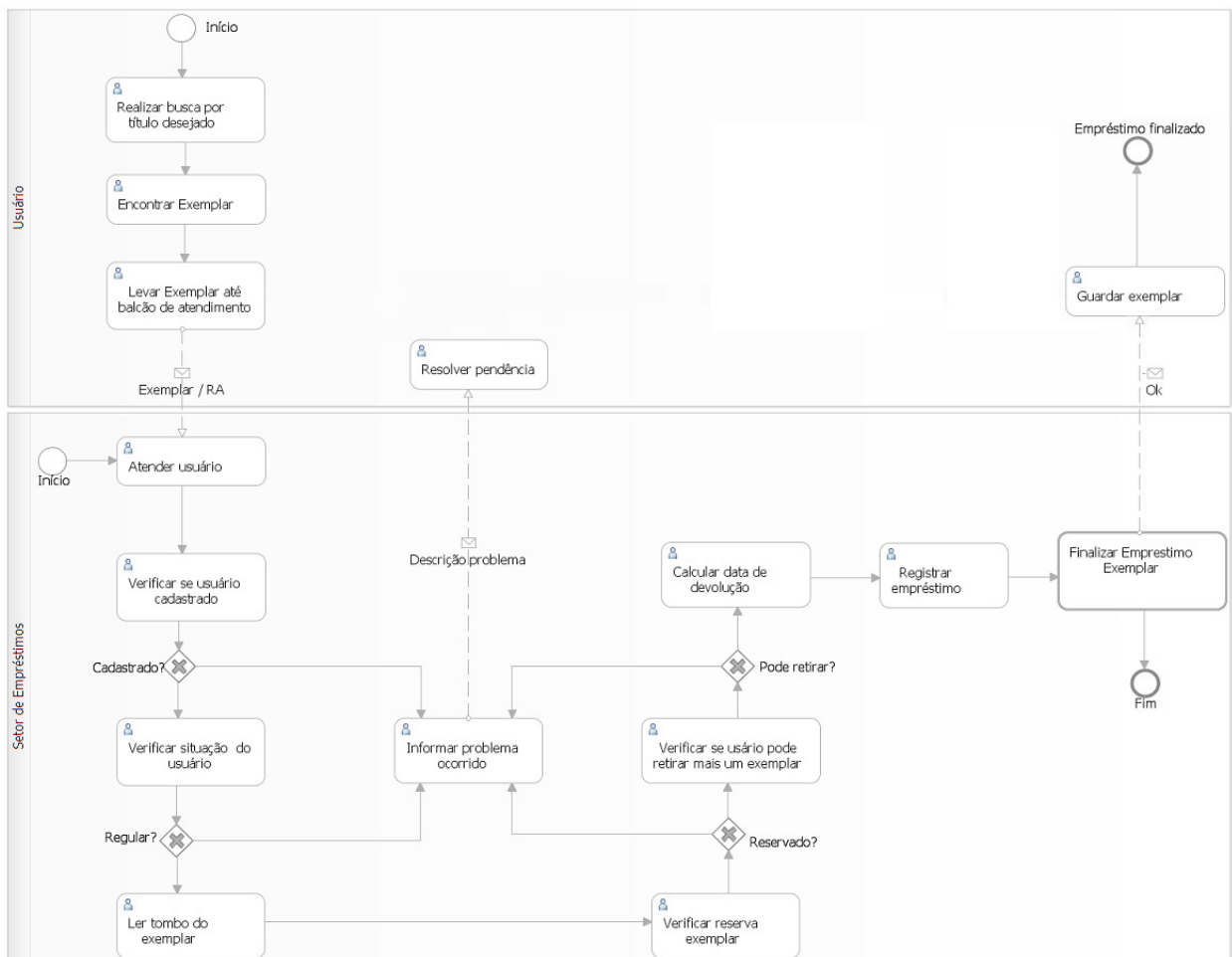


Figura 3.9: Modelagem do processo de empréstimo de exemplares de biblioteca usando BPMN.

Nesta seção foram apresentados resumidamente os principais elementos do BPMN para representação de processos. A especificação completa do BPMN define

trinta e oito elementos, sendo apresentados aqui apenas os pertinentes ao desenvolvimento desta dissertação. Sua especificação completa pode ser encontrada em OMG (2009a).

3.3 Considerações Finais

Este capítulo abordou a Modelagem de Processos de Negócio e a notação BPMN como técnica para realizar essa modelagem. Nota-se que o BPMN possui recursos e uma variedade de elementos capazes de representar processos de negócio adequadamente, contribuindo para que eles sejam mais bem compreendidos pelas pessoas envolvidas.

Dentre as principais características do BPMN podemos citar: a modelagem utilizando elementos visuais; a existência de elementos que permitem representar atores e papéis de um processo de negócio; a capacidade de representação de sequências de atividades de um processo de negócio; extensibilidade; a capacidade de representar processos sob diferentes perspectivas e interações entre esses processos; e a capacidade de representar o fluxo de informação dentro de um processo de negócio.

A escolha do BPMN para modelagem de processos de negócio nesta dissertação foi devido a: nos últimos anos, o BPMN ter se tornado uma técnica popular para modelagem de processos de negócio (RECHER *et al.*, 2006); o BPMN estar emergindo como uma linguagem padrão (reforçada pela organização internacional de padronização OMG) para representação de processos de negócio, especialmente no nível de análise de domínio e projeto de sistemas em alto nível (FERNÁNDEZ *et al.*, 2010). Além disso, Fernández *et al.* (2010) listam algumas vantagens do BPMN em comparação com outras linguagens ou notações utilizadas para modelagem de processos de negócio: o BPMN fornece uma técnica para modelagem do fluxo de um processo mais apropriada para uso pelos analistas de negócio; o BPMN possui uma base matemática sólida projetada para as linguagens de execução de processos de negócio; as dificuldades para utilizar outras linguagens de modelagem como UML.

De La Vara, Sánchez e Pastor (2008) afirmam que vários estudos avaliaram, por meio de comparações com outras notações de mesmo propósito, a adequação do BPMN para modelar processos de negócio e concluíram que seu uso oferece três vantagens principais: notação com maior poder de expressão; facilidade de uso e entendimento; e grande apoio de praticantes e empresas. Tais vantagens reforçam a adequação do uso do BPMN para modelagem de processos de negócio.

É importante ressaltar que além do BPMN, outras abordagens ou técnicas, como Diagrama de Atividades da UML, podem ser usadas para modelagem dos processos de negócio. Independente da abordagem ou técnica utilizada, o objetivo é representar os processos de negócio para que informações relacionadas ao processos possam ser facilmente visualizadas e compreendidas quando necessário.

Capítulo 4

IASWS- ABORDAGEM ITERATIVA PARA DESENVOLVIMENTO DE *SOFTWARE* UTILIZANDO *WEB* *SERVICES*

4.1 Considerações Iniciais

Os métodos/processos para desenvolvimento de sistemas de *software* orientados a serviço, apresentados no Capítulo 2, tem enfoque no desenvolvimento de sistemas compostos exclusivamente por serviços. Esses métodos/processos não mencionam a possibilidade de desenvolvimento de sistemas com incorporação gradual de serviços, ou seja, a criação de sistemas compostos tanto por serviços quanto por elementos tradicionais (código OO). A incorporação gradual de serviços favorece a adoção progressiva da Computação Orientada a Serviços (COS), possibilitando que conceitos, técnicas, ferramentas e tecnologias envolvidas sejam compreendidos e utilizados de forma adequada.

A Abordagem Iterativa para Desenvolvimento de *Software* utilizando Web Services (IASWS, em inglês, *Iterative Approach for Software Development using Web Services*) apoia o desenvolvimento de *software* orientado a objetos que faz uso de *web services*, de forma iterativa e interativa. Essa abordagem incorpora os princípios e conceitos da OO e da COS; a modelagem de processos de negócio; e ideias, conceitos e técnicas dos métodos ágeis, do modelo de processo Incremental

e dos métodos para desenvolvimento orientado a serviços. O conceito de *backlog* do *Scrum* (SCHWABER e BEEDLE, 2001) é utilizado, preservando-se esse termo, para priorizar a implementação dos processos de negócio e auxiliar no planejamento dos incrementos do *software*. Do modelo de processo Incremental é utilizado o conceito de incremento para possibilitar entregas constantes e a obtenção de *feedbacks* do cliente. Além disso, em todas as etapas da abordagem cuidou-se para que a documentação não fosse extensiva, considerando-se os modelos essenciais. Um diferencial desta abordagem em relação às demais existentes para desenvolvimento de *software* orientado a serviços é sua estratégia de incorporação gradual de serviços que permite que funções potencialmente reusáveis sejam publicadas sob a forma de serviços enquanto outras funções cujo potencial de reuso seja menor (ou inexistente) sejam implementadas de maneira tradicional. Outro diferencial consiste em favorecer o reuso de serviços nas fases de entendimento do problema (necessidade do cliente) e proposição de solução (*software*) para o problema. Além desses diferenciais, outro é a utilização da modelagem de processos de negócio para identificar e representar os requisitos do sistema. O ciclo de desenvolvimento idealizado na IASWS consiste de nove fases abrangem desde a identificação de requisitos até a verificação e aceitação do sistema produzido. Esta dissertação foca nas primeiras quatro fases - da identificação de requisitos à especificação de serviços e da solução – pois essas fases foram identificadas, neste trabalho, como essenciais para a utilização da Computação Orientada a Serviços nos sistemas de *software*.

Este capítulo está organizado da seguinte forma: na Seção 4.2 é apresentada a visão geral da abordagem IASWS, na Seção 4.3 são apresentadas as fases da IASWS abordadas nesta dissertação e na Seção 4.4 apresentam-se as considerações finais.

4.2 Abordagem IASWS

O ciclo de desenvolvimento representado no diagrama da Figura 4.1 apresenta as fases da abordagem IASWS. Elementos do BPMN do tipo tarefa (retângulos arredondados) representam as fases da IASWS e são interligados por

elementos do tipo fluxo de sequência (setas). O retorno a fases anteriores é indicado por um elemento do tipo fluxo de sequência rotulado com o número das fases para as quais é possível retornar. Em cada fase existem etapas (não mostradas no diagrama para simplificar a representação) que especificam o trabalho a ser realizado.

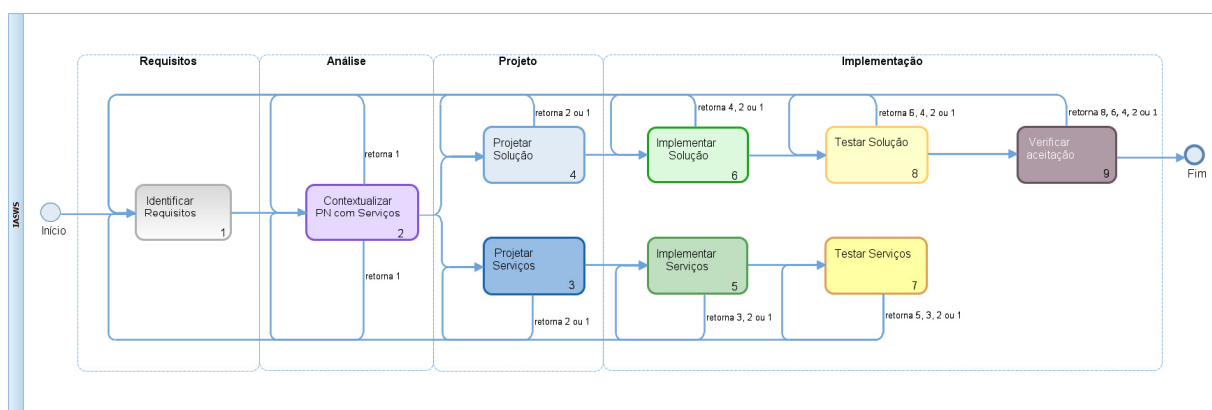


Figura 4.1 Ciclo de desenvolvimento da abordagem IASWS.

Na Tabela 4.1 as fases da IASWS são agrupadas a fim de relacioná-las com os estágios existentes em um ciclo de vida clássico para desenvolvimento de *software* (Requisitos, Análise, Projeto e Implementação). Na fase “Identificar Requisitos” os processos de negócio (PN) e os requisitos do *software* devem ser identificados, entendidos e documentados com o apoio da modelagem de processos de negócio. Na fase “Contextualizar PN com Serviços” é realizado o entendimento inicial do sistema de *software* (solução) a ser construído, enquanto nas fases “Projetar Solução” e “Projetar Serviços” são realizadas a especificação do sistema e dos novos serviços, respectivamente, que serão implementados nas fases “Implementar Solução” e “Implementar Serviços”, respectivamente. A solução proposta é testada na fase “Testar Solução” enquanto que os serviços desenvolvidos são testados na fase “Testar Serviços”. Por fim, na fase “Verificar aceitação” o cliente avalia se o sistema de *software* entregue está de acordo com o esperado. Iterações são criadas para cada processo de negócio a ser implementado, englobando as fases citadas anteriormente conforme necessário. Os serviços desenvolvidos a cada iteração podem ser catalogados em um repositório de serviços, caso exista, a fim de facilitar o seu reúso.

A abordagem foi concebida visando permitir que o desenvolvimento da solução e o de novos serviços ocorram de maneira independente, podendo ser realizados em paralelo. Essa decisão foi tomada tendo como base o trabalho de Haines e Rothenberger (2010) que afirmam que o desenvolvimento de *software* deve considerar a infraestrutura de serviços (incluindo seu desenvolvimento) e o desenvolvimento de aplicações compostas como duas tarefas distintas, com requisitos potencialmente diferentes.

Tabela 4.1: Fases da IASWS versus estágios do ciclo de vida clássico para desenvolvimento de *software*.

Estágio	Fase da IASWS	Descrição
Requisitos	Fase 1: Identificar Requisitos	Entender, identificar e documentar processos de negócio e os requisitos do sistema de <i>software</i> .
Análise	Fase 2: Contextualizar PN com Serviços	Entender o problema e propor uma solução utilizando serviços (disponíveis ou a serem desenvolvidos).
Projeto	Fase 3: Projetar Serviços	Projetar e especificar os serviços a serem desenvolvidos para fornecer funcionalidades requeridas na solução.
	Fase 4: Projetar Solução	Projetar o sistema de <i>software</i> .
Implementação	Fase 5: Implementar Serviços	Implementar os novos serviços que serão utilizados na solução.
	Fase 6: Implementar Solução	Implementar o sistema de <i>software</i> que irá automatizar os processos de negócio do cliente.
	Fase 7: Testar Serviços	Testar os novos serviços desenvolvidos.
	Fase 8: Testar Solução	Testar o sistema desenvolvido assegurando a correta implementação do processo de negócio.
	Fase 9: Verificar Aceitação	Entregar o sistema ao cliente e verificar se a solução atende suas necessidades.

A cada iteração da IASWS os processos de negócio implementados são entregues ao cliente. Essas entregas parciais permitem ao cliente perceber o progresso de desenvolvimento além de permitir ajustes nos processos de negócio. Em decorrência da iteratividade e das mudanças que ocorrem, o código fonte produzido necessita ser refatorado constantemente para: garantir a consistência dos conceitos de orientação a objetos e dos padrões de projeto usados; garantir que o desempenho da solução seja adequado às necessidades do cliente; e, principalmente, evitar inconsistências que levem o sistema de *software* a falhar.

4.3 Fases de Requisitos, Análise e Projeto da Abordagem Iterativa para Desenvolvimento de *Software* utilizando *Web Services* - IASWS

Esta dissertação trata detalhadamente das quatro primeiras fases da IASWS, abrangendo atividades de requisitos, análise e projeto pois nessas fases ocorrem a especificação dos serviços e a incorporação deles à solução.

Para a descrição das etapas das fases, o formato utilizado é o seguinte:

- Objetivo: informa qual o problema a ser resolvido;
- Descrição: apresenta a contextualização do problema;
- Solução: apresenta a proposição de uma solução para o problema.

O diagrama da Figura 4.2 apresenta, por meio do BPMN, as quatro fases da IASWS que serão abordadas nas próximas seções e suas respectivas etapas. Cada etapa é representada por elementos do tipo tarefa do BPMN e elementos do tipo fluxo de sequência interligam essas etapas, estabelecendo uma sequência de execução. As fases são representadas como agrupamentos dessas etapas por meio do elemento grupo do BPMN.

Na Tabela 4.2 é fornecida uma visão geral das quatro fases iniciais da IASWS. Na primeira coluna está apresentado o nome da etapa, na coluna dois, o objetivo pretendido com a etapa, na coluna três é listado o que deve ser executado para alcançar o objetivo de cada etapa (descrito como uma lista de passos a serem executados) e na última coluna são listados os artefatos gerados na etapa.

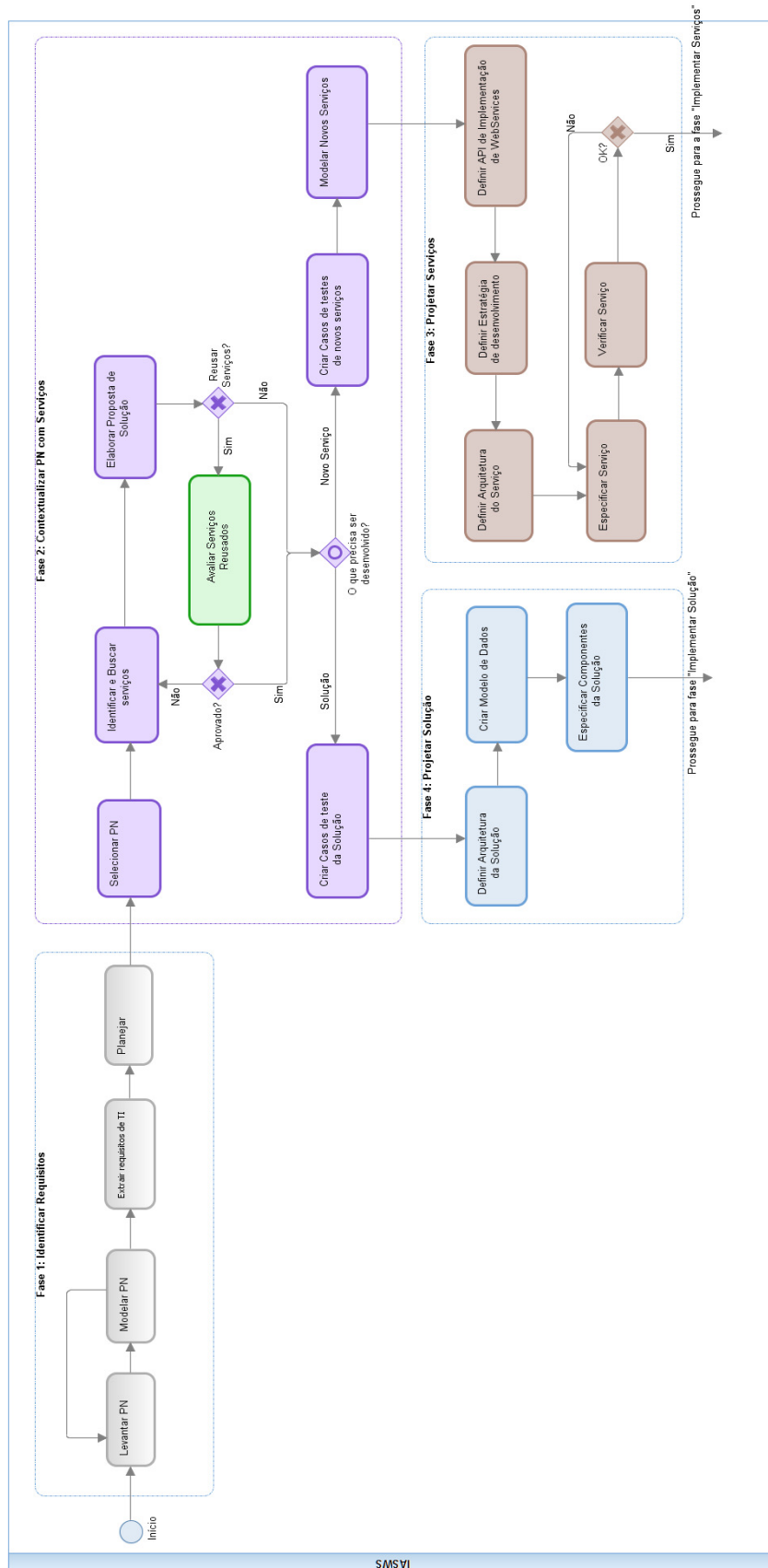


Figura 4.2 Etapas da Abordagem IASWS.

Tabela 4.2 Visão geral das fases e etapas da IASWS com as respectivas atividades.

Etapa	Objetivo	O que deve ser executado	Artefatos de saída	
Fase 1: Identificar Requisitos	1.1: Levantar PN	Identificar os processos de negócio que necessitam ser informatizados.	<ul style="list-style-type: none"> ▪ P1: Interagir com o cliente para obter informações sobre seu negócio, objetivos de negócio, processos de negócio, suas necessidades e dificuldades; ▪ P2: Identificar quais processos de negócio o cliente espera que sejam implementados no sistema; ▪ P3: Para cada processo de negócio: elencar suas atividades e determinar a sequência dessas atividades; ▪ P4: Obter informações detalhadas sobre esses processos. 	<ul style="list-style-type: none"> ▪ Processos de negócio a serem informatizados; ▪ Informações sobre esses processos.
	1.2: Modelar PN	Modelar e validar os processos de negócios a serem informatizados.	<ul style="list-style-type: none"> ▪ P1: Criar o Diagrama de Visão Geral de Processos; ▪ P2: Modelar processos (modelo <i>as-is</i>); ▪ P3: Apresentar modelo <i>as-is</i> ao cliente; ▪ P4: Realizar modificações na modelagem do processo a partir dos <i>feedbacks</i> do cliente. 	<ul style="list-style-type: none"> ▪ Diagrama de Visão Geral de Processos; ▪ Modelagem do processo de negócio (modelo <i>as-is</i>).
	1.3: Extrair requisitos de TI	Identificar os requisitos do sistema que será desenvolvido.	<ul style="list-style-type: none"> ▪ P1: Analisar o modelo <i>as-is</i> do processo de negócio; ▪ P2: Identificar atividades que podem ser informatizadas pelo sistema; ▪ P3: Refinar a modelagem, incluindo melhorias no processo (modelo <i>to-be</i>); ▪ P4: Documentar requisitos do sistema. 	<ul style="list-style-type: none"> ▪ Modelo <i>to-be</i> do processo de negócio (requisitos do sistema); ▪ Documento de Requisitos ou Diagrama de Casos de Uso ou outro artefato utilizado para registrar as informações sobre os processos de negócio.
	1.4: Planejar	Estabelecer o cronograma de entrega das implementações dos processos de negócio.	<ul style="list-style-type: none"> ▪ P1: Classificar os processo de negócio segundo sua prioridade de implementação; ▪ P2: Criar <i>Backlog</i> de Processos de Negócio; ▪ P3: Estimar data de entrega da implementação de cada processo de negócio. 	<ul style="list-style-type: none"> ▪ <i>Backlog</i> de Processos de Negócio; ▪ Cronograma de entregas parciais.

Etapa	Objetivo	O que deve ser executado	Artefatos de saída
2.1: Selecionar PN	Selecionar o processo de negócio a ser implementado na iteração.	<ul style="list-style-type: none"> ▪ P1: Remover o processo com a maior prioridade no <i>Backlog</i> de Processos de Negócio 	<ul style="list-style-type: none"> ▪ Próximo Processo de Negócio a ser implementado.
2.2: Identificar e buscar serviços	Identificar funcionalidades do sistema que podem ser fornecidas por serviços e verificar a disponibilidade desses serviços.	<ul style="list-style-type: none"> ▪ P1: Analisar as funcionalidades do sistema representadas no modelo <i>to-be</i> do processo de negócio; ▪ P2: Identificar serviços candidatos; ▪ P3: Buscar serviços; ▪ P4: Analisar provedores dos serviços encontrados; ▪ P5: Documentar serviços candidatos identificados e os serviços encontrados. 	<ul style="list-style-type: none"> ▪ Lista de serviços candidatos identificados.
2.3: Elaborar Proposta de Solução	Definir a estrutura da solução que atenderá os requisitos do sistema.	<ul style="list-style-type: none"> ▪ P1: Avaliar a viabilidade e a relação custo-benefício: reusar o serviço (caso encontrado); desenvolver um novo serviço; implementar a funcionalidade usando classes OO; ▪ P2: Elaborar proposta de solução definindo os serviços que serão reusados e os que serão desenvolvidos e as funcionalidades que serão implementadas por classes OO; ▪ P3: Analisar o tipo do sistema que implementará a solução proposta; ▪ P4: Revisar data de entrega; ▪ P5: Documentar serviços. 	<ul style="list-style-type: none"> ▪ Serviços a serem desenvolvidos; ▪ Serviços que serão reusados.
2.4: Avaliar serviços reusados	Avaliar se o serviço a ser reusado atende às necessidades da solução.	<ul style="list-style-type: none"> ▪ P1: Elaborar casos de teste; ▪ P2: Documentar casos de teste; ▪ P3: Construir protótipo para invocar o serviço; ▪ P4: Executar testes. 	<ul style="list-style-type: none"> ▪ Avaliação sobre adequação do serviço; ▪ Casos de testes para serviços que serão reusados; ▪ Protótipo para invocar o serviço.
2.5: Criar casos de testes de novos serviços	Elaborar casos de teste para testar os serviços que serão desenvolvidos. Consolidar o entendimento sobre o novo serviço.	<ul style="list-style-type: none"> ▪ P1: Elaborar casos de teste; ▪ P2: Documentar casos de teste. 	<ul style="list-style-type: none"> ▪ Casos de teste de novos serviços
2.6: Modelar Novos Serviços	Criar modelo, em nível de análise, para representar os serviços que serão desenvolvidos.	<ul style="list-style-type: none"> ▪ P1: Atribuir um nome ao serviço que identifique sua funcionalidade; ▪ P2: Identificar componentes do serviço; ▪ P3: Definir a interface do serviço; ▪ P4: Modelar serviço. 	<ul style="list-style-type: none"> ▪ Modelo de análise de novos serviços.
2.7: Criar Casos de Teste da Solução	Elaborar casos de teste para avaliar a solução desenvolvida. Consolidar o entendimento dos requisitos do sistema.	<ul style="list-style-type: none"> ▪ P1: Elaborar cenários de utilização do processo de negócio ▪ P2: Selecionar aqueles cenários que exercitam o processo de negócio de maneiras diferentes; ▪ P3: Elaborar casos de teste; ▪ P4: Documentar casos de teste. 	<ul style="list-style-type: none"> ▪ Casos de teste da solução

	Etapa	Objetivo	O que deve ser executado	Artefatos de saída
Fase 3: Projetar Serviços	3.1: Definir API de Implementação de Web Services	Definir a API de implementação de <i>Web Services</i> a ser adotada.	<ul style="list-style-type: none"> ▪ P1: Escolher a API de implementação de <i>Web Services</i> a ser utilizada na implementação dos serviços. 	<ul style="list-style-type: none"> ▪ API a ser utilizada para implementar os serviços.
	3.2: Definir Estratégia de Desenvolvimento	Definir a estratégia de desenvolvimento a ser adotada para implementar cada serviço.	<ul style="list-style-type: none"> ▪ P1: Escolher a estratégia de desenvolvimento mais adequada para implementação de cada um dos serviços. 	<ul style="list-style-type: none"> ▪ Estratégia de desenvolvimento dos serviços.
	3.3: Definir Arquitetura dos Serviços	Estabelecer a arquitetura dos serviços.	<ul style="list-style-type: none"> ▪ P1: Elaborar a arquitetura dos serviços que fornecem funcionalidades complexas; ▪ P2: Representar graficamente a arquitetura. 	<ul style="list-style-type: none"> ▪ Representação gráfica da arquitetura dos serviços.
	3.4: Especificar Serviços	Especificar os serviços a serem desenvolvidos.	<ul style="list-style-type: none"> ▪ P1: Refinar a modelagem dos serviços; ▪ P2: Detalhar operações da interface; ▪ P3: Modelar mensagens; ▪ P4: Modelar componentes internos. 	<ul style="list-style-type: none"> ▪ Modelagem dos serviços contendo a especificação dos serviços.
	3.5: Verificar Serviços	Verificar conformidade dos serviços especificados com relação aos princípios da orientação a serviços.	<ul style="list-style-type: none"> ▪ P1: Verificar se a especificação do serviço está em conformidade com os princípios da orientação a serviços. 	<ul style="list-style-type: none"> ▪ Avaliação sobre conformidade dos serviços com conceitos SOA.
Fase 4: Projetar Solução	4.1: Definir Arquitetura da Solução	Estabelecer a arquitetura da solução.	<ul style="list-style-type: none"> ▪ P1: Analisar o tipo do sistema que implementará a solução proposta; ▪ P2: Identificar os componentes necessários a esse sistema; ▪ P3: Organizar os componentes e determinar seus inter-relacionamentos; ▪ P4: Representar graficamente a arquitetura. 	<ul style="list-style-type: none"> ▪ Representação gráfica da arquitetura da solução.
	4.2: Criar Modelo de Dados	Modelar entidades de dados.	<ul style="list-style-type: none"> ▪ P1: Identificar os campos da entidade de dados e relacionamentos com outras entidades; ▪ P2: Mapear a entidade e seus relacionamentos para o modelo relacional; ▪ P3: Criar tabelas no banco de dados conforme com base no mapeamento. 	<ul style="list-style-type: none"> ▪ Modelos de dados da solução.
	4.3: Especificar Componentes da Solução	Especificar componentes da solução.	<ul style="list-style-type: none"> ▪ P1: Identificar as principais classes e seus relacionamentos; ▪ P2: Definir os métodos e atributos dessas classes; ▪ P3: Criar diagrama de classes da UML; ▪ P4: Refinar classes identificando interfaces, generalizações, polimorfismos, heranças e aplicando outros conceitos da orientação a objetos; ▪ P5: Aplicar os padrões de projeto adotados e representar elementos pertencentes ao <i>framework</i> (quando adotado). 	<ul style="list-style-type: none"> ▪ Especificações dos componentes da solução.

4.3.1 Fase 1: Identificar Requisitos

Nesta fase os processos de negócio do cliente são analisados para determinar quais desses devem ser informatizados no sistema de *software*. Esses processos são então modelados (utilizando BPMN) e seu entendimento junto ao cliente é verificado. Em seguida, os requisitos do sistema são extraídos e documentados. Por fim, é elaborado um cronograma com as datas de entrega da implementação de cada processo de negócio.

Nesta abordagem adotou-se a definição de processo de negócio fornecida por Yan *et al.* (2007), que definem processo de negócio como uma sequência de atividades que agregam valor a uma entrada para produzir uma saída com um determinado objetivo de negócio.

Quatro etapas compõem esta fase: “Levantar PN”, “Modelar PN”, “Extrair requisitos de TI” e “Planejar”. Essas etapas consomem tempo considerável durante a primeira iteração do ciclo de desenvolvimento pois há uma grande quantidade de informações a ser analisada. Nas iterações seguintes as etapas dessa fase são executadas apenas quando houver modificações nos requisitos.

4.3.1.1 Etapa 1.1: Levantar PN

Objetivo: Identificar os processos de negócio que necessitam ser informatizados.

Descrição: Identificar, a partir de informações fornecidas pelo cliente, quais são seus processos de negócio e quais necessitam ser informatizados.

Solução: Interagir com o cliente (**P1**) para: identificar sua área de negócios, seus objetivos de negócio e seus processos de negócio; entender seu negócio; e identificar suas dificuldades e necessidades. A obtenção dessas informações é feita por meio de entrevistas, reuniões, aplicação de questionários, observação da execução de atividades ou quaisquer outras técnicas que auxiliem a extração de informações do cliente. Caso o cliente não conheça seus processos de negócio, deve-se realizar o mapeamento (descoberta e definição) dos processos de negócio antes do início do desenvolvimento usando a IASWS.

Analisar as informações obtidas do cliente para identificar quais processos de negócio devem fazer parte do sistema a ser desenvolvido (**P2**). De forma geral, cada um desses processos define uma funcionalidade do sistema a ser desenvolvido.

Para cada um desses processos, elencar as atividades envolvidas no processo e determinar a sequência lógica que combina apropriadamente essas atividades (**P3**) para fornecer a funcionalidade do processo de negócio.

Obter informações detalhadas sobre as atividades, regras de negócios, condições de exceção e entidades de dados envolvidas nesses processos de negócio (**P4**).

Na Tabela 4.3 estão listados os artefatos de entrada e saída desta etapa.

Tabela 4.3 Artefatos de entrada e saída da etapa “Levantar PN”.

Artefatos de entrada	Artefatos de saída
<ul style="list-style-type: none"> ▪ Informações sobre o negócio do cliente. 	<ul style="list-style-type: none"> ▪ Processos de negócio a serem informatizados pelo sistema e informações sobre eles.

4.3.1.2 Etapa 1.2: Modelar PN

Objetivo: Modelar e validar os processos de negócios que necessitam ser informatizados.

Descrição: Os processos de negócio elicitados na etapa anterior são modelados com BPMN e apresentados ao cliente para validação ou alterações.

Solução: A partir dos processos de negócio que devem ser implementados, deve-se criar o Diagrama de Visão Geral de Processos (**P1**), contendo todos esses processos de negócio e os atores associados a eles. Um ator é a entidade que desempenha o papel de executar as atividades do processo. Um exemplo hipotético desse diagrama é exibido na Figura 4.3. Os processos PN1 e PN2 são executados pelo ator A1 enquanto PN3 é executado pelo ator A2. Elementos do tipo subprocesso representam cada um desses processos, enquanto que elementos do tipo *lane* foram utilizados para agrupar os processos de acordo com o ator.

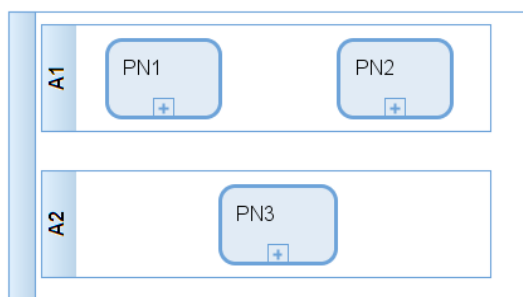


Figura 4.3 Exemplo de Diagrama de Visão Geral de Processos construído com BPMN.

Cada processo representado no Diagrama de Visão Geral de Processos, deve ser modelado em um diagrama de processo do BPMN (**P2**). Nesse diagrama, cada atividade do processo de negócio é representada por meio do elemento do tipo tarefa. A sequência das atividades de um processo é representada por meio de elementos do tipo fluxo de sequência interligando as atividades. Decisões e ramificações são representadas por meio do elemento do tipo *gateway* (ou, e, ou-exclusivo). Essa modelagem é o modelo *as-is* do processo de negócio do cliente.

