Uma abordagem de apoio à avaliação e melhoria de
processo de software baseada em metamodelagem
e transformações de modelos

Daniel Fernando Galego Feloni

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Daniel Fernando Galego Feloni

Uma abordagem de apoio à avaliação e melhoria de
processo de software baseada em metamodelagem e
transformações de modelos

Dissertação apresentada ao Instituto de Ciências
Matemáticas e de Computação – ICMC-USP,
como parte dos requisitos para obtenção do título
de Mestre em Ciências – Ciências de Computação e
Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e
Matemática Computacional

Orientadora: Profa. Dra. Rosana Teresinha
Vacarre Braga

USP – São Carlos
Maio de 2016

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

F323a Feloni, Daniel Fernando Galego
 Uma abordagem de apoio à avaliação e melhoria
de processo de software baseada em metamodelagem
e transformações de modelos / Daniel Fernando
Galego Feloni; orientadora Rosana Teresinha
Vacarre Braga. - São Carlos - SP, 2016.
 159 p.

 Dissertação (Mestrado - Programa de Pós-Graduação
em Ciências de Computação e Matemática Computacional)
- Instituto de Ciências Matemáticas e de Computação,
Universidade de São Paulo, 2016.

 1. Processo. 2. Modelos de Maturidade. 3. MPS.Br.
4. CMMI. Engenharia de Software Orientada a Modelos.
5. Metamodelos. 6. Metamodelagem. 7. Avaliação de
Processos de Software. 8. Melhoria de Processos de
Software. 9. Transformações de Modelos. I. Braga,
Rosana Teresinha Vacarre, orient. II. Título.

Daniel Fernando Galego Feloni

**An approach to support assessment and improvement of
software processes based on metamodeling and model
transformations**

Master dissertation submitted to the Instituto de
Ciências Matemáticas e de Computação – ICMC-
USP, in partial fulfillment of the requirements for the
degree of the Master Program in Computer Science
and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and
Computational Mathematics

Advisor: Profa. Dra. Rosana Teresinha Vacarre Braga

USP – São Carlos
May 2016

À minha família, amigos, professores e principalmente à Deus, pois, sem eles este trabalho não teria sido concluído.

AGRADECIMENTOS

A minha querida orientadora, Prof. Dr^a. Rosana Teresinha Vaccare Braga pela dedicação, empenho, compreensão e contribuição constantes, para a conclusão deste trabalho.

Aos meus pais e minhas irmãs pelo apoio e compreensão nos momentos difíceis que tivemos durante a elaboração deste trabalho.

Aos meus colegas de trabalho pela compreensão pelos momentos de ausência para que este trabalho pudesse ser concretizado.

A CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pelo apoio financeiro concedido de 01/2014 a 02/2015 para realização deste trabalho.

“O homem é do tamanho do seu sonho.”
(Fernando Pessoa)

RESUMO

FELONI, DANIEL. **Uma abordagem de apoio à avaliação e melhoria de processo de software baseada em metamodelagem e transformações de modelos.** 2016. 159 f. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

Melhoria de processo de software (SPI) é uma prática de engenharia de software motivada pela necessidade de aumentar a qualidade e a produtividade no desenvolvimento de software. Um fato amplamente reconhecido é que a qualidade do produto de software pode ser, em grande parte, determinada pela qualidade do processo utilizado para desenvolvê-lo e mantê-lo. A avaliação do processo de software ajuda as organizações de software a amadurecerem seus processos, identificando problemas críticos para estabelecer prioridades de melhoria. Essa avaliação pode ser feita por meio da comparação do estado dos processos da organização em relação a um modelo de referência que estabeleça estágios de melhoria. Uma avaliação geralmente se baseia em um modelo de processo de software que fornece um roteiro para melhorias. Este trabalho tem como objetivo estabelecer uma abordagem que: (i) define um conjunto de modelos de abstração (metamodelos) de modelos de maturidade de processo de software para apoiar uma metodologia de avaliação/melhoria de processo de software com o objetivo de certificação; e (ii) permite avaliar os processos de uma organização em comparação com um modelo de maturidade por meio de transformações desses metamodelos. A abordagem é instanciada por meio de um estudo de caso utilizando os modelos MPS.Br e CMMI para exemplificar sua aplicação. Como resultado, é apresentado um comparativo entre as limitações encontradas nas metodologias encontradas na literatura e como a abordagem sugere superá-las.

Palavras-chave: Processo, Modelos de Maturidade, MPS.Br, CMMI. Engenharia de Software Orientada a Modelos, Metamodelos, Metamodelagem, Avaliação de Processos de Software, Melhoria de Processos de Software, Transformações de Modelos.

ABSTRACT

FELONI, DANIEL. **An approach to support assessment and improvement of software processes based on metamodeling and model transformations.** 2016. 159 f. Master dissertation (Master student Program in Computer Science and Computational Mathematics) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

Software process improvement (SPI) is a software engineering practice motivated by the need to increase the quality and productivity in software development. A fact widely recognized is that the quality of the software product can be largely determined by the quality of the process used to develop and maintain it. The assessment of software process helps software organizations to improve themselves, identifying their critical problems to establish priorities for improvement. This assessment can take place by comparing the state of the organization on their software processes to a reference model that shows stages of improvement in scales. An assessment is usually based on a software process model that provides a roadmap for improvement. This work aims to establish an approach that: (i) defines a set of abstraction models (metamodels) of software process maturity models to support an assessment/improvement methodology aiming software process certification; and (ii) evaluates the organization processes in comparison with the maturity models through transformations of those metamodels. The approach is instantiated through a case study using the MPS.Br and CMMI models to illustrate its application. As a result, a comparison between the limitations found in the methodologies identified in the literature and how the approach suggested overcome them is presented.

Key-words: Process, Maturity Models, MPS.Br, CMMI, Model-Based Software Engineering, Metamodels, Metamodeling, Software Process Assessment, Software Process Improvement, Model Transformations.

LISTA DE ILUSTRAÇÕES

Figura 1 – Composição do modelo MPS.BR Softex (2016).	44
Figura 2 – Transformações de modelos ao longo do ciclo de vida (BRAMBILLA; CABOT; WIMMER, 2012)	51
Figura 3 – Representação da estrutura do <i>framework</i> SPEM SPEM (2015).	52
Figura 4 – Metamodelo criado por Musat <i>et al.</i> (2010) que representa o modelo de maturidade CMMI (MUSAT <i>et al.</i> , 2010).	69
Figura 5 – Ambiente MATURE em funcionamento no Eclipse (MUSAT <i>et al.</i> , 2010). .	70
Figura 6 – Apresentação do modelo conceitual do processo CAR (MUSAT <i>et al.</i> , 2010).	70
Figura 7 – Apresentação da modelagem da visão de configuração na ferramenta MATURE (MUSAT <i>et al.</i> , 2010).	70
Figura 8 – Apresentação da modelagem das partes específicas do processo CAR na ferramenta MATURE (MUSAT <i>et al.</i> , 2010).	71
Figura 9 – Representação conceitual da abordagem utilizada para a harmonização de diferentes IRM (JENERS; LICHTER; ROSENKRANZ, 2013).	72
Figura 10 – Representação do metamodelo da ISM (JENERS; LICHTER; ROSENKRANZ, 2013).	72
Figura 11 – Representação do estudo de caso realizado pelos autores do artigo (JENERS; LICHTER; ROSENKRANZ, 2013).	73
Figura 12 – Arquitetura do repositório proposto SIR-CM (PARK; KIM; LEE, 2008). .	74
Figura 13 – Integração do ambiente SIR-CM com outras ferramentas (PARK; KIM; LEE, 2008).	74
Figura 14 – Estrutura proposta para o repositório MCM (WANGENHEIM <i>et al.</i> , 2011). .	75
Figura 15 – Representação das etapas de execução da abordagem.	96
Figura 16 – Representação das atividades executadas na etapa Modelagem Oráculo . .	97
Figura 17 – Representação das atividades executadas na etapa Criação Avaliações de Conformidade	97
Figura 18 – Representação das atividades executadas na etapa Modelagem Usuário . .	98
Figura 19 – Representação das atividades executadas na etapa Execução Avaliação de Conformidade	98
Figura 20 – Representação das atividades executadas nas etapas Análise Relatório e Sugestão de Melhorias	99
Figura 21 – Pacotes de metamodelos criados para dar suporte à aplicação da abordagem proposta.	100

Figura 22 – Visão geral da abordagem proposta.	101
Figura 23 – Modelagem conceitual da abordagem proposta sem atributos.	103
Figura 24 – Modelagem conceitual do pacote de metamodelo <i>Taxonomy</i>	105
Figura 25 – Modelagem conceitual do pacote de metamodelo <i>MaturityModel</i>	106
Figura 26 – Modelagem conceitual do pacote de metamodelo <i>Process</i>	107
Figura 27 – Modelagem conceitual do pacote de metamodelo <i>Project</i>	108
Figura 28 – Modelagem conceitual do pacote de metamodelo <i>StructureAndBehavior</i> . . .	110
Figura 29 – Modelagem conceitual do pacote de metamodelo <i>Assessment</i>	112
Figura 30 – Modelagem conceitual do pacote de metamodelo <i>Report</i>	113
Figura 31 – Modelagem conceitual do pacote de metamodelo <i>Improvement</i>	114
Figura 32 – Arquivo do modelo Ecore criado para o pacote de metamodelos <i>MaturityModel</i> .	115
Figura 33 – Modelagem no <i>framework</i> Ecore do pacote de metamodelo <i>MaturityModel</i> pelo editor visual (<i>plugin Ecore Diagram</i>).	116
Figura 34 – Arquivo de instância do modelo <i>MaturityModel</i> no Eclipse.	119
Figura 35 – Aba de propriedades de instância do modelo <i>MaturityModel</i> no Eclipse. . .	119
Figura 36 – Diagrama de casos de uso representando a etapa de definição dos modelos <i>Ta-</i> <i>xi</i> <i>onomy</i> , <i>Process Template</i> , <i>ProcessInstance</i> , <i>MaturityModel</i> e os respectivos <i>StructureAndBehavior</i> da abordagem proposta.	125
Figura 37 – Diagrama de casos de uso representando a etapa da avaliação e sugestão de melhorias dos modelos criados.	126
Figura 38 – Arquivo de instância do modelo <i>StructureAndBehavior</i> criado para a fase de iniciação do processo de GRE no Eclipse.	131
Figura 39 – Arquivo de instância do modelo <i>Taxonomy</i> criado para o processo de GRE no Eclipse.	132
Figura 40 – Arquivo de instância do modelo <i>MaturityModel</i> criado para exemplificar uma modelagem do CMMI.	132
Figura 41 – Arquivo de instância do modelo <i>Project</i> e <i>ProcessInstance</i> criado para o processo de GRE no Eclipse.	133
Figura 42 – Arquivo de instância do modelo <i>StructureAndBehavior</i> criado para o processo de GRE no Eclipse.	133
Figura 43 – Arquivo de instância do modelo <i>Assessment</i> criado para o processo de GRE no Eclipse.	134
Figura 44 – Arquitetura do ambiente <i>AIRES</i> , retirado de (BRAGA <i>et al.</i> , 2016).	142
Figura 45 – Questão do quadro <i>Perfil da Empresa</i> do formulário aplicado ao <i>survey</i> realizado.	147
Figura 46 – Questão do quadro <i>Perfil da Empresa</i> do formulário aplicado ao <i>survey</i> realizado.	147
Figura 47 – Questão do quadro <i>Perfil da Empresa</i> do formulário aplicado ao <i>survey</i> realizado.	148

Figura 48 – Questão do quadro <i>Perfil da Empresa</i> do formulário aplicado ao <i>survey</i> realizado.	148
Figura 49 – Questão do quadro <i>Perfil da Empresa</i> do formulário aplicado ao <i>survey</i> realizado.	148
Figura 50 – Questão do quadro <i>Perfil da Empresa</i> do formulário aplicado ao <i>survey</i> realizado.	148
Figura 51 – Questão do quadro <i>Perfil do Participante</i> do formulário aplicado ao <i>survey</i> realizado.	149
Figura 52 – Questão do quadro <i>Perfil do Participante</i> do formulário aplicado ao <i>survey</i> realizado.	149
Figura 53 – Questão do quadro <i>Perfil do Participante</i> do formulário aplicado ao <i>survey</i> realizado.	149
Figura 54 – Questão do quadro <i>Perfil do Participante</i> do formulário aplicado ao <i>survey</i> realizado.	149
Figura 55 – Questão do quadro <i>Perfil do Participante</i> do formulário aplicado ao <i>survey</i> realizado.	149
Figura 56 – Questão do quadro <i>Perfil do Participante</i> do formulário aplicado ao <i>survey</i> realizado.	150
Figura 57 – Questão do quadro <i>Perfil do Participante</i> do formulário aplicado ao <i>survey</i> realizado.	150
Figura 58 – Questão do quadro <i>Perfil do Participante</i> do formulário aplicado ao <i>survey</i> realizado.	150
Figura 59 – Questão do quadro <i>Perfil do Participante</i> do formulário aplicado ao <i>survey</i> realizado.	150
Figura 60 – Questão do quadro <i>Perfil do Participante</i> do formulário aplicado ao <i>survey</i> realizado.	150
Figura 61 – Questão do quadro <i>Experiência com Implementação de Modelos de Maturidade e Processo de Software</i> do formulário aplicado ao <i>survey</i> realizado. . .	151
Figura 62 – Questão do quadro <i>Experiência com Implementação de Modelos de Maturidade e Processo de Software</i> do formulário aplicado ao <i>survey</i> realizado. . .	151
Figura 63 – Questão do quadro <i>Experiência com Implementação de Modelos de Maturidade e Processo de Software</i> do formulário aplicado ao <i>survey</i> realizado. . .	151
Figura 64 – Questão do quadro <i>Experiência com Implementação de Modelos de Maturidade e Processo de Software</i> do formulário aplicado ao <i>survey</i> realizado. . .	152
Figura 65 – Questão do quadro <i>Experiência com Implementação de Modelos de Maturidade e Processo de Software</i> do formulário aplicado ao <i>survey</i> realizado. . .	152
Figura 66 – Questão do quadro <i>Experiência com Implementação de Modelos de Maturidade e Processo de Software</i> do formulário aplicado ao <i>survey</i> realizado. . .	152

Figura 67 – Questão do quadro <i>Experiência com Implementação de Modelos de Maturidade e Processo de Software</i> do formulário aplicado ao survey realizado. . .	152
Figura 68 – Questão do quadro <i>Experiência com Implementação de Modelos de Maturidade e Processo de Software</i> do formulário aplicado ao survey realizado. . .	152
Figura 69 – Questão do quadro <i>Experiência com Implementação de Modelos de Maturidade e Processo de Software</i> do formulário aplicado ao survey realizado. . .	153
Figura 70 – Questão do quadro <i>Experiência com Implementação de Modelos de Maturidade e Processo de Software</i> do formulário aplicado ao survey realizado. . .	153
Figura 71 – Questão do quadro <i>Experiência com Implementação de Modelos de Maturidade e Processo de Software</i> do formulário aplicado ao survey realizado. . .	153
Figura 72 – Questão do quadro <i>Experiência com Implementação de Modelos de Maturidade e Processo de Software</i> do formulário aplicado ao survey realizado. . .	153
Figura 73 – Questão do quadro <i>Experiência com Implementação de Modelos de Maturidade e Processo de Software</i> do formulário aplicado ao survey realizado. . .	154
Figura 74 – Questão do quadro <i>Experiência com Implementação de Modelos de Maturidade e Processo de Software</i> do formulário aplicado ao survey realizado. . .	154
Figura 75 – Questão do quadro <i>Experiência com Implementação de Modelos de Maturidade e Processo de Software</i> do formulário aplicado ao survey realizado. . .	154
Figura 76 – Questão do quadro <i>Experiência com Implementação de Modelos de Maturidade e Processo de Software</i> do formulário aplicado ao survey realizado. . .	154
Figura 77 – Questão do quadro <i>Experiência com Implementação de Modelos de Maturidade e Processo de Software</i> do formulário aplicado ao survey realizado. . .	155
Figura 78 – Questão do quadro <i>Experiência com Implementação de Modelos de Maturidade e Processo de Software</i> do formulário aplicado ao survey realizado. . .	155
Figura 79 – Questão do quadro <i>Experiência com Implementação de Modelos de Maturidade e Processo de Software</i> do formulário aplicado ao survey realizado. . .	155
Figura 80 – Questão do quadro <i>Experiência com Implementação de Modelos de Maturidade e Processo de Software</i> do formulário aplicado ao survey realizado. . .	155

LISTA DE ALGORITMOS

Algoritmo 1 – Algoritmo de transformação <i>model-to-model</i> Avaliação do questionário.	123
Algoritmo 2 – Algoritmo de transformação <i>model-to-model</i> Avaliação de termos da taxonomia.	124

LISTA DE CÓDIGOS-FONTE

Código-fonte 1 – Arquivo em <i>XMI</i> do diagrama Ecore referente ao pacote de metamodelos <i>MaturityModel</i>	116
--	-----