O modelo *as-is* do processo de negócio PN1, exemplificado anteriormente, é mostrado na Figura 4.4, com as atividades que o compõem bem como a sequência em que devem ser executadas. Nesse exemplo observa-se que após a execução da Atividade 2 deve-se decidir por apenas um dos caminhos (ou-exclusivo) que leva à execução da Atividade 3.1 ou da Atividade 3.2. Por fim, a Atividade 4 é executada.

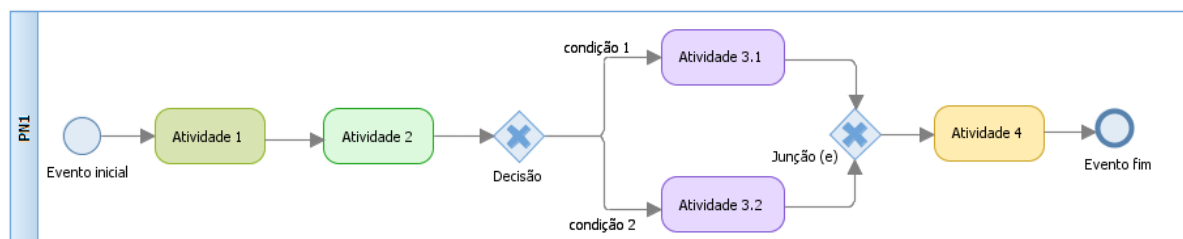


Figura 4.4 Modelo *as-is* do processo PN1 usando BPMN.

O nível de detalhamento (granularidade) das atividades pode variar conforme a experiência do desenvolvedor ou a complexidade do processo de negócio, mas deve ser suficiente para permitir que cliente e desenvolvedores compreendam o processo de negócio.

Em seguida, apresentar a modelagem dos processos de negócio ao cliente (**P3**) para corrigir possíveis falhas quanto ao entendimento dos processos e permitir

a identificação de pontos de melhorias. Modificações na modelagem do processo são realizadas a partir do *feedback* do cliente (**P4**) até que a modelagem do processo represente fielmente a forma como o cliente trabalha.

Na Tabela 4.4 estão listados os artefatos de entrada e saída desta etapa.

Tabela 4.4 Artefatos de entrada e saída da etapa “Modelar PN”.

Artefatos de entrada	Artefatos de saída
<ul style="list-style-type: none"> ▪ Processos de negócio a serem informatizados pelo sistema e informações sobre esses processos. 	<ul style="list-style-type: none"> ▪ Diagrama de Visão Geral de Processos; ▪ Modelos <i>as-is</i> dos processos de negócio.

4.3.1.3 Etapa 1.3: Extrair Requisitos de TI

Objetivo: Identificar os requisitos do sistema que será desenvolvido.

Descrição: Identificar quais atividades do processo de negócio (identificadas na etapa 1.1) podem/necessitam ser informatizadas. Em seguida, com base nessas atividades, definir os requisitos do sistema a ser implementado e documentar esses requisitos em um documento de requisitos (ou em um diagrama de casos de uso).

Solução: Para cada processo de negócio que necessita ser informatizado, analisar seu modelo *as-is* (**P1**) e identificar as atividades que podem ser completamente informatizadas e as que podem ser auxiliadas por um sistema de informação (**P2**). Nesse último caso o sistema implementará apenas parte da atividade, sendo necessária intervenção humana para completar tal atividade.

Uma vez que essas atividades foram identificadas, é preciso refinar a modelagem do processo (**P3**) para representá-las como requisitos do sistema. Para isso, adiciona-se à modelagem um elemento do tipo linha, rotulado como “Sistema”. Em seguida movem-se todas as atividades que serão executadas pelo sistema para sua linha. Nos casos em que o sistema apenas auxilia na execução de parte da atividade, deve-se especificar o que é realizado pelo sistema e o que é realizado por uma pessoa (ou outro sistema). O resultado do refinamento da modelagem é o modelo *to-be*. Melhorias no modelo *to-be* podem ser realizadas a fim de otimizar o processo considerando as atividades que serão informatizadas. As atividades do processo de negócio que serão executadas pelo sistema definem as funções que o sistema deverá possuir para ser capaz de fornecer a funcionalidade do processo de negócio.

Na Figura 4.5 é apresentado o modelo *to-be* do processo PN1, com dois elementos do tipo linha: um para representar o “Sistema” e outro para o ator A1. As

atividades 1 e 4 devem ser executadas pelo ator A1 enquanto as atividades 2 e 3.2 podem ser automatizadas pelo sistema. Já a atividade 3.1 é executada parcialmente pelo sistema (3.1 A) e é finalizada pelo ator A1 (3.1 B). Além dessas, outras atividades extras devem ser adicionadas para que atores e sistema interajam apropriadamente no decorrer da execução do processo, como é o caso da atividade “Interação com ator A1”. Esse tipo de interação ocorre quando o sistema necessita que alguma informação seja fornecida pelo ator ou deve indicar ao ator que é de sua responsabilidade a execução de uma atividade.

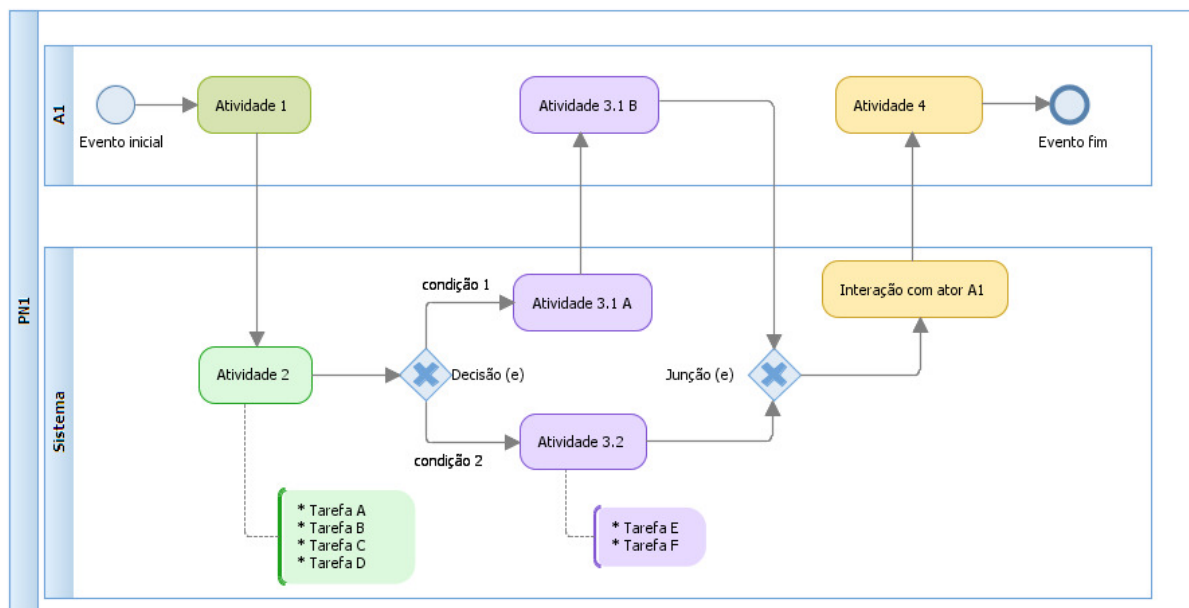


Figura 4.5: Modelo *to-be* do processo PN1.

O modelo *to-be* deve conter informações suficientes para subsidiar o desenvolvimento do sistema. Caso haja necessidade de especificar detalhes de uma atividade do processo de negócio podem ser utilizados textos, ou seja, o elemento anotação conectada à atividade. De forma análoga, anotações podem ser usadas para explicitar regras de negócio e entidades de dados envolvidas no processo de negócio e requisitos não funcionais.

Além de representar graficamente as atividades que serão executadas pelo sistema é preciso documentar detalhadamente os requisitos do sistema (**P4**). Assim, pode ser criado um documento de requisitos que contém informações textuais sobre as atividades que serão implementadas pelo sistema. Outra possibilidade é a criação de um diagrama de casos de uso, sendo que um caso de uso representa um processo de negócio e sua descrição irá detalhar suas atividades,.

Na Tabela 4.5 estão listados os artefatos de entrada e saída desta etapa.

Tabela 4.5 Artefatos de entrada e saída da etapa “Extrair Requisitos de TI”.

Artefatos de entrada	Artefatos de saída
<ul style="list-style-type: none"> ▪ Modelos <i>as-is</i> dos processos de negócio. 	<ul style="list-style-type: none"> ▪ Modelos <i>to-be</i> dos processos de negócio, representando requisitos do sistema. ▪ Documento de Requisitos com detalhamento dos requisitos do sistema ou Diagrama de Casos de Uso e as respectivas descrições ou outro artefato utilizado para registrar as informações sobre os requisitos dos sistema.

4.3.1.4 Etapa 1.4: Planejar

Objetivo: Estabelecer o cronograma de entrega das implementações dos processos de negócio.

Descrição: A equipe de desenvolvimento e o cliente trabalham em conjunto para definir a ordem de implementação dos processos de negócio. Para cada processo de negócio, a equipe de desenvolvimento realiza uma análise dos requisitos e determina uma data provável para a entrega da sua implementação. Para determinar essas datas devem ser considerados: a experiência da equipe, a complexidade do processo de negócio, a complexidade da implementação, as tecnologias, o ambiente de desenvolvimento, entre outros.

Solução: A equipe de desenvolvimento, com ajuda do cliente, classifica os processos de negócio segundo sua prioridade de implementação (**P1**). A priorização considera o valor de negócio do processo e a urgência do cliente para sua implementação. Quanto maior a prioridade, mais cedo o processo será implementado. Em projetos complexos, pode-se minimizar os riscos atribuindo uma prioridade alta aos processos mais complexos ou com maior probabilidade de levar a riscos durante o desenvolvimento.

Em seguida, cria-se o *Backlog* de Processos de Negócio (**P2**) - uma lista contendo os processos de negócio ordenados segundo suas prioridades.

Para cada processo no *Backlog* de Processos de Negócio, estima-se a data de entrega (**P3**) da primeira versão da sua implementação. Essas datas são determinadas pela equipe de desenvolvimento a partir de discussões sobre o esforço de desenvolvimento necessário para implementar o processo e levam em consideração a experiência dos membros da equipe de desenvolvimento, os requisitos do sistema, a complexidade do trabalho a ser realizado, possíveis

dificuldades que possam existir, tecnologias e o ambiente de desenvolvimento. Caso a empresa já utilize alguma técnica para estimar prazos, esta pode ser utilizada.

Caso o cliente solicite modificações na implementação de um processo de negócio após a sua entrega, os requisitos e a modelagem também devem ser alterados e o planejamento deve ser revisado. A priorização do *backlog* também poderá ser modificada, considerando que um processo que já foi retirado do *backlog* (implementado pela equipe), deverá ser inserido novamente para que as correções possam ser implementadas.

Na Tabela 4.6 estão listados os artefatos de entrada e saída desta etapa.

Tabela 4.6 Artefatos de entrada e saída da etapa “Planejar”

Artefatos de entrada	Artefatos de saída
<ul style="list-style-type: none"> ▪ Diagrama de Visão Geral de Processos; ▪ Modelos <i>to-be</i> dos processos de negócio, representando requisitos do sistema; ▪ Documento de Requisitos com detalhamento dos requisitos do sistema ou Diagrama de Casos de Uso e as respectivas descrições ou outro artefato utilizado para registrar as informações sobre os requisitos do sistema. 	<ul style="list-style-type: none"> ▪ <i>Backlog</i> de Processos de Negócio; ▪ Cronograma de entregas parciais

4.3.2 Fase 2: Contextualizar PN com Serviços

Uma vez identificados os requisitos do sistema a ser desenvolvido, é necessário propor uma solução que atenda a esses requisitos. Nesse contexto, solução é um conjunto de funções do sistema que atendem os requisitos do sistema.

A partir da funcionalidade do sistema são identificados os serviços que podem ser reusados na solução e os que podem ser desenvolvidos. Após essa identificação, pode-se ter uma visão geral da estrutura da solução, sendo possível uma das seguintes combinações:

- Solução OO sem uso de serviços;
- Solução OO reusando serviços já desenvolvidos;
- Solução OO utilizando serviços a serem desenvolvidos;
- Solução OO reutilizando serviços já desenvolvidos e serviços a serem desenvolvidos;
- Solução completamente orientada a serviços.

Sete etapas compõem esta fase: “Selecionar PN”, “Identificar e Buscar Serviços”, “Elaborar Proposta de Solução”, “Avaliar Serviços Reusados”, “Criar

Casos de Teste de Novos Serviços”, “Modelar Novos Serviços” e “Criar Casos de Teste da Solução”.

4.3.2.1 Etapa 2.1: Selecionar PN

Objetivo: Selecionar o processo de negócio a ser implementado na iteração.

Descrição: Consultar o *Backlog* de Processos de Negócio para obter o próximo processo a ser implementado.

Solução: Remover o processo com a maior prioridade no *Backlog* de Processos de Negócio (**P1**) para ser implementado na iteração atual. É importante ressaltar que o número de processos de negócio a serem implementados a cada iteração depende exclusivamente da decisão da equipe de desenvolvimento e deve levar em consideração o tempo de duração da iteração.

Na Tabela 4.7 estão listados os artefatos de entrada e saída desta etapa.

Tabela 4.7 Artefatos de entrada e saída da etapa “Selecionar PN”

Artefatos de entrada	Artefatos de saída
▪ <i>Backlog</i> de Processos de Negócio.	▪ Próximo processo de negócio a ser implementado.

4.3.2.2 Etapa 2.2: Identificar e Buscar Serviços

Objetivo: Identificar funções do sistema que podem ser fornecidas por serviços e verificar a disponibilidade desses serviços.

Descrição: Dentre as funções do sistema, algumas podem ser fornecidas por serviços. Essas funções devem ser identificadas para em seguida buscar por serviços que as forneçam, em repositórios de serviços, internet, portfólios de provedores de serviços conhecidos ou qualquer outra fonte disponível para buscar por serviços.

Solução: Analisar os requisitos do sistema (funções do sistema) representados no modelo *to-be* (**P1**) do processo de negócio que está sendo implementado. Em seguida, identificam-se os serviços candidatos (**P2**). Serviços candidatos são funções com potencial de reuso e que tem grandes chances de serem fornecidas por serviços (já implementados ou a serem desenvolvidos). Haines e Rothenberger (2010) ressaltam que é preciso equilibrar a granularidade de um serviço, uma vez que o uso de uma grande quantidade de serviços de granularidade pequena pode afetar o desempenho do sistema devido às inúmeras invocações

necessárias enquanto que serviços de granularidade maior acabam limitando seu reúso devido às suas especificidades.

Na Tabela 4.8 estão listadas as principais categorias de funções a serem consideradas na identificação de serviços candidatos. Cada uma das categorias listadas representa um nível de granularidade diferente e por este motivo, é recomendável iniciar a identificação pela primeira linha da tabela (maior granularidade) e prosseguir sequencialmente, conforme a granularidade diminui.

Tabela 4.8 Fontes para identificação de serviços.

Escopo	Descrição
Processo de negócio	Funcionalidade fornecida pela execução do processo de negócio.
	Funcionalidade de granularidade maior que engloba a funcionalidade do processo de negócio sendo implementado.
Atividade de um processo de negócio	Função desempenhada por uma atividade (elemento do tipo tarefa, subprocesso ou <i>gateway</i> do BPMN) do processo de negócio.
Agrupamento de atividades	Função desempenhada por um agrupamento de atividades do processo de negócio.
Regras de negócio	Função associada a regras de negócio.
Entidades de dados	Função relacionada à manipulação de uma entidade de dados.
Requisitos não-funcionais	Função que pode ser utilizada para endereçar alguns tipos de requisitos não-funcionais.

Técnicas (DWIVEDI e KULKARNI, 2008; INAGANTI e BEHARA, 2007) para identificação de serviços a partir de modelos de processo de negócio podem ser utilizadas ou adaptadas para melhorar os resultados da identificação de serviços candidatos.

Para cada função identificada (serviço candidato) devem-se buscar serviços capazes de atendê-la (**P3**). Para essa busca são utilizadas palavras-chave relacionadas à função identificada e é realizada de forma manual em repositórios de serviços, internet, portfólios de provedores de serviços conhecidos ou em qualquer outro local onde os serviços foram publicados, tais como páginas *web*, *wikis*, *blogs*, etc. A busca por serviços deve ser realizada primeiramente no escopo mais interno, por exemplo a intranet da empresa, passando por provedores de serviço conhecidos, até chegar ao escopo mais geral, a internet. Essa diretriz permite que

serviços desenvolvidos por outros departamentos da empresa possam ser reutilizados, maximizando o retorno do investimento da adoção de serviços.

Os possíveis cenários para os resultados das buscas são:

1. A função é fornecida por uma ou mais operações de um mesmo *web service*;
2. A função é fornecida por um conjunto de operações de *web services* distintos que devem ser invocados de forma coordenada e sincronizada (reúso parcial com composição de serviços);
3. Parte da função é fornecida por uma ou mais operações de um ou mais *web services*;
4. A função não é fornecida por nenhum serviço.

A abordagem proposta considera que resultados como os do cenário três devem ser descartados pois os benefícios de um reúso deste tipo são questionáveis devido a apenas parte da função ser fornecida. O caso número dois representa o cenário em que a composição de serviços deve ser utilizada para fornecer uma funcionalidade. Apesar de tecnologias como coreografia e orquestração permitirem a organização e sincronização dos serviços que participam da composição, a abordagem proposta considera que os serviços participantes da composição podem ser organizados e sincronizados utilizando-se código orientado a objetos de forma a reduzir custos com *middleware*, treinamento e pessoal qualificado, inerentes ao uso das tecnologias citadas anteriormente.

Os provedores dos serviços devem ser analisados (**P4**) quanto: à idoneidade, à segurança dos dados, aos custos, à disponibilidade e manutenção dos serviços, entre outros. Em casos de serviços fornecidos por outro departamento da empresa, deve-se assegurar a existência de uma política de governança de serviços para evitar problemas futuros como descontinuidade do serviço ou não disponibilidade do serviço em algum momento.

A identificação de serviços candidatos e a busca de serviços são altamente iterativos, dinâmicos e fortemente influenciados pela experiência dos desenvolvedores. Desenvolvedores com experiência em desenvolvimento utilizando serviços são capazes de identificar as funções para as quais a busca por um serviço pode retornar bons resultados, simplificando o trabalho de identificação e busca de serviços.

Documentar os serviços candidatos identificados e os serviços encontrados (P5). Na Tabela 4.9 é apresentado um exemplo de artefato para documentação dessas informações. Nessa tabela são mostrados os resultados (fictícios) da identificação de serviços candidatos e respectiva busca para o processo PN1. Para cada serviço candidato foi adicionada uma nova linha na tabela contendo o nome da funcionalidade (coluna um) identificada, o resultado da busca (coluna dois) e informações sobre o serviço (colunas três), caso este seja encontrado.

Tabela 4.9 Documentação de Serviços candidatos identificados.

	Funcionalidade	Serviço encontrado?	Informações sobre serviço (caso encontrado)
Processo de negócio PN1	Funcionalidade do processo PN1	Não	
	Função desempenhada pela Atividade 2	Sim	Link para o WSDL do <i>web service</i> . Custo: R\$10,00 a cada 100 transações
	Função desempenhada pela Atividade 3.1 A	Não	
	Função desempenhada pela Tarefa F da Atividade 3.2	Sim	Link para a página com a descrição do <i>web service</i> Custo: gratuito (intranet)

Na Tabela 4.10 estão listados os artefatos de entrada e saída desta etapa.

Tabela 4.10 Artefatos de entrada e saída da etapa “Identificar e Buscar Serviços”.

Artefatos de entrada	Artefatos de saída
<ul style="list-style-type: none"> ▪ Modelo <i>to-be</i> do processo de negócio, representando requisitos do sistema; ▪ Documento de Requisitos com detalhamento dos requisitos do sistema ou Diagrama de Casos de Uso e as respectivas descrições ou outro artefato utilizado para registrar as informações sobre os requisitos dos sistema. 	<ul style="list-style-type: none"> ▪ Lista de serviços candidatos identificados.

4.3.2.3 Etapa 2.3: Elaborar Proposta de Solução

Objetivo: Definir a estrutura da solução que atenda aos requisitos do sistema.

Descrição: A Equipe de desenvolvimento e o cliente trabalham em conjunto para decidir quais funções do sistema serão fornecidas por serviços (reusando ou desenvolvendo o serviço) e quais serão fornecidas por classes OO tradicionais.

O cliente deve participar da elaboração da solução, pois o reúso de um serviço pode envolver custos para sua utilização enquanto o desenvolvimento de um

novo serviço pode envolver custos extras de desenvolvimento (comparado ao desenvolvimento OO tradicional) .

Solução: Para cada serviço candidato identificado na etapa anterior deve-se avaliar a viabilidade e a relação custo-benefício (**P1**) entre as seguintes opções: reusar o serviço disponível (caso encontrado); desenvolver um novo serviço que forneça a funcionalidade; ou implementar a função usando classes OO.

A avaliação da viabilidade e da relação custo-benefício para o desenvolvimento de novos serviços deve considerar: o tempo necessário para implementar o serviço; a complexidade da implementação; os riscos associados; a infraestrutura necessária; a governança de serviços; a granularidade do serviço; o nível de reusabilidade esperado; a importância em fornecer a função como um serviço; as ferramentas de desenvolvimento; os custos relacionados à capacitação técnica necessária para implementação de serviços; entre outros.

Analogamente, durante a análise da viabilidade de reuso dos serviços encontrados é importante considerar: o acordo de nível de serviço (SLA); o custo pela utilização do serviço; a confiabilidade; a escalabilidade; a disponibilidade; a qualidade da documentação existente; o suporte técnico; entre outros.

Com base nas avaliações obtidas elabora-se uma proposta de solução (**P2**), definindo os conjuntos de serviços que serão reusados, de serviços que serão desenvolvidos e das funções que serão implementadas por classes OO (incluindo as que não foram identificadas como serviços candidatos na etapa anterior). Além disso, essa solução deve definir o tipo do sistema (*client-server*, *web*, *batch*, distribuído, *desktop*, etc.) (**P3**) que irá ser implementado.

Após a elaboração da solução é necessário revisar a data de entrega (**P4**) da implementação do processo de negócio considerando as decisões tomadas nessa etapa em relação à utilização e desenvolvimento de serviços.

Todos os serviços que farão parte da solução devem ser documentados (**P5**), por exemplo, como mostrado na Tabela 4.11, que contém a identificação do serviço, uma breve descrição desse serviço, as descrições das funções do sistema que são fornecidas pelo serviço, a estimativa de custos (custos para a implementação do novo serviço ou custos relacionados ao reuso do serviço) e a URL do arquivo de descrição de *web services* (caso o serviço a ser reusado possua um).

Tabela 4.11 Documentação para serviços.

Serviço:	Nome do serviço a ser desenvolvido ou reusado.
Descrição do serviço:	Breve descrição do serviço.
Funcionalidade atendida:	Nome da função fornecida por esse serviço.
Descrição da funcionalidade atendida:	Descrição da função fornecida por esse serviço.
Custos	Para um novo serviço que será desenvolvido: informar a quantidade de horas necessárias para implementar o serviço. Para um serviço que será reusado: informar custos envolvidos no uso do serviço.
URL para o WSDL do serviço	Endereço para o arquivo de descrição do serviço (WSDL)

Na Tabela 4.12 estão listados os artefatos de entrada e saída desta etapa.

Tabela 4.12 Artefatos de entrada e saída da etapa “Elaborar Proposta de Solução”.

Artefatos de entrada	Artefatos de saída
<ul style="list-style-type: none"> ▪ Lista de serviços candidatos identificados; ▪ Modelo <i>to-be</i> do processo de negócio, representando requisitos do sistema; ▪ Documento de Requisitos com detalhamento dos requisitos do sistema ou Diagrama de Casos de Uso e as respectivas descrições ou outro artefato utilizado para registrar as informações sobre os requisitos dos sistema. 	<ul style="list-style-type: none"> ▪ Serviços a serem desenvolvidos; ▪ Serviços que serão reusados.

4.3.2.4 Etapa 2.4: Avaliar Serviços Reusados

Objetivo: Avaliar se o serviço a ser reusado atende às necessidades da solução.

Descrição: A busca por serviços foi realizada manualmente e com base nas descrições textuais dos serviços. Essas descrições não possuem formatação ou estrutura específicas, o que pode levar a interpretações incorretas sobre a funcionalidade fornecida por um serviço. Por esse motivo é preciso avaliar os serviços que serão reusados para assegurar que sua funcionalidade atende às necessidades da solução.

A avaliação consiste na construção de um protótipo para verificar o comportamento do serviço e que permita avaliar a viabilidade do uso desse serviço na solução.

Solução: Para cada serviço que será reusado deve-se elaborar casos de teste (P1) para validar a funcionalidade fornecida pelo serviço visando assegurar que seu comportamento está de acordo com sua descrição. A elaboração dos casos de testes deve considerar os cenários em que o serviço pode falhar e os cenários

em que não há falhas. É importante ressaltar que os testes devem verificar se o serviço retorna resultados que sejam compatíveis com o comportamento esperado para a função do sistema que irá usar o serviço. Caso os resultados sejam incompatíveis, avaliar a possibilidade de tratar os dados de entrada e/ou saída de forma a possibilitar o reuso do serviço mediante adequação dos dados.

Os casos de testes para serviços focam na verificação da funcionalidade fornecida pelo serviço, não considerando detalhes de sua implementação, podendo, portanto, ser comparados a testes funcionais ou de caixa-preta. Por esse motivo, técnicas para elaboração de casos de teste do tipo caixa-preta (PRESSMAN, 2006) podem ser utilizadas para facilitar o trabalho de elaboração dos casos de teste e garantir maior cobertura.

Após a elaboração dos casos de teste, deve-se documentá-los (**P2**), por exemplo, como mostrado na Tabela 4.13. A primeira linha identifica o serviço a ser reusado. Para cada caso de teste, uma nova linha deve ser adicionada contendo o número do caso de teste, uma breve descrição do caso de teste, os valores de entrada e as saídas esperadas em resposta à entrada fornecida.

Tabela 4.13 Modelo para documentar casos de testes de serviço

Serviço: WebServiceExemplo1			
# caso de teste	Descrição	Valores de entrada	Saída esperada
1	Testa resposta a valor de entrada válido	123	123
2	Testa resposta a valor de entrada inválido	Abc	Mensagem informando erro.

Em seguida passa-se para a construção do protótipo (**P3**) que acionará o serviço. Esse protótipo é um trecho de código capaz de invocar o serviço fornecendo-lhe entradas e recebendo os resultados do processamento. Sua codificação pode ser realizada em qualquer linguagem/plataforma, no entanto, o uso da linguagem/plataforma adotada para implementação do sistema possibilita o reuso do código na fase de implementação. Pode ser necessário criar código extra para adaptar ou tratar os dados de entrada/saída para viabilizar a reutilização do serviço no contexto da solução em desenvolvimento.

Executar os casos de teste utilizando o protótipo construído (**P4**). Caso o serviço falhe em fornecer a funcionalidade, pode-se voltar à etapa “Identificar e Buscar Serviços” para tentar encontrar outros serviços.

Nesse ponto do trabalho de desenvolvimento, a arquitetura da solução ainda não foi definida, portanto, o foco dessa etapa é verificar se é possível reusar o serviço e como usá-lo. Caso seja possível reusar o serviço, este será encaixado na arquitetura da solução na etapa de projeto da solução.

Na Tabela 4.14 estão listados os artefatos de entrada e saída desta etapa.

Tabela 4.14 Artefatos de entrada e saída da etapa “Avaliar Serviços Reusados”.

Artefatos de entrada	Artefatos de saída
<ul style="list-style-type: none"> ▪ Serviços que serão reusados; ▪ Modelo <i>to-be</i> do processo de negócio, representando requisitos do sistema; ▪ Documento de Requisitos com detalhamento dos requisitos do sistema ou Diagrama de Casos de Uso e as respectivas descrições ou outro artefato utilizado para registrar as informações sobre os requisitos dos sistema. 	<ul style="list-style-type: none"> ▪ Avaliação sobre adequação do serviço; ▪ Casos de testes para serviços que serão reusados; ▪ Protótipo para invocar o serviço.

4.3.2.5 Etapa 2.5: Criar Casos de Teste de Novos Serviços

Objetivo: Elaborar casos de teste para testar os serviços que serão desenvolvidos e consolidar o entendimento sobre os novos serviços.

Descrição: Serviços que serão desenvolvidos para fornecer funcionalidades com potencial de reúso devem ser amplamente testados pois a ocorrência de um erro pode impactar diversos sistemas que utilizem o serviço.

Nesta etapa são criados os casos de testes que serão utilizados na fase de testes para avaliar o serviço em relação a aspectos como funcionalidade, desempenho, resposta a condições de erro, resiliência, entre outros.

Solução: Para cada serviço que será desenvolvido deve-se elaborar casos de teste (**P1**) para validar sua funcionalidade. Além de testar a funcionalidade do serviço, pode-se verificar a estrutura lógica do código que implementa o serviço, garantindo cobertura de código em nível adequado.

Caso haja necessidade, pode-se interagir com o cliente para obter informações adicionais ou esclarecer dúvidas sobre os requisitos ou processos de negócio.

Os casos de teste elaborados devem ser documentados (**P2**), podendo-se utilizar a Tabela 4.13, apresentada anteriormente para documentar os casos de testes de serviços reusados.

Na Tabela 4.15 estão listados os artefatos de entrada e saída desta etapa.

Tabela 4.15 Artefatos de entrada e saída da etapa “Criar Casos de Teste de Novos Serviços”.

Artefatos de entrada	Artefatos de saída
<ul style="list-style-type: none"> ▪ Serviços a serem desenvolvidos; ▪ Modelo <i>to-be</i> do processo de negócio, representando requisitos do sistema; ▪ Documento de Requisitos com detalhamento dos requisitos do sistema ou Diagrama de Casos de Uso e as respectivas descrições ou outro artefato utilizado para registrar as informações sobre os requisitos dos sistema. 	<ul style="list-style-type: none"> ▪ Casos de testes de novos serviços.

4.3.2.6 Etapa 2.6: Modelar Novos Serviços

Objetivos: Criar o modelo de análise de novos serviços.

Descrição: Após identificar as funções do sistema que possuem potencial de reúso e decidir quais dessas devem ser fornecidas por meio de serviços, é necessário nomear os serviços, identificar seus componentes e definir suas interfaces. A interface de um serviço define a forma para se invocar o serviço. Os componentes de um serviço fornecem uma visão geral da estrutura interna do serviço, definindo a organização do código que implementará o serviço.

Solução: Nesta etapa devem ser complementadas as informações documentadas na etapa “Elaborar proposta de solução”, sobre os serviços que serão desenvolvidos. Para cada serviço que será desenvolvido deve-se: atribuir um nome (**P1**) relacionado à funcionalidade fornecida; identificar seus componentes principais (**P2**); e definir sua interface (**P3**). Componentes de um serviço são funções de granularidade menor e de escopo limitado. A interface de um serviço especifica o contrato utilizado para comunicação com o serviço.

O perfil SoaML é utilizado para modelar os serviços que serão desenvolvidos, suas interfaces, seus componentes e inter-relacionamentos entre esses elementos. O elemento “*Capability*” do perfil SoaML é o mais apropriado para representar um serviço e seus componentes, permitindo representar a funcionalidade fornecida pelo serviço. Na Figura 4.6 é mostrada a modelagem do Serviço XY e dos seus respectivos componentes. O Serviço XY fornece sua funcionalidade por meio da operação acionarFuncionalidadeXY(). Duas funções são necessárias para o Serviço XY: a FunçãoX e a FunçãoY, ambas representadas utilizando-se elementos “*Capability*” e conectados ao Serviço XY por meio de uma associação com o estereótipo “*use*”.

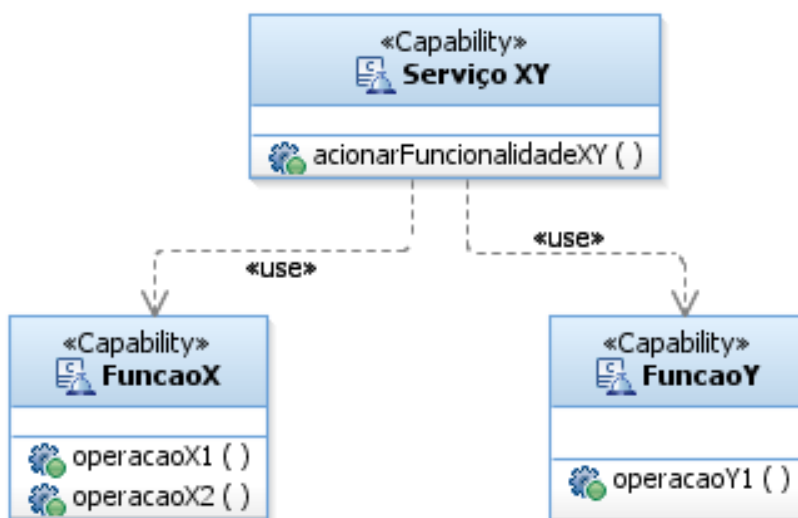


Figura 4.6 Exemplo de um modelo de análise para o Serviço XY.

A interface de um serviço pode ser modelada por meio do elemento “*ServiceInterface*” do perfil SoaML. Esse elemento define as operações de serviço que especificam como a funcionalidade fornecida pelo serviço pode ser acionada. Cada operação possui parâmetros de entrada e saída. Um serviço só pode ser invocado por meio das operações definidas em sua interface de serviço, podendo possuir várias interfaces de serviço. A interface de um serviço deve estar conectada ao elemento “*Capability*” que representa o serviço. Nesta etapa não é obrigatório especificar os tipos de dados dos parâmetros das operações em uma interface de serviços. Na Figura 4.7 é exemplificado o uso do elemento “*ServiceInterface*” para definir a interface do Serviço XY e a operação pela qual o serviço fornece sua funcionalidade.

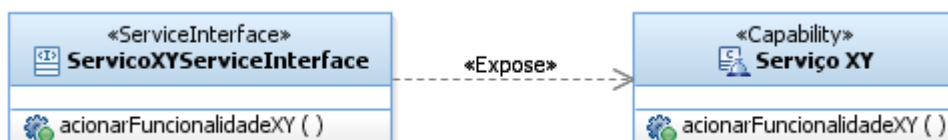


Figura 4.7 Exemplo do uso do elemento *ServiceInterface*.

Na Tabela 4.16 estão listados os artefatos de entrada e saída desta etapa.

Tabela 4.16 Artefatos de entrada e saída da etapa “Modelar Novos Serviços”

Artefatos de entrada	Artefatos de saída
<ul style="list-style-type: none"> ▪ Serviços a serem desenvolvidos; ▪ Modelo <i>to-be</i> do processo de negócio, representando requisitos do sistema; ▪ Documento de Requisitos com detalhamento dos requisitos do sistema ou Diagrama de Casos de Uso e as respectivas descrições ou outro artefato utilizado para registrar as informações sobre os requisitos dos sistema. 	<ul style="list-style-type: none"> ▪ Modelo de análise de novos serviços.

4.3.2.7 Etapa 2.7: Criar Casos de Teste da Solução

Objetivos: Elaborar casos de teste para avaliar a solução desenvolvida. Consolidar o entendimento dos requisitos do sistema.

Descrição: Nesta etapa são elaborados os casos de testes que serão utilizados na fase de testes para avaliar se o sistema de *software* implementa o processo de negócio conforme especificado nos requisitos do sistema. Aspectos como conformidade com o processo de negócio, funcionalidade do sistema, desempenho, interface com o usuário, tratamento de erros, resiliência, entre outros devem ser considerados durante a elaboração dos casos de teste.

Solução: Com base na modelagem do processo de negócio e nos requisitos do sistema deve-se elaborar cenários de utilização do processo de negócio (**P1**). Esses cenários correspondem à execução das atividades do processo de negócio utilizando dados de entrada específicos. Diferentes cenários podem exercitar diferentes caminhos ou sequências de atividades dentro de um processo de negócio.

Selecionar aqueles cenários que exercitam o processo de negócio de maneiras diferentes (**P2**) a fim de que todas as atividades do processo de negócio sejam executadas pelo menos uma vez.

Os casos de teste da solução visam avaliar a funcionalidade do sistema e por esse motivo a elaboração dos casos de testes da solução pode ser complementada com a utilização de técnicas para elaboração de casos de teste do tipo caixa-preta (PRESSMAN, 2006).

Caso haja necessidade, pode-se interagir com o cliente para obter informações adicionais ou esclarecer dúvidas sobre os requisitos ou processos de negócio.

Os casos de teste elaborados devem ser documentados (**P3**), utilizando, por exemplo, uma tabela similar à Tabela 4.17.

Tabela 4.17 Modelo para documentar casos de testes da solução.

Processo de negócio: PN1			
# caso de teste	Descrição	Valores de entrada	Saída esperada
1	Testa resposta a valor de entrada válido	123	123
2	Testa resposta a valor de entrada inválido	Abc	Mensagem informando erro.

Na Tabela 4.18 estão listados os artefatos de entrada e saída desta etapa.

Tabela 4.18 Artefatos de entrada e saída da etapa “Criar Casos de Teste da Solução”.

Artefatos de entrada	Artefatos de saída
<ul style="list-style-type: none"> ▪ Modelo <i>to-be</i> do processo de negócio, representando requisitos do sistema; ▪ Documento de Requisitos com detalhamento dos requisitos do sistema ou Diagrama de Casos de Uso e as respectivas descrições ou outro artefato utilizado para registrar as informações sobre os requisitos dos sistema. 	<ul style="list-style-type: none"> ▪ Casos de teste da solução.

4.3.3 Fase 3: Projetar Serviços

Na fase “Projetar Serviços” são projetados e especificados os serviços que serão implementados nesta iteração para fornecer as funções requeridas pela solução. Como serviços são unidades independentes e utilizam uma interface bem definida para fornecer sua funcionalidade, esta fase pode ser conduzida em paralelo com a fase “Projetar Solução”.

Cinco etapas compõem a fase “Projetar Serviços”: “Definir API de Implementação de *Web Services*”, “Definir Estratégia de Desenvolvimento”, “Definir Arquitetura dos Serviços”, “Especificar Serviço” e “Verificar Serviço”.

Uma situação interessante ocorre quando se deseja expor o processo de negócio que está sendo implementado, por meio de um serviço. Nesse cenário particular, em “Projetar Solução” é especificado como o processo de negócio será implementado enquanto que o serviço que irá expor o processo de negócio é definido na fase “Projetar Serviços”. Nessa situação específica, a arquitetura do serviço e a arquitetura da solução são idênticas e o detalhamento do serviço é o próprio detalhamento da solução.

4.3.3.1 Etapa 3.1: Definir API de Implementação de Web Services

Objetivo: Definir API de implementação de *Web Services* a ser adotada.

Descrição: Plataformas de desenvolvimento como Java e .Net possuem bibliotecas (APIs) específicas para implementação de serviços utilizando a tecnologia *Web Services*, adotada nessa abordagem. Nesta etapa seleciona-se qual API deve ser utilizada para implementar cada um dos serviços sendo desenvolvidos.

Solução: Para cada serviço a ser desenvolvido, deve-se escolher a API de implementação de *Web Services* a ser utilizada (**P1**). Essa escolha é realizada com base no conhecimento técnico e experiência dos desenvolvedores, no ambiente adotado para desenvolvimento de serviços e na infraestrutura na qual os serviços serão executados. O ambiente de desenvolvimento de serviços e a infraestrutura devem prover suporte para a API que se deseja utilizar, sendo que utilizar o mesmo ambiente para desenvolver serviços e solução e a mesma infraestrutura para executar serviços e solução reduz custos relacionados ao treinamento técnico, à aquisição de ferramentas de desenvolvimento, ao *software* e *hardware* usados nos servidores, aos custos de manutenção, entre outros.

Serviços implementados por diferentes APIs podem apresentar diferenças em suas características como desempenho e segurança. Assim, para cada serviço, deve-se considerar qual API fornece melhor desempenho, maior segurança, maior facilidade de desenvolvimento, menor custo de manutenção, etc.

As principais APIs da plataforma Java consideradas nesta abordagem para implementação de *Web Services* são JAX-RPC (JCP, 2002), JAX-WS (JCP, 2006), Axis (APACHE, 2006) e JAX-RS (JCP, 2008). As três primeiras são conhecidas como XML *Web Services*, pois utilizam a tecnologia XML.

Na Tabela 4.19 estão listados os artefatos de entrada e saída desta etapa.

Tabela 4.19 Artefatos de entrada e saída da etapa “Definir API de Implementação de Web Services”.

Artefatos de entrada	Artefatos de saída
<ul style="list-style-type: none"> ▪ Serviços a serem desenvolvidos; ▪ Modelo <i>to-be</i> do processo de negócio, representando requisitos do sistema; ▪ Documento de Requisitos com detalhamento dos requisitos do sistema ou Diagrama de Casos de Uso e as respectivas descrições ou outro artefato utilizado para registrar as informações sobre os requisitos dos sistema. 	<ul style="list-style-type: none"> ▪ API a ser utilizada para implementar os serviços.

4.3.3.2 Etapa 3.2: Definir estratégia de desenvolvimento

Objetivo: Definir a estratégia de desenvolvimento a ser adotada para implementar os serviços.

Descrição: Após definir a API a ser utilizada, deve-se definir qual será a estratégia de desenvolvimento do serviço: *top-down* ou *bottom-up*. Essa estratégia de desenvolvimento estabelece se o serviço será desenvolvido especificando-se primeiramente seus componentes e gerando código a partir dessa especificação ou se o serviço será criado a partir de código existente (ou a ser criado).

Solução: Para cada serviço a ser desenvolvido, escolher a estratégia de desenvolvimento mais adequada (**P1**). A IASWS adota duas estratégias de desenvolvimento de serviço: *top-down* e *bottom-up*. Na estratégia *top-down*, o desenvolvimento de um serviço inicia-se com a especificação do serviço, de seus componentes, interfaces e mensagens utilizando conceitos e padrões da engenharia de *software* e da orientação a serviços. Os artefatos gerados durante a especificação do serviço são transformados em um arquivo de descrição do serviço (WSDL), que é usado para gerar o esqueleto do código que implementará o serviço. Por fim, desenvolvedores preenchem o esqueleto com código que implementa as lógicas necessárias.

Na estratégia *bottom-up* o serviço é criado a partir do código (existente ou que será criado), sendo que as operações do serviço são determinadas pela estrutura do código. A adoção de uma estratégia *bottom-up* implica em não realizar algumas etapas tradicionais da engenharia de *software* (como especificar o serviço, suas interfaces e mensagens) em detrimento do aproveitamento de código existente e da agilidade na criação de um serviço.

A escolha da estratégia de desenvolvimento depende de fatores como: complexidade da funcionalidade fornecida pelo serviço, necessidade de projetar detalhadamente os componentes do serviço, existência de código que forneça a funcionalidade, API de implementação selecionada, peculiaridades da funcionalidade, ferramentas de desenvolvimento disponíveis, entre outros. Algumas ferramentas de desenvolvimento modernas possuem recursos que aceleram o desenvolvimento de serviços utilizando a abordagem *top-down*, a *bottom-up* ou ambas. Essas ferramentas automatizam a geração de esqueletos de código a partir

de modelos (que especificam o serviço) e permitem expor métodos de uma classe como um serviço de forma automatizada.

A API de implementação de *Web Services* escolhida para implementar o serviço influencia fortemente a definição da estratégia de desenvolvimento. Em se tratando de XML *Web Services* (JAX-RPC, JAX-WS, Axis) é possível escolher a estratégia *top-down* (especificar o serviço, gerar o arquivo WSDL de descrição do serviço, gerar o esqueleto do código) ou a estratégia *bottom-up*, na qual o serviço é criado automaticamente a partir do código existente ou que será desenvolvido. No caso da API JAX-RS a adoção de uma estratégia *top-down* requer esforço adicional visto que as ferramentas de desenvolvimento não fornecem recursos que permitam especificar o serviço e gerar o esqueleto do código automaticamente a partir dessa especificação. A adoção de uma estratégia *bottom-up* é mais adequada pois permitiria criar o código utilizando o recurso de anotações que definiria o serviço de forma automática.

Na Tabela 4.20 estão listados os artefatos de entrada e saída desta etapa.

Tabela 4.20 Artefatos de entrada e saída da etapa “Definir Estratégia de Desenvolvimento”.

Artefatos de entrada	Artefatos de saída
<ul style="list-style-type: none"> ▪ API a ser utilizada para implementar o serviço; ▪ Serviços a serem desenvolvidos; ▪ Modelo <i>to-be</i> do processo de negócio, representando requisitos do sistema; ▪ Documento de Requisitos com detalhamento dos requisitos do sistema ou Diagrama de Casos de Uso e as respectivas descrições ou outro artefato utilizado para registrar as informações sobre os requisitos do sistema. 	<ul style="list-style-type: none"> ▪ Estratégia de desenvolvimento dos serviços.

4.3.3.3 Etapa 3.3: Definir Arquitetura dos Serviços

Objetivo: Estabelecer a arquitetura dos serviços.

Descrição: Uma arquitetura genérica aplicável a qualquer *web service* pode ser descrita como um *software* executando em um servidor de aplicações, recebendo requisições de processamento por meio da interface do serviço e de protocolos de comunicação específicos. No entanto, dependendo da complexidade da funcionalidade do serviço, arquiteturas complexas podem ser necessárias para que o serviço possa ser implementado. Nesta etapa é definida a arquitetura desses tipos de serviço, sendo documentada utilizando recursos gráficos.

Solução: Para cada serviço que fornece uma funcionalidade complexa, elaborar sua arquitetura (**P1**). Essa arquitetura pode englobar informações que vão desde os componentes da infraestrutura e sua respectiva organização até informações da organização interna do código que implementa a funcionalidade fornecida pelo serviço.

A arquitetura do serviço deve descrever o relacionamento dos componentes de infraestrutura (por exemplo banco de dados, *middleware* de mensagens, servidor de aplicações) e de *software* (código, componentes, *web services*) que, em conjunto, fornecem a funcionalidade do serviço. Caso haja necessidade deve-se descrever a organização interna do código (organização do código em camadas, padrões de projeto, etc.) de forma a direcionar o desenvolvimento.

Essa arquitetura do serviço deve ser representada graficamente (**P2**) para facilitar seu entendimento. Um exemplo de representação da arquitetura de um serviço utilizando a ferramenta *Rational Software Architect* 8 (IBM, 2010a) é mostrado na Figura 4.8. É utilizado um diagrama de *deployment* para representar o *web service* “Serviço XY” que invoca outros dois *web services* (“WebService A” e “WebService B”) e o “Banco de dados X”. O *web service* “Serviço XY” é executado em um “Servidor de Aplicações SA” enquanto o “Banco de dados X” reside no “Servidor de Banco de Dados SBD”. Ambos os servidores (*middleware*) são executados no “Servidor S” (servidor físico). Estereótipos são usados para qualificar os elementos usados nesse diagrama, conforme necessidade.

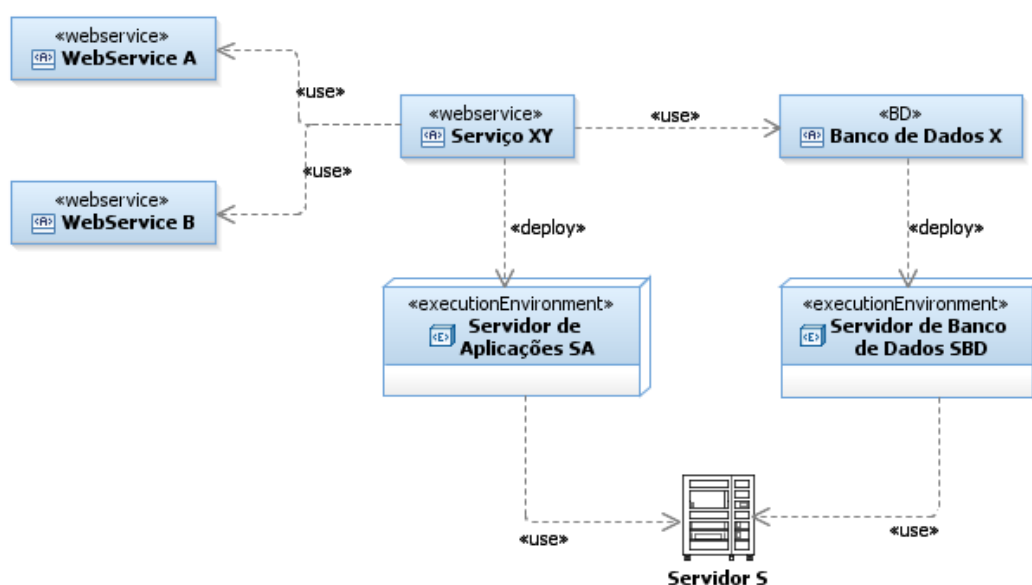


Figura 4.8 Exemplo de representação da arquitetura do Serviço XY.

Na Tabela 4.21 estão listados os artefatos de entrada e saída desta etapa.

Tabela 4.21: Artefatos de entrada e saída da etapa “Definir Arquitetura do Serviço”

Artefatos de entrada	Artefatos de saída
<ul style="list-style-type: none"> ▪ API a ser utilizada para implementar o serviço; ▪ Serviços a serem desenvolvidos; ▪ Modelo <i>to-be</i> do processo de negócio, representando requisitos do sistema; ▪ Documento de Requisitos com detalhamento dos requisitos do sistema ou Diagrama de Casos de Uso e as respectivas descrições ou outro artefato utilizado para registrar as informações sobre os requisitos do sistema. 	<ul style="list-style-type: none"> ▪ Representação gráfica da arquitetura dos serviços.

4.3.3.4 Etapa 3.4: Especificar Serviço

Objetivo: Especificar os serviço a serem desenvolvidos.

Descrição: Serviços são especificados apenas quando a estratégia de desenvolvimento é *top-down*. Essa especificação inclui a definição da interface pela qual o serviço será invocado, definição do formato das mensagens utilizadas e definição dos componentes internos que implementarão a funcionalidade fornecida pelo serviço.

Solução: Para cada novo serviço a ser desenvolvido, refinar sua modelagem (**P1**) (realizada na etapa 2.6 - “Modelar novos serviços”), detalhando as operações da interface (**P2**), modelando as mensagens (**P3**) e componentes internos (**P4**) (classes, *web services*, etc.).

Cada uma das operações do serviço deve ter seu retorno e seus parâmetros especificados, incluindo o tipo de dado ou mensagem. Mensagens são agrupamentos de informações enviadas a um serviço ou recebidas de um serviço. Cada tipo de mensagem utilizada nas operações de um serviço deve ser especificada utilizando o elemento do tipo *MessageType*. Classes, *web services* e outros componentes que farão parte da implementação da funcionalidade do serviço devem ser representados utilizando-se elementos adequados. Além desses elementos, é preciso representar o provedor de serviço que irá fornecer a implementação do serviço (*Capability*), disponibilizando-a por meio da interface (*ServiceInterface*) definida. No SoaML esse provedor de serviço é representado com o uso do elemento do tipo *Participant*.

Na Figura 4.9 é mostrado o refinamento da modelagem do “Serviço XY”. Dois tipos de mensagem foram especificados: MensagemEntrada e MensagemRetorno,

representados por meio do elemento do tipo *MessageType*. O “Serviço XY” fornece sua funcionalidade por meio da operação *acionarFuncionalidadeXY()* que recebe como parâmetro de entrada uma mensagem do tipo *MensagemEntrada* e retorna o resultado do processamento por meio de uma mensagem do tipo *MensagemRetorno*. As operações da *FunçãoX* e *FunçãoY*, são utilizadas na implementação do *web service* *Serviço XY* e foram detalhadas, definindo-se seus parâmetros de entrada, respectivos tipos e o tipo do valor retornado após sua execução. Por fim, um elemento do tipo *Participant* representa o provedor de serviço que irá fornecer a implementação da funcionalidade especificada no serviço.

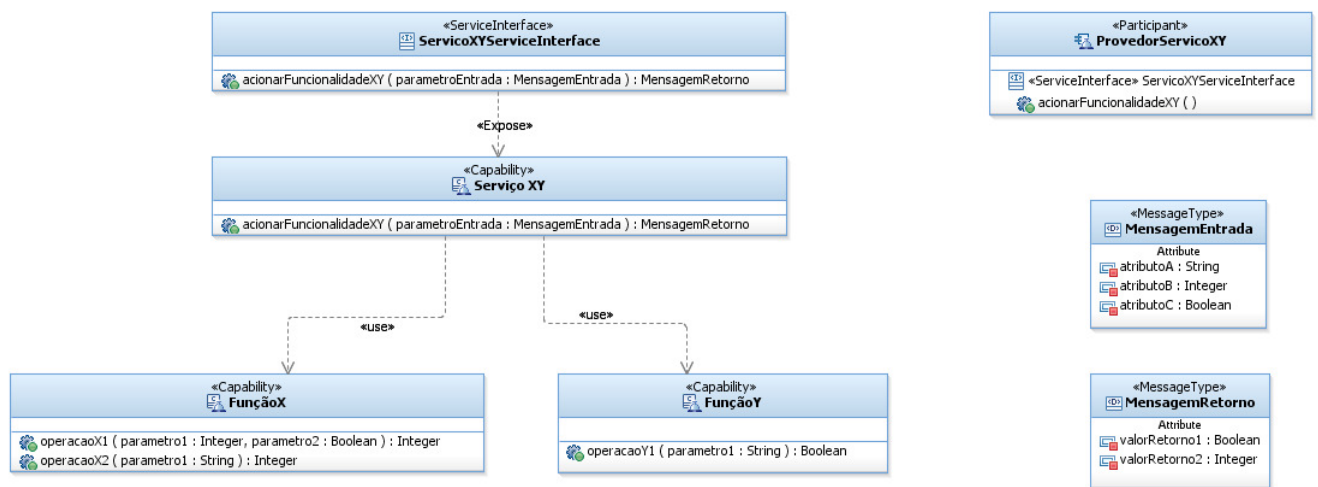


Figura 4.9 Exemplo de especificação de um serviço usando elementos do perfil SoaML.

Na Tabela 4.22 estão listados os artefatos de entrada e saída desta etapa.

Tabela 4.22: Artefatos de entrada e saída da etapa “Especificar Serviço”

Artefatos de entrada	Artefatos de saída
<ul style="list-style-type: none"> ▪ Arquitetura do serviço; ▪ Estratégia a ser adotada no desenvolvimento do novo serviço; ▪ Serviços a serem desenvolvidos; ▪ Modelo <i>to-be</i> do processo de negócio, representando requisitos do sistema; ▪ Documento de Requisitos com detalhamento dos requisitos do sistema ou Diagrama de Casos de Uso e as respectivas descrições ou outro artefato utilizado para registrar as informações sobre os requisitos dos sistema. 	<ul style="list-style-type: none"> ▪ Serviços modelados e suas respectivas especificações.

4.3.3.5 Etapa 3.5: Verificar Serviço

Objetivo: Verificar a conformidade dos serviços especificados com relação aos princípios da orientação a serviços.

Descrição: Os princípios da orientação a serviços norteiam o desenvolvimento de soluções orientadas a serviço para que os benefícios prometidos pela arquitetura SOA sejam alcançados. Consequentemente cada serviço desenvolvido deve adotar esses conceitos para contribuir na adoção de SOA de forma efetiva e eficaz. Nesta etapa, é verificada a conformidade dos serviços especificados na etapa anterior, com os princípios da orientação a serviços.

Solução: Para cada serviço especificado na etapa anterior, verificar se sua especificação está em conformidade com os princípios da orientação a serviços (P1). Os princípios que devem ser verificados são: reusabilidade do serviço; granularidade do serviço; baixo acoplamento entre serviços; contrato padronizado de serviço; abstração; autonomia; e independência de estado. Não há necessidade de o serviço especificado adotar todos os princípios. Cabe ao desenvolvedor avaliar quais princípios utilizar de modo que o serviço desenvolvido seja o melhor possível.

Na Tabela 4.23 estão listados os artefatos de entrada e saída desta etapa.

Tabela 4.23: Artefatos de entrada e saída da etapa “Verificar Serviço”.

Artefatos de entrada	Artefatos de saída
<ul style="list-style-type: none"> ▪ Especificação dos serviços. 	<ul style="list-style-type: none"> ▪ Avaliação sobre conformidade dos serviços com conceitos SOA

4.3.4 Fase 4: Projetar Solução

Nas fases iniciais da IASWS (“Identificar Requisitos” e “Contextualizar PN com serviços”), os processos de negócio do cliente foram entendidos, os requisitos do sistema foram elicitados, os serviços (a desenvolver e que serão reusados) foram identificados e uma solução foi proposta. Nesta fase, essa solução é projetada para garantir que a implementação do sistema atenda aos requisitos.

Durante as etapas desta fase, os elementos da solução (classes, *web services*, componentes OO, infraestrutura) são definidos, especificados e organizados.

Três etapas compõem esta fase: “Definir Arquitetura da Solução”, “Criar Modelo de Dados” e “Especificar Componentes da Solução”.

4.3.4.1 Etapa 4.1: Definir Arquitetura da Solução

Objetivo: Estabelecer a arquitetura da solução.

Descrição: Elabora-se a arquitetura da solução com os componentes, a organização desses componentes e inter-relacionamentos necessários para implementar o processo de negócio.

Solução: Para elaborar a arquitetura da solução, primeiramente deve-se analisar o tipo do sistema (*client-server*, *web*, distribuído, *desktop*, etc.) (**P1**) que implementará a solução proposta. Em seguida, identificam-se os componentes (**P2**) desse sistema, que podem ser: *web services*, componentes OO, *hardware* (servidores, dispositivos móveis, sensores, etc.), *middlewares* (SGBDs, servidor de aplicação, etc.) e *softwares*.

O próximo passo é organizar esses componentes e determinar seus inter-relacionamentos (**P3**), representando o papel que cada componente desempenha na solução proposta.

Por fim, a arquitetura elaborada deve ser representada graficamente (**P4**), por exemplo, como mostrado na Figura 4.10, na qual foi utilizado um diagrama de *deployment* da ferramenta *Rational Software Architect 8* (IBM, 2010a). Nesse exemplo, a arquitetura da solução que implementa o processo de negócio PN1 (exemplificado anteriormente) é composta por: um servidor remoto no qual o servidor de banco de dados e o “Banco de Dados A” estão alocados; um servidor local no qual o servidor de aplicações está alocado e que constitui o ambiente de execução do código que implementa a funcionalidade da solução; o *web service* “Serviço XY” cuja funcionalidade é usada na solução. Estereótipos são definidos e usados para qualificar os elementos usados nesse diagrama, conforme a necessidade.

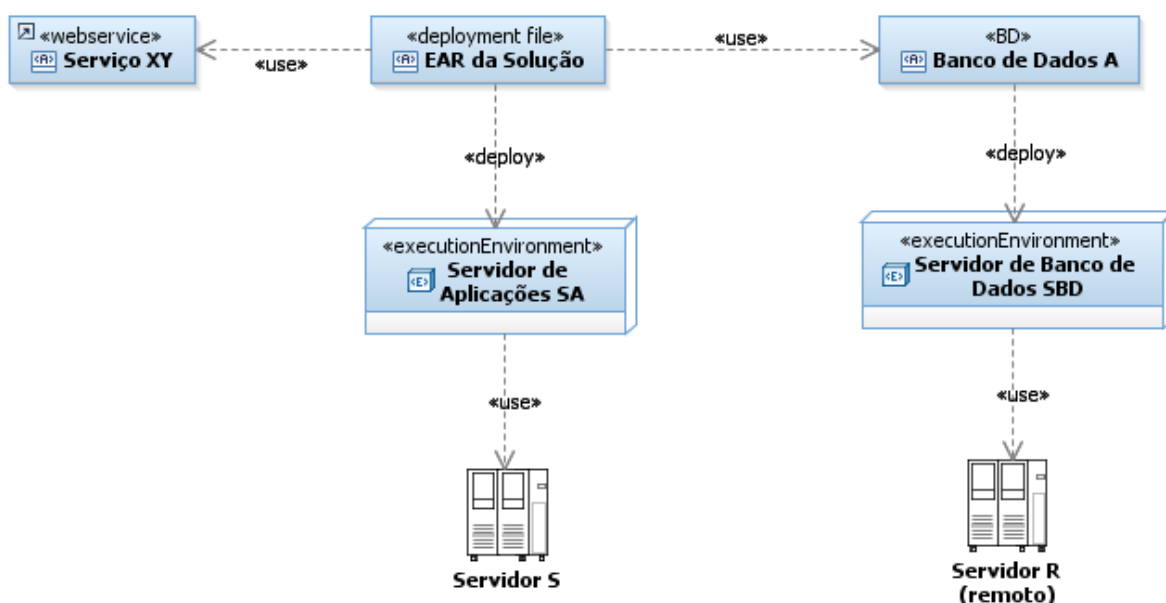


Figura 4.10 Exemplo de representação da arquitetura de uma solução.

Na Tabela 4.24 estão listados os artefatos de entrada e saída desta etapa.

Tabela 4.24: Artefatos de entrada e saída da etapa “Definir Arquitetura da Solução”.

Artefatos de entrada	Artefatos de saída
<ul style="list-style-type: none"> ▪ Serviços a serem desenvolvidos; ▪ Serviços que serão reusados; ▪ Modelo <i>to-be</i> do processo de negócio, representando requisitos do sistema; ▪ Documento de Requisitos com detalhamento dos requisitos do sistema ou Diagrama de Casos de Uso e as respectivas descrições ou outro artefato utilizado para registrar as informações sobre os requisitos dos sistema; ▪ Casos de testes da solução. 	<ul style="list-style-type: none"> ▪ Representação gráfica da arquitetura da solução.

4.3.4.2 Etapa 4.2: Criar Modelo de Dados

Objetivo: Modelar entidades de dados.

Descrição: As entidades de dados envolvidas no processo de negócio devem ser identificadas, analisadas, mapeadas para o modelo relacional e modeladas. O modelo de banco de dados relacional foi adotado considerando a sua ampla utilização, o grande número de SGBDs disponíveis e os recursos das IDEs existentes que trabalham com tal modelo.

Solução: Para cada entidade de dados, analisar suas informações para identificar seus atributos e seus relacionamentos com outras entidades (P1). Em

seguida, mapear a entidade e seus relacionamentos para o modelo relacional (**P2**), e criar uma tabela no banco de dados para cada relação mapeada (**P3**). A definição dessa tabela deve incluir os nomes dos atributos, seus respectivos tipos de dados, tamanho máximo (quando aplicável) e obrigatoriedade do atributo.

Existem ferramentas de desenvolvimento que permitem elaborar um diagrama físico de dados, que representa graficamente as tabelas de um banco de dados relacional e seus inter-relacionamentos, podendo ser usado, posteriormente, para a criação do banco de dados. Um exemplo, construído na ferramenta *Rational Software Architect 8*, é mostrado na Figura 4.11. A entidade P possui os atributos P1, P2 e P3, sendo que o atributo P1 identifica univocamente um indivíduo dessa entidade. A entidade T, por sua vez, possui os atributos T1 e T2, sendo o atributo T1 seu identificador. As entidades P e T relacionam-se por meio de uma relação “é do tipo”, ou seja, um indivíduo de P “é do tipo” do indivíduo em T.

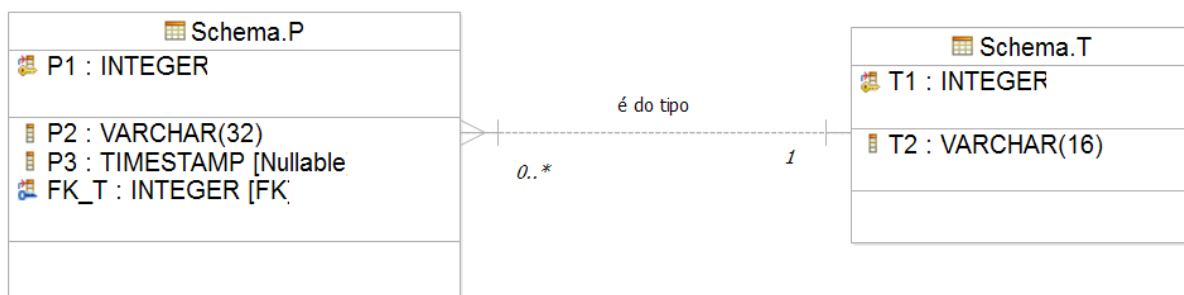


Figura 4.11 Exemplo de modelo de dados.

Na Tabela 4.25 estão listados os artefatos de entrada e saída desta etapa.

Tabela 4.25: Artefatos de entrada e saída da etapa “Criar Modelo de Dados”.

Artefatos de entrada	Artefatos de saída
<ul style="list-style-type: none"> ▪ Modelo <i>to-be</i> do processo de negócio, representando requisitos do sistema; ▪ Documento de Requisitos com detalhamento dos requisitos do sistema ou Diagrama de Casos de Uso e as respectivas descrições ou outro artefato utilizado para registrar as informações sobre os requisitos dos sistema; ▪ Casos de testes da solução. 	<ul style="list-style-type: none"> ▪ Modelos de dados da solução.

4.3.4.3 Etapa 4.3: Especificar Componentes da Solução

Objetivo: Especificar componentes da solução.

Descrição: Define-se a estrutura do código que implementará o processo de negócio. Essa estrutura pode ser composta por interfaces, classes, atributos, métodos, componentes OO, *web services*, relacionamentos, generalizações, polimorfismos, heranças, *frameworks* e padrões de projeto.

Solução: Deve-se identificar as principais classes e seus relacionamentos (P1), necessários para que o sistema implemente o processo de negócio selecionado. Serviços usados na solução não necessitam ter uma classe associada, bastando fazer uso de sua interface. Em seguida, analisa-se o fluxo das atividades do processo de negócio sendo implementado, representado no modelo *to-be*, para definir os métodos e atributos dessas classes (P2). Depois disso, representar as classes identificadas, seus respectivos atributos, métodos e relacionamentos usando um diagrama de classes da UML (P3). Nesse diagrama também devem constar as interfaces dos serviços que fazem parte da solução.

O próximo passo é refinar o diagrama de classes identificando interfaces, generalizações, polimorfismos, heranças e aplicando outros conceitos da orientação a objetos às classes desse diagrama (P4). Por fim, aplicar os padrões de projeto adotados e representar elementos pertencentes ao *framework* (quando adotado) e respectivos pontos de interação (P5).

Um exemplo de especificação do componente de código para a solução que implementa o processo de negócio PN1 é mostrado no diagrama de classes da Figura 4.12. Nesse diagrama são representadas as classes ClasseA e ClasseP e seu relacionamento todo-parte, a classe abstrata ClasseAbstrataB e seus descendentes (ClasseB1 e ClasseB2) e a interface do *web service* (ServicoXYServiceInterface) usado na solução. Os atributos e métodos necessários para a implementação do processo de negócio PN1 foram especificados.

Outros diagramas da UML, como de sequência, de atividades, de colaboração, de estados, entre outros, podem ser criados dependendo do nível de detalhamento necessário.

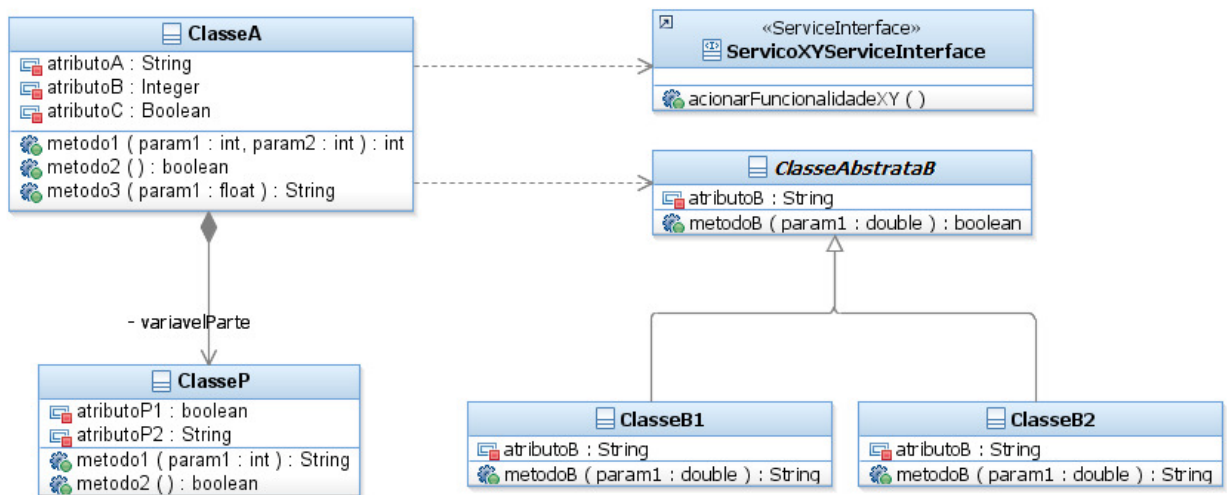


Figura 4.12: Exemplo de Diagrama de classes.