LISTA DE TABELAS

Tabela 1 – Representação do modelo CMMI na forma <i>Continuous</i> e <i>Staged</i>	43
Tabela 2 – Níveis de maturidade, processos e atributos de processos (SOFTEX, 2016) .	47
Tabela 3 – Termos utilizados na construção da string de busca.	61
Tabela 4 – Fases do processo de seleção dos trabalhos.	62
Tabela 5 – Distribuição dos critérios de inclusão e de exclusão.	62
Tabela 6 – Artigos selecionados na fase 4.	63
Tabela 7 – Demonstração do indicador Implementação de Prática (HOMCHUENCHOM <i>et al.</i> , 2012).	65
Tabela 8 – Indicador Característica de Prática (HOMCHUENCHOM <i>et al.</i> , 2012). . . .	66
Tabela 9 – Indicador Objetivos (HOMCHUENCHOM <i>et al.</i> , 2012).	66
Tabela 10 – Indicador Satisfação de área de Processos (HOMCHUENCHOM <i>et al.</i> , 2012). .	66
Tabela 11 – Sumarização dos estudos identificados e suas principais características. . . .	79
Tabela 12 – Práticas equivalentes entre MPS.Br e CMMI para o resultado de processo GRE1.	132
Tabela 13 – Cenários identificados ao responder as questões definidas no modelo <i>SubjectiveAssessment</i>	135
Tabela 14 – Sugestões para os cenários identificados ao responder as questões definidas no modelo <i>SubjectiveAssessment</i>	135
Tabela 15 – Sugestões para os cenários identificados ao responder as questões definidas no modelo <i>SubjectiveAssessment</i> . (Continuação)	136
Tabela 16 – Descrição resumida dos casos de uso presentes na Figura 36.	157
Tabela 17 – Descrição resumida dos casos de uso presentes na Figura 36. (Continuação)	158
Tabela 18 – Descrição resumida dos casos de uso presentes na Figura 37.	158
Tabela 19 – Descrição resumida dos casos de uso presentes na Figura 37. (Continuação)	159

LISTA DE ABREVIATURAS E SIGLAS

API		Application Programming Interface
ATL		Model Transformation Language
CASE		Computer-Aided Software Engineering
CMMI	...	Capability Maturity Model Integration
CMMI-DEV		Capability Maturity Model Integration: for Development
CMMI-SVC		Capability Maturity Model Integration: for Services
EMF		Eclipse Modeling Framework
ISO/IEC	..	International Organization of Standardization/International Electrotechnical Commission
MDD		Model Driven Development
MDE		Model Driven Engineering
MOF		Meta Object Facility
MPS.Br	..	Melhoria do Processo de Software Brasileiro
MR-MPS-SW		Modelo de Referência - Melhoria do Processo de Software - Software
OMG		Object Management Group
SEI		Software Engineering Institute
SOFTEX	..	Associação para Promoção da Excelência do Software Brasileiro
SPA		Software Process Assessment
SPEM		Software & Systems Process Engineering Metamodel
SPI		Software Process Improvement
SPICE		Software Process Improvement and Capability
UML		Unified Modeling Language
XMI		XML Metadata Interchange

SUMÁRIO

1	INTRODUÇÃO	31
1.1	Contextualização	31
1.2	Motivação e Justificativa	32
1.3	Objetivos	35
1.4	Metodologia	35
1.5	Organização	36
2	FUNDAMENTAÇÃO TEÓRICA E TÉCNICA	37
2.1	Considerações Iniciais	37
2.2	Qualidade de Software	38
2.3	Modelos de Maturidade	41
2.3.1	CMMI	41
2.3.2	MPS.Br (Melhoria do Processo de Software Brasileiro)	43
2.3.3	Programa MPS - Atributos de Processo e seus Resultados Esperados de Atributos de Processo	45
2.3.4	Base Técnica MPS.Br	47
2.4	Engenharia Orientada a Modelos	48
2.4.1	Processos MDE	49
2.5	SPEM - Software & Systems Process Engineering Metamodel	51
2.6	Ferramentas de Apoio Utilizadas	53
2.7	Considerações Finais	55
3	MAPEAMENTO SISTEMÁTICO	57
3.1	Considerações Iniciais	57
3.2	Planejamento do Mapeamento Sistemático	58
3.2.1	Objetivos da Pesquisa	58
3.2.2	Critérios de Inclusão e de Exclusão	59
3.2.3	Seleção da Fonte de Busca	60
3.2.4	Construção da String de Busca	60
3.3	Condução do Mapeamento Sistemático	61
3.4	Síntese dos Trabalhos Identificados	64
3.4.1	SPIALS-II: A light-weight software process improvement self assessment tool	64

3.4.2	<i>Taba workstation: Supporting software process deployment based on CMMI and MR-MPS.BR</i>	67
3.4.3	<i>MATURE: A model driven based tool to automatically generate a language that supports CMMI process areas specification</i>	69
3.4.4	<i>Efficient adoption and assessment of multiple process improvement reference models</i>	71
3.4.5	<i>Software repository for software process improvement</i>	73
3.4.6	<i>Building a Maturity & Capability Model repository</i>	74
3.5	Análise Comparativa dos Trabalhos Seleccionados	76
3.6	Considerações Finais	80
4	SURVEY - INDÚSTRIA DE SOFTWARE E PROCESSOS	83
4.1	Considerações Iniciais	83
4.2	Planejamento	83
4.3	Análise dos Dados	85
4.4	Considerações Finais	91
5	ABORDAGEM PROPOSTA	93
5.1	Considerações Iniciais	93
5.2	Visão Geral da abordagem	95
5.3	Modelos Conceituais	101
5.3.1	<i>Taxonomy</i>	104
5.3.2	<i>MaturityModel</i>	105
5.3.3	<i>Process</i>	106
5.3.4	<i>Project</i>	107
5.3.5	<i>StructureAndBehavior</i>	108
5.3.6	<i>Assessment</i>	110
5.3.7	<i>Report</i>	113
5.3.8	<i>Improvement</i>	113
5.4	Metamodelos	114
5.5	Exemplo de Modelagem	119
5.6	Métodos de Avaliação	119
5.7	Algoritmos de Transformações	121
5.8	Modelagem de Casos de Uso	125
5.9	Considerações Finais	127
6	ESTUDO DE CASO	129
6.1	Considerações Iniciais	129
6.2	Planejamento	129
6.3	Modelos <i>MaturityModel</i> , <i>StructureAndBehavior</i> e <i>Taxonomy</i>	131

6.4	Modelos <i>Project, Process e StructureAndBehavior</i>	132
6.5	Modelo <i>Assessment, Report e Improvement</i>	133
6.6	Considerações Finais	136
7	CONCLUSÕES E TRABALHOS FUTUROS	137
7.1	Contribuição	137
7.2	Limitações e Riscos à Validação	139
7.3	Trabalhos Futuros	141
REFERÊNCIAS		143
APÊNDICE A FORMULÁRIO APLICADO AO <i>SURVEY</i>		147
A.1	Quadro de Questões: Perfil da Empresa	147
A.2	Quadro de Questões: Perfil do Participante	148
A.3	Quadro de Questões: Experiência com Implementação de Modelos de Maturidade e Processo de Software	150
A.4	Quadro de Questões: Expectativas em Relação a Ferramentas de Avaliação e Melhoria de Processos de Software	154
APÊNDICE B GLOSSÁRIO CASOS DE USO		157

INTRODUÇÃO

1.1 Contextualização

Software pode ser considerado como uma das tecnologias de grande importância no cenário mundial, na qual negócios, governos, ciência, exploração espacial e comportamento são controlados por sistemas de computadores cada vez mais complexos, robustos e interconectados. Mas, para chegar ao que se conhece hoje sobre desenvolvimento de software, muitas crenças e formas de trabalhar precisaram ser reinventadas para que se chegasse ao que hoje é conhecido como Engenharia de Software.

No início da década de 70, a indústria de software estava com sérios problemas relacionados aos produtos desenvolvidos. O termo *Crise do Software* foi utilizado largamente para designar o desenvolvimento de software em que os produtos desenvolvidos eram falhos e não cumpriam seus cronogramas de entrega e orçamentos estimados. Tais problemas ocorriam em razão de falhas relacionadas ao projeto do produto, como: (i) coleta/análise ineficaz de requisitos; (ii) custos que ultrapassavam o esperado; (iii) estimativas incorretas gerando atrasos nas entregas; e (iv) produtos de baixa qualidade e de difícil manutenção. Com isso, a indústria entrou em um estado de alerta por parte dos fabricantes de software, o que estimulou profissionais da área de desenvolvimento a iniciar uma série de pesquisas e discussões sobre boas práticas de projeto, de desenvolvimento e de implantação. Uma nova forma de engenharia estava sendo descoberta, introduzindo-se o conceito Engenharia de Software ([PRESSMAN, 2014](#)).

A introdução do software no cotidiano das pessoas e a grande escala de diversidade de software disponível para facilitar as mais diversas atividades, fez com que essa poderosa ferramenta mudasse a forma da sociedade pesquisar, comprar e se relacionar. Entretanto, como toda nova área de conhecimento, muitos desafios precisaram ser superados. Com o passar do tempo, foi identificada a necessidade constante de corrigir, adaptar e melhorar diversos softwares. Porém, o ônus de realizar essas atividades de manutenção faria uso de mais pessoas e demais

recursos do que todo o trabalho despendido na criação do software.

1.2 Motivação e Justificativa

A maioria das empresas de desenvolvimento de software enfrentam problemas em seus projetos por causa da falta de implementação de melhores práticas de processos e de padrões (ALI; IBRAHIM, 2011). Em seu artigo, Ali e Ibrahim (2011) citam um estudo realizado e que identificou que 50% das aplicações falham em atender seus objetivos de negócio, enquanto que 40% dos projetos de TI (*Tecnologia da Informação*) falham em alcançar o retorno financeiro esperado pela falta de utilização de boas práticas nos processos de software utilizados no desenvolvimento.

Entre as causas da falta de implementação de melhores práticas de processos, está o baixo nível de consciência e de conhecimento das empresas de desenvolvimento de software. Existe também uma questão de custo envolvido para a implementação de normas ou de modelos de referência, que se tornou um dos fatores de impedimento importante para pequenas e médias empresas (ALI; IBRAHIM, 2011).

Além dos fatores citados anteriormente, o que causa problemas nas organizações para a escolha de um processo de desenvolvimento de software é a variedade de modelos disponíveis. Outro fator identificado é a introdução das metodologias ágeis, que têm sido tentadoras para as empresas levando em consideração seus objetivos. Entretanto, o modelo de processo ou método de desenvolvimento utilizado pela organização deve ser escolhido levando em consideração o contexto de cada projeto.

As organizações que implementam padrões de processo internacionais de maturidade de processos, tais como, CMMI, são capazes de identificar pontos fortes e fracos de uma forma sistemática. Por outro lado, as organizações que falham na implementação de boas práticas de desenvolvimento de projetos de software não são capazes de avaliar a sua força e fraquezas de forma eficaz e isso vai afetar a eficácia do processo a longo prazo. As organizações e os profissionais vão encontrar-se em uma implementação desestruturada de projeto que resultará em problemas como a falta de documentação. Com base nas declarações de problemas acima identificados, Ali e Ibrahim (2011) *apud* Rout e Tuffley (2007) reafirma que há necessidade de harmonizar e de integrar diferentes abordagens do processo de software no contexto de avaliação de processos e modelos de maturidade (ALI; IBRAHIM, 2011). Nesse contexto, no capítulo seguinte apresenta-se um mapeamento sistemático conduzido com o intuito de investigar o estado da arte em relação ao apoio computacional à melhoria de processo nas organizações de software.

Para atender a essa necessidade, os modelos de maturidade de processo de software começaram a surgir, como por exemplo o Projeto SPICE do inglês *Software Process Improvement and Capability* (ISO, 2013) e CMMI-DEV (do inglês *Capability Maturity Model - Integration for Development*) (SEI, 2013). Os modelos de maturidade funcionam como metamodelos de

processos que surgiram para avaliar a qualidade dos processos de software aplicados em uma organização. A maturidade de um processo pode ser determinada pelos fatores planejamento, controle de modificações e capacidade de evolução contínua de seus subprocessos (SOFTEX, 2016). Outras formas para identificar problemas e corrigi-los foram sugeridas, como o PDCA (criado por William Edwards Deming em seus trabalhos com melhoria de projetos, do inglês *Plan, Do, Check, Act*) (PRESSMAN, 2014).

Cada organização possui fatores distintos que devem ser levados em consideração ao elaborar um processo de software a ser utilizado em seu dia-a-dia de desenvolvimento, tais como, foco na funcionalidade ou na documentação, entregas pequenas em iterações ou entregas de produtos completos e envolvimento dos interessados ao longo do processo ou somente no início, etc. Cada variação da combinação desses fatores pode resultar em um processo diferente para melhor atender aquele cenário específico de desenvolvimento.

Nem todas as organizações possuem recursos financeiros e/ou humanos para realizar um levantamento relacionando planejamento estratégico da empresa e as atividades relacionadas ao desenvolvimento de seus produtos, com o objetivo de elaborar um processo que atenda suas necessidades e produza software de qualidade (GARCÍA *et al.*, 2012). Independentemente dos fatores citados anteriormente, realizar a definição de processos não é uma tarefa trivial e deve ser realizada por pessoas qualificadas no assunto, o que demanda eventualmente a capacitação interna de recursos para a realização dessa tarefa ou até a contratação de consultoria externa.

Pensando nesses fatores, faz-se necessário buscar soluções para evitar problemas com a qualidade dos produtos de software produzidos, que repetidas vezes são encarados por vários profissionais de engenharia de software. A existência de uma base de conhecimento que concentre processos e artefatos de apoio à execução de processos poderia ser disponibilizada como uma ferramenta de apoio que pudesse ser auxiliar a indústria. Futuramente, a base de conhecimento pode se tornar ainda mais eficiente, concentrando artefatos que possam apoiar processos, certificando-os de acordo com algum modelo de maturidade desejado. Seu acesso poderia ser realizado de forma a ser uma ferramenta que pudesse ser acessada pela indústria para que, desse modo, fosse possível compartilhar processos, artefatos, experiências, lições aprendidas, métricas, modelos de análise de requisitos, entre outros. Esses artefatos reunidos podem ser transformados em uma base para referência da indústria de software, em que uma empresa pode descrever suas principais características e obter sugestões de artefatos que viabilizem seus processos e certifique-os.

Lepasaar e Makinen (2002) afirmam que a melhoria do processo de software (SPI, *Software Process Improvement*) é motivada pela necessidade de aumentar a qualidade e a produtividade do trabalho em projetos de software. Um fato amplamente aceito é a qualidade do produto de software ser em grande parte, determinada pela qualidade do processo utilizado para desenvolvê-lo e mantê-lo. Com isso, SPI exige o conhecimento do que é necessário, a consciência do que é viável, e um compromisso de longo prazo para o seu investimento. Por

outro lado, avaliação de processo de software ajuda as organizações de software melhorar a si mesmas, identificando seus problemas críticos e estabelecendo prioridades de melhoria. A avaliação é realizada por meio da comparação do estado do processo de software da organização contra o modelo e a escala de melhoria desejada. Uma avaliação geralmente se baseia em um processo de software executado nos projetos desenvolvidos pela organização e um roteiro de melhoria (modelos de maturidade). O interesse na melhoria do processo de software criou várias normas, padrões e modelos internacionais de processo de software. O CMMI e SPICE são modelos aceitos pelas organizações para avaliação de maturidade de processos.

Lepasaar e Makinen (2002) citam que, a fim de ajudar as empresas de software, metodologias de suporte à execução de atividades de SPI devem ser desenvolvidas. Essas metodologias devem ser criadas para orientar as organizações na melhoria de seus processos de software, diminuindo a necessidade de consultorias externas e avaliadores terceirizados. Modelos de processos de software formariam a parte central dessas metodologias de apoio. Com o intuito de implementar vários modelos de processos, a estrutura dos modelos deveria ser analisada e os modelos integrados em um metamodelo comum. O metamodelo contribuiria para uma avaliação mais completa do processo, uma vez que incluiria elementos de modelos de processos variados.

Dentre os trabalhos relacionados identificados na literatura, (PIYABUNDITKUL; WONG-CHINGCHAI; METHAWACHANANONT, 2010) propõe uma abordagem baseada na ferramenta chamada *CMMIbySCRUM* para melhorar os processos baseados em CMMI combinados com técnicas de métodos ágeis como o Scrum. Se as organizações de software estiverem cientes da situação atual da capacidade de seus processos de software e tiverem uma diretriz de melhoria com base em suas metas de qualidade, elas podem ser capazes de melhorar substancialmente seus processos. Para apoiar essas organizações em seu caminho para melhores processos, o artigo apresenta o projeto de uma ferramenta genérica (SPIALS, *Software Process Improvement Adaptive Learning System*) aplicável para medir o processo de recuperação de *status* da capacidade dos processos das organizações. Microempresas podem usar a ferramenta para realizar uma auto-avaliação, reduzindo assim o processo de avaliação complexa e dependente de consultorias externas. A estratégia de avaliação da ferramenta é baseada no padrão *CMMI Appraisal Method* para Melhoria de Processos (SCAMPI), bem reconhecido por ser o padrão CMMI de avaliação (HOMCHUENCHOM *et al.*, 2011).

A avaliação do processo de software ajuda as organizações de software a amadurecerem seus processos, identificando problemas críticos para estabelecer prioridades de melhoria. Essa avaliação pode ser feita por meio da comparação do estado dos processos da organização e de um modelo de referência que estabeleça estágios de melhoria. Uma avaliação geralmente se baseia em um modelo de processo de software que fornece um roteiro para melhorias.

Portanto, seria um grande avanço para a comunidade de engenharia de software a criação de uma abordagem que permitisse descrever as características de processos e de modelos de maturidade de software, fornecendo mecanismos para que seja possível: (i) descrever os proces-

sos das organizações; (ii) descrever os modelos de maturidade disponíveis; (iii) relacionar os elementos contidos no processo de software que atendam a requisitos do modelo de maturidade; (iv) avaliar o grau de aderência entre processo e modelo de maturidade; e (v) criar uma base de conhecimento que possa ser compartilhada com os profissionais de engenharia de software.