Na Tabela 4.26 estão listados os artefatos de entrada e saída desta etapa.

Tabela 4.26: Artefatos de entrada e saída da etapa “Especificar Componentes da Solução”.

Artefatos de entrada	Artefatos de saída
<ul style="list-style-type: none"> ▪ Serviços a serem desenvolvidos; ▪ Serviços que serão reusados; ▪ Modelo <i>to-be</i> do processo de negócio, representando requisitos do sistema; ▪ Documento de Requisitos com detalhamento dos requisitos do sistema ou Diagrama de Casos de Uso e as respectivas descrições ou outro artefato utilizado para registrar as informações sobre os requisitos dos sistema; ▪ Casos de testes da solução. 	<ul style="list-style-type: none"> ▪ Especificações dos componentes da solução.

4.4 Considerações Finais

Neste capítulo foi descrita a abordagem IASWS, suas fases e etapas que compõem o ciclo de desenvolvimento proposto. Cada uma das etapas das quatro fases iniciais foi descrita detalhadamente, incluindo seu objetivo, sua descrição, os passos para se atingir o objetivo, seus artefatos de entrada e saída e exemplos.

Na Tabela 4.27 é mostrada uma comparação da abordagem IASWS com os métodos e processos estudados (Capítulo 2).

Tabela 4.27: Comparação da abordagem IASWS com outros métodos e processos.

	PAPAZOGLOU e van den Heuvel	MAPOS	SOMA 2.9	Erl	IASWS
Iterativo	Sim	Sim (pouco)	Não	Não	Sim
Incremental	Sim	Não	Não	Não	Sim
Adoção gradual da COS/Serviços	Não	Não	Não	Não	Sim
Reúso de serviços	Em nível de análise	Em nível de análise	Em nível de análise	Em nível de análise	Em nível de análise
Teste de serviços reusados	Não tem	Não tem	Não tem	Não tem	Realizado na análise
Prescreve ferramentas	Sim	Não	Sim	Não	Sim
Prescreve diagramas / artefatos	Sim	Sim	Sim	Não	Sim
Possui exemplos	Usa modelo de referência	Não	Poucos e não relacionados	Sim	Sim
Em algum ponto do ciclo permite realizar alterações ao que já foi implementado	É possível mas não está explícito no método	Não	É possível mas não está explícito no método	Não	Sim
Estabelece parâmetros para definir o tamanho de um incremento	Não	Não	Não	Não	Parcialmente
Estabelece tecnologias de implementação	Sim (WS, BPEL, WSDL)	Sim (BPMN, UML, BPEL, WSDL)	Não	Sim (WS, WSDL, BPEL, XSD)	Sim (WS, BPMN, SoaML, UML, Java)

Dentre as características da abordagem IASWS, destacam-se as seguintes:

- Adoção gradual da COS, possibilitando que serviços sejam incorporados à solução de maneira gradual, conforme a equipe de desenvolvimento e o cliente ganham confiança no paradigma e nas tecnologias;

- Possibilita reuso de serviços em nível de análise, reduzindo a complexidade da solução a ser desenvolvida com o encapsulamento de lógica de negócios nos serviços;
- Adota a modelagem de processos de negócio que favorece a identificação de requisitos do sistema. A análise e especificação de requisitos é uma das etapas críticas do desenvolvimento de sistemas de *software*, sendo que a qualidade desses requisitos é crucial para o sucesso dos projetos de desenvolvimento (ERFURTH e KIRCHNER, 2010). Problemas como a falta de entendimento do negócio por parte da equipe de desenvolvimento, falta de foco em relação ao propósito do sistema e a má comunicação entre a equipe de desenvolvimento e os analistas de negócios são alguns dos problemas mais comuns encontrados quando se realiza o levantamento de requisitos (DE LA VARA, SÁNCHEZ, PASTOR, 2008). Na IASWS, esses problemas são tratados utilizando uma adaptação da técnica proposta por Cardoso, Almeida e Guizzardi (2009), para engenharia de requisitos baseada em modelos de processo de negócio;
- Propõe a avaliação de serviços reusados para verificar a reusabilidade e adaptabilidade dos serviços encontrados ao contexto da solução. Essa estratégia foi inspirada na elaboração de soluções de ponta do XP (PRESSMAN, 2006);
- Propõe que casos de testes sejam elaborados antes da implementação, assim como realizado no XP, contribuindo para que a equipe de desenvolvimento melhore seu entendimento acerca dos requisitos do sistema;
- Adapta o conceito de *backlog* do Scrum, para priorizar processos de negócio e planejar os incrementos de *software*;
- Por fim, a estratégia de realizar entregas constantes, implementando um ou mais processos a cada iteração, foi influenciada pelo XP, Scrum e pelo modelo Incremental que adotam essa estratégia como forma de coletar *feedbacks* do cliente e demonstrar progresso do trabalho de desenvolvimento.

Capítulo 5

EXEMPLOS DE APLICAÇÃO DA IASWS

5.1 Considerações Iniciais

Para exemplificar a aplicação da abordagem IASWS foram desenvolvidos dois sistemas hipotéticos de *software*: Sistema Gerenciador de Biblioteca e Sistema de Atendimento de uma Unidade Saúde Escola (USE), sendo o papel de desenvolvedor assumido pelo autor desta dissertação e o de cliente, assumido pelos orientadores. As etapas da abordagem foram aplicadas no desenvolvimento desses dois sistemas exemplo, possibilitando ao leitor o entendimento de quais atividades/tarefas (etapas da abordagem) devem ser realizadas e quais os artefatos foram utilizados e produzidos em cada uma delas.

Devido à familiarização do autor desta dissertação com a ferramenta de desenvolvimento IBM Rational *Software Architect* (RSA), versão 8, ela foi adotada neste estudo e, também, devido à facilidade em se obter sua licença para uso acadêmico por meio do programa IBM Academic Initiative.

Na Seção 5.2 é apresentada a descrição do Sistema Gerenciador de Biblioteca bem como a aplicação da IASWS no desenvolvimento desse sistema. O desenvolvimento desse sistema serviu para o refinamento das fases da abordagem. Na Seção 5.3 são apresentados a descrição e o desenvolvimento do Sistema de Atendimento de uma Unidade Saúde Escola que possibilitou a validação da abordagem proposta. Na Seção 5.4 são feitas as considerações finais.

5.2 Exemplo 1: Sistema Gerenciador de Biblioteca

O Sistema Gerenciador de Biblioteca (SGBi) é um sistema simples, com funcionalidade limitada, para auxiliar no empréstimo e devolução de exemplares, reserva de títulos, gerenciamento de títulos (cadastro, exclusão), cadastro de usuários e consulta de títulos do acervo de uma biblioteca hipotética. Dois tipos de relatórios podem ser gerados quando solicitados por um usuário administrativo: um com informações sobre a quantidade de empréstimos realizados em um determinado período e outro com as informações sobre os empréstimos com devolução em atraso.

Durante a concepção e aplicação da abordagem foram implementados dez processos de negócio da biblioteca, sendo apresentada nesta seção apenas a primeira iteração (implementação do processo “Cadastrar usuário”). As demais iterações para implementação dos processos de negócio restantes foram documentadas em um documento de trabalho (NAKAGAWA, PENTEADO e SANTOS, 2011a).

5.2.1 Primeira iteração

A execução da fase “Identificar Requisitos” na primeira iteração de desenvolvimento demandou tempo considerável pois foi preciso coletar informações sobre todos os processos a serem implementados, modelá-los e documentá-los, como é mostrado nas subseções seguintes.

5.2.1.1 Fase 1: Identificar Requisitos

Etapa 1.1: Levantar PN

Foram realizadas reuniões com o cliente para que o desenvolvedor entendesse o negócio do cliente e identificasse os processos e objetivos de negócio, as dificuldades e as necessidades do cliente. O papel do cliente foi assumido pelos orientadores que forneceram as explicações detalhadas sobre o funcionamento da biblioteca e o autor desta dissertação assumiu o papel da equipe de desenvolvimento, coletando informações, e registrando-as como mostrado na Tabela 5.1.

Tabela 5.1 Informações coletadas com o cliente.

Área de negócio	Acadêmico
Objetivos de negócio	<ul style="list-style-type: none"> ▪ Manter acervo atualizado; ▪ Beneficiar o maior número possível de usuários; ▪ Facilitar o atendimento da biblioteca.
Processos de negócio considerados	<ul style="list-style-type: none"> ▪ Cadastrar usuário; ▪ Excluir usuário; ▪ Realizar empréstimo de exemplar; ▪ Devolver exemplar emprestado; ▪ Reservar título não disponível para empréstimo no momento; ▪ Consultar acervo; ▪ Cadastrar título; ▪ Excluir título; ▪ Alterar título; ▪ Gerar relatórios.
Dificuldades/necessidades	<ul style="list-style-type: none"> ▪ A geração de relatórios é trabalhosa pois requer que dados sejam extraídos manualmente; ▪ Há uma necessidade de agilizar o atendimento a usuários: reduzir tempo para realizar operações como Empréstimo, Devolução e Reserva.

Após a identificação dos processos de negócio do cliente, foi determinado, com sua ajuda, que os processos de negócio “Excluir Usuário” e “Alterar título” não seriam informatizados neste momento. Para os demais processos listados na Tabela 5.1 foram elencadas as atividades realizadas durante a execução de cada um deles. Essas atividades foram então organizadas em sequências lógicas e estão mostradas na Tabela 5.2, juntamente com o ator envolvido.

Para facilitar o entendimento dos processos listados na Tabela 5.2, a documentação de cada um deles é ilustrada na Tabela 5.3 de acordo com as informações obtidas com o cliente.

Tabela 5.2 Atividades dos processos de negócio do cliente

Processo de negócio	Atividades	Ator envolvido
Cadastrar usuário	<ol style="list-style-type: none"> 1. Obter informações do usuário; 2. Validar informações; 3. Registrar usuário. 	Usuário Administrativo
Realizar empréstimo de exemplar	<ol style="list-style-type: none"> 1. Encontrar exemplar reservado (caso seja empréstimo de exemplar reservado); 2. Obter informações do empréstimo (exemplar e usuário); 3. Realizar verificações das condições para empréstimo; 4. Registrar empréstimo. 	Usuário leitor e Usuário Administrativo
Devolver exemplar emprestado	<ol style="list-style-type: none"> 1. Obter informações do exemplar sendo devolvido; 2. Registrar devolução; 3. Tratar reserva (caso exista reserva para o título). 	Usuário leitor de posse de um empréstimo e Usuário Administrativo
Reservar título não disponível para empréstimo no momento	<ol style="list-style-type: none"> 1. Obter informações (título e usuário) para realizar reserva; 2. Confirmar informações do título e sua disponibilidade; 3. Realizar verificações; 4. Criar reserva. 	Usuário leitor e Usuário Administrativo
Consultar acervo	<ol style="list-style-type: none"> 1. Obter parâmetros de busca; 2. Consultar títulos do acervo; 3. Mostrar títulos encontrados e status dos exemplares. 	Usuário leitor
Cadastrar título	<ol style="list-style-type: none"> 1. Obter informações cadastrais do título; 2. Adicionar título ao acervo; 3. Adicionar exemplares do título; 4. Preparar e disponibilizar exemplar para empréstimo. 	Usuário Administrativo
Excluir título	<ol style="list-style-type: none"> 1. Identificar título a ser excluído; 2. Realizar verificações para excluir título; 3. Remover título do acervo e respectivos exemplares. 	Usuário Administrativo
Gerar relatórios	<ol style="list-style-type: none"> 1. Coletar dados necessários para elaborar o relatório; 2. Criar relatório: gerar gráficos e tabelas. 	Usuário Administrativo

Tabela 5.3 Informações dos processos de negócio para o SGBi.

Processo de negócio	Informações
Cadastrar usuário	Alunos e funcionários (docentes e técnico-administrativos) da universidade podem se cadastrar para realizar empréstimos na biblioteca. Suas informações cadastrais podem ser obtidas diretamente do banco de dados existente na Secretaria da Graduação (Secgrad) e do banco de dados existente na Secretaria de Recursos Humanos (RH) da universidade, respectivamente. Após a obtenção das informações cadastrais dos respectivos banco de dados, deve-se validá-las com o usuário no momento do cadastro. O email será a principal forma de comunicação com o usuário. Com a concordância do usuário de que as informações estão corretas, registra-se o usuário gerando um novo código que o identifica univocamente na biblioteca.
Realizar empréstimo de exemplar	Um usuário cadastrado pode emprestar um ou mais exemplares, caso haja disponibilidade. Para realizar o empréstimo é necessário que o usuário leve o exemplar até o balcão de atendimento, o entregue ao atendente e informe seu código de usuário. O empréstimo é realizado se o usuário não tiver empréstimo com devolução em atraso, não estiver suspenso e não tiver atingido o seu limite de empréstimos. O número máximo de empréstimos que um usuário pode realizar depende do tipo de usuário. Cada empréstimo possui uma data de devolução que é calculada com base no tipo de usuário.
Devolver exemplar emprestado	Quando um usuário entrega o exemplar no balcão de atendimento e informa ao atendente da devolução, o empréstimo é finalizado. Caso a devolução seja realizada após a data de devolução estipulada, o usuário é informado do período de suspensão. No momento da devolução do exemplar é necessário verificar se existe reserva do título desse exemplar. Caso haja, o Usuário Administrativo notifica, via email, o usuário que realizou a reserva, sendo o exemplar armazenado em área especial para os títulos reservados até que o usuário o retire.
Reservar título não disponível para empréstimo no momento	Se nenhum exemplar do título desejado estiver disponível para empréstimo, o usuário pode reservá-lo. Quando um exemplar desse título for devolvido, o primeiro usuário que o reservou é notificado da possibilidade de empréstimo. Para reservar um título, o usuário deve ir até o balcão de atendimento e informar o código de catalogação do título desejado juntamente com o seu código de usuário. A reserva é realizada se o usuário não possuir empréstimo com devolução em atraso, não estiver suspenso e também não tiver atingido o seu limite de empréstimos.
Consultar acervo	Permite que usuários encontrem títulos e seus respectivos exemplares na biblioteca. Informações que auxiliam na identificação do título procurado são fornecidas sob a forma de uma ou mais palavras-chave. Combinações das palavras-chave são permitidas usando-se os seguintes conectores: “Todas as palavras”, “Qualquer palavra” ou “Frase exata”. A combinação das palavras-chave fornecidas é procurada nos campos título e autor de cada título do acervo. O resultado de uma consulta é uma lista dos títulos (e respectivas informações) que atendem aos critérios de busca informados pelo usuário.

Processo de negócio	Informações
Cadastrar título	Permite que um novo título e seus exemplares sejam incluídos no acervo da biblioteca pelo Usuário Administrativo. Para que o título seja cadastrado é preciso fornecer seus dados cadastrais (ISBN, título, autor, área de conhecimento relacionada, categoria, ano de publicação, número da edição, editora, localização física dos exemplares na biblioteca). No momento do cadastro, o título recebe um código de catalogação que o identifica de maneira única dentro do acervo da biblioteca. Cada um dos exemplares deste título recebe um identificador único conhecido como Tombo. Assim que o processo de cadastro é finalizado, o exemplar é colocado na sua localização física e disponibilizado para empréstimo. No acervo não há revistas, revistas em quadrinhos, monografias, dissertações e teses de doutorado.
Excluir título	Permite que um Usuário Administrativo exclua um título pertencente ao acervo, utilizando seu código de catalogação. Quando um título é excluído, todos os exemplares desse título devem ser excluídos também. Dessa forma, é obrigatório que todos os exemplares do título não estejam emprestados para que o título possa ser excluído. É importante ressaltar que informações acerca de empréstimos envolvendo o título que está sendo excluído assim como outras informações pertinentes ao título devem ser mantidas.
Gerar relatórios	Dois tipos de relatórios são possíveis no gerenciamento da biblioteca: o relatório de empréstimos realizados por data e o de empréstimos com devolução em atraso.

Etapa 1.2: Modelar PN

Um Diagrama de Visão Geral de Processos, Figura 5.1, foi construído com os processos de negócio identificados pelo desenvolvedor, após o contato com o cliente. O processo de negócio “Gerar relatórios” foi dividido em dois processos específicos, para facilitar o entendimento: “Gerar relatório de empréstimos por data” e “Gerar relatório de empréstimos com devolução em atraso”.

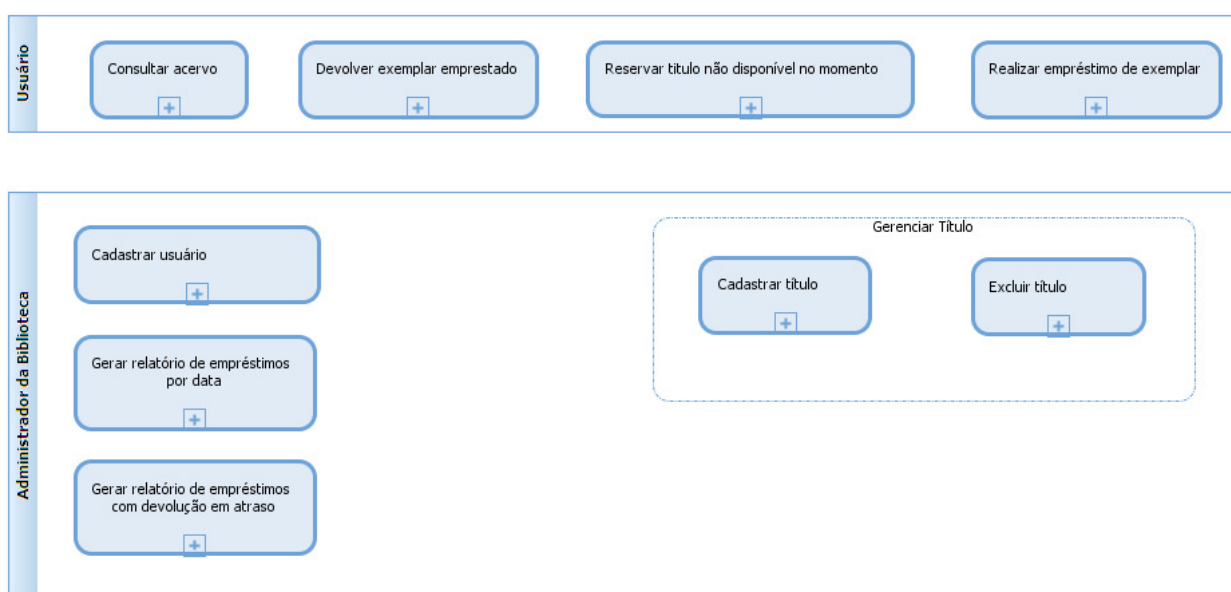


Figura 5.1 Diagrama de Visão Geral de Processos que serão implementados no SGBi.

Cada um dos processos representados no Diagrama de Visão Geral de Processos foi modelado em um diagrama de processo do BPMN. As atividades da Tabela 5.2 foram traduzidas em atividades no modelo *as-is* do processo de negócio correspondente. Para exemplificar, o modelo *as-is* do processo de negócio “Cadastrar usuário” é mostrado na Figura 5.2, com as atividades: Obter informações do usuário (quando são obtidas as informações pessoais do usuário a ser cadastrado), Validar informações (quando as informações obtidas são validadas, por exemplo, CPF) e Registrar usuário (quando as informações do usuário são inseridas no banco de dados da biblioteca). Os modelos *as-is* dos demais processos que serão implementados neste exemplo são mostrados a partir da Figura 5.3 até a Figura 5.10.

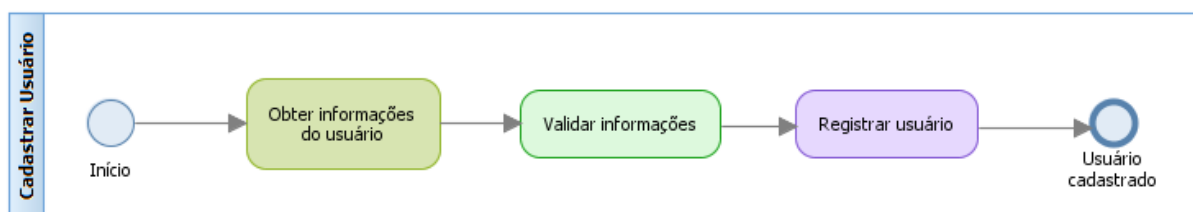


Figura 5.2 Modelo *as-is* do processo de negócio “Cadastrar usuário” do SGBi.

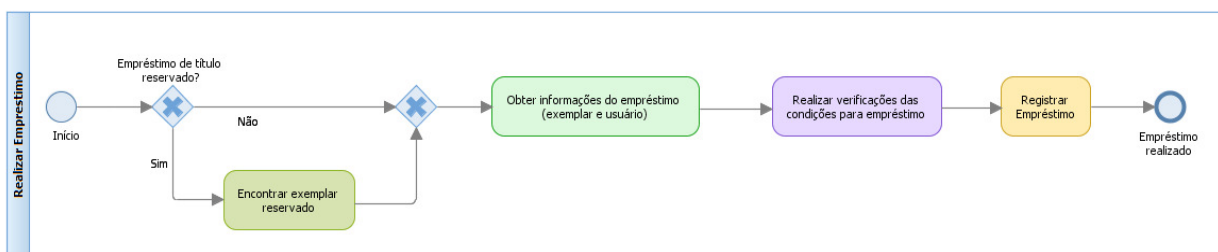


Figura 5.3 Modelo *as-is* do processo de negócio “Realizar empréstimo de exemplar” do SGBi.

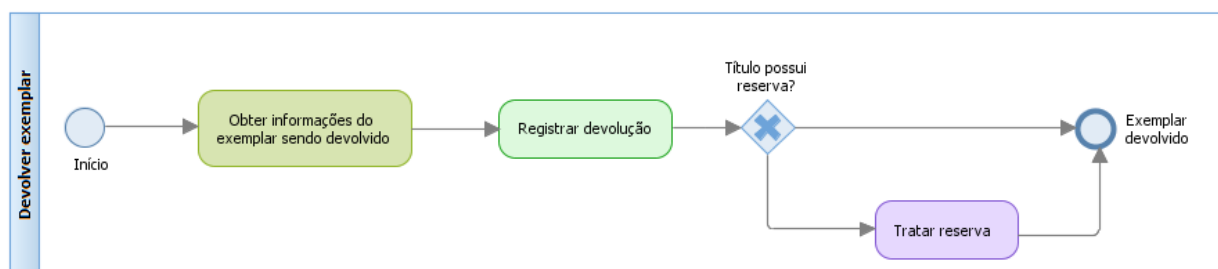


Figura 5.4 Modelo *as-is* do processo de negócio “Devolver exemplar emprestado” do SGBi.

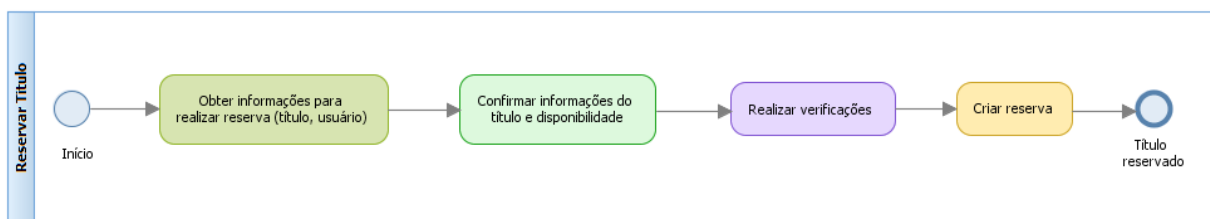


Figura 5.5 Modelo *as-is* do processo de negócio “Reservar título não disponível para empréstimo no momento” do SGBi.

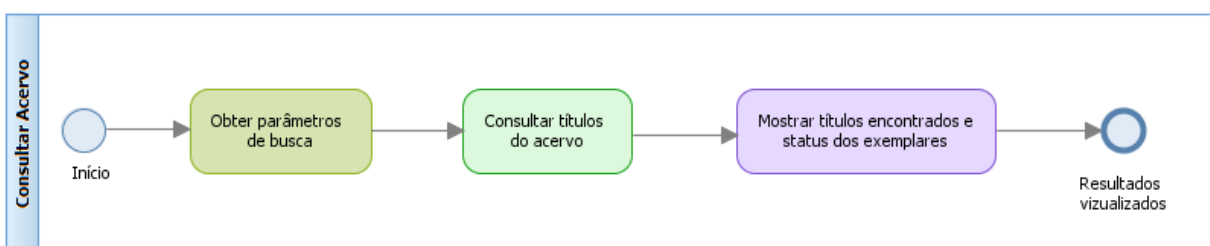


Figura 5.6 Modelo *as-is* do processo de negócio “Consultar acervo” do SGBi.

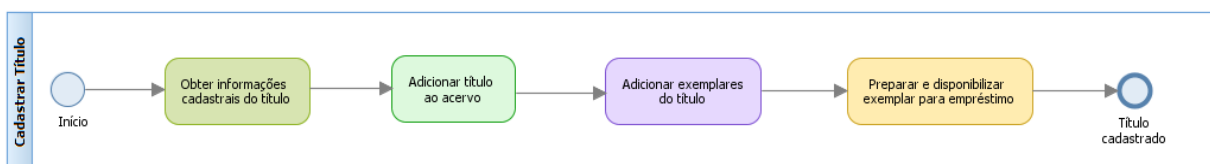


Figura 5.7 Modelo *as-is* do processo de negócio “Cadastrar título” do SGBi.

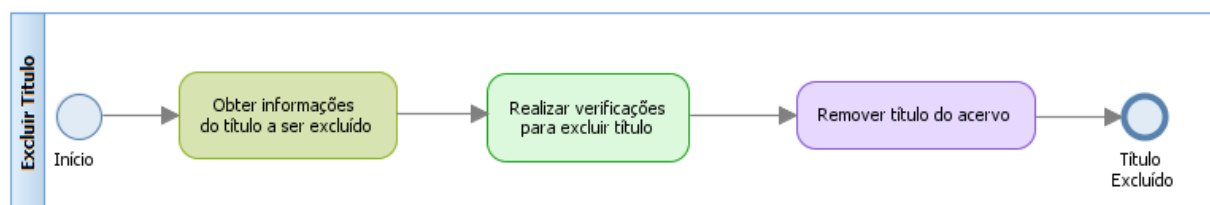


Figura 5.8 Modelo *as-is* do processo de negócio “Excluir título” do SGBi.

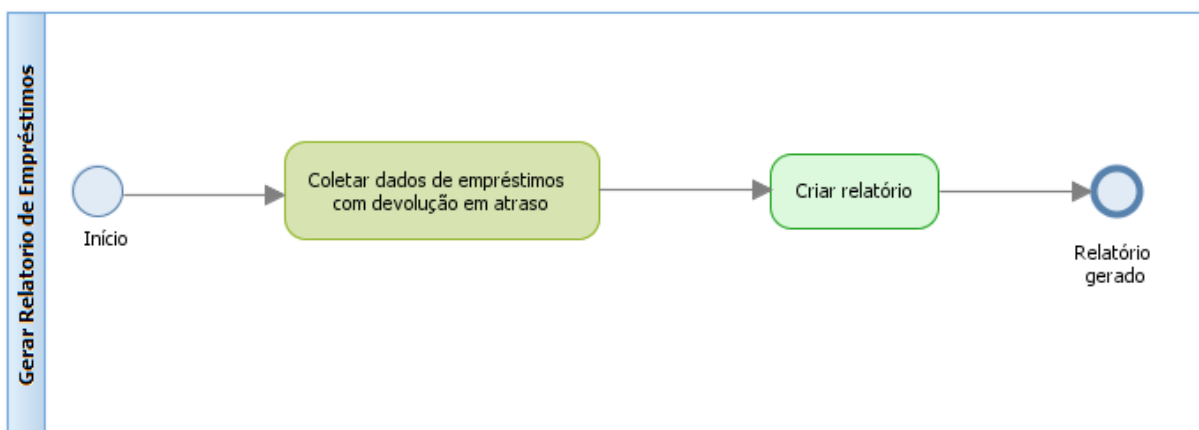


Figura 5.9 Modelo *as-is* do processo de negócio “Gerar relatório de empréstimos em com devolução em atraso” do SGBi.

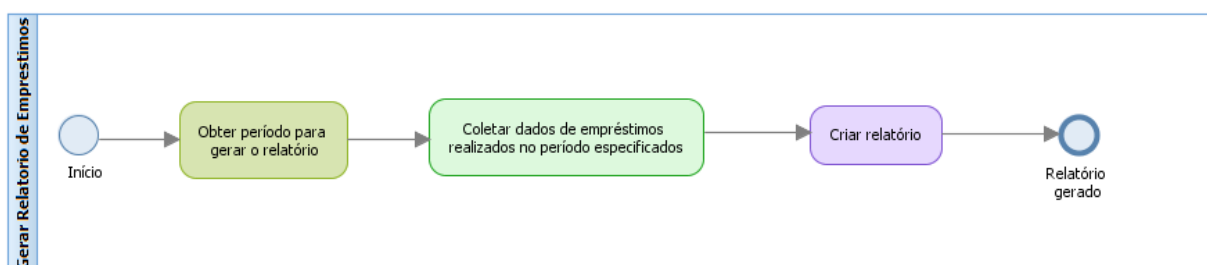


Figura 5.10 Modelo *as-is* do processo de negócio “Gerar relatório de empréstimos por data” do SGBi.

Etapa 1.3: Extrair Requisitos de TI

Após a elaboração e análise do modelo *as-is* do processo de negócio “Cadastrar usuário” e análise de seu contexto de negócio, o desenvolvedor determinou, com base em sua experiência técnica e em conversas com o cliente, que todas as três atividades (“Obter informações do usuário”, “Validar informações” e “Registrar usuário”) poderiam ser informatizadas. Por exemplo, informatizar a atividade “Obter informações do usuário” agilizaria a obtenção dos dados cadastrais da respectiva base de dados da universidade. Assim, essa atividade foi dividida em um conjunto de atividades executadas pelo usuário, pelo atendente e pelo sistema, de forma a permitir a informatização dessa atividade. De forma análoga, a atividade “Validar informações” foi dividida em cinco atividades (três do sistema, uma do atendente e uma do usuário) que realizam a validação do CPF, do email do usuário e dos dados cadastrais obtidos. Por fim, a atividade “Registrar usuário” foi dividida em quatro atividades, duas das quais serão implementadas pelo sistema: “Gerar identificação do usuário” e “Armazenar dados cadastrais”.

O modelo *to-be* resultante é mostrado na Figura 5.11, em que estão representados três atores: usuário (Usuário), atendente (Usuário Administrativo) e Sistema, e suas respectivas atividades. A correlação das atividades do modelo *to-be* com as do modelo *as-is* é explicitada pelas mesmas cores dos elementos nos dois modelos.

De maneira semelhante ao descrito anteriormente, os modelos *as-is* dos demais processos de negócio foram analisados e as atividades que poderiam ser informatizadas foram identificadas, assim como as que eram de responsabilidade do Usuário ou do Usuário Administrativo da biblioteca. Os modelos *to-be* resultantes são mostrados a partir da Figura 5.12 até a Figura 5.19.

Os requisitos do sistema para cada processo de negócio foram documentados textualmente em um documento de requisitos (NAKAGAWA, PENTEADO e SANTOS, 2011a).

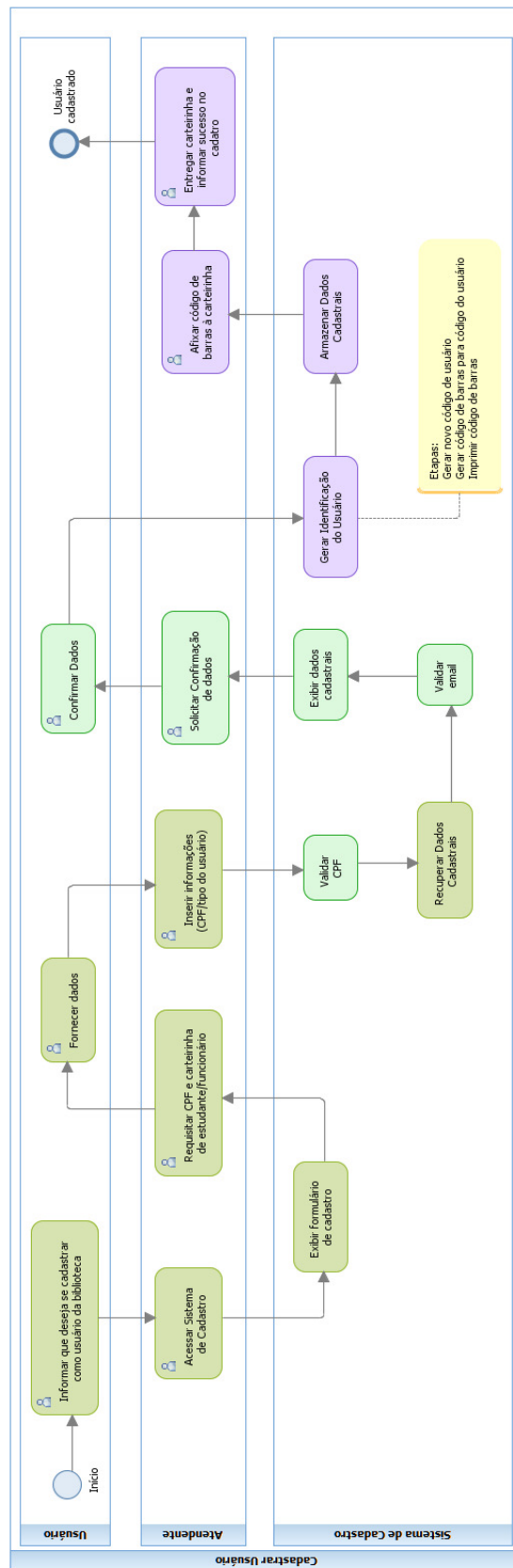


Figura 5.11 Modelo *to-be* do processo de negócio “Cadastrar usuário” do SGBi.

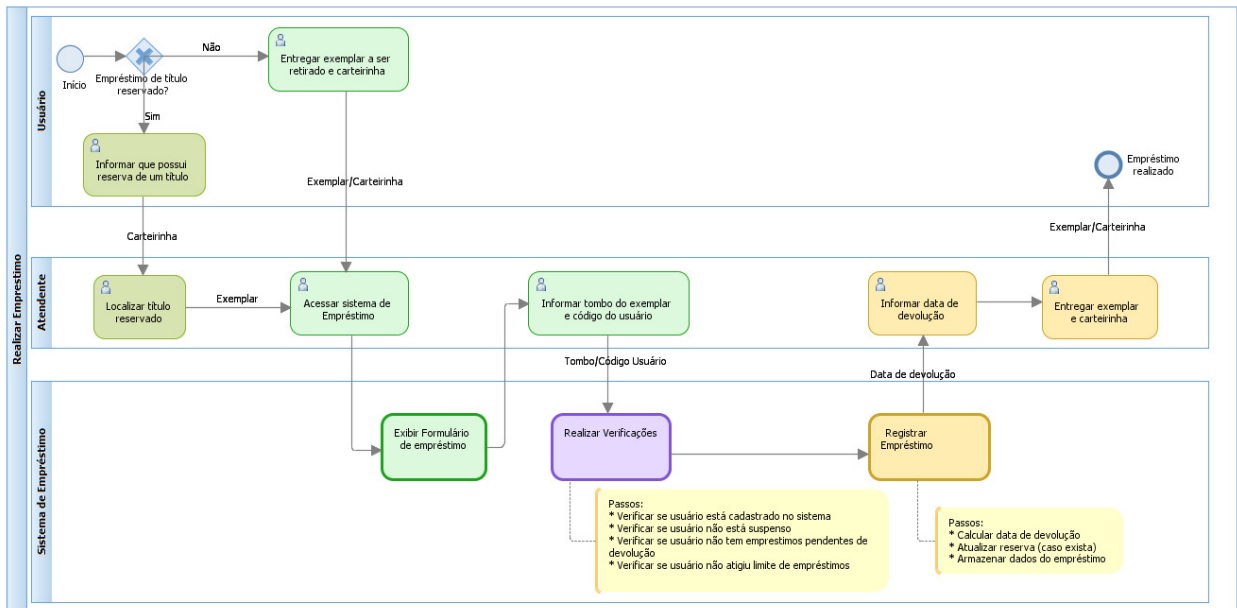


Figura 5.12 Modelo *to-be* do processo de negócio “Realizar empréstimo de exemplar” do SGBi.

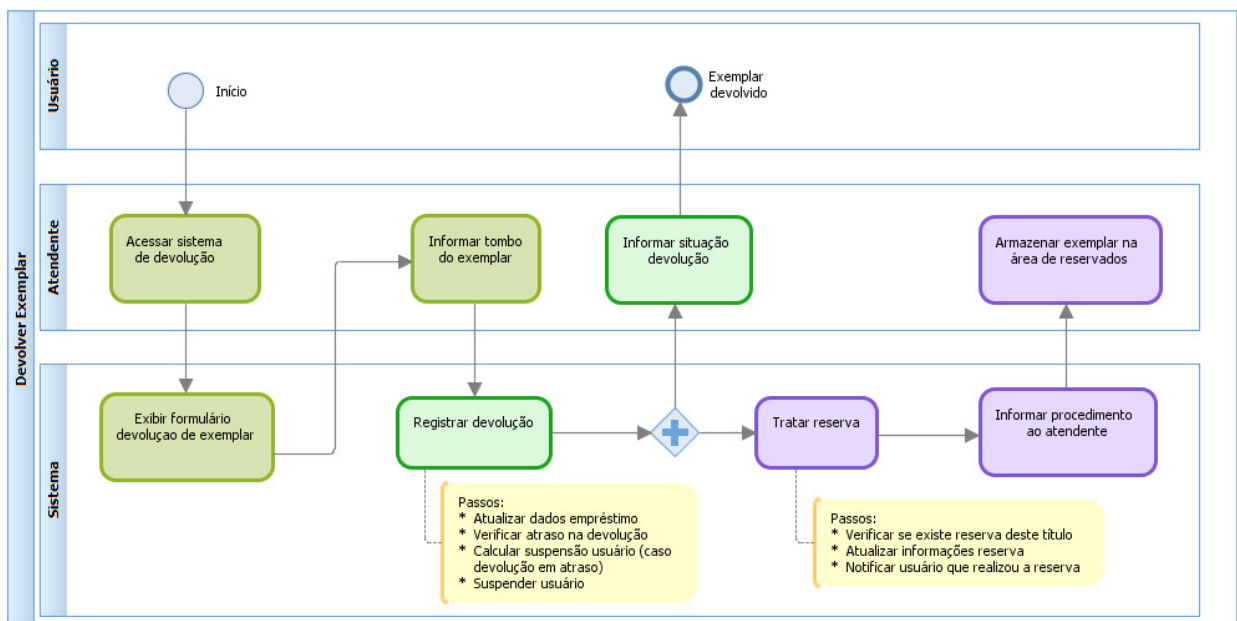


Figura 5.13 Modelo *to-be* do processo de negócio “Devolver exemplar emprestado” do SGBi.

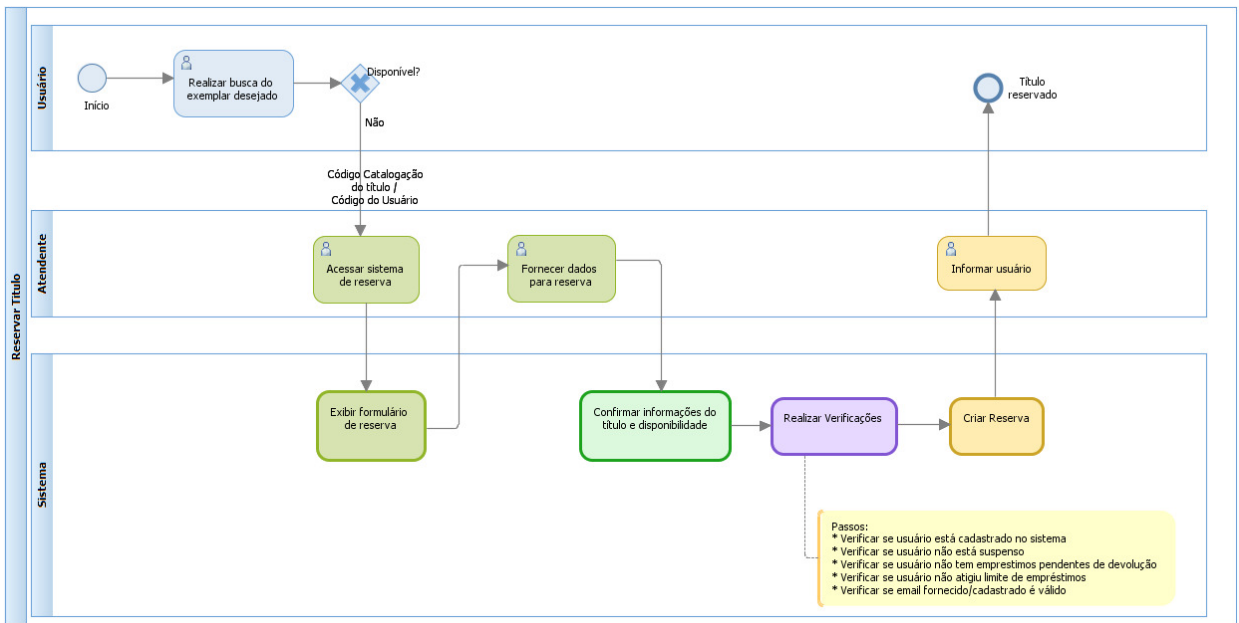


Figura 5.14 Modelo *to-be* do processo de negócio “Reservar título não disponível para empréstimo no momento” do SGBi.

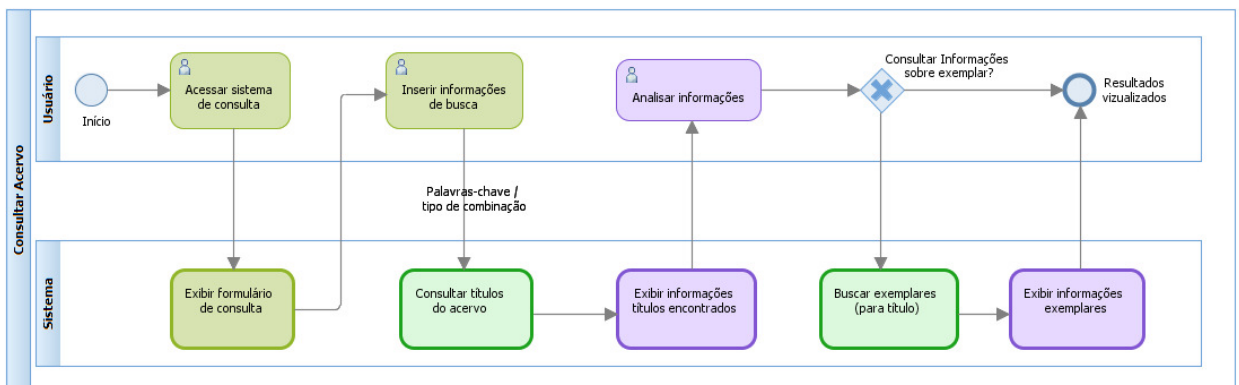


Figura 5.15 Modelo *to-be* do processo de negócio “Consultar acervo” do SGBi.

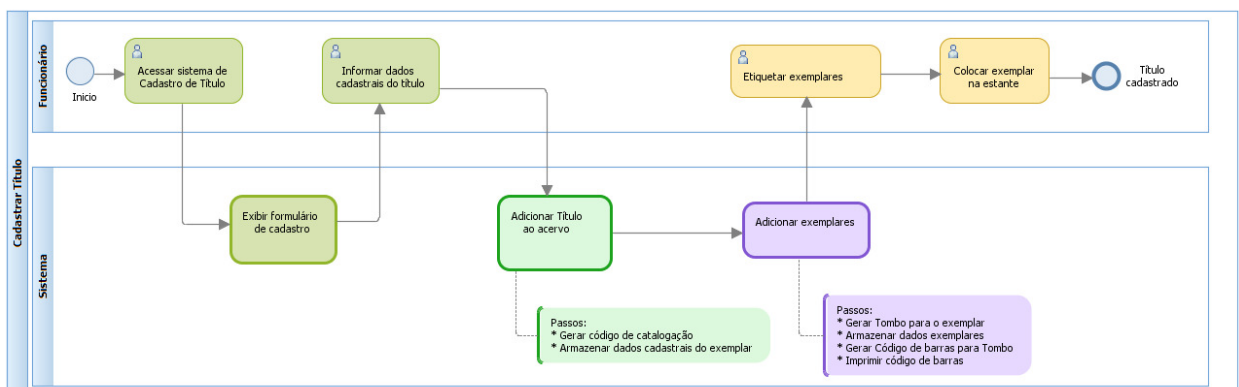


Figura 5.16 Modelo *to-be* do processo de negócio “Cadastrar título” do SGBi.

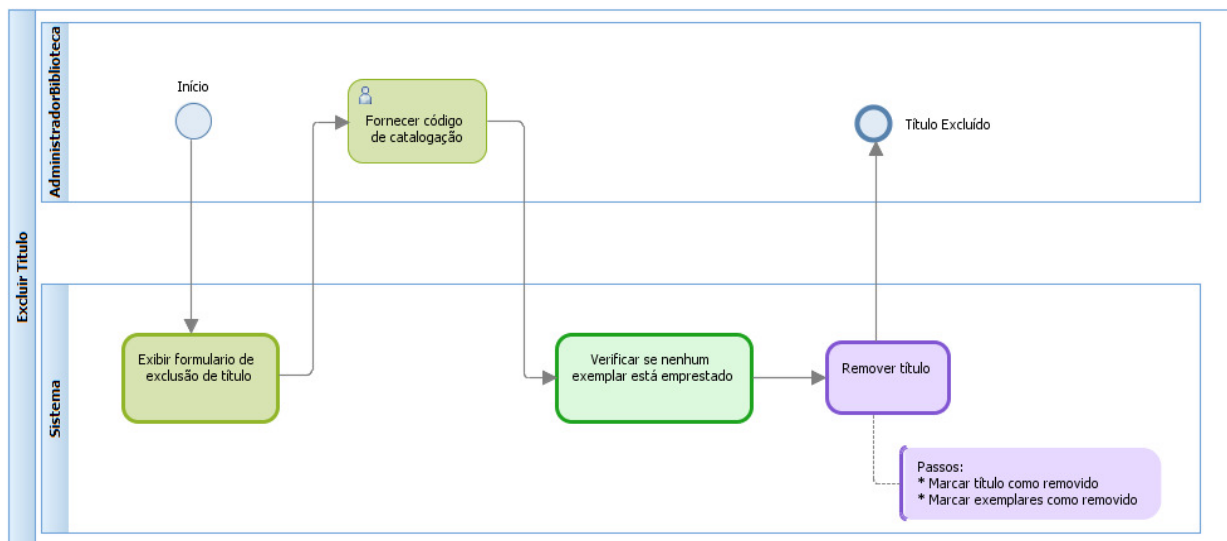


Figura 5.17 Modelo *to-be* do processo de negócio “Excluir título” do SGBi.

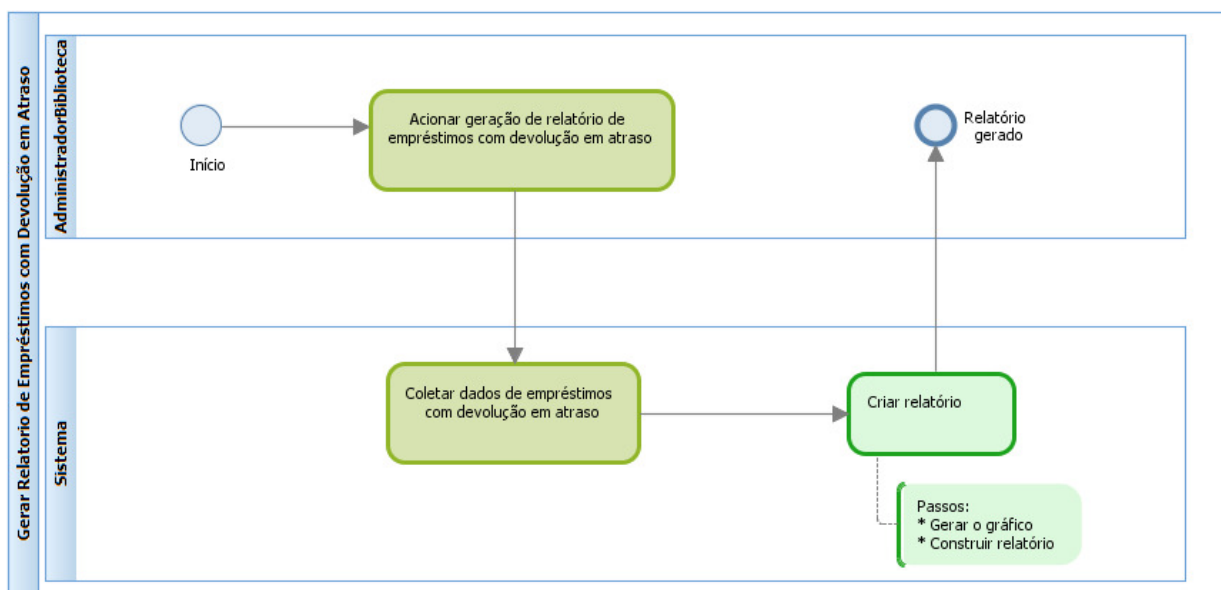


Figura 5.18 Modelo *to-be* do processo de negócio “Gerar relatório de empréstimos com devolução em atraso” do SGBi.

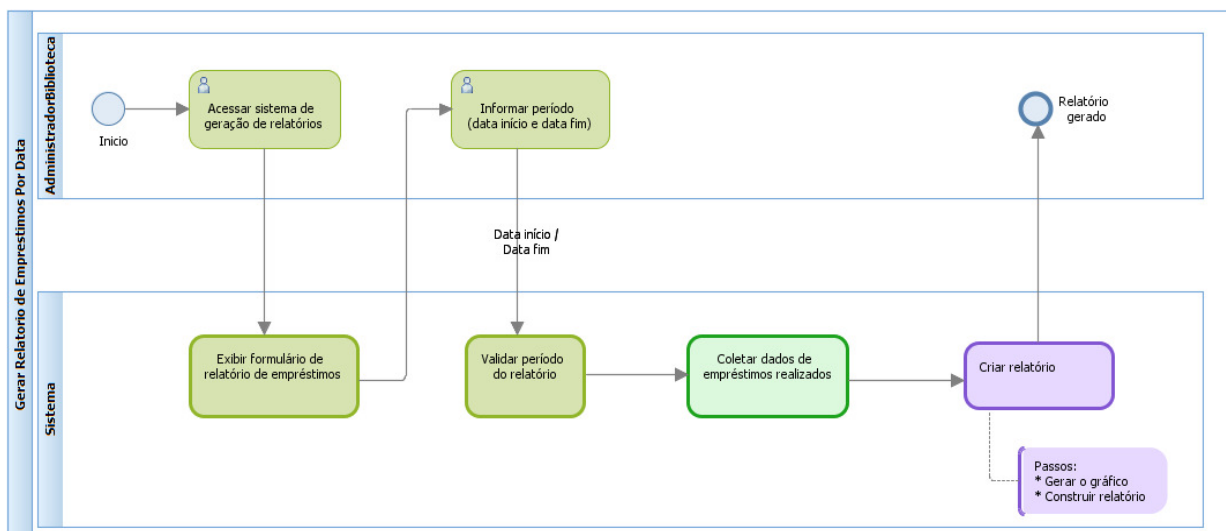


Figura 5.19 Modelo *to-be* do processo de negócio “Gerar relatório de empréstimos por data” do SGBi.

Etapa 1.4: Planejar

Com o auxílio do cliente, os processos de negócio foram classificados de acordo com a sua prioridade para a implementação. Como esta aplicação é hipotética, não há diferenças na prioridade de implementação dos processos, assim, classificação dos processos foi realizada de modo que minimizasse os esforços para geração de dados de teste. A ordem de implementação dos processos é mostrada na Tabela 5.4, que representa o *Backlog* de processos de negócio.

Tabela 5.4 *Backlog* de Processos de Negócio do SGBi com a respectiva estimativa de tempo.

Prioridade	Processo de negócio	Estimativa do número de horas	Comentários
1	Cadastrar usuário	80	
2	Cadastrar título	80	
3	Realizar empréstimo de exemplar	80	
4	Devolver exemplar emprestado	80	
5	Reservar título não disponível para empréstimo no momento	80	
6	Consultar acervo	60	
7	Gerar relatório de empréstimos realizados por data	60	
8	Gerar relatório de empréstimos com devolução em atraso	60	
9	Excluir título	60	

Com base na experiência do desenvolvedor e nas informações obtidas estimou-se o número de horas necessárias para implementar cada processo de negócio.

5.2.1.2 Fase 2: Contextualizar PN com Serviços

Etapa 2.1: Selecionar PN

Como a ordem apresentada na Tabela 5.4 é utilizada para implementação dos processos, o processo “Cadastrar usuário” é o processo selecionado, e portanto considerado nas etapas subsequentes da aplicação da IASWS neste exemplo de aplicação da IASWS.

Etapa 2.2: Identificar e Buscar Serviços

Após analisar os requisitos do sistema, representados no modelo *to-be* do processo “Cadastrar usuário”, foram identificados os serviços candidatos com base nas diretrizes propostas na IASWS para tal fim. O primeiro serviço identificado foi o próprio processo de negócio “Cadastrar usuário”, porém nenhum serviço foi encontrado na busca realizada. A busca por serviços foi realizada nos repositórios *Amazon Web Services* (AWS, 2011) e *Yahoo! Developer Network* (YDN, 2011). Optou-se também por buscar serviços disponibilizados na internet (qualquer serviço que estivesse operacional, incluindo os gratuitos) com o intuito de aumentar as possibilidades de obter sucesso na busca e possibilitar o reúso de serviços na solução. O mecanismo de busca embutido nesses repositórios foi utilizado, enquanto que o Google foi utilizado para buscar serviços publicados na internet. A busca por serviços foi realizada textualmente, utilizando palavras-chave relacionadas à funcionalidade requerida.

O próximo passo foi procurar por uma funcionalidade mais genérica do que a fornecida pelo processo de negócio. As funcionalidades “Cadastrar dados” ou “Cadastrar entidade” poderiam ser utilizadas para “Cadastrar dados de usuário” ou “Cadastrar entidade usuário”, mas também não foi encontrado nenhum serviço que fornecesse tais funcionalidades. Passou-se então para a análise das atividades desse processo de negócio que poderiam ser executadas por um serviço. As atividades “Exibir formulário de cadastro”, “Validar CPF”, “Validar email” e “Recuperar dados cadastrais” foram identificadas como serviços candidatos. Foram

encontrados serviços disponíveis na internet apenas para as funções “Validar CPF” e “Validar email”.

Analisando a atividade “Gerar Identificação de usuário”, verificou-se que esta atividade necessita “Gerar código de barras”. Assim, essa tarefa foi identificada como um serviço candidato. A busca por serviços que fornecessem essa funcionalidade não retornou nenhum serviço. Não foi possível realizar nenhum agrupamento de atividades para esse processo de negócio. Nenhuma função relacionada a regras de negócios, entidades de dados ou requisitos não-funcionais foi considerada como serviço candidato.

Na Tabela 5.5 estão mostradas as funcionalidades que foram identificadas como serviço candidato e o resultado da busca por um serviço que fornecesse a funcionalidade. A URL para o arquivo de descrição do serviço foi documentada nos casos em que um serviço foi encontrado.

Tabela 5.5 Serviços candidatos identificados para o SGBi.

	Funcionalidade	Serviço encontrado?	Informações sobre serviço (caso encontrado)
	Cadastrar usuário	Não	N/A
	Cadastrar dados	Não	N/A
	Cadastrar entidade	Não	N/A
Cadastrar usuário	Exibir formulário de cadastro	Não	N/A
	Recuperar dados cadastrais	Não	N/A
	Validar CPF	Sim	CPF <i>Service</i> http://www.bronzebusiness.com.br/webservices/valida.asmx? (acesso em 16 Fev. 2014) Custo: gratuito
	Validar email	Sim	Tower Data <i>Email Validation Service</i> http://soap.towerdata.com/validate.wsdl (acesso em 16 Fev. 2014) Custo: gratuito
	Gerar código de barras	Não	N/A

Etapa 2.3: Elaborar Proposta de Solução

Para cada serviço candidato identificado anteriormente, avaliaram-se a viabilidade e a relação custo-benefício quanto a desenvolver um novo serviço que fornecesse a funcionalidade desejada, quanto a implementar a funcionalidade usando classes OO e quanto a reusar serviços encontrados.

Nesta implementação, um novo serviço será desenvolvido para desempenhar a função da atividade “Recuperar dados cadastrais” visto que outras aplicações

poderiam fazer uso dessa funcionalidade quando necessitarem obter dados de alunos, docentes e funcionários das bases de dados da universidade. Informações sobre esse serviço são mostradas na Tabela 5.6.

Tabela 5.6 Serviço que fornece a funcionalidade “Recuperar dados cadastrais” para o SGBi.

Serviço:	Recuperar dados cadastrais
Descrição do serviço:	Permite recuperar todas as informações cadastrais de uma entidade de dados do respectivo banco de dados (RH ou Secgrad). A entidade deve necessariamente possuir um campo que a identifique univocamente (chave primária) na tabela.
Funcionalidade atendida:	Função desempenhada pela tarefa “Recuperar dados cadastrais”
Descrição da funcionalidade atendida:	Recuperar dados cadastrais do Sistema de RH ou da Secgrad, conforme o tipo de usuário.
Esforço estimado de desenvolvimento	Tempo de desenvolvimento: 20 horas

Outro serviço a ser desenvolvido desempenhará a função da tarefa “Gerar código de barras” da atividade “Gerar identificação do usuário”. Esse serviço irá gerar a imagem do código de barras referente a uma cadeia de caracteres fornecida como parâmetro de entrada, como mostrado na Tabela 5.7.

Tabela 5.7 Serviço que fornece a funcionalidade “Gerar código de barras” para o SGBi.

Serviço:	Gerar imagem contendo código de barras
Descrição do serviço:	Gerar a imagem contendo o código de barras referente a uma <i>string</i> fornecida.
Funcionalidade atendida:	Função desempenhada pela tarefa “Gerar código de barras” da atividade “Gerar identificação de usuário”.
Descrição da funcionalidade atendida:	Gera o código de barras a partir de uma cadeia de caracteres alfanuméricos fornecida.
Esforço estimado de desenvolvimento	Tempo de desenvolvimento: 20 horas

Os serviços encontrados para as funções “Validar CPF” e “Validar email” serão reutilizados nesta aplicação e são apresentados na Tabela 5.8 e na Tabela 5.9, respectivamente. Cabe ressaltar que em uma situação real ambos os serviços não seriam utilizados, pois apesar de serem gratuitos, não há informações quanto à estabilidade e confiabilidade do provedor de serviço.

O cliente solicitou que o processo de negócio “Cadastrar usuário” fosse implementado como uma aplicação acessível a partir de qualquer computador conectado à rede da biblioteca. Para atender a essa solicitação, o desenvolvedor analisou as características de diversos tipos de sistema e optou por um sistema *web*

devido à facilidade para acessá-lo por meio de um navegador de internet. Assim as demais funcionalidades do sistema (não fornecidas por serviços) serão implementadas utilizando classes OO, páginas HTML/JSP e *Servlets*.

Tabela 5.8 Informações do serviço de validação de CPF “CPF Service” para o SGBi.

Serviço:	CPF Service
Descrição do serviço:	Valida se o número do CPF está correto. Não é verificado se o CPF fornecido pertence a alguma pessoa.
Funcionalidade atendida:	Função desempenhada pela tarefa “Validar CPF”
Descrição da funcionalidade atendida:	Verifica se o CPF informado pelo atendente é um número de CPF válido.
Custos do uso do serviço	Gratuito, será utilizado por necessidade de validar o reúso de serviços na abordagem proposta.
URL para o WSDL do serviço	http://www.bronzebusiness.com.br/webservices/valida.asmx? (acesso em 16 Fev. 2014)

Tabela 5.9 Informações do serviço de validação de endereços de email “Tower Data Email Validation Service” para o SGBi.

Serviço:	Tower Data Email Validation Service
Descrição do serviço:	Valida um endereço de email. Pode-se validar a sintaxe de um endereço de email, validar o domínio, confirmar se o domínio pode receber emails, confirmar se o endereço de email pode receber mensagens.
Funcionalidade atendida:	Função desempenhada pela tarefa “Validar email”
Descrição da funcionalidade atendida:	Faz a validação do email armazenado nas informações cadastrais do usuário.
Custos do uso do serviço	Gratuito para testes, será utilizado por necessidade de validar o reúso de serviços na abordagem proposta.
URL para o WSDL do serviço	http://soap.towerdata.com/validate.wsdl (acesso em 16 Fev. 2014)

Dessa forma, o processo de negócio “Cadastrar usuário” será implementado como uma solução *web* composta dos seguintes elementos:

- Páginas *web* para: exibir o formulário de informações do usuário que se deseja cadastrar; exibir as informações desse usuário (recuperadas das bases de dados da universidade); exibir eventuais erros no processamento; exibir o resultado do cadastro do usuário, permitindo a impressão do código de barras referente ao código atribuído ao usuário;
- *Servlets* atuando como controladores de fluxo, invocando as funcionalidades fornecidas por serviços e classes OO;
- Serviços;
- Classes OO que implementam as demais tarefas do processo de negócio “Cadastrar usuário”.

Etapa 2.4: Avaliar Serviços Reusados

Esta etapa é necessária para que os serviços que serão reusados sejam validados. Esses serviços são do tipo XML *Web Services* e possuem o arquivo de descrição do serviço (WSDL) disponibilizado na internet. Neste exemplo esses arquivos foram importados na ferramenta RSA (IBM, 2010a) que gerou automaticamente as classes cliente desses *web services*.

O código para testar o *web service* “Validar CPF” pôde ser criado rapidamente em decorrência da simplicidade do serviço e de seus parâmetros de entrada e saída. Os casos de teste criados para testar esse *web service* são mostrados na Tabela 5.10.

Tabela 5.10 Casos de teste para o serviço de validação de CPF “CPF Service” do SGBi.

# caso de teste	Descrição	Valores de entrada	Saída esperada
1	CPF válido	035.921.809-16	Válido
2	CPF válido	03592180916	Válido
3	CPF válido	035.921.80916	Válido
4	CPF válido	035921809-16	Válido
5	CPF inválido	123.123.123-12	Inválido
6	CPF inválido	12312312312	Inválido
7	CPF inválido	123.123.12312	Inválido
8	CPF inválido	123123123-12	Inválido
9	CPF inválido	xyz.abc.xyz-dd	Inválido
10	CPF inválido	123.xyz.123.xx	Inválido

Após a execução desses casos de teste, nenhum problema foi detectado e o serviço foi considerado reusável na solução.

A criação do código para invocar o *web service* “*Tower Data Email Validation*” demandou mais esforço para sua construção visto que o serviço possui diversos tipos de parâmetros e configurações que podem ser usados em sua invocação, além do fato que não há uma documentação apropriada para o *web service*. Durante os testes foi detectada certa instabilidade no serviço, mas, como serviço é gratuito, foi considerado como reusável. Os casos de teste utilizados são apresentados na Tabela 5.11.

Tabela 5.11 Casos de teste para o serviço de validação de endereços de email “Tower Data Email Validation Service” do SGBi.

# caso de teste	Descrição	Valores de entrada	Saída esperada
1	Email válido	hiromiti@gmail.com	Válido
2	Email válido	hiron@br.ibm.com	Válido
3	Email válido	santos@dc.ufscar.br	Válido
4	Email inválido (domínio inválido)	hiromiti@gmail.com.br	Inválido
5	Email inválido (mailbox inválido)	hiromiti123456@gmail.com	Inválido
6	Email inválido (domínio inválido)	hiromiti@ufscar.com.br	Inválido

Etapa 2.5: Criar Casos de Testes de Novos Serviços

Os casos de teste para avaliar os serviços “Recuperar dados cadastrais” (Tabela 5.12) e “Gerar código de barras” (Tabela 5.13), que serão desenvolvidos nessa iteração, foram elaborados a partir da análise das informações do serviço e das funcionalidades fornecidas.

Tabela 5.12 Casos de teste para serviço “Recuperar dados cadastrais” do SGBi.

Serviço: Recuperar Dados Cadastrais			
# caso de teste	Descrição	Valores de entrada	Saída esperada
1	Recuperar dados de aluno cadastrado na base de dados da universidade	CPF: 11 dígitos sem pontos nem traço Tipo usuário: aluno	Dados cadastrais do aluno.
2	Recuperar dados de aluno que não está cadastrado na base de dados da universidade	CPF: 11 dígitos sem pontos nem traço Tipo usuário: aluno	Mensagem informando que aluno não está cadastrado no sistema Secgrad.
3	Recuperar dados de funcionário cadastrado na base de dados da universidade	CPF: 11 dígitos sem pontos nem traço Tipo usuário: Funcionário	Dados cadastrais do funcionário.
4	Recuperar dados de funcionário que não está cadastrado na base de dados da universidade.	CPF: 11 dígitos sem pontos nem traço Tipo usuário: Funcionário	Mensagem informando que funcionário não está cadastrado na base de dados da universidade.

Tabela 5.13 Casos de teste para serviço “Gerar código de barras” do SGBi.

Serviço: Gerar Código de barras			
# caso de teste	Descrição	Valores de entrada	Saída esperada
1	Gerar código de barras referente a uma sequência de caracteres alfabéticos	Abcxyz	Imagem contendo o código de barras
2	Gerar código de barras referente a uma sequência de caracteres numéricos	123456	Imagem contendo o código de barras
3	Gerar código de barras referente a uma sequência de caracteres alfanuméricos	abc123xyz456	Imagem contendo o código de barras
4	Gerar código de barras referente a uma sequência de caracteres que não pode ser representada com código de barras	A1&X123@XYZ	Mensagem informando que código de barras não pode ser gerado

Etapa 2.6: Modelar Novos Serviços

Para elaborar o modelo de análise de novos serviços, Figura 5.20, foi utilizada a ferramenta RSA, o perfil SoaML (OMG, 2009b) e as informações dos serviços que serão desenvolvidos (“Recuperar dados cadastrais” e “Gerar código de barras”). Nesse modelo, o elemento “*Capability*” do SoaML foi usado para representar os serviços. Os elementos “*ServiceInterface*” e respectivas operações especificam como as funcionalidades fornecidas por esses serviços podem ser invocadas.

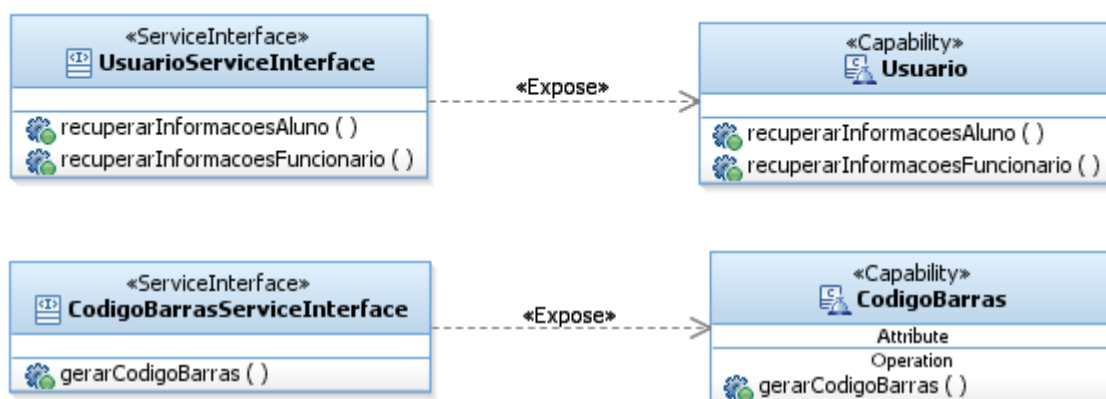


Figura 5.20 Modelo de análise de novos serviços.

Etapa 2.7: Criar Casos de Teste da Solução

Analisando as informações do processo de negócio e as atividades representadas no modelo *to-be* foi possível identificar os diferentes cenários para execução do processo “Cadastrar usuário”. Desses cenários, foram selecionados aqueles que permitem formar um conjunto mínimo, capaz de exercitar cada atividade do processo de negócio ao menos uma vez. Na Tabela 5.14 estão listados os casos de teste elaborados.