1.3 Objetivos

Neste trabalho, objetivo é apresentar uma abordagem, que no contexto deste trabalho, é definida como a especificação de um ambiente baseado em metamodelagem que: (i) defina um conjunto de modelos de abstração (metamodelos) de modelos de maturidade de processo de software para apoiar uma metodologia de avaliação/melhoria de processo de software com o objetivo de certificação; e (ii) permita avaliar os processos de uma organização em comparação com um modelo de maturidade por meio de transformações desses metamodelos. Para avaliar a abordagem, apresenta-se um estudo de caso utilizando o modelo MPS.Br e CMMI em que a abordagem proposta é aplicada. Adicionalmente, é apresentado um comparativo entre as limitações existentes nas metodologias de execução de atividades de relacionadas a avaliação e melhoria de processos de software encontradas na literatura e como a abordagem sugere superá-las. Além disso, a abordagem proposta pode ser utilizada como uma especificação (por meio da modelagem conceitual disponibilizada) que possibilita a implementação de uma ferramenta computacional que facilita a experiência do usuário na manipulação dos diversos modelos e processos.

1.4 Metodologia

Para elaboração da abordagem proposta nesta dissertação de mestrado, são executadas as seguintes etapas:

1. Estudos dos conceitos relacionados ao projeto, na qual foi realizada uma pesquisa sobre os temas e os resultados são apresentados no Capítulo 2;
2. Elaboração de um mapeamento sistemático, realizado com o intuito de encontrar trabalhos relacionados que serviriam de base para a elaboração do projeto e a identificação de *gaps* de pesquisa na área de SPA e SPI. Os resultados obtidos são apresentados no Capítulo 3;
3. Realização de um *Survey* com a comunidade de engenharia de software, com o intuito de: (i) identificar quais dificuldades seriam apontadas pela indústria na adoção de modelos de maturidade; (ii) entender a opinião da indústria em relação a parcerias com a academia para apoio em iniciativas de SPA e SPI; e (iii) criar um lista com perfis de empresas e de profissionais que tivessem interesse em participar da avaliação do projeto. Infelizmente, como apresentado no Capítulo 4, não foi possível selecionar empresas que pudessem

participar da avaliação da abordagem. No Capítulo 6, é apresentado um estudo de caso, criado para exemplificar a aplicação da abordagem para certificação no nível G do modelo de maturidade MPS.Br;

4. Identificação das entidades que fazem parte do domínio do problema e da modelagem conceitual dos elementos envolvidos e suas relações. Nessa etapa, várias reuniões de projeto foram realizadas em conjunto com a orientadora do projeto e outros alunos de pós-graduação com experiência em engenharia de software para avaliar as soluções propostas e a modelagem realizada. Por fim, o modelo alcançado foi largamente debatido, prevendo possíveis situações do dia-a-dia dos usuários da abordagem e sua aderência no contexto, principalmente, de SPA e SPI.

1.5 Organização

Neste capítulo, foi apresentado o contexto no qual este trabalho de pesquisa está inserido, bem como a motivação e os objetivos.

O restante deste trabalho está organizado da seguinte forma. No Capítulo 2, é apresentada a revisão da literatura acerca dos conceitos relacionados com o tema. No Capítulo 3, é detalhado o mapeamento sistemático realizado em relação às questões de pesquisas levantadas neste trabalho. No Capítulo 4, são apresentados os resultados obtidos na aplicação de um *Survey* com profissionais da indústria de software, no qual foram coletados dados do perfil de empresas e de profissionais que trabalham com modelos de maturidade e suas dificuldades e suas experiências na implementação dos processos. No Capítulo 5, é apresentada a abordagem proposta, descrevendo a metodologia utilizada, o conjunto de metamodelos criados, as formas de avaliação e seus algoritmos. No Capítulo 6, é apresentado um estudo de caso mostrando a utilização da abordagem proposta. No Capítulo 7, são apresentados as conclusões, as limitações e os trabalhos futuros identificados.

FUNDAMENTAÇÃO TEÓRICA E TÉCNICA

2.1 Considerações Iniciais

Desenvolver software com qualidade é um problema conhecido pela indústria de software há muito tempo. Segundo [Sommerville \(2011\)](#), os problemas com qualidade de software começaram na década de 1960 com o desenvolvimento do primeiro grande sistema de software e tais problemas continuaram a incomodar os praticantes de engenharia de software até os dias atuais com problemas semelhantes aos daquela época, softwares lentos, de pouca confiabilidade e difíceis de dar manutenção e de reutilizar.

Em resposta a esses problemas, a indústria de software passou a adotar técnicas de controle de qualidade vindas da indústria manufatureira, utilizando de técnicas formais de gerenciamento de qualidade em conjunto com o desenvolvimento de novas tecnologias e abordagens mais apuradas de teste ([SOMMERVILLE, 2011](#)).

Sendo assim, além da necessidade de garantir a qualidade do software adotando técnicas formais de verificação (atividades de teste), foi de grande importância a definição de processos de software para definir um conjunto de tarefas bem estruturadas e sequenciais, que visam contribuir para alcançar os objetivos do projeto. Analogamente, pode-se concluir que o processo de software estrutura o conhecimento para ser transformado em um software de qualidade. O processo de software define a abordagem a ser utilizada quando o software é elaborado, porém a engenharia de software inclui algumas analogias que constituem o processo, como métodos, técnicas e ferramentas ([PRESSMAN, 2014](#)).

Entretanto, a aplicação de conceitos de engenharia de software no processo de desenvolvimento de software de uma organização tem por objetivo garantir a qualidade dos produtos produzidos. As tarefas de definição, de execução e de acompanhamento de um processo de software demandam recursos (humanos e de máquinas) e, conseqüentemente, consiste em custos para empresa. Nem sempre a organização possui uma equipe madura o suficiente para realizar

tais tarefas, o que leva a implementação de um modelo de maturidade de processo na organização.

Os modelos de maturidade de software têm como objetivo orientar as organizações a tornar o seu processo de software maduro, de acordo com pré-requisitos definidos pelo modelo. Quando satisfeitos esses pré-requisitos, é concedida à organização uma certificação mediante avaliação da aplicação do modelo.

Com o avanço das técnicas em engenharia de software, uma nova abordagem vem sendo difundida, apesar de ser utilizada despercebidamente e intuitivamente dentro da indústria de software, a *Engenharia Orientada a Modelos* (MDE, *Model Driven Engineering*). A MDE é uma disciplina baseada na utilização de modelos de desenvolvimento de software por meio de transformações desses modelos até a produção de código executável. A idéia de transformações de modelos, a modelagem e o modelo são a base de um conjunto de métodos de desenvolvimento de software conhecidos como *Desenvolvimento Orientado a Modelos* (MDD, *Model Driven Development*) (MUSAT *et al.*, 2010; SELIC, 2003; SCHMIDT, 2006). Os conceitos de MDE são utilizados para o desenvolvimento da proposta deste trabalho conforme descrito no Capítulo 5.

Neste capítulo, são apresentados os principais conceitos relacionados à qualidade de software, bem como a diferença entre qualidade de software e qualidade de produto, com o intuito de esclarecer as diferenças de definição. Também, são apresentados os ciclos de vida de software mais conhecidos e utilizados, bem como as novas tendências das metodologias ágeis. O objetivo é ilustrar, de forma mais específica, as diferentes abordagens entre os métodos clássicos de ciclo de vida em contraste com as propostas das metodologias ágeis. Em seguida, são apresentados os modelos de maturidade mais difundidos na indústria de software nos ambientes nacionais e internacionais para mostrar as diferenças entre eles e salientar a importância de representá-los de forma genérica para facilitar a adoção de múltiplos modelos em organizações com diferentes necessidades de processos e maturidades organizacionais. Por fim, são apresentados os conceitos de forma mais aprofundada relacionados ao MDE para ilustrar as possibilidades de aplicação no contexto de representação de diferentes modelos de referência de processos e modelos de maturidade.

2.2 Qualidade de Software

Essa disciplina da engenharia de software visa acompanhar e auditar o processo de desenvolvimento de software do ponto de vista da qualidade do produto de software em desenvolvimento. Consiste em um conjunto de funções para auditar e relatar a avaliação realizada sobre a efetividade e a completeza das atividades de controle de qualidade. Sua principal meta é fornecer, à gerência do projeto, os indicadores necessários para que fique ciente da qualidade do produto desenvolvido. Casos os indicadores fornecidos identifiquem problemas, cabe à gerência tomar medidas de correção dos problemas e aplicar os recursos necessários para resolução destes problemas (PRESSMAN, 2014).

De acordo com [Sommerville \(2011\)](#), o gerenciamento da qualidade de software possui três pontos de atenção, sendo eles:

- Na esfera organizacional, o gerenciamento de qualidade possui como objetivo a definição de processos e de padrões organizacionais que conduzam a um software de alta qualidade. Dessa forma, fica sob responsabilidade da equipe de qualidade a definição de processos de desenvolvimento, os padrões a serem adotados no software e de toda a documentação relacionada (requisitos de sistema, planos de projeto e código fonte);
- Na esfera de projeto, deve ser garantida pela equipe de qualidade a aplicação dos processos específicos de qualidade, visando à garantia de que os processos definidos foram executados e que os produtos de trabalho produzidos estejam em conformidade com os padrões aplicáveis ao projeto;
- Ainda na esfera de projeto, o gerenciamento de qualidade possui como objetivo o estabelecimento de um plano de qualidade que defina as metas de qualidade para o projeto, quais processos e padrões deverão ser aplicados.

A implementação da garantia de qualidade em uma organização pode ser realizada a partir da criação de um grupo responsável por colher os indicadores de qualidade a partir de algumas atividades executadas durante o processo de desenvolvimento. Esse grupo é denominado SQA (*Software Quality Assurance*, Garantia da Qualidade de Software) e possui o seguinte papel dentro da organização ([PRESSMAN, 2014](#)):

- Elaborar um plano de SQA para um projeto;
- Participar no desenvolvimento da descrição do processo de software do projeto;
- Revisar as atividades de engenharia de software para validar a execução do processo de software definido;
- Garantir que os desvios do processo foram devidamente documentados e solucionados;
- Reportar à gerência do projeto todos os itens de não conformidade encontrados na execução do processo de software definido;
- Realizar atividades que auxiliam a gerência na coleta e na análise de métricas de software.

Mesmo estabelecendo processos e padrões de qualidade, garantir que um produto atenda a todas as expectativas de seu público alvo é uma atividade que demanda tempo, revisão e, ainda assim, está sujeito a mudanças a qualquer momento de seu desenvolvimento. Sendo assim, é difícil criar especificações de software completas e precisas, o que está sempre exposto ao risco de diferentes interpretações por parte de programadores e de usuários com relação aos requisitos

especificados. Diferentes pontos de vista acabam sendo incorporados em um projeto de software, o que muitas vezes causa a impressão por parte de alguns indivíduos envolvidos no projeto que suas solicitações não foram incorporadas ao software e, por conseguinte, o classificam como de má qualidade. Dessa forma, pode ser afirmado que a qualidade de software é um processo subjetivo, em que a equipe de gerenciamento de qualidade deve usar bom senso para atestar que um nível aceitável de qualidade foi alcançado no projeto (SOMMERVILLE, 2011).

Existe um pressuposto do gerenciamento de qualidade que alega que a qualidade do software estaria diretamente relacionada com a qualidade do processo de desenvolvimento utilizado. Esse conceito vem da indústria manufatureira, em que a qualidade do produto está fortemente relacionada com o processo de produção. Pode-se dizer que haja certa influência entre ambos, entretanto um processo de fabricação exige somente a configuração e a calibragem de máquinas envolvidas na produção, garantindo uma linha de produção de qualidade. Porém, produtos de software não são produzidos de forma automática, mas criados a partir do conhecimento sobre um domínio específico (SOMMERVILLE, 2011). Em seu livro, Sommerville (2011) afirma que não existem dúvidas de que o processo de desenvolvimento de software exerça influência sobre a qualidade do software produzido e que a utilização de técnicas de gerenciamento e de garantia de qualidade levem a produção de software com menos defeitos.

Outro ponto importante a ser esclarecido é a diferença entre os conceitos de qualidade de processo e de produto. A qualidade de produto consiste na aplicação de padrões de documentos, tais como, a estrutura dos documentos, os padrões de documentação, o cabeçalho padrão de classes de objetos e os padrões de codificação. A qualidade de processo está associada à definição dos processos organizacionais que devem ser seguidos durante o desenvolvimento de software. Os processos devem encapsular boas práticas e padrões de processo e incluir definições de especificação, de projeto e processos de validação, de ferramentas de suporte ao processo e de descrição dos documentos que devem ser produzidos durante a execução desses processos (SOMMERVILLE, 2011).

No caso da qualidade de produtos de software, pode-se citar a norma ISO/IEC 25.010 que se enquadra no modelo de qualidade das normas da família ISO/IEC 9000 (a norma brasileira correspondente é a NBR ISO/IEC 13.596). A norma ISO/IEC 25.010 estabelece um modelo de qualidade composto de três componentes, sendo eles: (i) **Processo de desenvolvimento**, cuja qualidade afeta a qualidade do produto de software gerado e é influenciado pela natureza do produto desenvolvido; (ii) **Produto**, compreendendo os atributos de qualidade do produto de software, divididos entre atributos internos e atributos externos, que se diferenciam pela forma como são aferidos (interna ou externamente ao produto de software) e em conjunto compõem a qualidade do produto de software em si; e (iii) **Qualidade em uso**, que consiste na aferição da qualidade do software em cada contexto específico de usuário, ou seja, a qualidade percebida pelo usuário. Para a norma ISO/IEC 25.010, a qualidade externa e interna do software é medida a partir de características de qualidade, tais como, funcionalidade, confiabilidade, usabilidade,

eficiência, manutenibilidade e portabilidade (ISO/IEC, 2001).

Por outro lado, existe a norma ISO/IEC 12.207 que define um processo de desenvolvimento de software. Ela tem como objetivo principal estabelecer uma estrutura comum para os processos de ciclo de vida e de desenvolvimento de software visando auxiliar as organizações a compreenderem os componentes presentes na aquisição e no fornecimento de software e, assim, conseguirem firmar contratos e executarem projetos de forma mais eficaz. Essa norma estabelece uma arquitetura de alto nível do ciclo de vida de software construída a partir de um conjunto de processos e de seus inter-relacionamentos. Os processos são descritos em nível de propósito/saídas e em termos de atividades, não possuindo ligação com métodos, ferramentas, treinamentos, métricas ou tecnologias utilizadas. Essa não ligação é útil para permitir que a norma seja utilizada mundialmente e possa acompanhar a evolução da engenharia de software nas diversas culturas organizacionais. Os processos da ISO/IEC 12.207 são modulares, ou seja, são fortemente coesos e fracamente acoplados. Isto significa que as partes de um processo são fortemente relacionadas, mas a quantidade de interfaces entre os processos é mínima. Os processos são agrupados, por uma questão de organização de acordo com a sua natureza, ou seja, o seu objetivo principal no ciclo de vida de software. Esse agrupamento resultou em 4 diferentes classes de processos, que são: (i) **Processos fundamentais**; (ii) **Processo de apoio**; (iii) **Processos organizacionais**; (iv) **Processos de Reúso de Software**; e (v) **Processos de adaptação** (ISO/IEC, 2008).

2.3 Modelos de Maturidade

Atualmente, a preocupação com a qualidade do software produzido é um fator importante para o plano estratégico das empresas de TI que utilizam da Garantia da Qualidade como um diferencial de mercado para a captação e manutenção de usuários de seus produtos (SOFTEX, 2016).

Nesse contexto, a adoção de modelos de maturidade para a melhoria de processos organizacionais vem sendo uma prática cada vez mais comum entre as empresas de tecnologia, que buscam nesses modelos um guia para a definição de seus processos e a garantia de que, ao aplicar as práticas definidas, seus produtos possuirão ganho de qualidade e de produtividade (SOFTEX, 2016).

Dessa forma, neste trabalho, são descritos alguns dos principais modelos de maturidade de processos de software, como o modelo MPS.Br, que foi criado com foco nas micro, pequenas e médias empresas de tecnologia do Brasil. Também, é descrito o modelo CMMI nesta seção.

2.3.1 CMMI

O modelo CMMI (*Capability Maturity Model Integration*, Modelo de Maturidade em Capacitação - Integração), foi desenvolvido pela SEI (*Software Engineering Institute*, Instituto de Engenharia de Software) e é definido como um metamodelo de processo baseado em um

conjunto de capacidades de engenharia de software que devem estar presentes nos processos de desenvolvimento à medida que diferentes níveis de capacidade e maturidade são alcançados pelas empresas (PRESSMAN, 2014). O CMMI-DEV, que consiste da versão do modelo para desenvolvimento de software, foi criado a partir da definição do modelo SW-CMM¹ a pedido do Departamento de Defesa dos Estados Unidos pela SEI². Em 1991, começaram desenvolver CMM's para diversas disciplinas, entre elas a Engenharia de Sistemas, Engenharia de Software, Aquisição de Software, Gerência e Desenvolvimento da Força de Trabalho e Desenvolvimento Integrado do Processo e do Produto. O CMMI surgiu para resolver o problema da utilização de vários modelos e é o resultado da evolução do SW-CMM, SECM³ e IPD-CMM⁴ sendo considerado o sucessor desses modelos (SOFTEX, 2016). O modelo CMMI pode ser classificado como:

- um Modelo Descritivo, pois são descritos atributos essenciais (ou chaves) que caracterizam uma organização em um nível de maturidade particular;
- um Modelo Normativo, pois as práticas detalhadas caracterizam os tipos normais de comportamento que se espera em uma organização que desenvolve projetos de grande escala.

O modelo CMMI é abstrato, pois não determina como o processo de software é implementado pela organização. Também descreve o que normalmente se espera em um processo de software e não como o processo é implementado. Uma característica do CMMI é a possibilidade de implementação do modelo de duas formas (SEI, 2013):

- **Staged**, nessa representação do CMMI o modelo disponibiliza uma sequência pré-determinada para melhoria baseada em áreas de processos em estágios, pois cada estágio serve de base para o próximo sendo caracterizado por *Níveis de Maturidade (Maturity Levels)*. Nessa representação, a maturidade é medida por um conjunto de processos. Dessa forma, é necessário que todos os processos atinjam um nível de maturidade para que a empresa seja certificada em um nível estabelecido pelo modelo;
- **Continuous**, possibilita a organização utilizar a ordem de melhoria que melhor atende os objetivos de negócio da empresa, sendo caracterizado por *Níveis de Capacidade (Capability Levels)*. Nessa representação, a capacidade é medida por processos separadamente, em que é possível ter um processo em um nível e outro processo em outro nível, variando de acordo com as necessidades estratégicas da empresa.

¹ Software Capability Maturity Model

² Software Engineering Institute

³ System Engineering Capability Model

⁴ Integrated Product Development Capability Maturity Model

Os níveis de maturidade do CMMI e as áreas de processo de cada nível para a representação *Staged* (SEI, 2013) são: (i) Nível 1 (Incompleto ou inicial) — não possui áreas de processo; (ii) Nível 2 (Gerenciado ou gerido) — define as áreas de processo Acompanhamento de Projeto de Software, Garantia da Qualidade de Software de Processo e Produto, Gerenciamento de Configuração, Gerenciamento de Requisitos, Planejamento de Projeto, Gerenciamento de Acordo com Fornecedor e Medição e Análise; (iii) Nível 3 (Definido) — define as áreas de processo Desenvolvimento de Requisitos, Solução Técnica, Integração de Produto, Verificação, Validação, Foco de Processo Organizacional, Definição de Processo Organizacional, Treinamento Organizacional, Gerenciamento Integrado de Projeto, Gerenciamento de Riscos e Análise de Decisão e Resolução; (iv) Nível 4 (Quantitativamente gerenciado) — define as áreas de processo Desempenho de Processo e Organizacional e Gerenciamento Quantitativo de Projeto; (v) Nível 5 (Otimizado) — define as áreas de processo Gestão de Processo Organizacional e Análise Causal e Resolução.

Para a representação *Continuous* (SEI, 2013), os níveis de maturidade do CMMI são: (i) Nível 0 (Incompleto) realizado de forma *ad-hoc*; (ii) Nível 1 (Executado) — O processo é executado de modo a completar o trabalho necessário para a execução de um processo; (iii) Nível 2 (Gerenciado/Gerido) — Planejar a execução e confrontar o executado com o que foi planejado; e (iv) Nível 3 (Definido) — O processo é construído sobre as diretrizes do processo existente e é mantido uma descrição do processo. Na Tabela 1, são apresentadas as duas representações do CMMI, SEI (2013).

Tabela 1 – Representação do modelo CMMI na forma *Continuous* e *Staged*.

Nível	Representação <i>Continuous</i>	Representação <i>Staged</i>
Nível 0	Incompleto	
Nível 1	Executado	Inicial
Nível 2	Gerenciado	Gerenciado
Nível 3	Definido	Definido
Nível 4		Gerenciado Quantitativamente
Nível 5		Otimizado

Outra vertente do CMMI é o CMMI-SVC (CMMI *for Services*) lançado em 2009, o modelo da série SEI mais recente. O CMMI-SVC é um modelo para a aplicação de práticas de melhoria de processos em empresas prestadoras de serviços de TI. O modelo serve como um guia para a aplicação das melhores práticas do CMMI em organizações provedoras de serviços. Seus níveis de maturidade seguem a mesma estrutura do CMMI, partindo do nível 1 até o nível 5.

2.3.2 MPS.Br (Melhoria do Processo de Software Brasileiro)

O programa MPS foi criado em dezembro de 2003 e é coordenado pela Associação para Promoção da Excelência do Software Brasileiro (SOFTEX), que possui o apoio do Ministério da Ciência e Tecnologia (MCT), Financiadora de Estudos e Projetos (FINEP), Serviço Brasileiro de

Apoio as Micro e Pequenas Empresas (SEBRAE) e o Banco Interamericano de Desenvolvimento (BID) ([SOFTEX, 2011](#)).

O modelo MPS possui como base técnica as normas ISO/IEC 12.207 e o modelo CMMI-DEV. Sua aplicação é compatível com organizações de diferentes áreas de atuação como adquirentes de software, fábricas de software, fábricas de teste e prestadores de serviços. Na Figura 1, é apresentada a estrutura técnica do modelo MPS.Br. Nela, pode-se notar que as caixas ligadas por linhas azuis representam os modelos de referência e as normas utilizados como base técnica para a criação do modelo MPS.Br. As caixas ligadas por linhas contínuas apresentam os manuais de referência que descrevem o modelo MPS.Br.

Figura 1 – Composição do modelo MPS.Br [Softex \(2016\)](#).

O modelo MPS define níveis de maturidade como uma combinação entre processos e sua capacidade. Essa definição dos processos segue os requisitos para um modelo de referência de processos tal como é apresentado pela norma ISO/IEC 15.504-2, descrevendo seu propósito e os resultados esperados após sua execução. Uma vantagem do modelo consiste nas atividades e nas tarefas necessárias para atender ao propósito e aos resultados esperados dos processos serem definidas pelos usuários do modelo, possibilitando a instituição implementadora analisar as políticas e as culturas da organização e definir processos que não venham com grande impacto cultural para os recursos humanos envolvidos ([SOFTEX, 2016](#)).

Os *níveis de maturidade* são definidos pelo modelo como patamares para evolução de processos, caracterizando estágios de melhoria da implementação dos processos nas organizações. O modelo MPS está dividido em sete níveis de maturidade, sendo eles o nível A (Em Otimização), nível B (Gerenciado Quantitativamente), nível C (Definido), nível D (Largamente Definido),

nível E (Parcialmente Definido), nível F (Gerenciado) e nível G (Parcialmente Gerenciado), conforme definido no MR-MPS-SW (Modelo de Referência - Melhoria do Processo de Software - Software) (SOFTEX, 2016).

A escala de maturidade inicia no nível G e progride até o nível A sendo que, para cada um dos níveis, é atribuído um perfil de processos que visa indicar em quais áreas a organização deve concentrar esforços de melhoria de processos. O progresso e o alcance de um nível de maturidade no modelo MPS são satisfeitos no momento que os processos atendem os propósitos e todos os resultados esperados dos respectivos processos e os resultados esperados dos atributos de processo estabelecidos para aquele nível. Para facilitar a implementação em micro, pequenas e médias empresas, o modelo sugere os sete níveis para possibilitar a implementação dos níveis em prazos mais curtos (SOFTEX, 2016).

Um processo no modelo MPS é definido em termos de propósito e de resultados, no qual o propósito define o objetivo geral a ser atingido durante a execução do processo e os resultados esperados de processo descrevem os resultados a serem obtidos com a efetiva implementação do processo, que podem ser evidenciados por um *produto de trabalho*⁵ ou por uma mudança significativa de estado da organização ao executar o processo (SOFTEX, 2016).

A capacidade do processo é representada por um conjunto de atributos de processo descritos em termos de resultados esperados, nos quais expressam o quão refinado e institucionalizado é o processo executado na organização. À medida que a organização evolui nos níveis de maturidade, um maior nível de capacidade para desempenhar o processo deve ser atingido. Ao atender um atributo de processo (AP), o atendimento aos resultados esperados do processo (RAP) é requerido para todos os processos no nível correspondente ao nível de maturidade. Outro detalhe importante do modelo MPS é os níveis de maturidade serem cumulativos, ou seja, se a organização está no nível F, ela possui o nível de capacidade do nível F que inclui os atributos de processo dos níveis G e F para todos os processos relacionados no nível F, o que também inclui os processos do nível G. Sendo assim, ao passar do nível G para o nível F, os processos do nível G passam a ser executados no nível de capacidade correspondente ao nível superior (SOFTEX, 2016).

2.3.3 Programa MPS - Atributos de Processo e seus Resultados Esperados de Atributos de Processo

Os diferentes níveis de capacidade dos processos estão divididos em nove atributos de processo (AP), e o alcance de cada atributo de processo é avaliado utilizando os respectivos resultados esperados de atributo de processo (RAP), conforme descrito abaixo (SOFTEX, 2016).

- AP 1.1. O processo é executado: esse atributo evidencia o quanto o processo atinge o seu

⁵ Pode ser um documento, e-mail ou artefato de software que comprove a execução e a obtenção do resultado esperado do processo.

propósito;

- AP 2.1. O processo é gerenciado: esse atributo é uma medida do quanto a execução do processo é gerenciada;
- AP 2.2. Os produtos de trabalho do processo são gerenciados: este atributo é uma medida do quanto os produtos de trabalho produzidos pelo processo são gerenciados apropriadamente;
- AP 3.1. O processo é definido: esse atributo é uma medida do quanto um processo padrão é mantido para apoiar a implementação do processo definido;
- AP 3.2. O processo está implementado: este atributo é uma medida do quanto o processo padrão é efetivamente implementado como um processo definido para atingir seus resultados;
- AP 4.1. O processo é medido: esse atributo é uma medida do quanto os resultados de medição são usados para assegurar que a execução do processo atinge os seus objetivos de desempenho e apoia o alcance dos objetivos de negócio definidos;
- AP 4.2. O processo é controlado: esse atributo é uma medida do quanto o processo é controlado estatisticamente para produzir um processo estável, capaz e previsível dentro de limites estabelecidos;
- AP 5.1. O processo é objeto de melhorias e inovações: esse atributo é uma medida do quanto as mudanças no processo são identificadas a partir da análise de defeitos, de problemas e de causas comuns de variação do desempenho e da investigação de enfoques inovadores para a definição e implementação do processo;
- AP 5.2. O processo é otimizado continuamente: esse atributo é uma medida do quanto as mudanças na definição, na gerência e no desempenho do processo têm impacto efetivo para o alcance dos objetivos relevantes de melhoria do processo.

Como o modelo MPS.BR foi baseado na especificação técnica do modelo CMMI, uma analogia entre cada nível de maturidade pode ser realizada. A equivalência entre os níveis de maturidade dos modelos CMMI e MPS.Br pode ser elaborada comparando os dois modelos uma vez que todos os requisitos das áreas de processo do CMMI estão presentes no MPS.Br (SOFTEX, 2016).

Na Tabela 2 são descritos para cada nível de maturidade dentro do modelo MPS quais processos e atributos de processos são definidos para satisfazer aquele nível. No campo “Atributos de Processo (AP)”, são apresentadas as áreas de processo que cada nível de maturidade exige estar em conformidade com o modelo. A cada nível de maturidade, AP’s podem ou não ser adicionados, sendo que os AP’s são acumulados a cada nível de maturidade em que a empresa avançar.

Tabela 2 – Níveis de maturidade, processos e atributos de processos (SOFTEX, 2016)

Nível	Processos	Atributos de Processo (AP)
A	Evolução contínua	Adicionado AP's 5.1 e 5.2
B	Gerência de Projetos - GPR (evolução)	Adicionado AP's 4.1 e 4.2
C	Gerência de Riscos - GRI Desenvolvimento para Reutilização - DRU Gerência de Decisões - GDE	Nenhum AP adicionado
D	Verificação - VER Validação - VAL Projeto e Construção do Produto - PCP Integração do Produto - ITP Desenvolvimento de Requisitos - DRE Gerência de Decisões - GDE	Nenhum AP adicionado
E	Gerência de Projetos - GPR (evolução) Gerência de Reutilização - GRU Gerência de Recursos Humanos - GRH Definição do Processo Organizacional - DFP Avaliação e Melhoria do Processo Organizacional - AMP	Adicionado AP's 3.1 e 3.2
F	Medição - MED Garantia da Qualidade - GQA Gerência de Portfólio de Projetos - GPP Gerência de Configuração - GCO Aquisição - AQU	Adicionado AP 2.2
G	Gerência de Requisitos - GRE Gerência de Projetos - GPR	AP 1.1 e AP 2.1

2.3.4 Base Técnica MPS.Br

Nesta subseção é descrita a norma ISO/IEC 12.207 que foi utilizada como alicerce para a criação do modelo MPS.Br. Na Seção 2.3.1 foi apresentado o modelo CMMI que também foi utilizado como base técnica. A norma ISO/IEC 12.207 foi criada pela ISO - *International Organization for Standardization* e o IEC *International Electrotechnical Commission* em um esforço conjunto (SOFTEX, 2016).

A norma foi proposta em 1988 e publicada em agosto de 1995 como uma Norma Internacional; em 1998, foi publicada a sua versão brasileira que tem o mesmo nome que a internacional, somente acrescida das iniciais NBR. Entre outubro de 2002 e 2004, foram realizadas atualizações na Norma Internacional ISO/IEC 12.207, denominadas de emendas 1 e 2, nas quais foram inseridas diversas melhorias. Tais melhorias criaram novos processos ou expandiram o escopo de alguns existentes. Para cada processo, foram descritos o seu propósito e os resultados esperados; para os novos processos, foram definidas suas atividades e suas tarefas. Essas modificações tinham o objetivo de representar a evolução da Engenharia de Software e as necessidades vivenciadas pelos usuários da norma e de harmonizar com a série ISO/IEC 15.504 (SOFTEX, 2016).

Novamente, em 2008, a norma sofreu alterações e foi reformulada, incorporando as melhorias que já apareciam nas emendas 1 e 2 e harmonizando sua estrutura a Norma Internacional ISO/IEC 15288. A norma ISO/IEC 12.207:2008 também foi publicada como padrão IEEE e

como norma brasileira (SOFTEX, 2016).

Em linhas gerais, a norma estabelece uma arquitetura comum para o ciclo de vida de processos de software com uma terminologia bem definida, contendo processos, atividades e tarefas que devem ser aplicadas durante o fornecimento, a aquisição, o desenvolvimento, a operação, a manutenção e o descarte de produtos de software aplicando-se também a aquisição de sistemas, produtos e de serviços (SOFTEX, 2016).

2.4 Engenharia Orientada a Modelos

De acordo com Sommerville (2011) apud Kent (2002), Schmidt (2006), a engenharia orientada a modelos é uma metodologia de desenvolvimento de software na qual a principal saída do processo de desenvolvimento é os modelos que representam o produto de software que se quer desenvolver e não o software executável em si. Nessa metodologia, os produtos de software a serem executados são gerados automaticamente a partir dos modelos que o representam em suas diferentes instâncias. Outra definição encontrada para *Model Driven Engineering* (MDE) consiste na ideia de uma disciplina baseada no uso de modelos para o desenvolvimento de software por meio de transformações Musat *et al.* (2010) apud Beydeda, Book e Gruhn (2005) e a utilização de modelos, de transformações de modelagem e de modelos são a base de um conjunto de abordagens de desenvolvimento de software conhecidos como *Model Driven Development* (MDD) Musat *et al.* (2010) apud Selic (2003), Schmidt (2006).

A engenharia orientada a modelos surgiu nos conceitos de arquitetura orientada a modelos (MDA *Model Driven Architecture*) proposta pela *Object Managment Group* (OMG), em 2001, com o objetivo de se tornar um novo paradigma de desenvolvimento de software. Os conceitos de arquitetura e de engenharia orientada a modelos muitas vezes são vistos como atividades similares, entretanto, de acordo com (BRAMBILLA; CABOT; WIMMER, 2012) a MDE possui uma abrangência maior que a MDA, sendo que a MDA concentra-se nos estágios de projeto e de implementação do processo de desenvolvimento de software e a MDE possui como foco todos os aspectos do processo de engenharia de software. Dessa forma, as disciplinas de engenharia de requisitos, de processos de software e de testes seriam baseados em modelos e fariam parte do contexto de MDE e não de MDA (BRAMBILLA; CABOT; WIMMER, 2012).

Modelos são fundamentais para a compreensão e disseminação de conhecimentos sobre sistemas de software complexos. MDE pode ser concebido como uma ferramenta para fazer essa suposição de uma forma concreta de trabalhar e de pensar, transformando os modelos artefatos de destaque em engenharia de software. Em razão dos diversos possíveis usos para MDE, seu principal objetivo é definir a engenharia de software pela definição de modelos, de transformações e de suas combinações dentro de um processo de desenvolvimento de software (BRAMBILLA; CABOT; WIMMER, 2012).

Dessa forma, a MDE pode ser definida como uma metodologia composta por conceitos,

notações, processos, papéis e ferramentas. Conceitos são representados pelos componentes que constroem a metodologia, abrangendo artefatos de linguagens a atores. Notações são a forma com que os conceitos são representados, ou seja, as linguagens utilizadas na metodologia. Processos e papéis são as atividades que conduzem para o produto final. Papéis e seus controles e coordenadas são afirmações em propriedades desejadas (corretitude, consistência, etc) dos produtos ou dos processos. Ferramentas são aplicações que facilitam a execução das atividades em suas respectivas coordenadas, fornecendo cobertura ao processo de produção e de suporte ao desenvolvedor na utilização das notações (BRAMBILLA; CABOT; WIMMER, 2012). Para os paradigmas de desenvolvimento tradicionais, é possível deduzir que a combinação entre *Algoritmos* e *Estruturas de Dados* produzem o *Software* desejado; na engenharia de software orientada a modelos, a fórmula para a criação de software pode ser representada como combinação de *Modelos* e *Transformações* que produzem o *Software* desejado (BRAMBILLA; CABOT; WIMMER, 2012).