Tabela 5.14 Casos de teste para o PN “Cadastrar usuário” do SGBi

Processo de negócio: Cadastrar Usuário			
# caso de teste	Descrição	Valores de entrada	Saída esperada
1	Tentar cadastrar usuário do tipo Aluno sem fornecer o CPF	CPF: não fornecer Tipo usuário: aluno	Mensagem de erro informando que o CPF deve ser informado
2	Tentar cadastrar usuário do tipo Aluno fornecendo um CPF inválido	CPF: 11 dígitos sem pontos e traço e inválido Tipo usuário: aluno	Mensagem de erro informando que o CPF informado é inválido
3	Tentar cadastrar usuário sem informar o tipo do usuário	CPF: 11 dígitos sem pontos e traço Tipo usuário: não fornecer	Mensagem de erro informando que o tipo do usuário deve ser informado
4	Cadastrar usuário do tipo aluno que esteja cadastrado na base de dados da universidade	CPF: 11 dígitos sem pontos e traço e existente na base de dados da universidade Tipo usuário: aluno	Código de barras impresso Mensagem informando que o cadastro foi concluído
5	Cadastrar usuário do tipo aluno com email inválido	CPF: 11 dígitos sem pontos e traço e existente na base de dados da universidade com email inválido Tipo usuário: aluno	Mensagem informando que o email do usuário é inválido e solicitando a inserção de um novo email
6	Cadastrar usuário do tipo aluno que não esteja cadastrado na base de dados da universidade	CPF: 11 dígitos sem pontos e traço e não existente na base de dados da universidade Tipo usuário: aluno	Mensagem de erro informando que o CPF informado não consta na base de dados de alunos da universidade
7	Cadastrar usuário do tipo Docente que esteja cadastrado na base de dados da universidade	CPF: 11 dígitos sem pontos e traço e existente na base de dados da universidade Tipo usuário: Docente	Código de barras impresso Mensagem informando que o cadastro foi concluído
8	Cadastrar usuário do tipo Docente com email inválido	CPF: 11 dígitos sem pontos e traço e existente na base de dados da universidade com email inválido Tipo usuário: Docente	Mensagem informando que o email do usuário é inválido e solicitando a inserção de um novo email
9	Cadastrar usuário do tipo Docente que não esteja cadastrado na base de dados da universidade	CPF: 11 dígitos sem pontos e traço e não existente na base de dados da universidade Tipo usuário: Docente	Mensagem de erro informando que o CPF informado não consta na base de dados de funcionários da universidade
10	Cadastrar usuário do tipo Técnico-Administrativo que esteja cadastrado na base de dados da universidade	CPF: 11 dígitos sem pontos e traço e existente na base de dados da universidade Tipo usuário: Técnico-Administrativo	Código de barras impresso Mensagem informando que o cadastro foi concluído
11	Cadastrar usuário do tipo Técnico-Administrativo com email inválido	CPF: 11 dígitos sem pontos e traço e existente na base de dados da universidade com email inválido Tipo usuário: Técnico-Administrativo	Mensagem informando que o email do usuário é inválido e solicitando a inserção de um novo email
12	Cadastrar usuário do tipo Técnico-Administrativo que não esteja cadastrado na base de dados da universidade	CPF: 11 dígitos sem pontos e traço e não existente na base de dados da universidade Tipo usuário: Técnico-Administrativo	Mensagem de erro informando que o CPF informado não consta na base de dados de funcionários da universidade

5.2.1.3 Fase 3: Projetar serviços

Etapa 3.1: Definir API de Implementação de Web Services

O desenvolvedor optou pela API JAX-WS para implementar os serviços “Recuperar Dados Cadastrais” e “Gerar Código de barras” por se tratar da especificação mais recente para implementação de *web services* na plataforma Java e pelo suporte nativo fornecido pela ferramenta RSA, adotada neste exemplo de aplicação da IASWS.

Etapa 3.2: Definir Estratégia de Desenvolvimento

O desenvolvedor decidiu implementar o serviço “Recuperar dados cadastrais” utilizando a estratégia *top-down*, pois a ferramenta RSA fornece recursos para o desenvolvimento de serviços seguindo essa estratégia. Nessa estratégia, o serviço deve, primeiramente, ser especificado para em seguida ser implementado. A especificação de um serviço apresenta duas vantagens principais: auxilia no seu desenvolvimento; e serve como documentação.

Já para o serviço “Gerar código de barras” adotou-se a estratégia *bottom-up*, considerando suas características que requerem que o serviço retorne a imagem do código de barras gerado. Como a imagem é um dado binário, sua representação pode dificultar o trabalho de especificação do serviço e por isso, a adoção de uma estratégia *bottom-up* seria interessante, permitindo criar o código do serviço e, em seguida, utilizar os recursos da ferramenta RSA para gerar o serviço e sua descrição a partir do código.

Etapa 3.3: Definir Arquitetura do Serviço

O serviço “Recuperar dados cadastrais” terá acesso restrito (permissão de leitura) aos bancos de dados ALUNOS e FUNCIONARIOS, existente na universidade. Esse serviço foi implementado na plataforma Java (usando a API JAX-WS), foi implantado no servidor de aplicações IBM *Websphere Application Server* versão 7 e teve acesso aos bancos de dados por meio de conexões JDBC, configuradas no servidor de aplicações.

A arquitetura do serviço foi representada por meio de um diagrama de *deployment* fornecido pela ferramenta RSA, mostrado na Figura 5.21 e foi construído com elementos UML e estereótipos definidos pelo autor para representar os

componentes da arquitetura: servidor de aplicações (“*JEE Container*”), servidor de banco de dados (“*DB2 Server*”), o banco de dados (“*Database*”) e o arquivo que contém o código-fonte implantado no servidor (“*EAR file*”). Nesse diagrama foi representado o servidor da universidade, onde os bancos de dados utilizados estão alocados, e o servidor “DC” que hospeda o servidor de aplicações que executa os artefatos de código.

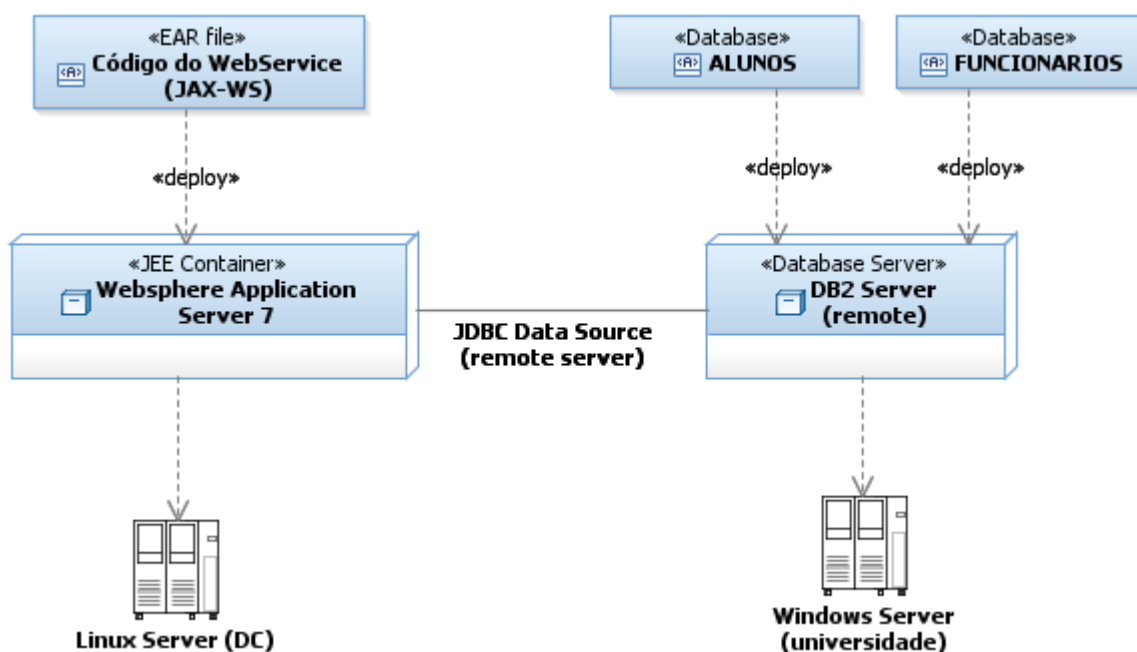


Figura 5.21 Arquitetura do serviço “Recuperar Dados Cadastrais” usado no SGBi.

O serviço “Gerar código de barras” não necessita realizar acessos a bancos de dados ou utilizar outros componentes, conseqüentemente sua arquitetura é bastante simples e é mostrada na Figura 5.22. Esse serviço necessita apenas do ambiente de execução fornecido pelo servidor de aplicações *Websphere Application Server* versão 7.

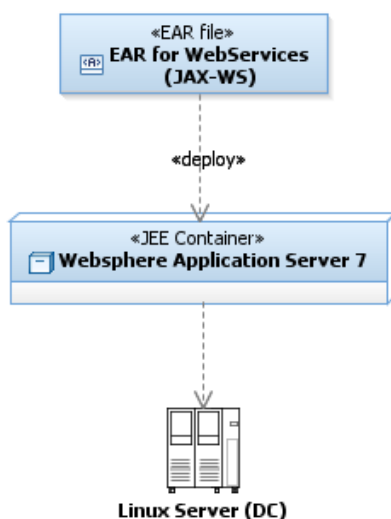


Figura 5.22 Arquitetura do serviço “Gerar código de barras” usado no SGBi.

Etapa 3.4: Especificar Serviço

O desenvolvedor realizou a especificação do serviço “Recuperar dados cadastrais” refinando a modelagem desse serviço, criada na etapa “Modelar Novos Serviços”. Para cada operação listada no elemento *ServiceInterface* “UsuarioServiceInterface”, foram especificados o tipo de dados (mensagem) retornado pela operação, os parâmetros de entrada e respectivos tipos de dados. As mensagens retornadas pelo serviço foram especificadas utilizando-se o elemento *MessageType* do SoaML. Por fim, foi especificado o provedor do serviço por meio do elemento *Participant* que foi adicionado à modelagem. Esse provedor disponibilizará o serviço usando uma porta, especificada pelo elemento *ServiceInterface* citado anteriormente. Na Figura 5.23 é mostrada a modelagem resultante dos refinamentos.

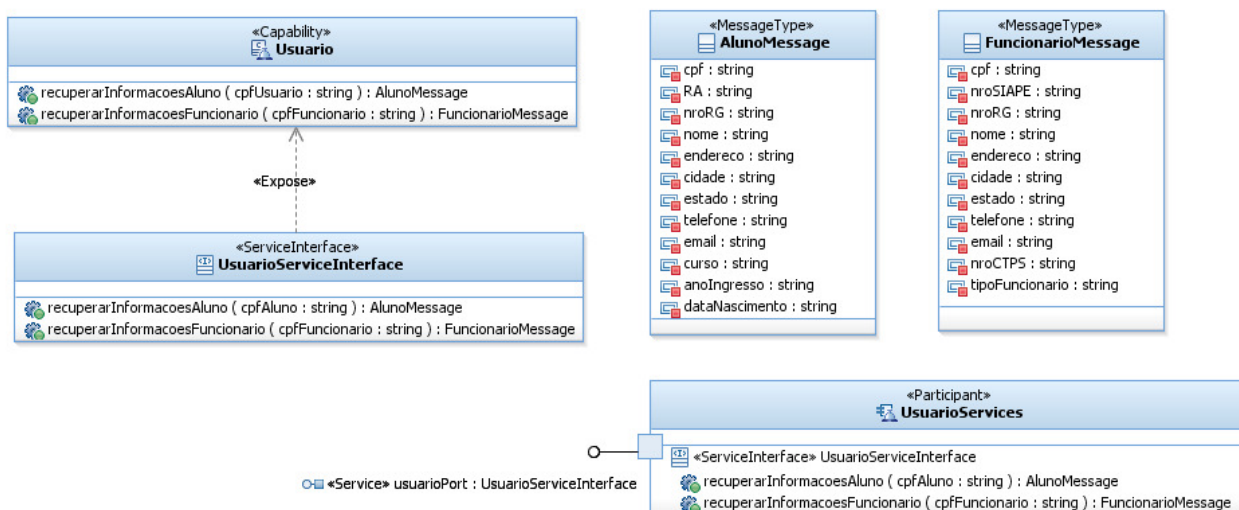


Figura 5.23 Especificação do serviço “Recuperar dados cadastrais” para o SGBi.

Etapa 3.5: Verificar serviço

Após a especificação do serviço “Recuperar dados cadastrais”, verificou-se que sua especificação estava em conformidade com os princípios da orientação a serviços. A granularidade desse serviço é adequada para permitir que a funcionalidade oferecida possa ser reusada em outros sistemas. A interface desse serviço estabelece um contrato padronizado para sua invocação, abstraindo os detalhes de sua implementação de forma adequada. A funcionalidade fornecida por esse serviço é independente de estado (*stateless*) e não utiliza nenhum outro serviço, não existindo, portanto, acoplamento entre serviços. Dessa forma, o serviço “Recuperar dados cadastrais” está de acordo com os princípios da orientação a serviços.

5.2.1.4 Fase 4: Projetar solução

Etapa 4.1: Definir Arquitetura da Solução

O sistema *web* que implementa a solução proposta será executado em um servidor de aplicações JEE (*web*), utilizará um servidor de banco de dados para armazenar as informações dos usuários e quatro *web services*, identificados anteriormente. Uma conexão JDBC configurada no servidor de aplicações permitirá que o sistema acesse o banco de dados.

A arquitetura definida para implementar a solução foi representada com um diagrama de *deployment* elaborado com a ferramenta RSA. Esse diagrama é mostrado na Figura 5.24 e foi construído usando elementos UML e estereótipos definidos pelo autor para representar os componentes da arquitetura: servidor de aplicações (“*JEE Container*”), servidor de banco de dados (“*Database Server*”), o banco de dados da aplicação (“*Application Database*”), *web services* (“*WebService*”) e o arquivo que contém o código-fonte no servidor (“*EAR file*”). A conexão JDBC está representada por uma associação entre os elementos que representam o servidor de aplicação e de banco de dados. Outra informação necessária é elemento “Linux Server (dc)”, que é o *hardware* para os dois servidores (de aplicação e de banco de dados).

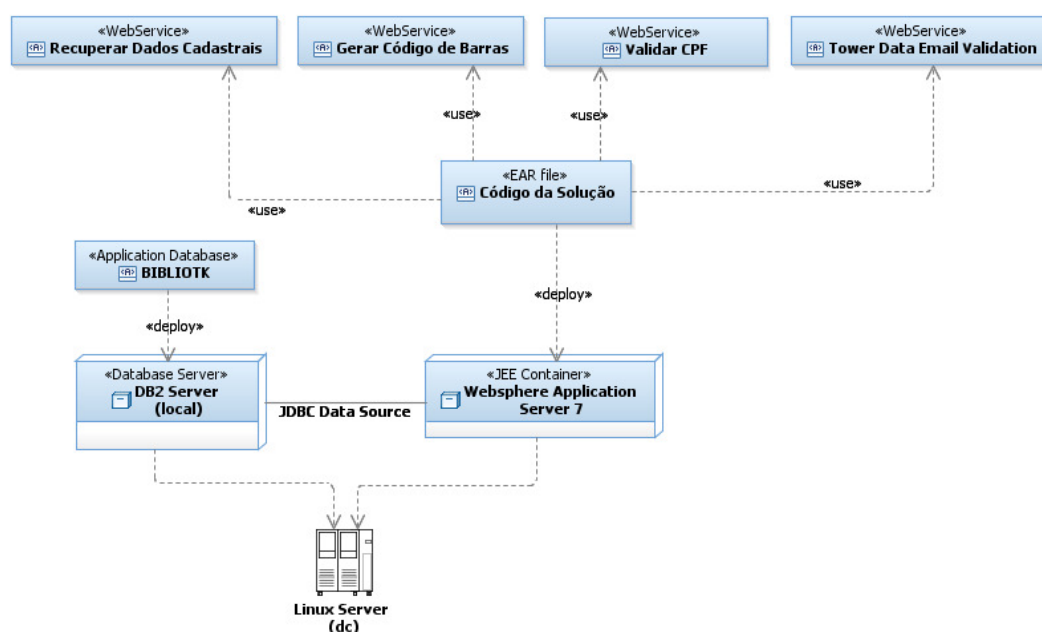


Figura 5.24 Arquitetura da solução para o PN “Cadastrar Usuário” do SGBi.

Etapa 4.2: Criar Modelo de Dados

O desenvolvedor analisou o modelo *to-be* do processo “Cadastrar usuário” e suas informações e identificou que a entidade de dados “Usuário”, está envolvida nesse processo. Após verificar os campos dessa entidade pôde-se observar que ela não se relaciona a nenhuma outra no contexto desse processo de negócio. Conforme definido anteriormente, os dados cadastrais de funcionários e alunos não serão replicados localmente (exceto o *email*), pois serão obtidos diretamente das respectivas bases de dados. O campo *email* foi replicado localmente para permitir

sua atualização de modo mais fácil considerando sua importância na implementação do SGBi.

Em seguida, o desenvolvedor realizou o mapeamento da entidade “Usuário” para o modelo relacional e criou as tabelas: USUARIO e TIPOUSUARIO que tem o relacionamento “é do tipo”. A tabela USUARIO contém as informações de um usuário enquanto a tabela TIPOUSUARIO armazena os tipos de usuário existentes (aluno, docente, técnico-administrativo). Na Figura 5.25 é mostrada a modelagem dessas tabelas, seu relacionamento e respectivos campos com seus respectivos tipos de dados, tamanho máximo e obrigatoriedade especificados. A modelagem apresentada foi criada na ferramenta RSA, utilizando o recurso de criação de modelo físico de dados.

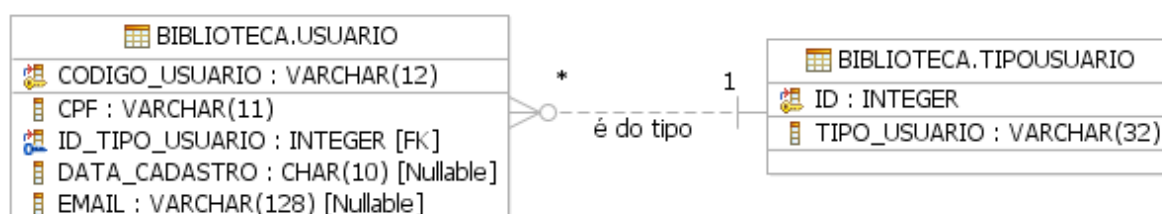


Figura 5.25 Modelagem de dados para implementação do PN “Cadastrar Usuário” do SGBi.

Para fins de documentação, as tabelas que contêm os dados cadastrais de alunos e funcionários nas respectivas bases de dados foram representadas no diagrama que é exibido na Figura 5.26.

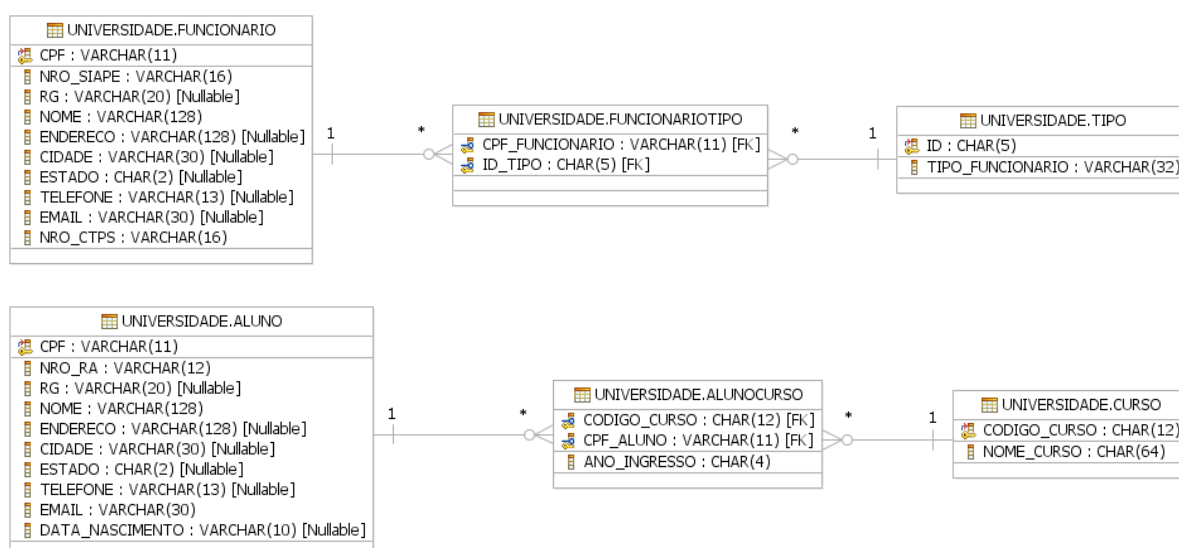


Figura 5.26 Tabelas que armazenam dados de alunos e funcionários do SGBi.

Etapa 4.3: Especificar Componentes da Solução

O desenvolvedor, com base no seu conhecimento das tecnologias adotadas e na sua experiência, identificou as classes necessárias para o componente de código que implementa o processo de negócio “Cadastrar usuário” sob a forma de um sistema *web*. Dois *servlets* (classes `ExibeDadosCadastroServlet` e `CadastroUsuarioServlet`) são necessários, uma classe representando a entidade de dados `Usuario` e uma classe auxiliar `UsuarioDAO` para comunicação com o banco de dados da aplicação. Há necessidade também de arquivos JSP para exibir o formulário inicial, os dados cadastrais do usuário, informações de processamento e eventuais erros; e quatro *web services* serão acionados pelo código para fornecer funções necessárias para execução do processo de negócio. Todos esses elementos e seus relacionamentos estão representados no diagrama de classes mostrado na Figura 5.27, elaborado utilizando a ferramenta RSA, a linguagem UML, o perfil SoaML e estereótipos definidos pelo desenvolvedor para identificar os arquivos JSP (“<<JSP>>”) e *servlets* (“<<Servlet>>”).

Em seguida, o desenvolvedor definiu os atributos e métodos das classes, identificadas anteriormente, a partir da análise das atividades do sistema (representadas no modelo *to-be* do processo), representando-os em um diagrama de classes que auxilia nos possíveis refinamentos.

O desenvolvedor identificou que a classe `Usuario` devia fazer parte de um agrupamento lógico de classes conhecido como camada de mapeamento objeto-relacional que consiste de classes que estabelecem o relacionamento entre objetos, que representam entidades de dados, com suas respectivas tabelas do banco de dados relacional. Dessa forma, objetos da classe `Usuario` podem ser usados para representar uma instância da entidade de dados “Usuário” sendo os dados dessa instância armazenados em uma linha da tabela `USUARIO`.

Ainda analisando a entidade de dados “Usuário”, o desenvolvedor identificou que devido à existência de três diferentes tipos de usuários (aluno, docente e técnico-administrativo), no contexto do processo de negócio sendo implementado, é preciso que a classe `Usuario` seja abstrata e possua classes especializadas para representar cada tipo de usuário (aluno, docente e técnico-administrativo). Dessa forma, foram criadas três novas classes: `UsuarioAluno`, `UsuarioDocente`, `UsuarioTecnicoAdministrativo`, cada uma com métodos encapsulando lógicas de negócio apropriadas conforme o tipo do usuário. Na classe `Usuario` foram mantidos

métodos e atributos comuns a todos os tipos de usuários, favorecendo o reuso de código.

As decisões de projeto tomadas pelo desenvolvedor levaram a classe Usuário a assumir dois papéis distintos: o de classe de mapeamento objeto-relacional e o de classe de negócios fornecendo o comportamento requerido pelas regras de negócio. O desenvolvedor optou por manter a classe Usuário da forma como foi definida para seguir os conceitos da orientação a objetos permitindo que uma classe possua características e comportamentos.

Analisando a classe UsuárioDAO, foi identificado que alguns métodos dessa poderiam ser reusados por outras classes e assim deveriam fazer parte de uma classe mais genérica, a qual foi denominada BaseDAO, possuindo um relacionamento do tipo generalização com a classe UsuarioDAO. Outros métodos da classe UsuarioDAO foram movidos para uma nova classe, SimpleDataSource, pois a função dos mesmos trata especificamente de configurações para se ter acesso ao banco de dados. Todos esses refinamentos são mostrados na Figura 5.27.

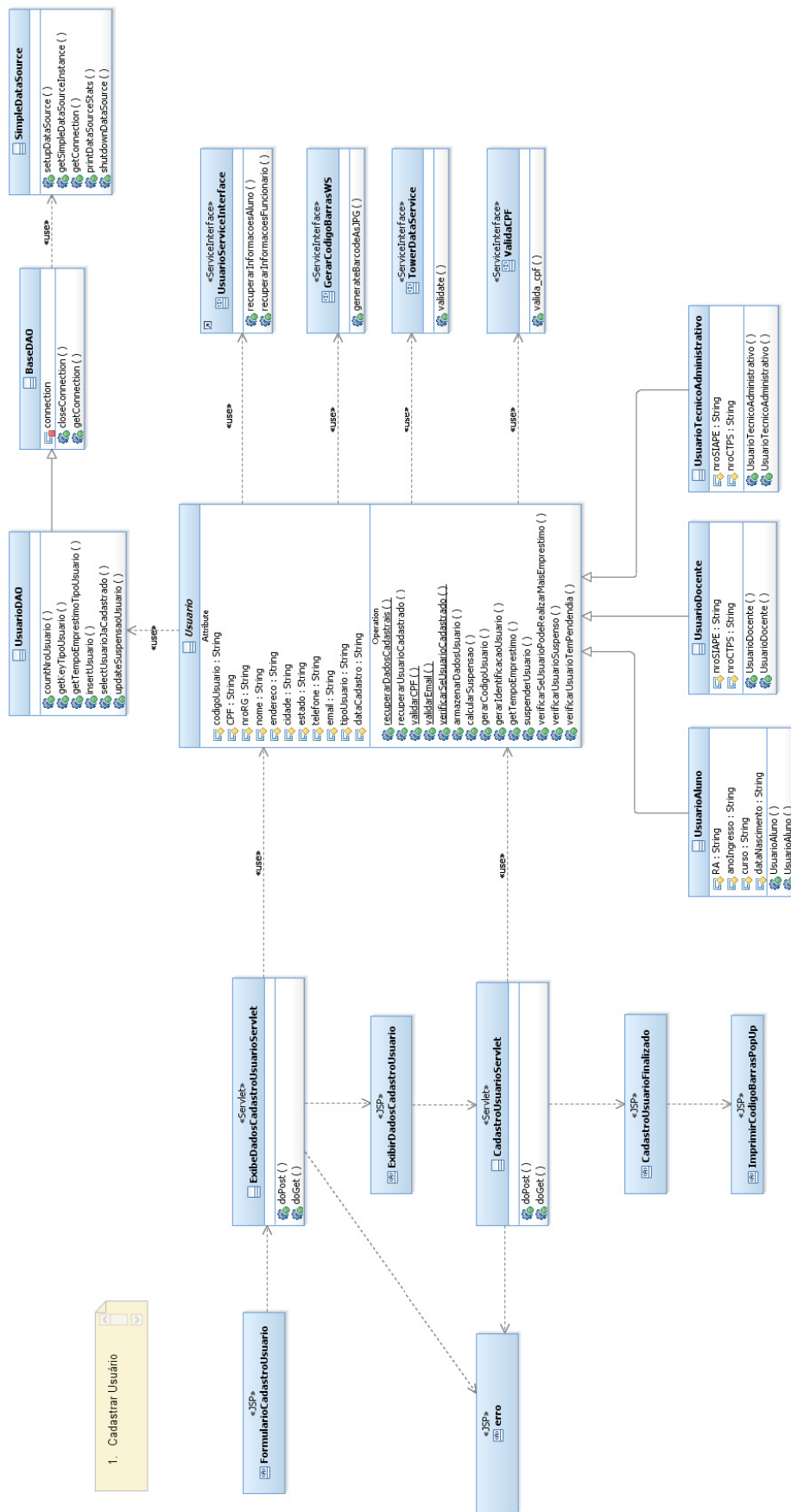


Figura 5.27 Diagrama de classes para implementação do processo “Cadastrar usuário” do SGBi.

A implementação e testes de serviços e da solução, assim como a verificação de aceitação não serão descritas aqui pois nesta dissertação foi dado ênfase às fases iniciais da IASWS. No entanto, o desenvolvedor finalizou o desenvolvimento dos serviços e da solução, realizou todos os testes especificados anteriormente e apresentou o sistema implementado ao cliente para verificação de aceitação.

5.3 Exemplo 2: Sistema de Atendimento da Unidade Saúde Escola

Uma Unidade Saúde Escola (USE), fictícia, é um espaço destinado ao desenvolvimento das atividades de ensino, pesquisa e extensão, na área de saúde podendo ser classificada como um ambulatório de média complexidade. Para o bom gerenciamento da USE, há a necessidade de desenvolver um sistema computacional que facilite o atendimento de usuários.

O processo de atendimento na USE inicia-se com o cadastro do usuário, caso ele ainda não seja cadastrado. Nesse cadastro são registrados os dados do usuário assim como as queixas que o levaram a procurar atendimento na unidade. Após o cadastro, o usuário aguarda em uma fila de espera pela triagem, que consiste em uma entrevista inicial feita por um profissional da unidade para analisar as queixas apresentadas e prover o encaminhamento para o especialista adequado. Após a triagem, o prontuário do usuário é criado e ele passa a ser um paciente. Sessões de atendimento, previamente agendadas, são realizadas para tratar o problema apresentado pelo paciente. Periodicamente são realizadas avaliações com o paciente a fim de avaliar os resultados do tratamento e para a realização de novos procedimentos. Sempre que novos sintomas forem diagnosticados ou uma nova queixa for apresentada, eles são registrados e o ciclo de atendimento se repete. Por haver grande procura pelos serviços oferecidos pela USE, o cadastro, a triagem e as consultas não são realizados no mesmo dia.

Considerando a aplicação da abordagem IASWS para desenvolver um sistema de *software* adequado para esse cenário, inicialmente foram implementados dois processos de negócio da USE, sendo descrita neste capítulo apenas a implementação de um desses processos, durante a primeira iteração. Informações

sobre as demais iterações foram documentadas em outro documento de trabalho (NAKAGAWA, PENTEADO e SANTOS, 2011b).

5.3.1 Primeira iteração

A execução da primeira iteração demandou mais tempo do que as demais iterações pois foi preciso coletar informações sobre todos os processos a serem implementados, modelá-los e documentá-los, como mostrado nas subseções seguintes.

5.3.1.1 Fase 1: Identificar Requisitos

Etapa 1.1: Levantar PN

Reuniões com o cliente foram realizadas para que o desenvolvedor pudesse compreender o negócio do cliente, identificar os objetivos de negócio, seus processos de negócio e suas necessidades. Novamente, o papel do cliente foi assumido pelos orientadores que forneceram as explicações detalhadas sobre o funcionamento da unidade saúde escola e o autor desta dissertação assumiu o papel da equipe de desenvolvimento, coletando informações, apresentadas na Tabela 5.15.

Tabela 5.15 Informações coletadas com o cliente sobre a Unidade Saúde Escola.

Área de negócio	Atendimento médico
Objetivos de negócio	<ul style="list-style-type: none"> ▪ Fornecer, a pacientes, atendimento de qualidade por médico ou outro profissional de saúde; ▪ Agilizar o atendimento de pacientes ▪ Diminuir o número de vezes que o usuário vai à USE até receber o primeiro atendimento.
Processos de negócio	<ul style="list-style-type: none"> ▪ Atendimento de pacientes
Dificuldades/necessidades	<ul style="list-style-type: none"> ▪ A USE requer um sistema de prontuário eletrônico que possa ser utilizado em todo o sistema unificado de saúde. Assim, esse exemplo de aplicação da IASWS refere-se somente ao cadastro de usuários, requisito inicial para o sistema de <i>software</i> desejado.

Com o auxílio do cliente, as atividades realizadas durante a execução do processo de negócio “Atendimento de pacientes” foram elencadas e organizadas em sequências lógicas, mostradas na Tabela 5.16.

Tabela 5.16 Atividades da operação de negócio “Atendimento de pacientes” da USE.

Processo de negócio	Atividades
Atendimento de pacientes	<ol style="list-style-type: none"> 1. Consultar cadastro de usuários <ol style="list-style-type: none"> A. Obter parâmetros de busca; B. Realizar busca por registro do usuário; C. Exibir resultados da busca; D. Analisar registros. 2. Cadastrar usuário <ol style="list-style-type: none"> A. Obter informações cadastrais; B. Obter a queixa do usuário; C. Realizar verificações; D. Registrar usuário; E. Notificar usuário. 3. Adicionar nova queixa ao prontuário 4. Executar triagem <ol style="list-style-type: none"> A. Recuperar informações do usuário; B. Realizar entrevista com usuário; C. Encaminhar usuário para atendimento. 5. Criar prontuário do paciente 6. Atender paciente <ol style="list-style-type: none"> A. Receber as queixas do paciente; B. Realizar avaliação do paciente; C. Diagnosticar; D. Recomendar tratamento. 7. Avaliar paciente <ol style="list-style-type: none"> A. Analisar o progresso do paciente em relação às queixas apresentadas antes do tratamento; B. Verificar evolução do quadro clínico; C. Avaliar necessidade de continuar com tratamento ou necessidade de mudança de tratamento ou “dar alta”.

Durante o trabalho para elencar as atividades do processo de negócio, identificou-se que algumas atividades (“Consultar cadastro de usuários”, “Cadastrar usuário”, “Executar triagem”, “Atender paciente” e “Avaliar paciente”) são compostas por subatividades, podendo, portanto, serem consideradas subprocessos. A partir dessa constatação foi decidido que a implementação de todos os componentes do processo “Atendimento de pacientes” de uma única vez seria inviável, pois implicaria em grande esforço para treinamento dos funcionários e adequação de suas rotinas de trabalho, impactando no atendimento ao público. Assim, nesse momento apenas o subprocesso “Consultar cadastro de usuários” e o subprocesso “Cadastrar usuário” seriam automatizados como parte do sistema.

Por fim, as informações sobre os subprocessos de negócio “Cadastrar usuário” e “Consultar cadastro de usuários” foram documentadas textualmente, como mostrado na Tabela 5.17.

Tabela 5.17 Informações dos processos de negócio que serão informatizados pelo Sistema de Atendimento da Unidade Saúde Escola.

Processo de negócio	Informações
Cadastrar usuário	<p>Para que um usuário possa ser atendido na USE, seus dados cadastrais devem estar registrados no sistema. Caso seja a primeira vez que o usuário procura atendimento, deve-se realizar o seu cadastro com todos os seus dados pessoais, informações de contato assim como as queixas apresentadas por ele.</p> <p>Dados como telefone celular e email devem ser validados uma vez que são a principal forma de contato da USE com o usuário para agendamentos de entrevistas ou atendimentos.</p> <p>Se todos os dados estiverem corretos, gera-se o código de usuário e as informações são armazenadas no sistema.</p>
Consultar cadastro de usuários	<p>Ao consultar o cadastro de usuários é possível saber se um determinado usuário já está cadastrado no sistema, realizando a busca por esse usuário na base de dados do sistema. Como a USE atende desde crianças até idosos, não há nenhum dado cadastral do usuário que permita identificá-lo univocamente. Assim essa busca é realizada a partir de uma combinação de informações (parâmetros e operadores). Em virtude disso, é possível que o resultado da busca não seja o registro do usuário em questão, sendo assim necessário que o atendente analise manualmente cada um dos registros retornados pela busca.</p>

Etapa 1.2: Modelar PN

O Diagrama de Visão Geral de Processos não foi criado, pois apenas o processo “Atendimento de pacientes” da USE é abordado. A modelagem desse processo com suas atividades (“Adicionar nova queixa ao prontuário” e “Criar prontuário do paciente”) e subprocessos (“Consultar cadastro de usuários”, “Cadastrar usuário”, “Executar triagem”, “Atender paciente” e “Avaliar paciente”) é apresentada na Figura 5.28.