Em MDE, o domínio do problema é definido com o campo ou a área de conhecimento que precisa ser examinado para resolver um problema. O modelo de domínio é o modelo conceitual do domínio do problema que representam contextos de trabalho específicos para a especificação, a implementação e a implantação de software. Para isso, são criadas linguagens de domínio específico (DSL *Domain Specific Language*). Essas linguagens são projetadas especificamente para um domínio ou contexto; as DSL têm sido amplamente utilizadas em ciência da computação. Para ser possível representar os modelos de si mesmos como instâncias de alguns modelos mais abstratos, foram criados os metamodelos, um nível a mais de abstração, com destaque para as propriedades que representam o próprio modelo. Metamodelos podem ser usados para definir novas linguagens e definir novas propriedades ou características da informação existente (metadados) (BRAMBILLA; CABOT; WIMMER, 2012).

MDE fornece linguagens adequadas para a definição de regras de transformação do modelo; as regras podem ser escritas manualmente a partir do zero por um desenvolvedor ou podem ser definidas como uma especificação refinada de uma existente. Alternativamente, as próprias transformações podem ser produzidas automaticamente fora de algumas regras de mapeamento de alto nível entre os modelos e definir um mapeamento entre elementos de um modelo de elementos para outro. O processo de construção do software baseia-se na transformação de modelos. As transformações podem ocorrer de modelos para modelos e de modelos para texto/código (BRAMBILLA; CABOT; WIMMER, 2012).

2.4.1 Processos MDE

Em qualquer mudança de tecnologia, aspectos organizacionais, gerenciais e sociais são os principais motivos do fracasso nas organizações. A adoção de MDE sem considerar esses aspectos é um caminho certo para o fracasso em organizações de software. Alguns conselhos de senso comum podem ser considerados pela gestão ao tomar a decisão de adotar MDE em seus

projetos (BRAMBILLA; CABOT; WIMMER, 2012):

- Primeiro projeto MDE não deve ser crítico;
- Certifique-se que a gestão está comprometida com a decisão;
- Apoio ao longo dos problemas que aparecerão no projeto;
- Ter alguém com experiência na equipe;
- Comece pequeno, com um projeto piloto e crescer a partir deste.

Segundo Brambilla, Cabot e Wimmer (2012), alguns fatores podem ser esperados como perdas e ganhos na adoção da modelagem de software:

- Modelagem introduz novas tarefas e funções no desenvolvimento do processo;
- Algumas são “dolorosas” (agora há mais trabalho a ser feito);
- Algumas proporcionam ganhos (a manutenção é mais fácil com os modelos);
- Se as pessoas que sentem a “dor” e as que sentem o ganho não fazem parte da mesma equipe:
 - Cuidado com a motivação e os problemas de percepção sobre o uso de modelagem;
 - Reconheça o trabalho doloroso.

No contexto de processo de desenvolvimento de software, a MDE pode ser identificada nos ciclos de vida mais antigos e utilizados, por exemplo, Cascata, que pode ser considerado como baseado em modelos. Os modelos são tipicamente utilizados em cada fase do processo: modelos de requisitos, modelos de análise, modelos de *design*, modelos de implantação, etc. Dessa forma, qual a contribuição de MDE no contexto do processo de desenvolvimento de software? Segundo Brambilla, Cabot e Wimmer (2012), a principal contribuição é a transição entre **baseado em modelos** para **orientado a modelos** e a possibilidade/oportunidade de (semi) automatizar as transições entre as diferentes fases do processo de desenvolvimento. Na Figura 2 (BRAMBILLA; CABOT; WIMMER, 2012), são representadas as transformações de modelos que ocorrem durante o ciclo de vida do projeto de software. Nela, pode-se notar que a partir de um modelo de origem, são aplicadas transformações (refinamentos) que resultam em modelos de domínio específicos dentro do desenvolvimento de software. Por fim, é aplicada uma transformação em um conjunto de modelos que represente os diversos estados do software resultando no código fonte desejado.

Figura 2 – Transformações de modelos ao longo do ciclo de vida (BRAMBILLA; CABOT; WIMMER, 2012)

2.5 SPEM - Software & Systems Process Engineering Metamodel

Baseado na UML 2.0, o *framework* SPEM foi definido com o objetivo de fornecer um pacote de infraestrutura para SPEM (2015): (i) fornecer uma representação padronizada e bibliotecas gerenciadas de conteúdo método reutilizável; (ii) apoiar o desenvolvimento sistemático, a gestão e o crescimento dos processos de desenvolvimento; (iii) apoiar a implantação de apenas o conteúdo de método e de processo necessário para definir configurações de processos e seus métodos de conteúdo; e (iv) apoiar a promulgação de um processo para projetos de desenvolvimento.

O *framework* possui uma arquitetura composta por sete pacotes de metamodelos, sendo que, para a sua implementação, somente o pacote *Core* é obrigatório, tornando totalmente customizável a aplicação dos pacotes convenientes ao domínio em que se pretende aplicar o modelo. No *framework*, estão disponíveis os pacotes *MethodPlugin*, *ProcessWithMethods*, *ProcessStructure*, *ProcessBehavior*, *MethodContent*, *ManagedContent* e *Core*. Na Figura 3, é ilustrada a estrutura de pacotes do *framework* SPEM. Em seguida, em uma listagem, são apresentadas as características de cada pacote do *framework* SPEM:

1. *Core*: é o núcleo do metamodelo e contém classes e abstrações que constroem a base para as classes nos outros pacotes. Dessa forma, as classes comuns entre os níveis de conformidade estão presentes neste pacote;
2. *ProcessStructure*: nesse pacote, está definida a base para os modelos de processo, fornecendo suporte à criação de processos simples e flexíveis;

Figura 3 – Representação da estrutura do *framework* SPEM [SPEM \(2015\)](#).

3. *ProcessBehavior*: os conceitos concentrados nesse pacote visam representar um processo como uma estrutura de divisão estática, permitindo aninhamentos de atividades e definição das dependências predecessoras entre eles. O pacote permite estender essas estruturas com modelos comportamentais. No entanto, ele não define a sua própria abordagem de modelagem de comportamento, mas fornece *links* para modelos de comportamento externamente definidos, permitindo a reutilização dessas abordagens de outras especificações OMG ou de terceiros;
4. *ManagedContent*: os processos de desenvolvimento são, em muitos casos, representados como modelos e documentados e gerenciados como descrições de linguagem natural. Para muitas abordagens de desenvolvimento de software, documentação humana fornece orientação compreensível para as melhores práticas de desenvolvimento podendo ser mais importantes do que modelos precisos. Em outras palavras, a praticidade de técnicas e de métodos expressa com essas práticas é, em muitos casos, percebida para proporcionar maior valor do que a obediência estrita a um processo formalmente definido. As razões para isso consistem na percepção existente de que abordagens de desenvolvimento de software são um processo criativo que exige constante reavaliação e aprovação, em vez de uma sequência estrita de atividades, por exemplo, para as equipes ágeis de desenvolvimento;
5. *MethodContent*: esse pacote fornece conceitos aos usuários e às organizações para construir uma base de conhecimento de desenvolvimento independente de quaisquer processos específicos e projetos de desenvolvimento. Compreende conteúdo de explicações passo-a-passo textuais, descrevendo como objetivos específicos de desenvolvimento são alcançados,

por quais papéis, com quais recursos e quais são os resultados esperados;

6. *ProcessWithMethods*: nesse pacote, são definidas as estruturas existentes para processos definidos com conceitos do pacote *ProcessStructure* e com instâncias do pacote *MethodContent* que integra os dois conceitos;
7. *MethodPlugin*: esse pacote introduz conceitos de *design* e gestão sustentável, em grande escala, reutilizável e bibliotecas configuráveis ou repositórios de conteúdo e de processos. Os conceitos introduzidos nesse pacote permitem organizar diferentes partes de uma biblioteca com base em diferentes níveis de abstração semelhantes às arquiteturas de software em camadas.

Pensando em usuários que não necessitam de todas as funções previstas pelo *framework* SPEM e que desejam modelar/documentar processos de forma mais simples, foram disponibilizados perfis de implantação diferentes. Os perfis de implantação foram divididos em três categorias: (i) o *SPEM Complete*, recomendado para usuários que precisam de todos os recursos definidos nesse meta-modelo, sendo destinado a fornecedores de ferramentas *CASE* que querem fornecer suporte para bibliotecas de grande escala de conteúdo textual e modelos de processos reutilizáveis; (ii) o *SPEM Process with Behavior and Content*, recomendado para usuários que querem se concentrar nos aspectos de modelagem de SPEM, destina-se ao público que se concentra em um processo de cada vez e trabalha principalmente em representações do modelo de processo, tais como estruturas de divisão de trabalho ou diagramas de fluxo de trabalho; e (iii) o *SPEM Method Content*, recomendado para usuários que se concentram principalmente na gestão da documentação de descrições de métodos de desenvolvimento, técnicas e melhores práticas, sendo considerado ideal para edição de livros e autores de relatórios técnicos que poderiam aumentar sua publicação de métodos de desenvolvimento, técnica ou melhores práticas, com a documentação com base em conteúdo SPEM semi-formal fornecendo um formato reutilizável e intercambiáveis.

2.6 Ferramentas de Apoio Utilizadas

A utilização de metamodelagem foi adotada por causa da flexibilidade que as transformações de metamodelos fornecem. Com base exatamente neste diferencial — flexibilidade —, foi aplicada a utilização de metamodelos para a representação de modelos de maturidade e de processos. Para desenvolver os metamodelos, foram utilizadas as tecnologias descritas a seguir ([ECLIPSE, 2015](#)):

1. Eclipse EMF: o projeto *EMF* (*Eclipse Modeling Framework*) consiste de um *framework* de modelagem e ambiente de geração de código para o desenvolvimento de ferramentas e outras aplicações baseadas em modelos de dados estruturados. A partir de uma especificação

de modelo descrita por um arquivo *XMI* (*XML Metadata Interchange*), a plataforma EMF fornece ferramentas e suporte em tempo de execução para a produção de um conjunto de classes Java a partir de modelos, juntamente com um conjunto de classes de adaptadores que habilite a criação por linha de comando e editores de seus modelos. O núcleo do EMF (*EMF core*) é um padrão comum para modelos de dados em que tecnologias e *frameworks* são baseados. Isto inclui soluções de servidores, *frameworks* de persistência, *frameworks* de interface de usuário e suporte para transformações;

2. *Ecore*: O núcleo do *EMF framework* inclui um metamodelo (*Ecore*) utilizado para a descrição de modelos e suporte em tempo de execução para modelos incluindo notificações de alterações, suporte à persistência utilizando serialização por meio de um arquivo *XMI* e uma API (*Application Programming Interface*) para reflexão e manipulação de objetos EMF genericamente;
3. Eclipse Mars: a distribuição *Mars* do Eclipse IDE foi utilizada para o desenvolvimento dos projetos *Ecore*, utilizados para a criação dos metamodelos e utilizados para criação dos modelos propostos pela abordagem:
 - a) Projeto *Ecore*: o primeiro passo para a criação de cada pacote de metamodelo definido pela abordagem é a criação de um projeto *Ecore* vazio. A partir disso, são criados os diagramas *Ecore* em que cada entidade identificada no pacote do modelo conceitual é transformada em uma classe e os atributos são mapeados novamente;
 - b) *Model Generator*: em seguida, é criado o *model generator* de cada metamodelo *Ecore*. O *model generator* é utilizado para a criação de um *plug-in* executável do metamodelo *Ecore*. A partir dele, é criado o código fonte do editor utilizado para a modelagem dos modelos a partir dos metamodelos criados;
 - c) *Plug-in Eclipse*: após criados os pacotes de código do editor gráfico do metamodelo criado, são criados os *plug-in* que devem ser instalados no Eclipse para que seja possível criar modelos utilizando a notação definida na metamodelagem. A partir desses disso, é possível criar um arquivo de modelagem no Eclipse que possuirá a extensão do pacote de metamodelos definidos no modelo conceitual;
4. *ATL - A Model Transformation Language*: ATL é uma linguagem de transformação de modelos. No campo da Engenharia Orientada a Modelos (MDE), ATL fornece maneiras de produzir um conjunto de modelos alvo a partir de um conjunto de modelos de origem. Desenvolvido sobre a plataforma Eclipse, o ATL fornece ferramentas de desenvolvimento, que visam facilitar o desenvolvimento de transformações em ATL;
5. *Accelleo*: Acceleo é uma implementação pragmática do *Object Management Group (OMG) MOF (Meta Object Facility)*, que consiste em um Modelo de texto de idioma padrão. O *framework* fornece mecanismos para se criar geradores de código, fazendo uso de modelos como a base para a criação do código-fonte.

2.7 Considerações Finais

Neste capítulo, foram apresentados os conceitos utilizados para o desenvolvimento da proposta descrita nesta dissertação de mestrado. Ações de avaliação e de melhoria de processos de desenvolvimento de software estão intrinsicamente relacionadas à utilização de modelos de maturidade. Sendo assim, os modelos MPS.Br e CMMI foram apresentados para formalizar a estrutura definida por esses modelos, quais os seus requisitos e fundamentos técnicos utilizados em sua criação. Adicionalmente aos modelos de maturidade, foram apresentadas as normas ISO/IEC 12.207 e ISO/IEC 15.504, internacionalmente reconhecidas na âmbito de qualidade de processo de software.

A engenharia de software orientada a modelos é uma metodologia utilizada a algum tempo pela indústria, as primeiras publicações de livros relacionados ao tema datam da década de 80 e vem sendo aplicada de forma intuitiva há algum tempo pela comunidade de engenharia de software, por exemplo, modelos de classes em *UML (Unified Modeling Language)* utilizados pelas ferramentas *CASE (Computer-Aided Software Engineering)* para geração de código fonte é um exemplo comum do uso de MDE. Para o desenvolvimento da proposta, o conceito de MDE exerce um papel central, pois, utilizando a metodologia definida por ela, foram definidos os modelos de abstração de modelos de maturidade e de processos de desenvolvimento de software, utilizados para a avaliação e a melhoria de processos. Complementando o aspecto de modelagem de processos, o *framework* SPEM foi apresentado como uma alternativa disponibilizada pela OMG para a modelagem e a documentação de processos. O modelo disponibilizado pelo SPEM não foi utilizado nesta proposta, entretanto alguns elementos criados na abordagem, presentes nos metamodelos criados (apresentados no Capítulo 5), foram inspirados no SPEM. Por fim, as tecnologias e as ferramentas de apoio utilizadas para o desenvolvimento da abordagem foram apresentadas com o intuito de fornecer uma referência para a reprodução do ambiente em que esse projeto foi elaborado.

MAPEAMENTO SISTEMÁTICO

3.1 Considerações Iniciais

De acordo com [García et al. \(2012\)](#) *apud* [Oktaba e Piattini \(2008\)](#), [Johnson \(2006\)](#), é de conhecimento que, infelizmente, o problema fundamental para muitas empresas é a incapacidade de gerir o seu processo de software. Desde o início da década de 70, tem havido uma demanda em curso de melhores serviços e funções em produtos de software. Mesmo com o desenvolvimento de vários métodos, técnicas e ferramentas, os produtos de software ainda sofrem com os custos excessivos, atrasos na entrega e baixa qualidade. Para satisfazer os vários requisitos para um processo de software, as grandes e empresas têm feito um esforço central no desenvolvimento de atividades voltadas para a **Melhoria de Processos de Software**, mais conhecido pela sigla em inglês *SPI (Software Process Improvement)*.

Como resultado, a SPI tem emergido como uma nova forma de resolver esses problemas. Um indicador disso é a quantidade crescente de iniciativas internacionais relacionadas com a SPI, como o *Capability Maturity Model Integration - Development (CMMI -DEV)*, ISO/IEC 15504:2004, SPICE, IDEAL e ISO/IEC 12207:2008. Estes modelos SPI têm sido apresentados como uma alternativa para aumentar a qualidade dos serviços e dos produtos que as organizações de software fornecem ([GARCÍA et al., 2012](#)).

No entanto, o problema é o elevado custo da implementação, independente do tamanho da empresa [García et al. \(2012\)](#) *apud* [Mondragn \(2006\)](#). Aparentemente, os principais impedimentos para as pequenas empresas, incluindo a gestão de processos de software e atividades, são: há princípios para a forma de realização de tarefas, incluindo os planos e procedimentos, o desempenho de tarefas *ad-hoc* confiando apenas na experiência de profissionais e a quantidade insuficiente de pessoal com competências adequadas.

Com base nas informações expostas, o desafio desta pesquisa é identificar a existência de ferramentas que dêem apoio à avaliação de processos de software e que auxiliem na identificação

de pontos de melhoria de processos com o objetivo de certificação em modelos de maturidade de processos de software como o CMMI e o MPS.Br, conforme discutido no Capítulo 2. Uma forma de fazer o levantamento das ferramentas existentes é por meio de um mapeamento sistemático sobre o assunto.

O mapeamento sistemático descrito neste capítulo foi publicado parcialmente em um artigo (FELONI; BRAGA, 2015). O resultado desse artigo é apresentado com mais detalhes, principalmente em relação aos trabalhos encontrados os quais são descritos com mais profundidade. Assim, na Seção 3.2, é apresentado o planejamento do mapeamento sistemático. Na Seção 3.3, é apresentada a metodologia da condução. Na Seção 3.4, as ferramentas/abordagens identificadas são sumarizadas. Na Seção 3.5, é feita a análise do mapeamento sistemático.