Cada um dos subprocessos que serão implementados pelo sistema foi modelado em um diagrama de processo do BPMN e as suas respectivas atividades, listadas na Tabela 5.16, foram traduzidas em atividades no modelo *as-is*. Para exemplificar tem-se o modelo *as-is* dos subprocessos “Cadastrar usuário” e “Consultar cadastro de usuários” exibidos, respectivamente, na Figura 5.29 e Figura 5.30.

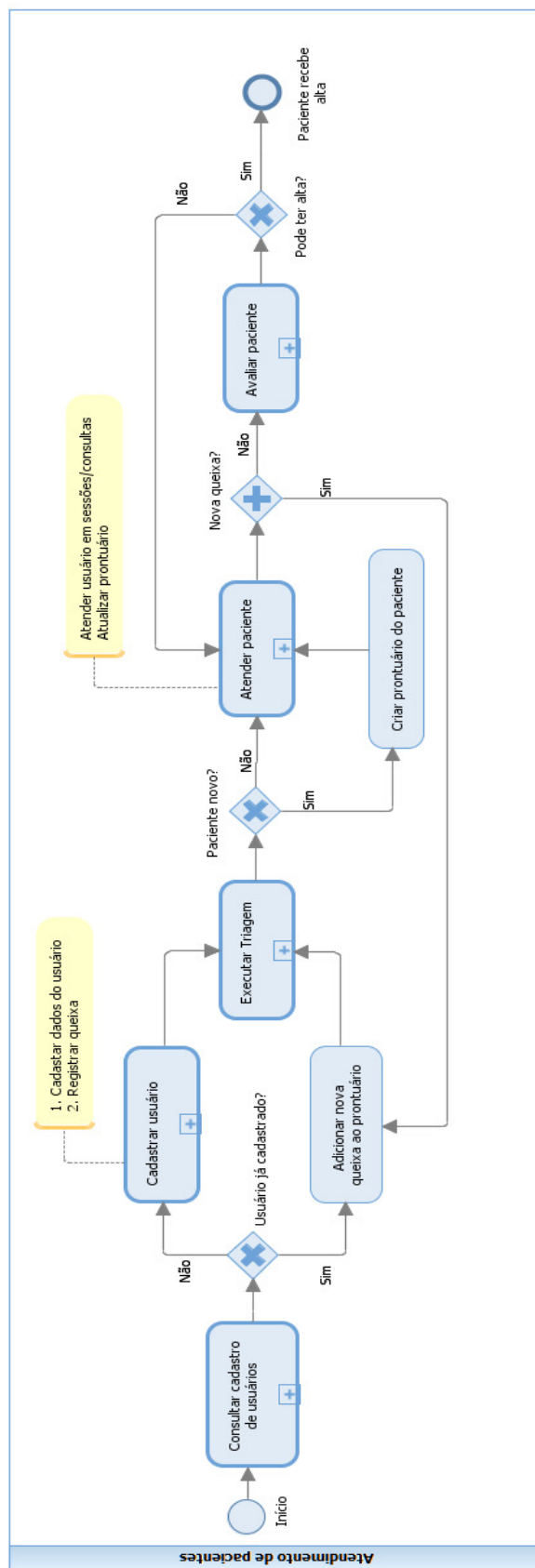


Figura 5.28 Modelagem em BPMN do processo “Atendimento de pacientes” do Sistema de Atendimento da Unidade Saúde Escola.

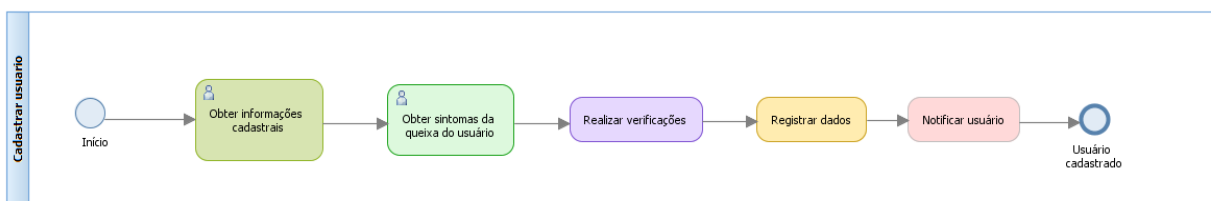


Figura 5.29 Modelo *as-is* do subprocesso “Cadastrar usuário” do Sistema de Atendimento da Unidade Saúde Escola.

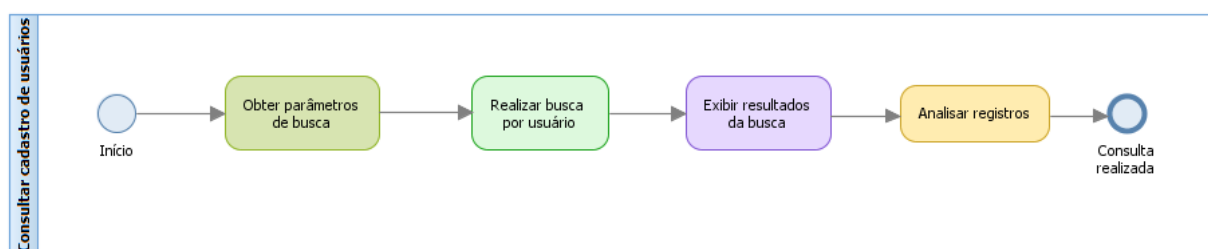


Figura 5.30 Modelo *as-is* do subprocesso “Consultar cadastro de usuários” do Sistema de Atendimento da Unidade Saúde Escola.

Etapa 1.3: Extrair Requisitos de TI

Após análise do subprocesso “Cadastrar Usuário”, o desenvolvedor identificou que todas as cinco atividades (“Obter informações cadastrais”, “Obter sintomas da queixa do usuário”, “Realizar verificações”, “Registrar dados”, “Notificar usuário”) representadas no modelo *as-is* podem ser informatizadas. Essa decisão foi tomada com base no conhecimento e na experiência do desenvolvedor. Por exemplo, as atividades “Obter informações cadastrais” e “Obter sintomas da queixa do usuário” não podem ser totalmente informatizadas, pois envolvem interação com o atendente para coletar as informações do usuário e transferi-las para o sistema.

O modelo *to-be* do subprocesso “Cadastrar Usuário” é mostrado Figura 5.31, com dois atores representados: atendente e sistema, bem como suas respectivas atividades. As atividades “Obter informações cadastrais” e “Obter sintomas da queixa do usuário” foram divididas em atividades executadas pelo atendente e pelo sistema para que cada atividade pudesse ser informatizada adequadamente. As atividades “Realizar verificações”, “Registrar dados” e “Notificar usuário” serão totalmente informatizadas pelo sistema e por isso estão vinculadas ao ator sistema. Duas atividades (“Gerar cartão do usuário” e “Plastificar e entregar cartão do usuário”) foram adicionadas após uma melhoria proposta pelo desenvolvedor e

aceita pelo cliente: cada paciente da USE possui seu cartão de usuário para facilitar e agilizar o atendimento, à medida que o novo sistema for implantado.

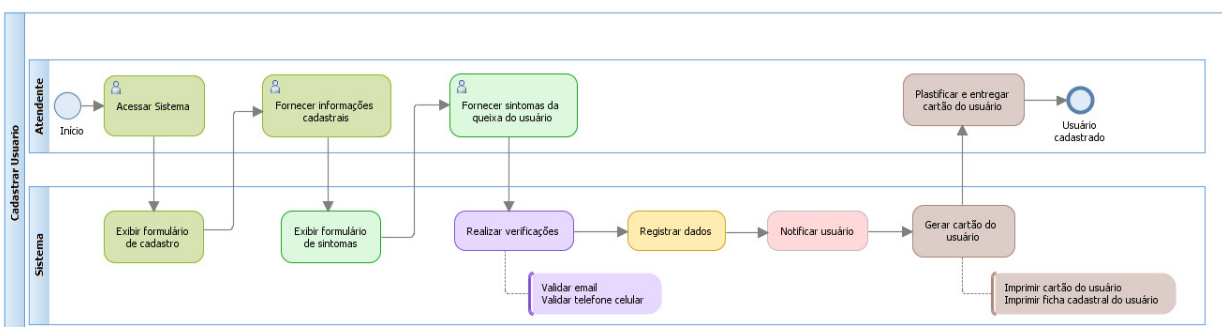


Figura 5.31 Modelo *to-be* do subprocesso “Cadastrar Usuário” do Sistema de Atendimento da Unidade Saúde Escola.

Similarmente, foram identificadas que três atividades do subprocesso “Consultar cadastro de usuários” podem ser informatizadas. A atividade “Obter parâmetros de busca” é parcialmente informatizada, por ser responsabilidade do atendente coletar os dados e fornecê-los ao sistema. A atividade “Analisar registros” não será informatizada, uma vez que ela não pode ser realizada sem interferência humana. O modelo *to-be* desse subprocesso é mostrado na Figura 5.32. Os requisitos do sistema para cada processo de negócio foram documentados textualmente em um documento de requisitos (NAKAGAWA, PENTEADO e SANTOS, 2011a).

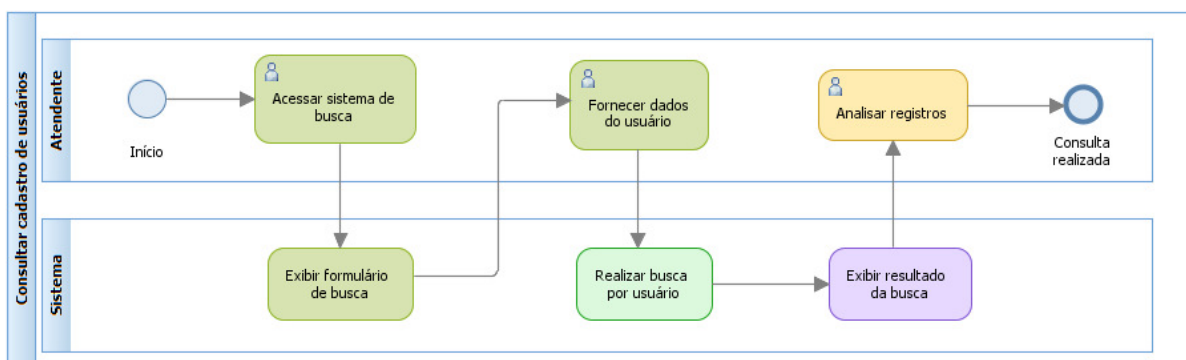


Figura 5.32 Modelo *to-be* do subprocesso “Consultar cadastro de usuários” do Sistema de Atendimento da Unidade Saúde Escola.

Etapa 1.4: Planejar

Neste exemplo de aplicação da IASWS, o cliente não indicou a ordem de prioridade para a implementação dos subprocessos. Assim, a classificação dos subprocessos, mostrada na Tabela 5.18, foi realizada de modo que minimizasse os

esforços para geração de dados de teste. Com base na experiência do desenvolvedor e nas informações obtidas estimou-se o número de horas necessárias para implementar cada processo de negócio.

Tabela 5.18 Backlog de Subprocessos de Negócio e respectiva estimativa de tempo do Sistema de Atendimento da Unidade Saúde Escola.

Prioridade	Subprocesso de negócio	Estimativa de horas	Comentários
1	Cadastrar usuário	120	
2	Consultar cadastro de usuários	80	

5.3.1.2 Fase 2: Contextualizar PN com Serviços

Etapa 2.1: Selecionar PN

O primeiro processo de negócio a ser implementado é o “Cadastrar Usuário”, segundo o *Backlog* de Processos.

Etapa 2.2: Identificar e Buscar Serviços

Após analisar a funcionalidade do sistema, representada no modelo *to-be* do processo “Cadastrar usuário”, os serviços candidatos foram identificados com base nas diretrizes propostas na IASWS. O mesmo procedimento adotado no exemplo de aplicação da IASWS anterior foi aplicado aqui. Ou seja, buscou-se por serviços que pudessem fornecer a funcionalidade do processo de negócio “Cadastrar Usuário”, porém nenhum serviço foi encontrado. O próximo passo seria buscar serviços que fornecessem uma funcionalidade mais genérica do que a do processo de negócio, porém, assim como ocorreu no exemplo anterior, não foi encontrado nenhum serviço que fornecesse esse tipo de funcionalidade (por exemplo “Cadastrar dados” ou “Cadastrar entidade”). Passou-se então para a análise das atividades desse processo de negócio que poderiam ter suas funcionalidades fornecidas por um serviço. Na atividade “Registrar dados”, os dados do usuário são armazenados em um banco de dados e o desenvolvedor tinha conhecimento de um serviço da *Amazon Relational Database Service* para persistir dados em banco de dados relacional. Dessa forma, esse serviço foi selecionado para avaliação de viabilidade. Na atividade “Notificar usuário”, identificada como serviço candidato, pode-se utilizar emails ou SMS para notificar usuários, pois existem serviços disponíveis na internet para ambos.

As atividades “Exibir formulário de cadastro”, “Realizar verificações” e “Registrar dados” também foram analisadas e foi identificado que na atividade “Exibir formulário de cadastro” pode-se ter o preenchimento automático dos campos endereço, cidade e estado a partir do CEP fornecido. Por se tratar de uma função muito utilizada em sistemas de *software*, um serviço disponível na internet foi encontrado. Outro serviço candidato foi identificado para realizar a validação de um endereço de email, podendo ser usado na implementação da atividade “Realizar verificações”.

Não foi possível realizar nenhum agrupamento de atividades para este processo de negócio. Nenhuma função relacionada a regras de negócios, entidades de dados ou requisitos não-funcionais pôde considerada como serviço candidato.

Na Tabela 5.19 estão mostradas as funcionalidades que foram identificadas como serviço candidato e o resultado da busca por um serviço que as fornecesse. A URL para página de descrição do serviço foi documentada nos casos em que um serviço foi encontrado.

Etapa 2.3: Elaborar Proposta de Solução

Para cada serviço candidato identificado na etapa anterior avaliou-se a viabilidade e a relação custo-benefício quanto a desenvolver um novo serviço que fornecesse a funcionalidade; quanto a implementar a funcionalidade usando classes OO; e no caso em que serviços foram encontrados, quanto ao reúso do serviço.

Analisando essas informações foi possível decidir quais serviços reusar, quais desenvolver e quais funcionalidades implementar usando classes, dando origem à proposta de solução. Na solução proposta, nenhum novo serviço será criado nesta iteração, pois nenhum dos serviços candidatos possui potencial de reúso por outros sistemas que justifique sua implementação como um web service. Os serviços encontrados para a atividade “Notificar usuário” serão reutilizados para que o usuário possa escolher como deseja ser notificado: via SMS ou email. Esses serviços foram documentados na Tabela 5.20 e na Tabela 5.21 respectivamente.

Tabela 5.19 Serviços candidatos identificados para o Sistema de Atendimento da Unidade Saúde Escola.

Funcionalidade	Serviço encontrado?	Informações sobre serviço (caso encontrado)
Cadastrar usuário	Não	N/A
Cadastrar dados	Não	N/A
Cadastrar entidade	Não	N/A
Registrar dados	Sim	Amazon <i>Relational Database Service</i> http://aws.amazon.com/rds/ (acesso em 16 Fev. 2014) Custo depende da quantidade de <i>megabytes</i> armazenado
Notificar usuário	Sim	Para notificar via SMS: Torpedo SMS <i>Service</i> http://www.byjg.com.br/site/xmlnuke.php?xml=smswebsevice (acesso em 16 Fev. 2014) Custo: R\$0,18 por SMS enviado Para notificar via email: Amazon <i>Simple Email Service</i> http://aws.amazon.com/sdkforjava (acesso em 16 Fev. 2014) Custo: \$0.10 a cada 1000 emails. \$0.10 para cada GB de dados recebidos \$0.15 para cada GB de dados enviados (até 10 GB) Pagamento com cartão de crédito.
Cadastrar usuário	Exibir formulário de cadastro	CEP <i>Service</i> http://www.byjg.com.br/site/xmlnuke.php?xml=onlinecep (acesso em 16 Fev. 2014) Custo: gratuito
	Realizar verificações	Tower Data <i>Email Validation Service</i> http://soap.towerdata.com/validate.wsdl (acesso em 16 Fev. 2014) Custo: gratuito

Tabela 5.20 Informações do serviço “Torpedo SMS Service” para o Sistema de Atendimento da Unidade Saúde Escola.

Serviço:	Torpedo SMS <i>Service</i>
Descrição do serviço:	Envia mensagens de texto via SMS para usuários das principais operadoras em atuação no território brasileiro.
Funcionalidade atendida:	Função desempenhada pela tarefa “Notificar usuário”
Descrição da funcionalidade atendida:	Notifica o usuário sobre algum acontecimento no processo de atendimento.
Custos do uso do serviço	R\$0,18 por SMS enviado.
URL para o WSDL do serviço	http://www.byjg.com.br/site/xmlnuke.php?xml=smswebservice (acesso em 16 Fev. 2014)

Tabela 5.21 Informações do serviço de envio de emails “Amazon Simple Email Service” para o Sistema de Atendimento da Unidade Saúde Escola.

Serviço:	<i>Amazon Simple Email Service</i>
Descrição do serviço:	Permite enviar mensagens via email
Funcionalidade atendida:	Função desempenhada pela tarefa “Notificar usuário”
Descrição da funcionalidade atendida:	Notifica o usuário sobre algum acontecimento no processo de atendimento.
Custos do uso do serviço	\$0.10 a cada 1000 emails. \$0.10 para cada GB de dados recebidos \$0.15 para cada GB de dados enviados (até 10 GB) Pagamento com cartão de crédito.
URL para o WSDL do serviço	Não é disponibilizado o arquivo WSDL. No lugar deste arquivo, é disponibilizado o Amazon AWS SDK que fornece as classes cliente necessárias para invocar o serviço. http://aws.amazon.com/sdkforjava (acesso em 16 Fev. 2014)

O serviço “CEP Service” será utilizado na atividade “Exibir formulário de cadastro”, para auxiliar o preenchimento dos campos endereço, cidade e estado com base no CEP fornecido. Informações sobre esse serviço estão documentadas na Tabela 5.22.

Tabela 5.22 Informações do serviço “CEP Service” para o Sistema de Atendimento da Unidade Saúde Escola.

Serviço:	<i>CEP Service</i>
Descrição do serviço:	Recupera o logradouro com base no CEP fornecido.
Funcionalidade atendida:	Função desempenhada pela tarefa “Exibir formulário de cadastro”
Descrição da funcionalidade atendida:	Exibe o formulário com os campos para o cadastro de um novo usuário
Custos do uso do serviço	Gratuito. Pelas informações no site do provedor, este serviço aparente ser bem estável. Segundo o provedor, cerca de 1.800.000 consultas são realizadas diariamente.
URL para o WSDL do serviço	WSDL em http://www.byjg.com.br/site/xmlnuke.php?xml=onlinecep (acesso em 16 Fev. 2014)

O serviço encontrado para validar endereços de email também será usado na solução proposta e foi documentado como mostrado na Tabela 5.23.

Tabela 5.23: Informações do serviço de validação de endereços de email “Tower Data Email Validation Service” para o Sistema de Atendimento da Unidade Saúde Escola.

Serviço:	<i>Tower Data Email Validation Service</i>
Descrição do serviço:	Permite validar um endereço de email. Pode-se validar a sintaxe de um endereço de email, validar de domínio, confirmar se o domínio pode receber emails, confirmar se o endereço de email pode receber mensagens.
Funcionalidade atendida:	Função desempenhada pela tarefa “Validar email”
Descrição da funcionalidade atendida:	Fazer a validação do email armazenado nas informações cadastrais do usuário.
Custos do uso do serviço	Gratuito para testes.
URL para o WSDL do serviço	WSDL em http://soap.towerdata.com/validate.wsdl (acesso em 16 Fev. 2014)

Por fim, o serviço “*Amazon Relational Database Service*” que havia sido listado como opção para fornecer a infraestrutura necessária para suportar um banco de dados via *Web Services* não será utilizado na solução por não se tratar de um *web service*.

O desenvolvedor optou por implementar o processo de negócio “Cadastrar usuário” como um sistema *web* para permitir que qualquer computador conectado à rede da USE pudesse acessar o sistema, reduzindo a complexidade da infraestrutura computacional. Assim as funcionalidades do sistema que não são fornecidas por serviços serão implementadas por meio de classes OO, páginas HTML/JSP e *Servlets*, sendo a solução *web* proposta composta dos seguintes elementos:

- Páginas *web* para: informar os dados do usuário que se deseja cadastrar; exibir eventuais erros no processamento; exibir o resultado do cadastro do usuário, permitindo a impressão dos dados do usuário;
- *Servlets* atuando como controladores de fluxo, invocando as funcionalidades fornecidas por serviços e classes OO;
- Serviços;
- Classes OO que implementam as demais tarefas do processo de negócio “Cadastrar usuário”.

Etapa 2.4: Avaliar Serviços Reusados

A solução proposta reusará quatro serviços já implementados, mas que devem ser testados nesta etapa. Dois desses serviços, “*Torpedo SMS Service*” e “*CEP Service*”, são *Restful Web Services* e possuem exemplos demonstrando como invocar o serviço. Os serviços “*Tower Data Email Validation Service*” e “*Amazon Simple Email Service*” são do tipo *XML Web Services*. O primeiro possui o arquivo de descrição do serviço (WSDL), que foi importado na ferramenta RSA (IBM, 2010) para gerar automaticamente as classes cliente para invocar o serviço. O segundo fornece um pacote contendo classes que devem ser utilizadas para invocar o *web service*.

Os casos de testes, para cada serviço, foram elaborados analisando-se o comportamento esperado da funcionalidade do serviço que será utilizada na solução.

Os casos de testes dos serviços “*Torpedo SMS Service*” e “*CEP Service*” são os mostrados na Tabela 5.24 e Tabela 5.25, respectivamente. Ambos os serviços fornecem uma interface HTTP (REST) e código-fonte em Java exemplificando a utilização do serviço, que puderam ser aproveitados na construção do protótipo para invocar esses serviços. Ao executar os casos de testes do serviço “*Torpedo SMS Service*”, detectou-se um problema ao enviar mensagens SMS para números de celulares da operadora Oi. Para as operadoras Vivo, Claro e TIM, as mensagens foram enviadas com sucesso. Como o objetivo deste exemplo de aplicação da IASWS é avaliar a abordagem proposta, decidiu-se que o problema não deveria impedir o reuso do serviço. A execução dos casos de testes do serviço “*CEP Service*” não indicou nenhum problema, sendo possível reusá-lo na solução.

Tabela 5.24 Casos de teste para o serviço “*Torpedo SMS Service*” para o Sistema de Atendimento da Unidade Saúde Escola.

# caso de teste	Descrição	Valores de entrada	Saída esperada
1	Enviar SMS para um número válido	(19) 9294-6355	SMS enviado ao destinatário; Mensagem de sucesso.
2	Enviar SMS para um número inexistente	(16) 9911-2233	Mensagem informando o erro.
3	Enviar SMS para número das principais operadoras (TIM, Oi, Vivo, Claro)	Oi: (19) 9294-6355 Vivo: (19) 9768-4537 Claro: (19) 9236-4670 Tim: (19) 8157-4064	SMS enviado ao destinatário; Mensagem de sucesso.
4	Enviar SMS para número de telefone fixo válido	(19) 4062-9398	Mensagem informando o erro.

Os casos de testes para o serviço “*Amazon Simple Email Service*” são mostrados na Tabela 5.26. Esse serviço não possui o arquivo WSDL e é acessado por meio de um pacote de classes (SDK), distribuído para seus consumidores. Exemplos de utilização das classes do SDK fazem parte desse pacote, auxiliando o desenvolvedor na criação do código para testar o serviço. A execução dos casos de teste não evidenciou nenhum problema, qualificando o serviço a ser reusado na solução.

Os casos de testes dos serviços “*Tower Data Email Validation Service*” são mostrados na Tabela 5.27. O código para invocar esse *web service* foi reusado do exemplo de aplicação da IASWS anterior. Os casos de testes foram executados e esse serviço foi considerado como reusável.

Tabela 5.25 Casos de teste para o serviço “CEP Service” para o Sistema de Atendimento da Unidade Saúde Escola.

# caso de teste	Descrição	Valores de entrada	Saída esperada
1	Recuperar logradouro para CEP válido	13560-300	Logradouro correto: Rua São Joaquim Vila Monteiro, São Carlos - SP
2	Recuperar logradouro para CEP válido	13560300	Logradouro correto: Rua São Joaquim Vila Monteiro, São Carlos - SP
2	Recuperar logradouro para CEP inválido	12345-678	Mensagem informando o erro.
3	Recuperar logradouro para CEP válido, mas incompleto (sem os três últimos dígitos)	13560	Mensagem informando o erro.

Tabela 5.26 Casos de teste para o serviço “Amazon Simple Email Service” para o Sistema de Atendimento da Unidade Saúde Escola.

# caso de teste	Descrição	Valores de entrada	Saída esperada
1	Enviar email para um endereço de email válido	Endereço de email válido	Email enviado; Mensagem de sucesso informando que email foi enviado.
2	Enviar email para um endereço de email inválido	Endereço de email inválido	Mensagem informando que email não foi enviado porque o email informado é inválido.

Tabela 5.27 Casos de teste para o serviço “Tower Data Email Validation Service” para o Sistema de Atendimento da Unidade Saúde Escola.

# caso de teste	Descrição	Valores de entrada	Saída esperada
1	Email válido	hiromiti@gmail.com	Válido
2	Email válido	hiron@br.ibm.com	Válido
3	Email válido	santos@dc.ufscar.br	Válido
4	Email inválido (domínio inválido)	hiromiti@gmail.com.br	Inválido
5	Email inválido (mailbox inválido)	hiromiti123456@gmail.com	Inválido
6	Email inválido (domínio inválido)	hiromiti@ufscar.com.br	Inválido

Etapa 2.5: Criar Casos de Testes de Novos Serviços

Não há necessidade de desenvolver um novo serviço nesta iteração e por esse motivo esta etapa não será executada.

Etapa 2.6: Modelar Novos Serviços

Não há necessidade de desenvolver um novo serviço nesta iteração e por esse motivo esta etapa não será executada.

Etapa 2.7: Criar Casos de Teste da Solução

O desenvolvedor analisou as informações do processo de negócio e as atividades representadas no modelo *to-be* para identificar os diferentes cenários

para execução desse processo. Desses cenários, foram selecionados aqueles que permitissem formar um conjunto mínimo capaz de validar cada atividade do processo de negócio ao menos uma vez. A Tabela 5.28 lista os casos de teste elaborados.

Tabela 5.28 Casos de teste para o PN “Cadastro de Usuário” do Sistema de Atendimento da Unidade Saúde Escola.

Processo de negócio: Cadastrar Usuário			
# caso de teste	Descrição	Valores de entrada	Saída esperada
1	Cadastrar usuário sem fornecer informações sobre sua queixa	Dados do usuário Não fornecer dados da queixa	Usuário não cadastrado; Mensagem de erro informando que informações sobre a queixa são obrigatórias.
2	Cadastrar usuário que não possui um email	Dados do usuário	Usuário cadastrado com o campo email em branco; Mensagem informando que usuário foi cadastrado.
3	Cadastrar usuário com email inválido	Dados do usuário	Usuário não cadastrado; Mensagem informando que o email do usuário é inválido.
4	Cadastrar usuário que não possui um número de telefone celular	Dados do usuário	Usuário cadastrado com o campo telefone celular em branco; Mensagem informando que usuário foi cadastrado.
5	Cadastrar usuário que não possui email nem um número de telefone celular	Dados do usuário	Usuário não cadastrado; Mensagem informando que email ou celular devem ser informados.
6	Cadastrar usuário fornecendo CEP com formato inválido	Dados do usuário	Usuário não cadastrado; Mensagem informando que o CEP não está no formato correto.
7	Cadastrar usuário fornecendo CEP inexistente	Dados do usuário	Usuário não cadastrado; Mensagem informando que o CEP não existe.
8	Cadastrar usuário e não selecionar a forma de notificação do usuário (SMS ou email)	Dados do usuário	Usuário não cadastrado; Mensagem informando que se deve selecionar a forma de notificação do usuário.
9	Cadastrar usuário com dados corretos	Dados do usuário	Usuário cadastrado; Mensagem informando que usuário foi cadastrado.

5.3.1.3 Fase 3: Projetar serviços

As etapas dessa fase não foram executadas, uma vez que não há um novo serviço a ser desenvolvido nesta iteração.

5.3.1.4 Fase 4: Projetar solução

Etapa 4.1: Definir Arquitetura da Solução

Analisando as características de um sistema *web*, a implementação da solução proposta requer: um servidor de aplicações JEE (*web*); um servidor de banco de dados para armazenar as informações dos usuários; os quatro *web services* identificados anteriormente; e uma conexão JDBC configurada no servidor de aplicações permitindo que o código acesse o banco de dados.

A arquitetura definida para implementar a solução é mostrada na Figura 5.33. Para representar essa arquitetura foram usados um diagrama de *deployment* (fornecido pela ferramenta RSA), elementos UML e estereótipos definidos pelo autor para representar os componentes da arquitetura como: servidor de aplicações (“*JEE Container*”), servidor de banco de dados (“*Database Server*”), o banco de dados da aplicação (“*Application Database*”), os *web services* (“*WebService*”) e o arquivo que contém o código-fonte Java (“*EAR file*”). A conexão JDBC foi representada por uma associação ligando o servidor de aplicações e o de banco de dados. Por fim, representou-se o *hardware* no qual ambos os servidores serão hospedados: Linux Server.

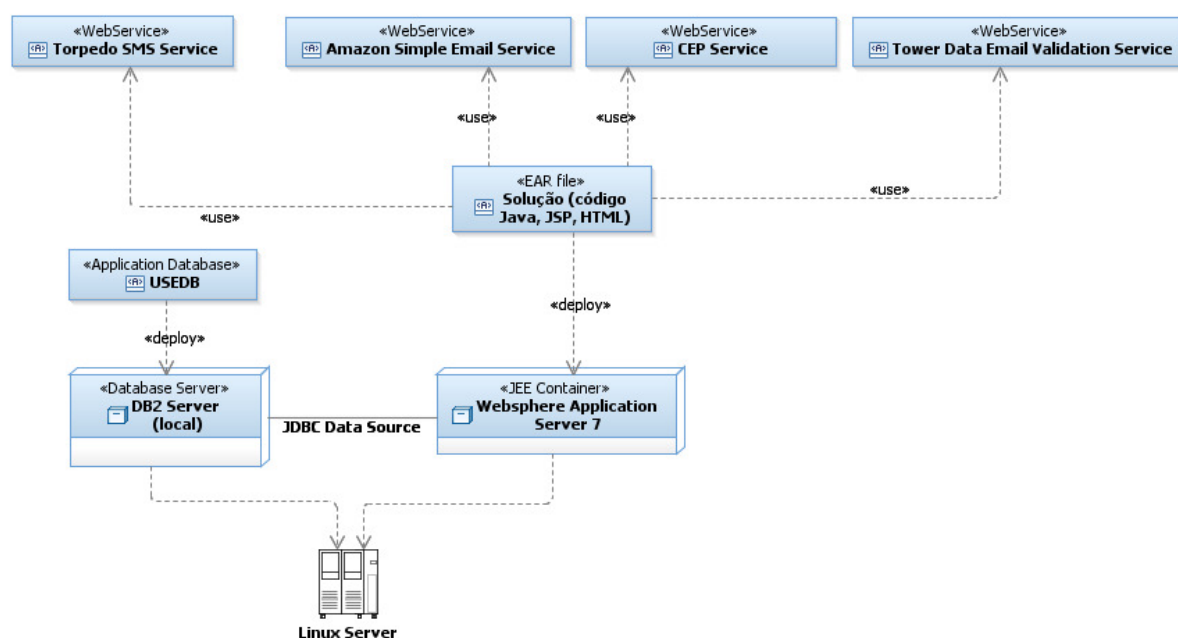


Figura 5.33 Arquitetura da solução para o PN “Cadastrar Usuário” do Sistema de Atendimento da Unidade Saúde Escola.

Etapa 4.2: Criar Modelo de Dados

O desenvolvedor analisou as informações do processo de negócio “Cadastrar usuário” e seu modelo *to-be* e identificou duas entidades de dados: “Usuário” e “Queixa”. Os campos dessas entidades foram elencados e dois relacionamentos envolvendo essas entidades foram identificados. O primeiro relacionamento identificado estabelece que uma “Queixa” é feita por um “Usuário”. Já o segundo é o relacionamento da entidade “Usuário” com a entidade “Paciente” (identificada após essa constatação) no contexto do processo de atendimento, que estabelece que todo paciente da USE é um usuário cadastrado no sistema.

Em seguida, o desenvolvedor realizou o mapeamento dessas entidades para o modelo relacional e criou quatro tabelas: USUARIO, PACIENTE, PROFISSAO e QUEIXA. A tabela USUARIO contém as informações cadastrais de um usuário, sendo a profissão do usuário armazenada na tabela PROFISSAO (para padronizar os nomes das profissões). A tabela QUEIXA armazena informações relacionadas a queixas de problemas relatados por um usuário. Por fim, a tabela PACIENTE armazena informações de um paciente. Na Figura 5.25 é mostrada a modelagem dessas tabelas, seus relacionamentos e respectivos atributos. Cada atributo tem seu respectivo tipo de dados, tamanho máximo e obrigatoriedade especificados. A modelagem apresentada foi criada na ferramenta RSA, utilizando um modelo físico de dados.