3.2 Planejamento do Mapeamento Sistemático

Conforme discutido na seção anterior, a garantia de qualidade de software pode ser alcançada com o uso de um processo de desenvolvimento de software bem definido institucionalizado na organização e, com esforços suplementares, uma certificação em modelo de qualidade de software pode ser obtido. No entanto, a implementação de atividades de SPI consiste em um processo que exige esforço, experiência, métodos e ferramentas para apoiar a organização durante um processo de certificação.

As ferramentas identificadas para auxiliar atividades de SPI geralmente abordam algumas atividades: (i) coletar e gerir informações gerais sobre a empresa, avaliação de metas e de projetos, processamento e avaliação de itens de gerenciamento; (ii) selecionar processos para revisões; (iii) gerar questionários estáticos analisando aspectos específicos de modelos de qualidade; e (iv) auto-avaliar tarefas atribuídas pelos profissionais de engenharia de software (que requerem a assistência de peritos) (GARCÍA *et al.*, 2012). Dessa forma, foi identificada a necessidade de avaliar as abordagens e as metodologias disponíveis na literatura, definindo quais os modelos de qualidade são mais utilizados e como essa informação é armazenada para reutilização futura.

O objetivo de um mapeamento sistemático é revelar o estado da arte em um tema, observando a quantidade de estudos selecionados, seus tipos, os resultados disponíveis, a frequência de publicações ao longo do tempo, entre outros fatores (PETERSEN *et al.*, 2008). O processo de mapeamento sistemático pode ser descrito como um conjunto de tarefas bem definidas e planejadas de acordo com um protocolo previamente estabelecido (KITCHENHAM *et al.*, 2009), composto das fases planejamento, condução e análise dos resultados, como mostrado nas seções a seguir.

3.2.1 Objetivos da Pesquisa

Neste mapeamento sistemático, o objetivo é identificar ferramentas que forneçam suporte às atividades de SPI. O intuito é verificar quais são os modelos de referência de maturidade

mais citados pela literatura e quais técnicas/abordagens são mais utilizadas em uma tentativa de harmonizar esses modelos. A harmonização de modelos de referência tem o objetivo de tornar as ferramentas independentes de um modelo específico como o CMMI ou o MPS.Br, para alcançar uma certificação e gerir os artefatos produzidos/consumidos ao longo da execução dos processos. Com base nesses objetivos, é possível identificar as seguintes *questões de pesquisa*:

1. Existem ferramentas de apoio para certificação de processos de software?
 - a) Quais são os tipos de abordagens que auxiliam a avaliação e a melhoria de processos de software (SPA e SPI)?
 - b) Essas abordagens permitem a execução da avaliação, considerando um ou mais modelos de qualidade?
 - c) Será que as abordagens permitem a reutilização de processos e modelos de qualidade?
 - d) Que critérios são usados por essas abordagens para sugerir atividades de melhoria de processo?
 - e) Como essas abordagens fazem o mapeamento entre os elementos do processo de software executado e os atributos necessários por modelos de qualidade e suas relações?

De acordo com essas questões, foram extraídas as especificidades dessa revisão, ou seja, a identificação do **PICO** (população, intervenção, comparação e resultados):

- População: processos de desenvolvimento de software e modelos de maturidade;
- Intervenção (o que será observado): técnicas, abordagens e critérios utilizados para avaliação de processos e modelos de maturidade. Desenvolvimento de ferramentas para apoio à implementação de modelos de maturidade e o desenvolvimento de ferramentas para definição de processos para certificação em modelos de maturidade;
- Comparação: comparar as diferentes abordagens e/ou ferramentas para avaliação de processos de software e quais os critérios/indicadores são utilizados para a indicação de atividades de SPI para a obtenção de certificação de modelos de maturidade;
- Resultados: ferramentas que implementem técnicas e critérios utilizados para sugestão de modelos de maturidade.

3.2.2 Critérios de Inclusão e de Exclusão

Diante do contexto da pesquisa, foram definidos cinco critérios de inclusão para verificar a pertinência dos trabalhos selecionados ao escopo do mapeamento sistemático.

- **CI-1:** Trabalhos que descrevam a utilização de ferramentas/métodos para definição de atividades para certificação de processos em modelos de maturidade;
- **CI-2:** Trabalhos que apresentem métodos/critérios para mapear processos entre modelos de maturidade de processo diferentes;
- **CI-3:** Trabalhos que apresentem os principais problemas na implementação de modelos de maturidade;
- **CI-4:** Trabalhos que apresentem definições de modelos de abstração para modelos de maturidade e processos;
- **CI-5:** Trabalhos que apresentem abordagens para a persistência de artefatos de engenharia de software no contexto de reúso.

Para completar e refinar a busca e a análise dos trabalhos, os seguintes critérios de exclusão foram definidos:

- **CE-1:** Trabalhos não relacionados a melhoria de processo de software;
- **CE-2:** Trabalhos que não apresentem ferramentas/técnicas/abordagens de apoio para a avaliação de processo de software;
- **CE-3:** Trabalhos que estejam em idiomas diferentes do inglês e português;
- **CE-4:** Trabalhos que não estejam disponíveis para *download* na web.

3.2.3 Seleção da Fonte de Busca

A fonte de busca escolhida está entre as principais na área computacional e possui grande impacto na área de pesquisa. Para a escolha da fonte, foram observadas as características da cobertura da máquina de busca (conferências, revistas, livros), da atualização do conteúdo, da disponibilidade dos trabalhos e da usabilidade da máquina de busca. Diante dessas características, a máquina de busca consultada foi a *Scopus*. Essa escolha foi feita por causa da quantidade de trabalhos relacionados ao assunto e da base *Scopus* indexar as principais fontes de pesquisa como *IEEE Xplore*, *ACM Digital Library* e *Science Direct*.

3.2.4 Construção da String de Busca

De acordo com os artigos bases para a pesquisa (indicados pelos especialistas) e com o objetivo do mapeamento sistemático, foram definidos os sinônimos para compor a *string* de busca. As palavras-chave que envolvem o núcleo da pesquisa são melhoria do processo de software (*software process improvement*) e avaliação do processo de software (*software process assessment*).

Os termos foram divididos em cinco partes para formar a expressão de busca. Conforme apresentado na Tabela 3, o termo A refere-se ao contexto de engenharia de software, o termo B refere-se às atividades de melhoria do processo de software (*SPI*) e avaliação do processo de software (*SPA*), o termo C refere-se ao método como foram definidas ferramentas/abordagens para a execução de *SPI* e *SPA*, o termo D refere ao contexto dos modelos de maturidade e normas para a avaliação de processos de software e o termo E refere-se as palavras que indicam o objetivo de criação das técnicas apresentadas pelos artigos.

Tabela 3 – Termos utilizados na construção da string de busca.

Termo	Sinônimos
A	“software”, “software engineering”
B	“process improvement”, “process software improvement”, “spi”, “self-assessment”, “assessment”, “process assessment”, “software process assessment”
C	“framework”, “tool”, “tool based”, “tool-based”, “tool oriented”, “tools”, “guideline”, “approach”, “method”, “criterion”, “techniques”, “methodologies”, “tool support”, “meta model”, “meta-model”, “metamodel”, “abstraction”, “abstract model”, “abstract models”
D	“maturity”, “maturity model”, “maturity models”, “capability model”, “capability models”, “maturity capability model”, “maturity capability models”, “software process standard”, “scampi”, “cmmi”, “spice”, “iso/iec 15504”, “iso/iec 20000”, “iso/iec 12207”, “cmmi-dev”
E	“aid”, “help”, “support”, “assist”
String	(A AND B AND C AND D AND E)

3.3 Condução do Mapeamento Sistemático

A realização deste mapeamento sistemático ocorreu por meio de quatro fases de análise antes da leitura integral dos documentos extraídos pelo motor de busca. Na primeira fase, foi realizada a identificação dos artigos, nas quais foram identificados 315 artigos; desses, 88 foram removidos pois estavam duplicados. Na segunda fase, 227 artigos foram analisados por leitura do *abstract* e título, resultando em 80 trabalhos selecionados para a próxima fase. A terceira fase consistiu na leitura da introdução e da conclusão dos 80 trabalhos selecionados na fase anterior, o que resultou na seleção de 17 trabalhos para leitura na íntegra. Finalmente, uma fase adicional foi conduzida para apontar os artigos que representam os principais aspectos identificados em todas as amostras analisadas, que resultou em 6 artigos selecionados entre aqueles selecionados na terceira fase. Na Tabela 4, é apresentada a sequência de passos para a seleção trabalhos relacionados.

Após a última fase da análise dos artigos selecionados, foi possível recuperar algumas estatísticas sobre os critérios de aceitação e de exclusão definidos para este mapeamento sistemático. Na Tabela 5, é apresentada a distribuição dos critérios de aceitação e de exclusão nos trabalhos analisados. O campo “# de Trabalhos” representa a quantidade de artigos analisados que se enquadram nos critérios indicados. Alguns artigos foram registrados em mais de um critério. Nessa Tabela, é fornecida uma visão da distribuição entre os critérios definidos para esse mapeamento sistemático e são destacados quais aspectos estão mais presentes na literatura.

Tabela 4 – Fases do processo de seleção dos trabalhos.

Fase	Análise Realizada	Analizados	Restantes
1	Identificação dos trabalhos no motor de pesquisa	315	227
2	Exclusão dos trabalhos com base no título e no <i>abstract</i>	227	80
3	Exclusão dos trabalhos com base na introdução e conclusão	80	17
4	Seleção de trabalhos que representam os aspectos principais da amostra analisada	17	6

Tabela 5 – Distribuição dos critérios de inclusão e de exclusão.

Critério	Tipo	# de Trabalhos
IC-1	Inclusão	69
IC-2	Inclusão	41
IC-3	Inclusão	14
IC-4	Inclusão	11
IC-5	Inclusão	6
EC-1	Exclusão	184
EC-2	Exclusão	181
EC-3	Exclusão	1
EC-4	Exclusão	9

A maioria dos artigos refere-se ao desenvolvimento e à utilização de ferramentas e *frameworks* para apoiar iniciativas de SPA e de SPI. Alguns estudos sugerem o uso de diferentes abordagens, tais como, a criação de ontologias e uma análise semântica dos artefatos do projeto. O objetivo é avaliar o estado atual de um processo de software, usado para determinar se e quanto cumpre as práticas e os objetivos definidos por um modelo maturidade. Entre os trabalhos selecionados, foram escolhidos 6 (selecionados na quarta fase), que representam uma síntese das abordagens propostas pela academia no esforço para apoiar as iniciativas de SPA e de SPI (Tabela 6). Também, foi possível apresentar uma categorização que divide os esforços encontrados para atividades SPI e SPA na literatura, conforme listado:

1. Ferramenta/Framework: esses trabalhos apresentam diferentes abordagens para a implementação do processo de avaliação. O principal objetivo dos métodos propostos consiste na obtenção de um relatório com as lacunas identificadas na aderência com o modelo de maturidade, sugerindo medidas para satisfazer um nível de maturidade em uma área de processo específico (HOMCHUENCHOM *et al.*, 2012), (MONTONI *et al.*, 2006);
2. Harmonização de modelos de maturidade: em alguns estudos (MUSAT *et al.*, 2010), (JENERS; LICHTER; ROSENKRANZ, 2013), é identificada a necessidade de algumas organizações em adotar modelos de qualidade diferentes para atender as necessidades comerciais específicas. Nesse sentido, alguns estudos propõem abordagens para harmonizar

esses modelos, ou seja, identificar práticas e requisitos semelhantes em modelos de maturidade diferentes. O objetivo é identificar áreas de processos, funções, atividades, artefatos e outros componentes de processo de software semelhantes em diferentes modelos de qualidade, tentando otimizar o processo de definição das atividades de SPI nas organizações. Entre as abordagens usadas, pode-se citar o uso de MDE como a principal ferramenta para definir modelos conceituais que possam representar os modelos de maturidade sem estar ligada a nenhum deles;

3. Repositórios: alguns estudos apresentam iniciativas para criar repositórios para a persistência de artefatos de engenharia de software (ou metadados dos artefatos) que poderiam ser reutilizados no âmbito da SPI e da SPA (PARK; KIM; LEE, 2008), (WANGENHEIM *et al.*, 2011).

Tabela 6 – Artigos selecionados na fase 4.

Categoria	Referência	Título
Tool	(HOMCHUENCHOM <i>et al.</i> , 2012)	SPIALS-II: A light-weight software process improvement self-assessment tool
Tool	(MONTONI <i>et al.</i> , 2006)	Taba workstation: Supporting software process deployment based on CMMI and MR-MPS.BR
Harmonization	(MUSAT <i>et al.</i> , 2010)	MATURE: A model driven based tool to automatically generate a language that supports CMMI process areas specification
Harmonization	(JENERS; LICHTER; ROSENKRANZ, 2013)	Efficient adoption and assessment of multiple process improvement reference models
Repository	(PARK; KIM; LEE, 2008)	Software repository for software process improvement
Repository	(WANGENHEIM <i>et al.</i> , 2011)	Building a Maturity & Capability Model repository

O mapeamento sistemático realizado indica que os trabalhos executados no contexto de SPI e de SPA apresentam deficiência na adoção de técnicas recentes de engenharia de software como a MDE na tentativa de harmonizar modelos de referência em maturidade e em processos de software. A questão de pesquisa trata de uma necessidade atual que busca identificar novas formas de avaliar processos de desenvolvimento de software em execução e de determinar um *roadmap* para a adoção de um modelo de maturidade sem estar de certa forma preso a algum tipo de modelo por causa de ferramentas de apoio não flexibilizarem a escolha do modelo a ser adotado. Na Seção 3.4, são apresentados os trabalhos identificados de acordo com a classificação apresentada anteriormente.

3.4 Síntese dos Trabalhos Identificados

Após identificar os trabalhos relacionados, foi realizada a extração dos dados e das características das ferramentas/abordagens descritas no contexto de SPI e de SPA. Nos trabalhos, foram identificados os modelos de referência mais comumente usados pela literatura como CMMI, ISO/IEC 15504 (SPICE) e a ISO/IEC 12207. Na maioria dos trabalhos, as abordagens utilizavam um modelo de referência como base para o desenvolvimento de ferramentas com o intuito de apoiar a avaliação de atividades de SPI. Como citado anteriormente, existe também uma vertente de trabalhos concentrados na tentativa de harmonização desses modelos de referência com o intuito de possibilitar flexibilização no uso de ferramentas. Entretanto, os trabalhos concentram-se em criar modelos de abstração para diferentes modelos de referência e não fornecem aparato computacional que instancie os modelos e possibilite validar o modelo com algum processo.

Para a sumarização dos trabalhos são apresentados os objetivos, a metodologia, a ferramenta/*framework* criado (ou não), os modelos de referência utilizados, se houve a criação de algum modelo de abstração, se foi realizado algum tipo de validação, os resultados obtidos e os trabalhos futuros.

3.4.1 *SPIALS-II: A light-weight software process improvement self assessment tool*

Nesse trabalho, é levantada a questão para as microempresas de como aplicar um modelo de maturidade de processos, como o CMMI, de forma leve e reduzir custos para realizar a sua aplicação. O *National Electronics and Computer Technology Center* (NECTEC) é uma organização tailandesa de apoio a pequenas e microempresas para desenvolver produtos de alta qualidade. A NECTEC visa à adoção de técnicas ágeis, como Scrum ou XP para resolver problemas conhecidos. Foram estabelecidas abordagens para comparar a complexidade entre processos baseados no modelo CMMI, utilizando diferentes ciclos de vida de desenvolvimento de software como Cascata e Scrum.

O trabalho apresenta o desenvolvimento de uma ferramenta Web com base no modelo CMMI para a realização de atividades de SPA que, na opinião dos autores, pode ser o ponto de partida para a melhoria de processos sem alta demanda com consultorias externas. A ferramenta deve ser utilizada no sentido de identificar qual o estado atual dos processos da organização e determinar quais os *gaps* nos processos para que possam ser melhorados. A ferramenta foi denominada *Software Process Improvement Adaptive Learning System*, SPIALS (Melhoria de Processo de Software Adaptivo à Aprendizagem) e tem o propósito de simplificar o processo de SPA para a identificação de atividades de SPI na organização.

A metodologia de avaliação utilizou as regras de avaliação definidas nos Requisitos de Avaliação para CMMI (*Appraisal Requirements for CMMI*) versão 1.2 e no *Standard CMMI*

Appraisal Method for Process Improvement (SCAMPI) A, versão 1.2 : *Method Definition Document - MDD*. No entanto, a abordagem utilizada é baseada no *Handbook for Conducting Standard CMMI Appraisal Method for Process Improvement (SCAMPI) B and C Appraisals*. Além disso, foi criada uma abordagem baseada em questionários para realização das auto-avaliações chamado *Self-assessment and the CMMI-A: A Guide for Government Program Managers*.

Nesse trabalho, foram identificadas algumas abordagens para fundir o modelo CMMI com métodos ágeis. O modelo *CMMIbyScrum* é focado nas áreas de processo Planejamento de Projeto, Monitoramento e Controle de Projeto e Gestão de Projeto Integrado e é baseado no CMMI utilizando práticas e artefatos mesclados entre o Cascata e o Scrum criados pela NECTEC.