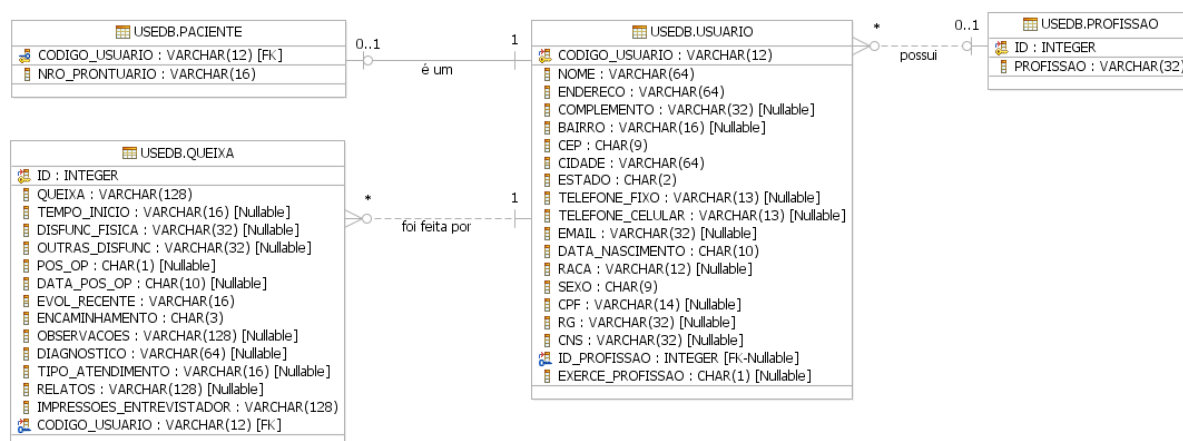


Figura 5.34 Modelo de dados do PN “Cadastrar Usuário” do Sistema de Atendimento da Unidade Saúde Escola.

Etapa 4.3: Especificar Componentes da Solução

O desenvolvedor utilizou seu conhecimento e experiência nas tecnologias adotadas para identificar as classes necessárias à implementação do sistema *web* que fornece a funcionalidade do processo de negócio “Cadastrar usuário”. Serão necessários dois *servlets* (classes `CadastroDadosUsuarioServlet` e `CadastroQueixaUsuarioServlet`), uma classe para representar a entidade Usuário, uma classe auxiliar `UsuarioDAO` para comunicação com o banco de dados da aplicação, uma classe representando a entidade Queixa e sua respectiva classe auxiliar `QueixaDAO`. Arquivos JSP serão necessários para exibir o formulário de cadastro do usuário e de sua queixa, para exibir informações de processamento e eventuais erros e para exibir e imprimir a ficha cadastral do usuário. Os *web services* identificados anteriormente serão acionados pelo código para fornecer suas funcionalidades. Todos esses elementos e seus relacionamentos foram representados no diagrama de classes mostrado na Figura 5.35, elaborado utilizando a ferramenta RSA, a linguagem UML, o perfil SoaML e estereótipos definidos pelo desenvolvedor para identificar os arquivos JSP (“<<JSP>>”) e *servlets* (“<<Servlet>>”).

Em seguida, o desenvolvedor analisou as atividades do sistema (representadas no modelo *to-be* do processo) para definir os atributos e métodos das classes, representados em um diagrama de classes que facilitará os possíveis refinamentos.

Foi identificado que alguns métodos da classe `UsuarioDAO` deveriam fazer parte de uma classe mais genérica, denominada `BaseDAO`, para permitir o reuso desses métodos por outras classes DAO. A classe `BaseDAO` foi relacionada à classe `UsuarioDAO` por meio de um relacionamento do tipo generalização. Métodos pertencentes à classe `UsuarioDAO` serão responsáveis por realizar a persistência dos dados contidos em objetos da classe `Usuario` (classe da camada de mapeamento objeto-relacional), incluindo a persistência de dados nas diversas tabelas utilizadas para representar a entidade de dados Usuário no modelo relacional. Outros métodos da classe `UsuarioDAO` foram movidos para uma nova classe, denominada `SimpleDataSource`, por causa da função dos métodos tratar especificamente de configurações para acessar o banco de dados. Todos esses refinamentos são mostrados na Figura 5.35.

5.4 Considerações Finais

A aplicação da abordagem IASWS para desenvolver o Sistema Gerenciador de Biblioteca teve papel fundamental no refinamento das etapas da abordagem, visto que, conforme suas etapas eram aplicadas para desenvolver o sistema, foi possível identificar deficiências (por exemplo: a necessidade de realizar testes e validar a operacionalidade dos serviços usados na implementação da solução em fases iniciais de sua especificação; a ordem das etapas de elaboração de casos de testes que não favorecia/agregava valor ao trabalho; a necessidade de haver um ciclo entre as etapas Levantar PN e Modelar PN para endereçar a natureza dinâmica desse trabalho; necessidade de alta iteratividade entre as etapas Identificar e Buscar Serviços, Elaborar Proposta de Solução e Avaliar Serviços Reusados para possibilitar a elaboração de uma solução adequada) da abordagem e corrigi-las. Já o segundo exemplo de aplicação da IASWS (desenvolvimento do Sistema de Atendimento da Unidade Saúde Escola), possibilitou validar a abordagem IASWS após os refinamentos e correções identificados durante a realização do primeiro exemplo de aplicação da IASWS. Ambos os sistemas desenvolvidos atenderam às expectativas do cliente..

O fluxo de trabalho proposto pela abordagem IASWS mostrou-se adequado para o desenvolvimento de sistemas de *software* que fazem uso (reusando ou desenvolvendo) de serviços. As fases dessa abordagem juntamente com suas respectivas etapas permitem: entender os requisitos; entender e formular a solução que combina elementos OO com serviços, identificando e encontrando esses serviços; desenvolver novos serviços; e desenvolver a solução. Com a realização dos estudos de caso foi possível constatar que a abordagem proposta permite que um sistema seja gerado a partir da análise e entendimento do processo de negócio implementado pelo sistema.

A estratégia de representar os requisitos do sistema sob a forma de um modelo de processos *to-be* mostrou-se válida, fornecendo uma representação gráfica dos requisitos e contribuindo para melhorar o entendimento do sistema a ser desenvolvido. Além disso, definir os requisitos do sistema a partir dos processos de negócio do cliente aproxima o mundo dos negócios (processos de negócio) do

mundo de TI (sistema), trazendo agilidade e flexibilidade para permitir ao sistema acompanhar as mudanças no processo de negócio.

O reúso (de serviços) em nível de análise auxilia a definir uma solução robusta e melhora a acuracidade das estimativas de tempo e custos pois o reúso de serviços permite que o escopo de funções da solução que necessitam ser implementadas seja reduzido. A documentação dos serviços existentes mostrou-se de extrema importância para encontrá-los e determinar se suas funcionalidades podem ser reusadas.

A realização de provas de conceito para serviços que foram identificados como reusáveis na solução auxilia a garantir que o serviço: está operacional; faz o que promete fazer; se comporta adequadamente para o contexto da solução; e pode ser reusado no contexto da arquitetura da solução.

Foi possível notar que a criação dos casos de teste antes do projeto e da implementação dos serviços e da solução permitem que desenvolvedores melhorem seu entendimento sobre o que precisa ser implementado no sistema, além de criarem vários cenários de utilização do processo, o que possibilita melhorar a qualidade do código produzido.

A especificação de serviços utilizando o perfil SoaML, além de servir como documentação formal do serviço, permite que o esqueleto do código do serviço seja gerado automaticamente por meio de transformações, economizando tempo e eliminando possíveis erros de codificação do desenvolvedor.

As etapas da fase “Projetar Solução” permitem projetar adequadamente a solução, definindo detalhes de sua arquitetura, dos dados e dos componentes (classes, interfaces, *web services*, etc.), que auxiliam no trabalho de criação de código.

Após a realização dos exemplos de aplicação da IASWS notou-se que apesar de existirem repositórios de serviços gratuitos na internet, sua confiabilidade nem sempre é significativa. Os repositórios pesquisados nos exemplos limitaram-se àqueles encontrados utilizando-se mecanismos de buscas na internet; caso outros repositórios gratuitos tivessem sido encontrados e utilizados, é possível que os resultados fossem diferentes. Notou-se também que poucos provedores de serviço oferecem uma grande gama de serviços, sendo normal que um provedor forneça apenas um ou dois serviços. Outro ponto observado foi que algumas funções necessárias para atender a funcionalidade do *software* tendem a ser mais facilmente

fornechas por um serviço do que outras (por exemplo, busca de endereço por CEP versus cadastro de usuário).

A utilização da ferramenta de desenvolvimento IBM Rational Software Architect (IBM, 2010a) facilitou a aplicação da IASWS, contribuindo para acelerar o desenvolvimento de serviços e da solução pois se constitui de uma ferramenta que integra funcionalidades essenciais para o desenvolvimento e reuso de serviços. A IASWS poderia ser aplicada utilizando-se outras ferramentas de desenvolvimento, estando desvinculada do IBM Rational Software Architect.

Capítulo 6

CONCLUSÕES

A Computação Orientada a Serviços (COS) (PAPAZOGLU, 2003; HUHNS e SINGH, 2005), demanda características específicas para o desenvolvimento de sistemas que utilizam serviços. Nesse tipo de desenvolvimento, o desenvolvedor precisa se preocupar com uma série de particularidades, tratadas aqui como requisitos, sendo alguns desses requisitos: identificação de serviços, busca de serviços em operação, composição de serviços, especificação, implementação e testes de serviços, além da aderência aos conceitos da COS.

Na literatura existem abordagens para desenvolvimento de *software* orientado a serviços, como comentado no Capítulo 2. Porém, notou-se que algumas dessas abordagens tem o enfoque no desenvolvimento de sistemas de *software* compostos exclusivamente por serviços e adotam a COS integralmente, fatores esses que podem dificultar o desenvolvimento desse tipo de *software* em empresas que estão iniciando a adoção da COS. A partir dessa constatação, nesta dissertação foi desenvolvida a abordagem IASWS (em inglês, *Iterative Approach for Software Development using Web Services*), apresentada no Capítulo 4, baseada nos conceitos da COS; na modelagem de processos de negócio, utilizando BPMN (OMG, 2009); no modelo de processo Incremental; no XP (BECK, 1999) e no Scrum (SCHWABER e BEEDLE, 2001). A abordagem criada permite que serviços e conceitos da COS sejam gradualmente incorporados à solução, assim suas atividades (etapas) e artefatos foram definidos de modo que o desenvolvimento de sistemas OO possa tanto reusar serviços já existentes e disponibilizados para uso quanto desenvolver novos serviços necessários.

Este capítulo apresenta as conclusões obtidas com o desenvolvimento desta dissertação e está organizado da seguinte forma: na Seção 6.1 são comentados os

resultados obtidos; na Seção 6.2 são apresentadas as limitações do trabalho desenvolvido; na Seção 6.3 destacam-se as contribuições deste trabalho. Por fim, na Seção 6.4, são discutidos alguns dos possíveis trabalhos futuros.

6.1 Resultados Obtidos

A IASWS, apresentada no Capítulo 4, foi desenvolvida iterativamente. Assim, inicialmente foram obtidos os principais requisitos necessários para uma abordagem de desenvolvimento de *software* que use serviços. Entende-se por uso de serviços tanto a construção de novos serviços quanto o reuso de serviços já existentes e disponibilizados. Em seguida, foi realizado um estudo dos modelos de desenvolvimento de *software* tradicionais e processos/métodos para desenvolvimento de *software* orientado a serviços. A partir das informações desse estudo, elaborou-se um esboço inicial da abordagem, definindo suas fases, etapas e o fluxo de trabalho a ser seguido. Esta abordagem foi aplicada ao desenvolvimento do sistema Gerenciador de Biblioteca (primeiro exemplo de aplicação da IASWS), apresentado no Capítulo 5, resultando em modificações e refinamentos na abordagem até se chegar a um conjunto de atividades e fluxo de trabalho capazes de tratar os problemas identificados. Nesse ponto, por diversas vezes foi preciso recorrer à literatura e ao estudo realizado para propor soluções para alguns desses problemas. Um outro exemplo de aplicação da IASWS foi realizado para avaliá-la. Nesses exemplos foram considerados sistemas hipotéticos, para contemplar a maioria das possibilidades existentes quando da aplicação a um sistema real. A realização dos exemplos de aplicação da IASWS teve por objetivo mostrar a adequação das etapas (com suas atividades) e do fluxo de trabalho propostos para o desenvolvimento de *software* OO que usa de serviços. Durante o desenvolvimento dos exemplos de aplicação da IASWS foi possível: identificar e entender os processos de negócio do cliente; identificar, analisar e documentar os requisitos do sistema; identificar serviços reusáveis e testá-los; identificar e especificar novos serviços a serem desenvolvidos; formular uma solução (utilizando serviços) para a construção do sistema; desenvolver o sistema; atender às expectativas do cliente

por meio de entregas rápidas; e adotar COS e serviços em nível adequado ao conhecimento e experiência da equipe de desenvolvimento.

A utilização da modelagem de processos de negócio com BPMN para identificar e representar os requisitos do sistema, mostrou-se eficaz e auxiliou no desenvolvimento dos sistemas, e também facilitou o entendimento das funcionalidades do sistema e do contexto de negócio, no qual o sistema está inserido. Além disso, essa estratégia permitiu estabelecer uma correlação direta entre requisitos do sistema e processos de negócio, dando agilidade para realizar mudanças no sistema, quando necessário, pois auxilia na identificação dos trechos de código afetados pelas mudanças exigidas. Os modelos (*as-is* e *to-be*) podem ser utilizados como principal artefato para comunicação entre a equipe técnica e analistas de negócio.

Na literatura é muito citado o reúso de serviços existentes. Esse reúso, em nível de análise, possibilita que o desenvolvedor, logo nas etapas iniciais do desenvolvimento, possa planejar o esforço estimando de modo mais preciso o tempo que demandará no desenvolvimento completo do *software*. Na IASWS o reúso de serviços em nível de análise é possível e contribui para que a solução proposta seja obtida em menor tempo e custo, além de propiciar o retorno do investimento com a reutilização de serviços desenvolvidos anteriormente. Esse reúso é complementado e fortalecido pela realização de provas de conceito dos serviços que podem reusados. A avaliação de serviços reusados mostrou-se eficaz na verificação da adaptabilidade do serviço ao contexto da solução, agilizando o desenvolvimento uma vez que, ainda durante a concepção da solução, é possível saber se o serviço é adequado ou adaptável à solução pretendida, não sendo necessário chegar à fase de implementação e testes para essa constatação.

Assim como no XP, a IASWS especifica que os casos de teste devem ser elaborados antes de iniciar a implementação, melhorando o entendimento sobre os processos de negócio e requisitos do sistema. Isso ocorre, pois durante a elaboração dos casos de teste, vários cenários de uso do processo são analisados pelo desenvolvedor, aprofundando e refinando o seu entendimento sobre os requisitos.

Ao desenvolver os dois sistemas apresentados no Capítulo 5 pode-se notar que a abordagem proposta cobre as etapas iniciais do desenvolvimento dos sistemas, sendo que a ênfase maior foi dada às fases de levantamento de requisitos

e análise. Da forma como IASWS foi concebida o desenvolvedor tem liberdade para utilizar recursos de implementação que ele está acostumado no seu dia a dia, não restringindo o uso da abordagem com a utilização de técnicas particulares.

Por fim, as etapas da abordagem para especificação de serviços e da solução mostraram ser eficientes na identificação, definição e especificação de serviços, da solução e seus respectivos componentes. As especificações dos serviços criadas foram utilizadas para gerar o arquivo WSDL de descrição do serviço, que, posteriormente, foi utilizado para geração automática dos esqueletos de código que implementam os serviços, reduzindo consideravelmente o tempo gasto no desenvolvimento e simplificando a tarefa. A especificação da solução possibilitou a geração das classes e a criação das tabelas envolvidas na solução de maneira automatizada, liberando o desenvolvedor dessas tarefas. Além disso, o trabalho de elaborar soluções que fazem uso de serviços é facilitada, em parte, devido ao encapsulamento da lógica de negócios nos serviços reusados, reduzindo a complexidade dessas soluções.

6.2 Limitações do Trabalho

A complexidade inerente ao desenvolvimento de *software*, a velocidade em que novas tecnologias são apresentadas à comunidade e o tempo para desenvolvimento deste trabalho fizeram com que algumas decisões fossem tomadas limitando a abordagem proposta como comentado a seguir.

A abordagem assume que os processos de negócio do cliente já foram previamente descobertos e mapeados e que o cliente possui conhecimentos sobre processos de negócio. A identificação e mapeamento de processos de negócio demanda trabalho considerável e não foi amplamente abordada nesta abordagem, por dois principais motivos: 1) existem diversas técnicas consolidadas para a identificação (descoberta) dos processos de negócio; 2) deixar a abordagem mais flexível, possibilitando que o desenvolvedor utilize algo que lhe seja familiar. Quanto à modelagem desses processos, a decisão tomada foi a de adotar o BPMN como linguagem de modelagem os processos de negócio por ser de fácil entendimento e com grande utilização pelos profissionais e empresas de *software*.

Outro fator que limita a abordagem diz respeito à adoção da plataforma Java e da tecnologia *web services* para implementação dos serviços. Atualmente, a tecnologia *web services* é a mais utilizada e aceita para implementação de serviços. No entanto, existem alguns tipos de serviço como, por exemplo, serviços de infraestrutura, que não podem ser fornecidos via *web services* em decorrência da sua natureza e características. Assim, para que esses tipos de serviço possam ser reusados ou desenvolvidos, é necessário adequar as etapas da IASWS e várias considerações devem ser feitas. Algumas dessas considerações envolvem o nível de conhecimento do desenvolvedor e características peculiares do serviço, o que gera uma discussão complexa sobre esses tipos de serviço. Assim, julgou-se que isso está fora do escopo do tema de pesquisa desta dissertação e será considerado como trabalho futuro. A adoção da tecnologia Java restringiu as APIs de desenvolvimento disponíveis para o desenvolvimento dos serviços, influenciando a abordagem nesse ponto.

Como comentado anteriormente, o maior enfoque foi dado às atividades de levantamento de requisitos, análise e projeto, deixando a cargo do desenvolvedor a decisão sobre implementação e testes. Essa decisão foi tomada, considerando que existem diversas técnicas e tecnologias que podem ser utilizadas e a apresentação e discussão de cada uma delas não estaria completa, pois alguma utilizada por um desenvolvedor poderia não ser apresentada.

A abordagem também se limita a propor atividades necessárias ao desenvolvimento de sistemas novos, ou seja, aqueles que serão desenvolvidos a partir do zero. A manutenção em sistemas de software não foram consideradas durante o desenvolvimento da abordagem apresentada, ficando como sugestão para os trabalhos futuros. Entretanto, a manutenção evolutiva é viabilizada com uma nova iteração realizada até que o usuário fique satisfeito com a solução entregue (por exemplo com uma nova funcionalidade) ou com a solução que é viável no momento.

Nesta dissertação não foi considerado o desenvolvimento específico de sistemas para a Web 2.0, no entanto, isso não impede que a abordagem seja adaptada para tratar o desenvolvimento desse tipo de sistema.

6.3 Contribuições

Algumas das contribuições desta dissertação com a elaboração da IASWS, uma abordagem para o desenvolvimento de *software* que utiliza serviços (desenvolvendo-os ou reusando-os), são:

- Há uma separação entre o desenvolvimento de serviços e o desenvolvimento da solução que faz reuso de serviços em estado operacional, o que permite que essas tarefas sejam realizadas (desenvolvidas) em paralelo;
- Favorece a adoção gradual de serviços e COS no desenvolvimento de um sistema. Isso permite que os serviços sejam desenvolvidos de acordo com a necessidade, sem a obrigatoriedade de que todas as funcionalidades do sistema sejam fornecidas por serviços. Essa estratégia também facilita a adoção dos conceitos da COS, pois os serviços vão sendo introduzidos gradualmente, dando tempo para a equipe e o cliente ganharem confiança na COS e nas tecnologias;
- Possibilita identificar e reusar serviços já existentes, maximizando o retorno de investimento;
- Utiliza a modelagem de processos de negócio com BPMN, que favorece a identificação de requisitos. O modelo *to-be* gerado permite não apenas representar os requisitos de forma gráfica, relacionando-os às atividades do processo (representadas no modelo *as-is*), como também permite que os desenvolvedores tenham uma visão mais ampla do sistema a ser implementado;
- Propõe a avaliação dos serviços que serão reusados. Essa avaliação, quando realizada em etapas iniciais do desenvolvimento, possibilita verificar, por meio de testes, se o serviço se comporta conforme sua descrição e documentação, evitando chegar até a implementação/teste para descobrir que o serviço não é adequado;
- Orienta de forma clara, por meio das etapas, como os serviços já desenvolvidos são incorporados à solução; como definir quais serviços devem ser criados e como definir o que será implementado de maneira tradicional (OO);

- Pode servir como referencial para equipes de desenvolvimento que estão começando a desenvolver soluções SOA;
- Utiliza o perfil SoaML (OMG, 2009b) para documentar e especificar serviços que serão desenvolvidos e para representar os serviços incorporados à solução. Essa especificação pode ser utilizada para gerar a descrição do serviço e posteriormente o código do serviço, poupando o desenvolvedor de realizar tais tarefas manualmente;
- Contribui com uma metodologia para elaboração de abordagens de desenvolvimento.

6.4 Trabalhos Futuros

Ao longo do desenvolvimento desta dissertação, algumas oportunidades de melhoria, tanto da abordagem quanto do suporte computacional, foram identificadas. Como comentado anteriormente algumas decisões tiveram que ser adotadas para que a conclusão do trabalho fosse possível. Assim, alguns dos possíveis trabalhos futuros envolvem:

- Complementar a abordagem IASWS, detalhando as fases: Fase 5 - Implementar Solução, Fase 6 - Implementar Serviços, Fase 7 - Testar Solução, Fase 8 - Testar Serviços, Fase 9 - Verificar Aceitação, suas etapas, técnicas e tecnologias a serem adotadas;
- Realizar um estudo sobre a aplicabilidade da abordagem para projetos reais com maior complexidade de desenvolvimento;
- Adaptar a abordagem para tratar com a manutenção de sistemas de *software*;
- Realizar um estudo quantitativo para avaliar os benefícios da utilização da IASWS;
- Adaptar a abordagem permitindo o uso de outras plataformas de desenvolvimento como Microsoft .Net;
- Adaptar a abordagem para permitir o desenvolvimento ou utilização de serviços que não utilizam a tecnologia *web services*;
- Utilizar outras técnicas para modelagem dos processos de negócio; e

- Realizar um estudo comparativo para avaliar as possíveis diferenças encontradas entre outras técnicas de modelagem e a usada nesta dissertação.

REFERÊNCIAS

AWS, AMAZON Web Services. Disponível em <<http://aws.amazon.com/>>. Acesso em 01 dez. 2011.

APACHE **Axis 1.4**. Disponível em <<http://www.jcp.org/en/jsr/detail?id=101>>, (Acesso em 25 nov. 2011), 2006.

BASILI, V. R.; TURNER, A. J. Iterative Enhancement: A Practical Technique for *Software Development*. **IEEE Transactions on Software Engineering**, v. 1, n. 4, p. 390-396, dezembro 1975.

BECK, K. **Extreme Programming Explained: Embrace Change**. 1. ed. Addison-Wesley, 1999. 224 p.

BIEBERSTEIN, N.; BOSE, S.; FIAMMANTE, M.; JONES, K.; SHAH, R. **Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap**. 1. ed. IBM Press, 2005. 272 p.

BOEHM, B. A Spiral Model for *Software Development and Enhancement*. **IEEE Computer**, v. 21, n. 5, p. 61-72, maio 1988.

BOOCH, G. Object-Oriented Development. **IEEE Transactions on Software Engineering**, v. 12, n. 2, p. 211-221, fevereiro 1986.

BROOKS JR, F. P. No Silver Bullet: Essence and Accidents of *Software Engineering*. **IEEE Computer**, v. 20, n. 4, p. 10-19, 1987.

BROWN, A. W.; WALLNAU, K. C. Engineering of Component Based Systems. In: IEEE INTERNATIONAL CONFERENCE ON ENGINEERING OF COMPLEX COMPUTER SYSTEMS, 2., 1996, Montreal, Canadá. **Proceedings...** IEEE Computer Society Press, 1996. p. 414-422.

CARDOSO, E.C.S.; ALMEIDA, J.P.A.; GUIZZARDI, G. Requirements engineering based on business process models: A case study. In: ENTERPRISE DISTRIBUTED OBJECT COMPUTING CONFERENCE WORKSHOPS, 12., 2009, Auckland, Nova Zelândia. **Proceedings...** IEEE Computer Society Press, 2009. p. 320-327.

CLEMENTS, P. C. **From Subroutines to Subsystems: Component Based Software Development**. Pittsburgh, Estados Unidos: The American Programmer, 1995, v. 8, n. 11, 8 p. White paper.

DAHMAN, K.; CHAROY, F.; GODART, C. From Business Process to Component Architecture: Engineering Business to IT Alignment. In: IEEE INTERNATIONAL ENTERPRISE DISTRIBUTED OBJECT COMPUTING CONFERENCE

WORKSHOPS, 15., 2011, Helsinki, Finlândia. **Proceedings...** IEEE Computer Society Press, 2011, p. 269-274.

DE LA VARA, J. L.; SÁNCHEZ, J.; PASTOR, O. Business Process Modeling and Purpose Analysis for Requirements Analysis of Information Systems. In: ADVANCED INFORMATION SYSTEMS ENGINEERING, 20., 2008, Montpellier, França. **Proceedings...** Springer Press, 2008, p. 213-227, v. 5074.

DWIVEDI, V.; KULKARNI, N. A Model Driven Service Identification Approach for Process Centric Systems. In: CONGRESS ON SERVICES PART II, 2008, Beijing, China. **Proceedings...** IEEE, 2008. p.65-72.

ERFURTH, I.; KIRCHNER, K. Requirements Elicitation with Adapted CUTA Cards: First Experiences with Business Process Analysis. In: INTERNATIONAL CONFERENCE ON ENGINEERING OF COMPLEX COMPUTER SYSTEMS, 15., 2010, Oxford, Reino Unido. **Proceedings...** IEEE Computer Society Press, 2010. p.215-223.

ERL, T. **Service Oriented Architecture: Concepts, Technology and Design**. 1. ed. Prentice Hall, 2005. 792 p.

ERL, T. **SOA: Principles of service design**. 1. ed. Prentice Hall, 2007. 608 p.

FERNÁNDEZ, H. F.; GONZÁLES, E. P.; DÍAZ, V. G., BUSTELO, B. C. P. G.; MARTÍNEZ, O. S.; LOVELLE J. M. C. SBPMN — An easier business process modeling notation for business users. **Computer Standards & Interfaces**, v. 32, n. 1-2, p. 18-28, 2010.

FUGITA, H. S. **MAPOS: Método de Análise e Projeto Orientado a Serviços**. 175 fl. Dissertação (Mestrado em Engenharia) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2009.

FUGITA, H.S.; HIRAMA, K. MAPOS: Método de Análise e Projeto Orientado a Serviços. In: CONGRESSO INTERNACIONAL DE GESTÃO DE TECNOLOGIA E SISTEMAS DE INFORMAÇÃO, 5., 2008(a), São Paulo. **Anais...** São Paulo, 2008 (a).

FUGITA, H.S.; HIRAMA, K. Method for Service-Oriented Analysis and Design. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING RESEARCH AND PRACTICE, 2008(b), Las Vegas, Estados Unidos. **Proceedings...** Las Vegas: CSREAD Press, 2008 (b).

HAINES, M. N.; ROTHENBERGER, M. A. How a service-oriented architecture may change the *software* development process. **Commun. ACM**, v. 53, n. 8, p. 135-140, 2010.

HEKMATPOUR, S. Experience with Evolutionary Prototyping in a Large *Software* Project. **ACM Software Engineering Notes**, v. 12, n. 1, p. 38-41, 1987.

HUHNS, M. N.; SINGH, M. P. Service-oriented computing: key concepts and principles. **IEEE Internet Computing**, v. 9, n. 1, p. 75-81, 2005.

IBM **Rational Method Composer**, versão 7.5.0. IBM, 2008.

IBM **Rational Software Architect**, versão 8.0. IBM, 2010a.

IBM **Rational Service-Oriented Modeling and Architecture 2.9**. Disponível em <http://www-01.ibm.com/support/docview.wss?rs=2360&uid=swg24024567&S_TACT=105AGX15&S_CMP=ART>, (Acesso em 20 mai. 2010), 2010b.

INAGANTI S.; BEHARA G. K. **Service Identification**: BPM and SOA handshake. Disponível em <<http://www.bptrends.com>>, (Acesso em 22 jan. 2012), 2007.

JCP JSR 101: Java™ APIs for XML based RPC. Disponível em <<http://www.jcp.org/en/jsr/detail?id=101>>, (Acesso em 25 nov. 2011), 2002.

JCP JSR 224: Java™ API for XML-Based Web Services (JAX-WS) 2.0. Disponível em <<http://jcp.org/en/jsr/summary?id=jax-ws>>, (Acesso em 25 nov. 2011), 2006.

JCP JSR 311: JAX-RS: The Java™ API for RESTful Web Services. Disponível em <<http://jcp.org/en/jsr/detail?id=311>>, (Acesso em 25 nov. 2011), 2008.

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, p. 744-791, 2009.

LANE S.; BUCCHIARONE A.; RICHARDSON, I. SOAdapt: A process reference model for developing adaptable service-based applications. **Information and Software Technology**, v. 54, n. 3, p. 299-316, março 2011.

LEAVITT, H. J.; WHISLER, T. L., Management in the 1980's. **Harvard Business Review**, p. 41-48, novembro 1958.

MARKS, E. A.; BELL, M. **Service Oriented Architecture**: A Planning and Implementation Guide for Business and Technology. 1. ed. Estados Unidos: John Wiley & Sons, 2006. 384 p.

NAKAGAWA, H.; PENTEADO, R. A. D.; SANTOS, A. C. **Aplicação da abordagem IASWS no desenvolvimento do Sistema Gerenciador de Biblioteca**. São Carlos: UFSCar/DC, 2011a. Documento de trabalho.

NAKAGAWA, H.; PENTEADO, R. A. D.; SANTOS, A. C. **Aplicação da abordagem IASWS no desenvolvimento do Sistema de Atendimento de Unidade de Saúde**. São Carlos: UFSCar/DC, 2011b. Documento de trabalho.

OASIS Web Services Business Process Execution Language Version 2.0. Disponível em <<http://www.omg.org/news/whitepapers/#SoaML>>, (Acesso em 18 mai. 2010), 2007.

OMG **Business Process Modeling and Notation 1.2 Specification**. Disponível em <<http://www.omg.org/spec/BPMN/1.2/>>, (Acesso em 26 abr. 2010), 2009a.

OMG Unified Modeling Language – UML 2.0 Specification. Disponível em <<http://www.omg.org/spec/UML/2.0/>>, (Acesso em 28 abr. 2010), 2005.

OMG Service oriented architecture Modeling Language – SoaML. Disponível em <<http://www.omg.org/spec/SoaML/1.0/Beta2/>>, (Acesso em 17 jun. 2010), 2009b.

PAPAZOGLU, M. P. Service-oriented computing: concepts, characteristics and directions. In: INTERNATIONAL CONFERENCE ON WEB INFORMATION SYSTEMS ENGINEERING, 4., 2003, Roma, Itália. **Proceedings...** IEEE Computer Society Press, 2003. p. 3-12.

PAPAZOGLU, M. P. **Web Services: Principles and Technology**. 1. ed. Prentice Hall, 2007. 784 p.

PAPAZOGLU, M. P.; van den HEUVEL, W. J. Service-oriented design and development methodology. **International Journal of Web Engineering and Technology**, v. 2, n. 4, p. 412-442, 2006.

PAPAZOGLU, M. P.; TRAVERSO, P.; DUSTDAR, S.; LEYMAN, F. Service-Oriented Computing: State of the Art and Research Challenges. **IEEE Computer**, v. 40, n. 11, p. 38-45, 2007.

PRESSMAN, R. S. **Engenharia de Software**. 6. ed. São Paulo: McGraw-Hill, 2006. 720 p.

PRESSMAN, R. S. **Software Engineering: a practitioner's approach**. 7. ed. Nova Iorque: McGraw-Hill, 2010. 895 p.

RAMOLLARI, E.; DRANIDIS D.; SIMONS A. J. H. A Survey of Service Oriented Development Methodologies. In: EUROPEAN YOUNG RESEARCHERS WORKSHOP ON SERVICE ORIENTED COMPUTING, 2., 2007, Leicester, Reino Unido. **Proceedings...** 2007. p.1-6.

RECKER, J.; INDULSKA, M.; ROSEMAN, M.; GREEN, P. F. How good is BPMN really? Insights from theory and practice. In: EUROPEAN CONFERENCE ON INFORMATION SYSTEMS, 14., 2006, Göteborg, Suécia. **Proceedings...** 2006, p. 1582-1593.

ROYCE, W. W. Managing the Development of Large *Software* Systems: Concepts and Techniques. In: IEEE WESCON, 1970. **Proceedings...**1970.

SCACCHI, W. Process Models in *Software* Engineering. In: MARCINIAK, J. J. **Encyclopedia of Software Engineering**. 2. ed. Wiley, 2001. p. 993-1005.

SCHWABER, K.; BEEDLE, M. **Agile Software Development with Scrum**, 1. ed. Prentice Hall, 2001. 158 p. (Series in *Agile Software* Development).

SOMMERVILLE, I. **Engenharia de Software**. 8. ed. São Paulo: Pearson Addison-Wesley, 2007. 549 p.

van der AALST, W. M. P.; ter Hofstede, A. H. M.; WESKE, M.; Business Process Management: A Survey. In: BUSINESS PROCESS MANAGEMENT, INTERNATIONAL CONFERENCE, 2003, Eindhoven, Países Baixos. **Proceedings...** Springer Press, 2003, p. 1019-1029, v. 2678.

W3C Web Services Architecture. Disponível em <<http://www.w3.org/TR/ws-arch/>>, (Acesso em: 11 mai. 2010), 2004.

W3C Web Services Architecture. Disponível em <<http://www.w3.org/TR/ws-arch/>>, (Acesso em: 11 mai. 2010), 2004a.

W3C Web Services Description Language (WSDL) version 2.0. Disponível em <<http://www.w3.org/TR/wsdl20-primer/>>, (Acesso em: 11 abr. 2010), 2007.

W3C. XML Schema. Disponível em: < <http://www.w3.org>>. (Acesso em: 24 mar 2010), 2004b .

WHITE, S. A. **Introduction to BPMN**. Disponível em <http://www.bpmn.org/Documents/Introduction_to_BPMN.pdf>, (Acesso em: 11 abr. 2010), 2004.

YAN, Z.; MAZZARA, M.; CIMPIAN, E.; URBANEC, A. Business Process Modeling: Classifications and Perspectives. In: BUSINESS PROCESS AND SERVICES COMPUTING: INTERNATIONAL WORKING CONFERENCE ON BUSINESS PROCESS AND SERVICES COMPUTING, 1., 2007, Leipzig, Alemanha. **Proceedings...** Leipzig, 2007, p. 222-227.

YDN, YAHOO! Developer Network. Disponível em <<http://developer.yahoo.com/>>. Acesso em 07 set. 2011.