A ferramenta deve ser utilizada por um representante da organização. Os primeiros projetos a serem avaliados devem ser criados na ferramenta; em seguida, os papéis associados aos projetos são atribuídos. Os elementos centrais da auto-avaliação SPIALS são questionários para serem preenchidos pelos representantes da organização. Os dados de entrada necessários são introduzidos e definidas as evidências para os respectivos processos de desenvolvimento de software. O resultado é relatado indicando automaticamente a aceitação nas áreas de processo selecionadas. No entanto, a coleta de documentos não é o foco da ferramenta, pois cada organização possui seus próprios tipos e padrões de documentos. Portanto, apenas o nome dos artefatos é utilizado como evidências coletadas pela ferramenta.

Para gerar o resultado da avaliação, muitas questões têm de ser consideradas. De acordo com SCAMPI, evidências são mais importantes para determinar o quão implementada está uma prática. Portanto, foram definidos precisamente o significado dos indicadores **Implementação de Prática**, **Característica de Prática**, **Objetivos** e **Satisfação de área de Processos**. Nas Tabelas 7, 8, 9 e 10, é apresentada a forma de avaliação dos indicadores criados. A quantidade de práticas fortes implica na quantidade de práticas implementadas para um objetivo. A quantidade de metas com elevada quantidade de práticas implementadas pode implicar a satisfação de uma área de processo.

Tabela 7 – Demonstração do indicador Implementação de Prática (HOMCHUENCHOM *et al.*, 2012).

Evidence Selection (for All Related Roles or Majority after an Affirmation)	Practice Implementation Indicator
All roles: Use	Strength
All roles: Not use	Weak
All roles: Not available to use	Not Rated
Majority: Combination of evidence selection result	Ask for affirmations for those evidences. Then assumes an indicator value from a majority. If there is no majority, uses Weak.

Com base nos valores dos indicadores de implementação da prática, as respectivas características de prática podem ser avaliadas. Dessa forma, são definidos os atributos W, *Weak*

(fraco), S, *Strength* (forte) e NR, *Not Rated* (não avaliado).

Tabela 8 – Indicador Característica de Prática (HOMCHUENCHOM *et al.*, 2012).

Practice Implementation Indicator Condition	Practice Characteristics
$W \geq NR$ and $S = 0$ and $W > 0$	Not implemented
$W > S$ and $W > NR$ and $S > 0$	Partially implemented
$S > W$ and $S > NR$ and $W > 0$	Largely implemented
$S \geq NR$ and $W = 0$ and $S > 0$	Fully implemented
$S = 0$ and $W = 0$ and $NR \geq 0$	Not Rated
$NR > W + S$ and $S \geq 0$ and $W \geq 0$	Not Capable

Dessa forma, podem ser utilizados os valores do indicador Implementação de Prática para avaliar a satisfação dos objetivos associados com base nas definições: (i) SI, representa a soma dos atributos *Largely* e *Fully Implemented* para o indicador Característica de Prática; e (ii) NSI, representa a soma dos atributos *Partially* e *Not Implemented* para o indicador Característica de Prática.

Tabela 9 – Indicador Objetivos (HOMCHUENCHOM *et al.*, 2012).

Practice Characteristics	Goal Satisfaction
$SI > NSI$	Satisfied
$SI < NSI$	Not Satisfied
$SI = NSI$	Not Capable

Por fim, é possível determinar a satisfação de uma área de processo a partir dos valores de satisfação dos Objetivos. Mais uma vez, são definidos os atributos: (i) SG, representa a quantidade de objetivos satisfeitos; (ii) NSG, representa a quantidade de objetivos não satisfeitos; e (iii) NCG, representa a quantidade de objetivos não possíveis de determinar.

Tabela 10 – Indicador Satisfação de área de Processos (HOMCHUENCHOM *et al.*, 2012).

Goal Satisfaction	Process Area Satisfaction
$SG > (NSG + NCG)$ and $SG \geq NCG$	Satisfied
$SG < (NSG + NCG)$ and $NSG \geq NCG$	Not Satisfied
$NCG > SG + NSG$	Not Capable

Os passos para a utilização da ferramenta seguem a seguinte ordem: (i) a organização deve inserir os dados do projeto; (ii) as informações coletadas são utilizadas para criar um conjunto de perguntas que os participantes têm de responder com base em suas funções, de acordo com cada área de processo sobre a definição de avaliação da organização; (iii) a ferramenta determina, de acordo com o modelo *CMMIbyScrum* a produção de um relatório com lacunas e sugestões de atividades SPI (o relatório é apresentado sob o ponto de vista da área de processo por categoria e o ponto de vista da área de processo por maturidade); e (iv) o relatório de SPI contém possíveis soluções para resolver as não-conformidades.

Em sua conclusão, o trabalho descreve uma próxima etapa que consiste em integrar outras técnicas e métodos, tais como, *eXtreme Programming*, e *Rational Unified Process* (RUP) por artefatos e atividades de mapeamento com base na abordagem utilizada com mapeamento do *CMMIbyScrum*.

3.4.2 *Taba workstation: Supporting software process deployment based on CMMI and MR-MPS.BR*

O *Taba Workstation* é um ambiente que proporciona a realização e o acompanhamento de atividades individuais e em grupo, atividades de gerenciamento de projetos, melhoria de produtos de software de qualidade visando ao aumento da produtividade, fornecendo os meios para os engenheiros de software para controlar projetos e medir a evolução das atividades com base em informações coletadas em todo o ciclo de desenvolvimento. Também, esse ambiente integra às atividades de gestão do conhecimento dentro de processos de software com o objetivo de preservar o conhecimento organizacional e promover a institucionalização de um organização de aprendizagem de software. Além disso, fornece a infraestrutura para o desenvolvimento e a integração de ferramentas para apoiar a execução de processos de software. Essa infraestrutura mantém um repositório útil contendo projetos de software e informações recolhidas ao longo do seu ciclo de vida. Com o intuito de apoiar a definição, a implantação a e melhoria de processos de software, o *Taba Workstation* apoia a definição de processos padrão da organização e a modelagem desses processos para projetos específicos com o objetivo de aumentar o controle e melhorar a qualidade de produtos de software.

Os principais objetivos da ferramenta *Taba Workstation* são (MONTONI *et al.*, 2006): (i) apoiar a configuração de ambientes de engenharia de software centrado em processos por diferentes organizações; (ii) apoiar a geração automática (ou seja, a instanciação) de processos de engenharia de software para projetos específicos; (iii) apoiar o desenvolvimento de software utilizando o ambiente instanciado; e (iv) apoiar a gestão do conhecimento organizacional relacionada aos processos de software.

A ferramenta vem sendo utilizada pela indústria brasileira de software desde 2003. Foi considerada durante três avaliações oficiais SCAMPI para o Nível 2 do CMMI como um dos pontos fortes da organização para facilitar o sucesso das iniciativas de implantação de processo de software e para superar as dificuldades inerentes, também em outras três avaliações oficiais MR-MPS.BR. O ambiente é constituído de ferramentas integradas para apoiar a definição de processos de software e implantação, sendo essas ferramentas aderentes às práticas dos níveis CMMI 2 e 3 (MONTONI *et al.*, 2006).

O ambiente é composto por várias ferramentas no qual a definição do processo padrão para uma organização específica é feita durante a configuração de um ambiente específico. A ferramenta responsável por apoiar essa configuração é chamada *Config*. Uma outra ferramenta

chamada *AdaptPro* dá apoio à institucionalização dos processos organizacionais. Usando a ferramenta *AdaptPro*, o engenheiro de software pode executar as seguintes atividades: (i) caracterizar o projeto; (ii) planejar o processo que irá orientar o projeto por meio da adaptação do processo padrão organizacional, considerando as características do projeto; e (iii) instanciar um PSEE (*Process-centered Software Engineering Environment*) para apoiar a execução do procedimento planejado.

Os atores do processo podem executar ferramentas associadas para realizar uma atividade específica do processo; outras funções existentes são baixar versões (*download*) controladas de artefatos a serem consumidos, a fim de concluir uma atividade do processo. Uma vez que a atividade foi iniciada ou concluída, o executor pode fazer o *upload* para o sistema dos artefatos produzidos durante essa atividade. A partir da interface principal, o usuário também pode consultar diretamente informações relacionados às atividades de processo, por exemplo, a padrões de projeto e procedimentos de inspeção de software. O sistema fornece ainda, meios para identificar caminhos críticos do processo, a fim de apoiar o acompanhamento e o controle da execução de processos.

Para realização da validação dos resultados obtivos ao longo do desenvolvimento e da utilização do ambiente, foram realizadas pesquisas com o objetivo de identificar fatores de sucesso e dificuldades relacionadas à implantação de processos de software. Participaram dessas pesquisas membros da COPPE/UFRJ, com vasta experiência em pesquisa de processo de software e implantação. Os fatores de sucesso e as dificuldades identificadas com as pesquisas foram agrupados de acordo com a categoria dos resultados, sendo que 12 categorias foram relacionadas aos fatores de sucesso e 16 categorias relacionadas à dificuldades no processo de software com base em experiências de implantação do CMMI e do MR-MPS.BR. Em seguida, uma análise comparativa desses resultados identificou fatores importantes que contribuíram significativamente para o sucesso de programas de melhoria de processo de software em pequenas, médias e grandes organizações.

O ambiente criado por [Montoni et al. \(2006\)](#) fornece os meios para assegurar que a maioria desses fatores estão fortemente presentes durante iniciativas de implantação de processo de software baseados em CMMI e MR-MPS.BR. Os próximos passos ressaltados pelos autores consistem em avaliar a adequação das ferramentas que forneçam apoio às áreas de processo do CMMI Nível 3 e definir/integrar outras ferramentas de apoio às áreas de processo do CMMI Nível 4 e 5 com o intuito de facilitar a evolução de maturidade de desenvolvimento de software das organização para níveis mais elevados.

3.4.3 **MATURE: A model driven based tool to automatically generate a language that supports CMMI process areas specification**

Neste trabalho, [Musat et al. \(2010\)](#) propõem uma abordagem utilizando MDE para a definição de um metamodelo para a modelagem do modelo de maturidade CMMI. Os autores salientam que todas as formas de engenharia dependem de modelos como ferramentas essenciais para compreender os sistemas complexos do mundo real.

O objetivo apresentado para o trabalho foi produzir um artefato gráfico, denominado *MATURE*, para modelar efetivamente modelos de processos organizacionais. A estratégia de pesquisa foi estruturada da seguinte forma ([MUSAT et al., 2010](#)): (i) estudo do CMMI a partir de um MDD e ponto de vista DSM; (ii) extração dos componentes principais do CMMI; (iii) composição de um conjunto de metamodelos que representem o CMMI; (iv) definição das metáforas gráficas para cada metaclassa do metamodelo; (v) introdução do metamodelo em uma ferramenta de gerenciamento de modelos; (vi) definição e geração da ferramenta de apoio CMMI para a definição de modelos; (vii) a aplicação dessa ferramenta para estudo de caso; (viii) derivação automática do DSL de acordo com cada modelo de referência; e (ix) avaliação dos resultados em relação aos objetivos iniciais. Na Figura 4, é apresentado o metamodelo criado por [Musat et al. \(2010\)](#) que representa o modelo de maturidade CMMI.

Figura 4 – Metamodelo criado por [Musat et al. \(2010\)](#) que representa o modelo de maturidade CMMI ([MUSAT et al., 2010](#)).

MATURE é uma ferramenta baseada no paradigma MDD e oferece como modelagem os conceitos necessários para definir um modelo específico de domínio do CMMI. A ferramenta permite que o usuário defina a especificidade do modelo, com um ambiente gráfico, e foi desenvolvida utilizando o *Graphical Modeling Framework* (GMF) a partir da plataforma Eclipse, com o objetivo de fornecer uma ferramenta de modelagem de software livre para o CMMI. Como resultado, o metamodelo CMMI foi especificado utilizando *Ecore* usando o *plug-in Eclipse*

Modeling Framework (EMF). Na Figura 5, é ilustrado o ambiente MATURE em funcionamento no Eclipse (MUSAT *et al.*, 2010).

Figura 5 – Ambiente MATURE em funcionamento no Eclipse (MUSAT *et al.*, 2010).

O trabalho apresenta um estudo de caso escolhido para apresentar a ferramenta *MATURE*, que corresponde a uma parte do modelo CMMI para a área de processo Análise e Resolução de Causas (CAR). Utilizando essa área de processo, os membros do projeto podem identificar as causas de defeitos selecionados e outros problemas e tomar medidas para impedir que elas ocorram no futuro. Entretanto, a abordagem somente indica como pode ser modelado uma área de processo utilizando a ferramenta *MATURE* (MUSAT *et al.*, 2010). A Figure 6, é apresentado um modelo conceitual da área de processo CAR (MUSAT *et al.*, 2010). Na Figura 7, é apresentada a modelagem da visão de configuração na ferramenta *MATURE* (MUSAT *et al.*, 2010). Por fim, na Figura 8, é apresentada a modelagem das partes específicas da área de processo CAR na ferramenta *MATURE*.

Figura 6 – Apresentação do modelo conceitual do processo CAR (MUSAT *et al.*, 2010).

Figura 7 – Apresentação da modelagem da visão de configuração na ferramenta *MATURE* (MUSAT *et al.*, 2010).

Figura 8 – Apresentação da modelagem das partes específicas do processo CAR na ferramenta *MATURE* (MUSAT *et al.*, 2010).

Como trabalho futuro, os autores pretendem especificar um metamodelo para cobrir alguma melhoria para outros processos e modelos existentes como a norma ISO/IEC 15504 ou ITIL.

3.4.4 *Efficient adoption and assessment of multiple process improvement reference models*

Nesse trabalho, Jeners, Lichter e Rosenkranz (2013) propõem uma abordagem para a integração de diferentes modelos de referência de melhoria de processos baseado em modelos. A ideia principal da abordagem de integração de IRM (*Improvement Reference Models*) da *Mosaic* é normalizar IRM baseado em uma estrutura comum e um conjunto comum de termos. Segundo descrito no trabalho, de acordo com a teoria de modelagem de megamodelos, pode-se normalizar diferentes IRM definindo metamodelos apropriados. Foram analisados metamodelos publicados de IRM, por exemplo, o do CMMI, extraído e adicionado apenas os elementos suficientes para atingir os objetivos definidos pelos autores. Para realizar a modelagem de diferentes IRM, o trabalho propõe uma “Estrutura de Integração de Metamodelos” (*IS Meta-Model*), responsável pela definição do núcleo e tipos de elementos adicionais introduzidos por IRM diferentes, tais como, atividades, artefatos ou papéis.

Enquanto o núcleo e os tipos de elementos adicionais permitem fornecer uma visão aproximada da maioria dos aspectos importantes da IRM, os tipos de elementos foram utilizados para permitir a integração entre conceitos concretos e abstratos de uma comparação detalhada das IRM. Na definição dos autores, um conceito IRM é uma palavra ou a menor combinação de palavras que tem significado único no contexto de uma IRM. Como exemplo, podem ser citados plano de projeto ou divisão de trabalho. Conceitos podem ser derivados de atividades, papéis, insumos e saídas de elementos de IRM como áreas de processo.

Para cada IRM, tais como CMMI, SPICE, COBIT ou ITIL, foram extraídos o núcleo, os conceitos adicionais e informações conceituais e as criou-se uma respectiva estrutura IRM de Integração Modelos (*IRM Integration Structure Models*). Na Figura 9, é representada conceitualmente a abordagem utilizada para a harmonização de diferentes IRM. Nela, pode-se observar

o propósito e a aplicação de ambos os metamodelos e seus respectivos modelos concretos o IRM-ISM (*IRM Integration Structure Models*) e o ICM (*Integration Concept Model*). As diferentes estruturas geométricas representam os IRM, enquanto as figuras geométricas pequenas representam as diferentes terminologias utilizadas em cada IRM. As ISM (*Integration Structure Meta-Model*) representam diferentes instâncias dos metamodelos IS (*Integration Structure*) de cada IRM. O ICM representa os conceitos e os *links* semânticos entre as IRM-ISM conectando os conceitos relacionados.

Figura 9 – Representação conceitual da abordagem utilizada para a harmonização de diferentes IRM (JENERS; LICHTER; ROSENKRANZ, 2013).

Os elementos das ISM foram agrupados em três categorias e foi utilizada pelos autores a notação UML (diagrama de classes) para modelar essas relações. O primeiro elemento é o *Núcleo* que contempla os mais comuns entre os IRM existentes. O segundo elemento é os *Add-ons* que oferecem uma representação para os elementos presentes nos IRM. O terceiro elemento é os *Conceitos*, que contêm elementos para modelos mais específicos. Na Figura 10, é representado o metamodelo da ISM.

Figura 10 – Representação do metamodelo da ISM (JENERS; LICHTER; ROSENKRANZ, 2013).

Com o intuito de avaliar a adequação dos metamodelos desenvolvidos, foi realizado um estudo de caso de médio porte, considerando algumas partes do COBIT, CMMI e FS. Na Figura 11, é apresentado o estudo de caso realizado.

Figura 11 – Representação do estudo de caso realizado pelos autores do artigo (JENERS; LICHTER; ROSENKRANZ, 2013).

Como trabalhos futuros os autores apresentam a necessidade de automatizar a geração dos modelos. Por a modelagem ser uma atividade manual, a modelagem está propensa a falhas e necessita de profissionais com experiência para realizar uma modelagem completa dos modelos e de criação das ICM.

3.4.5 Software repository for software process improvement

Nesse artigo, Park, Kim e Lee (2008) sugerem a criação de um repositório de melhoria de processo de software para gerenciamento e suporte de ativos/artefatos e de ferramentas heterogêneas produzidos por diferentes organizações na SPIC (*Software Process Improvement Center* na Coreia). No trabalho, também são descritas as principais questões de projeto e de modelagem para a organização e a gestão do SIR (*SPIC Integration Repository*). O objetivo é criar um repositório para apoiar a certificação do modelo CMMI, na qual o repositório será utilizado para catalogar os diferentes artefatos de engenharia de software criados ao longo da execução dos processos. O repositório persistiria metadados dos diferentes artefatos (arquivos, programas, modelos, etc); nos metadados, seria possível identificar as informações necessárias para a utilização do ativo.

Na Figura 12, é apresentada a arquitetura do repositório proposto SIR-CM (PARK; KIM; LEE, 2008). A arquitetura do repositório é composta de três camadas: (i) a camada de *Plataforma*, na qual está a estrutura de hardware e software que compõe o repositório; (ii) a camada intermediária que é a de *Negócio*, que contém as regras de manipulação e integração do

repositório com outros subsistemas; e (iii) camada de *Apresentação*, que representa a interface de usuário para a manipulação dos ativos/artefatos (PARK; KIM; LEE, 2008).

Figura 12 – Arquitetura do repositório proposto *SIR-CM* (PARK; KIM; LEE, 2008).

O ambiente possui quatro subsistemas: (i) *Registro*, responsável pelo registros dos ativos no repositório; (ii) *Recuperação*, responsável pela busca dos ativos registrados no ambiente; (iii) *Visão*, fornece informações gerenciais como relatórios, informações de ferramentas e ativos; e (iv) *Gerenciamento*, permite o gerenciamento dos ativos, metadados e ativos locais. O ambiente proporciona a integração de ferramentas de terceiros com o repositório a partir do consumo de serviços disponibilizados pela camada de *Plataforma* conforme descrito anteriormente (PARK; KIM; LEE, 2008). A Figura 13 ilustra como pode ser realizada a integração do ambiente com outras ferramentas (PARK; KIM; LEE, 2008).

Figura 13 – Integração do ambiente *SIR-CM* com outras ferramentas (PARK; KIM; LEE, 2008).

Segundo os autores, os próximos passos para esse trabalho são a continuação da evolução do ambiente e a utilização em um ambiente industrial para avaliar a contribuição no contexto de melhoria do processo de software.

3.4.6 Building a Maturity & Capability Model repository

Em seu artigo, Wangenheim *et al.* (2011) apontam a diversidade de modelos de capacidade e maturidade de processos existentes. Entretanto, não existem muitas ferramentas

disponíveis nas quais esses modelos ou *Model/Capability Models* (MCM), como são denominados pelos autores, estejam disponíveis para análise e comparação. Nesse sentido, é proposto no trabalho a criação de um repositório baseado na Web no qual seja possível armazenar a definição de diferentes MCM disponíveis para consulta pela comunidade de SPI (WANGENHEIM *et al.*, 2011).

Segundo os autores, um repositório de modelos de maturidade e capacidade representa um Sistema de Gestão do Conhecimento, para o gerenciamento de informações sobre os modelos apoiando o armazenamento e acesso de informações sobre eles. Seu objetivo básico é fornecer recursos de pesquisa e de navegação avançados baseados em necessidades de um engenheiro de processos para ajudá-los a compreender melhor a informação e o conhecimento disponível.

As operações disponíveis para a manipulação dos modelos de capacidade e maturidade são: (i) *Busca*, para pesquisar MCM de acordo com processos específicos; (ii) *Pesquisa* (baseado em taxonomias), para possibilitar a busca por modelos de capacidade e maturidade por meio de atributos específicos; (iii) *Gerenciamento de Metadados de Modelos*, para possibilitar um usuário alterar as informações dos metadados dos modelos registrados no repositório; e (iv) *Comentários*, para permitir um usuário registrar um comentário sobre um modelo em específico. Na Figura 14, representa a estrutura proposta para o repositório.

Figura 14 – Estrutura proposta para o repositório MCM (WANGENHEIM *et al.*, 2011).

Os MCM são representados no repositório por meio de metadados, enquanto os modelos reais residem no *site* do proprietário do modelo ou em um local informado. Metadados (“dados sobre dados”) são rótulos descritivos utilizados para indexar o conteúdo para o uso futuro. Sendo assim, o repositório gerencia atributos de informação sobre o objeto ao invés do próprio objeto. Metadados fornecem descrições do formato e dos atributos do objeto e os sistemas de metadados tornam o processo de recuperação de conteúdo mais eficiente e eficaz.

Em sua conclusão, os autores afirmam que estão finalizando a primeira versão do repositório e planejam a sua avaliação por um grupo de especialistas em 2011. Para a avaliação,

foi realizada uma revisão sistemática da literatura com o intuito de identificar MCMs para que pudessem ser disponibilizadas no ambiente e, como resultado, foram disponibilizados 52 modelos para avaliação no repositório. O propósito é avaliar aspectos do ambiente como a sua utilidade e usabilidade. Pesquisas futuras foram planejadas incluindo o reforço das capacidades de pesquisa em termos de buscas semânticas, bem como alternativas de apresentação gráfica da informação.

3.5 Análise Comparativa dos Trabalhos Selecionados

A maioria dos trabalhos propõem metodologias e abordagens (com o apoio de ferramentas computacionais ou modelos de abstração) para realizar a avaliação de processos de software, bem como a sugestão de ações de melhoria para essas atividades de processos. As propostas variam entre a utilização de ferramentas baseadas em questionários e em técnicas que utilizam MDE que procuram representar os modelos de maturidade e identificar semelhanças para facilitar a adoção de múltiplos modelos de maturidade, de acordo com necessidades específicas de negócio.

No entanto, os estudos parecem ser desconectados, utilizando abordagens distintas para fornecer os mesmos resultados com as mesmas limitações. Vale ressaltar que nenhum dos estudos analisados produziu ferramentas computacionais compostas por três conceitos: (i) representação (abstrações dos modelos de maturidade); (ii) avaliação; e (iii) persistência de modelos em um único ambiente para posterior reutilização.

Na Tabela 11, são apresentados os principais estudos identificados na terceira fase do mapeamento sistemático. Nela, pode-se notar (avaliando-se as colunas “Repositório”, “Processo”, “Projeto” e “Maturidade”) a falta de estudos que cubram o processo de modelagem, de avaliação e de persistência de modelos de maturidade e processos de software. O único trabalho identificado que conta com tais características foi *Taba workstation: Supporting software process deployment based on CMMI and MR-MPS.BR* (MONTONI *et al.*, 2006) e tem forte dependência em relação ao modelo de maturidade utilizado durante o seu desenvolvimento. Ele não faz uso de técnicas de MDE em um esforço de permitir a harmonização de modelos de qualidade, como é sugerido na literatura analisada. A presença das características citadas poderia torná-la mais flexível para ser utilizada com modelos de maturidade diferentes permitindo que os usuários não se restrinjam a um modelo de qualidade em particular.

Pode ser notado que os principais modelos utilizados na elaboração dos estudos são CMMI, ISO/IEC 15.504 e ISO/IEC 12.207. O modelo MPS.Br (SOFTEX, 2016) foi utilizado apenas no âmbito nacional. Outro ponto observado neste mapeamento sistemático foi a utilização de abordagens que utilizam MDE (em todos os itens analisados na íntegra) com o intuito de harmonizar diferentes modelos de qualidade de software, o que indica o potencial dessa metodologia para construir um caminho para a representação/abstração de modelos maturidade diferentes, utilizando uma estrutura única de metamodelos.

Por meio dessas observações, pode-se afirmar que os estudos relacionados com a SPI e SPA tem alguns *gaps* de pesquisa:

1. Dependência com Modelo de Maturidade

- Nenhum dos estudos apresentou uma metodologia independente de um modelo de maturidade em particular. Na maioria dos casos, apenas uma parte do modelo foi coberto pela abordagem estabelecida (ferramentas/*frameworks*/planilhas), indicando a necessidade da criação de extensões para cobrir mais níveis de maturidade sendo identificados como trabalhos futuros;

2. Reúso

- Reutilização de software, no contexto da SPA e da SPI, visa à criação de artefatos que podem ser usados por diferentes organizações e que têm contextos de aplicação semelhantes, tais como, modelos de documentos, padrões de documentação e documentos de gerenciamento de projetos. Um conceito pouco explorado no contexto da SPA e da SPI é a reutilização de processos e de modelos de maturidade. Um grande desafio é a criação de uma forma de representação de modelos de qualidade existentes, para ser reutilizados e instanciados por ferramentas diferentes, seja por meio de serviços *web* ou repositórios de base de conhecimento compartilhados pela comunidade de engenharia de software;

3. Envolvimento da Comunidade de Engenharia de Software

- A maioria dos estudos analisados visam à criação de abordagens para apoiar SPA e SPI em ambientes de empresas de pequeno e médio porte, geralmente sendo direcionados a uma necessidade específica. Por exemplo, o principal público-alvo dos estudos eram pequenas e médias empresas; no entanto, há estudos que indicam que as dificuldades e os desafios apresentados são muito semelhantes, independentemente do tamanho da empresa (GARCÍA *et al.*, 2012). Outro detalhe importante é as ferramentas e as metodologias encontradas na literatura fazem uso dos níveis iniciais de modelos de maturidade, deixando a evolução contínua dos processos inteiramente à responsabilidade das organizações. O envolvimento da comunidade de engenharia de software deve ser mais significativa, recolhendo *feedback* constante sobre a eficácia dos métodos disponibilizados pela academia. A eficácia das abordagens criadas não podem ser completamente comprovadas, pois, em muitos casos as ferramentas fornecidas devem ser utilizadas com o suporte de especialistas (JENERS; LICHTER; ROSENKRANZ, 2013). Portanto, os custos que deveriam ser reduzidos permaneceriam os mesmos;

4. Uso de MDE

- O conceito de combinar modelos de maturidade parece promissor, a fim de criar uma forma unificada para representá-los e identificar semelhanças para facilitar sua adoção. Se implementado com sucesso, esse conceito poderia ser usado para alavancar a reutilização de processos, uma vez que eles seriam modelados em uma estrutura homogênea e poderiam ser disponibilizados para a comunidade. Não só MDE pode ser melhor utilizado em SPA e SPI, mas a disponibilização de serviços que possam ser usados pela indústria para avaliar seus processos de forma independente;

5. Modelos de Referência em Maturidade de Processos de Software

- ISO/IEC 15.504 e ISO/IEC 12.207 têm sido amplamente utilizados como fundamentação técnica para o desenvolvimento de novos modelos de qualidade como CMMI-DEV e MPS.Br. No entanto, para cada novo modelo criado, novos termos, objetivos e práticas surgem e podem ser descritos de maneira diferente, mas com a mesma finalidade, em todos eles. Esse fato dificulta o uso de abordagens como ontologias e análise semântica no contexto da SPA e da SPI, porque os termos usados para descrever conceitos similares são, muitas vezes, identificados de forma diferente em modelos de maturidade. Portanto, a existência de um modelo padrão de referência para modelos de maturidade de software seria desejável, a fim de unificar termos, conceitos, práticas e objetivos que devem ser empregados por modelos de maturidade existentes. Tal instrumento facilitaria os esforços de harmonização desses modelos, bem como a criação de abordagens para ajudar na adoção de múltiplos modelos de maturidade;

6. Integração de Metodologias

- Uma deficiência constante nas abordagens para SPA e SPI é a falta de metodologias de integração para facilitar a implementação dessas iniciativas. Nesse sentido, ambientes integrados devem ser criados, o que poderia permitir a execução de SPA e de SPI, bem como compartilhar o conhecimento sobre a adoção de modelos de qualidade. Isso poderia ser alcançado com maior envolvimento da comunidade de engenharia e qualidade de software, distribuindo repositórios com ativos a serem reutilizados em processos. Esses elementos combinados podem levar a uma ferramenta integrada que permite a comunidade compartilhar experiências em diferentes contextos de engenharia de software. A existência de uma base de conhecimento que concentra processos, modelos de maturidade e artefatos para apoiar os processos seria um avanço considerável na área de SPI e SPA, porque poderia ser usado como um guia de referência para os conceitos relacionados à engenharia e qualidade de software.

Tabela 11 – Sumarização dos estudos identificados e suas principais características.

#	Título	Repositório	Processo	Projeto	Maturidade
1	SPIALS-II: A light-weight software process improvement self-assessment tool (HOMCHUENCHOM <i>et al.</i> , 2012)	Não	Não	Sim	Sim
2	Semantic based approach supporting CMMI gap analysis process (RUNGRATRI; USANAVASIN, 2012)	Não	Não	Não	Sim
3	Managing the software process with a software process improvement tool in a small enterprise (GARCÍA <i>et al.</i> , 2012)	Não	Não	Não	Sim
4	A software tool to support the integrated management of software projects in mature SMEs (MAS; MESQUIDA, 2011)	Não	Sim	Sim	Sim
5	MATURE: A model driven based tool to automatically generate a language that supports CMMI process areas specification (MUSAT <i>et al.</i> , 2010)	Não	Não	Não	Sim
6	A design of tool for software processes assessment and improvement (YEOM; HWANG, 2008)	Não	Não	Sim	Sim
7	Software repository for software process improvement/Frameworks of integration repository for software process improvement using SOA (PARK; KIM; LEE, 2008)	Sim	Não	Não	Não
8	An application tool to support the implementation of integrated software process improvement for Malaysia's SME (ALI; IBRAHIM, 2011)	Não	Sim	Não	Sim
9	Taba workstation: Supporting software process deployment based on CMMI and MR-MPS.BR (MONTONI <i>et al.</i> , 2006)	Sim	Sim	Sim	Sim
10	A framework for assisting the design of effective software process improvement implementation strategies (NIAZI; WILSON; ZOWGHI, 2005)	Não	Não	Não	Sim
11	An ontology based infrastructure to support CMMI-based software process assessment (GAZEL; SEZER; TARHAN, 2012)	Não	Não	Não	Sim
12	Efficient adoption and assessment of multiple process improvement reference models (JENERS; LICHTER; ROSENKRANZ, 2013)	Não	Não	Não	Sim
13	A self-assessment framework for finding improvement objectives with ISO 29119 test standard (KASURINEN <i>et al.</i> , 2011)	Não	Sim	Não	Sim
14	HPProcessTOOL: A support tool in the harmonization of multiple reference models (PARDO <i>et al.</i> , 2011)	Não	Sim	Não	Sim
15	Building a Maturity & Capability Model repository (WANGENHEIM <i>et al.</i> , 2011)	Sim	Não	Não	Sim
16	SPIALS: A light-weight Software Process Improvement Self-Assessment Tool (HOMCHUENCHOM <i>et al.</i> , 2011)	Não	Não	Sim	Sim
17	Creating software process capability/maturity models (HOMCHUENCHOM <i>et al.</i> , 2011)	Não	Sim	Não	Sim

Nos estudos selecionados no âmbito de modelos de qualidade harmonizados, foi possível identificar dois métodos mais utilizados: (i) a utilização de metamodelos para definir uma estrutura única, que represente apenas um guia de referência (MUSAT *et al.*, 2010); e (ii) uma extensão da abordagem anterior, porque, além de definir um metamodelo, ele também fornece um método para identificar similaridades entre diferentes modelos de maturidade, facilitando a adoção de modelos múltiplos (JENERS; LICHTER; ROSENKRANZ, 2013).

3.6 Considerações Finais

Este estudo permitiu avaliar a gama de pesquisas realizadas no contexto da SPA e da SPI para responder questões de pesquisa identificadas no planejamento deste mapeamento sistemático, que podem ser resumidas da seguinte forma:

1. Quais são os tipos de abordagens que auxiliam a avaliação e a melhoria de processos de software (SPA e SPI)?
 - Foi possível identificar várias abordagens para a realização de SPI e SPA, dentre elas, pode-se citar: (i) utilização de questionários; (ii) harmonização de modelos de maturidade por meio de metamodelagem; e (iii) criação de repositórios para armazenamento e gestão dos artefatos criados.
2. Essas abordagens permitem a execução da avaliação, considerando um ou mais modelos de qualidade?
 - As abordagens que utilizaram suporte computacional não fornecem a implementação de iniciativas de SPA/SPI, considerando mais de um modelo de maturidade. Somente iniciativas de harmonização dos modelos de maturidade utilizam modelos de abstração; no entanto, a abordagem utilizada tem algumas limitações, sendo elas: (i) modelagem é uma tarefa manual e sujeito a erros; (ii) as abordagens não evoluíram em direção a outros estudos, uma vez que não foram identificados estudos que continuaram a evolução de uma metodologia com a resolução de suas limitações e possibilidades de trabalhos futuros identificados pelos autores; e (iii) a usabilidade não foi avaliada, na maioria dos casos, a validação é uma forma de criar um cenário de desenvolvimento para aplicar o método proposto.
3. Será que as abordagens permitem a reutilização de processos e modelos de qualidade?
 - Todas as abordagens analisadas concentram as informações coletadas em seus próprios ambientes, sem possibilidade de extensões usando serviços da *web*, por exemplo.
4. Que critérios são usados por essas abordagens para sugerir atividades de melhoria de processo?

- A maioria das ferramentas de análise que fazem sugestões de atividades de SPI utilizam questionários em que, a partir das respostas de usuários, indicadores de medição são aplicados para saber o quanto a prática ou o objetivo exigido pelo modelo de maturidade é satisfatório. Em seguida, um relatório é fornecido a partir de uma base de conhecimento implementado pela ferramenta.
5. Como essas abordagens fazem o mapeamento entre os elementos do processo de software executado e os atributos necessários por modelos de qualidade e suas relações?
- A avaliação de um processo de software deve permitir o mapeamento entre as atividades executadas no processo e as atividades esperadas considerando um modelo de maturidade. Isso permite identificar o nível de conformidade e se sugerir ações de melhoria. No entanto, nenhum dos estudos apresentou uma abordagem que permite ao usuário criar uma instância do processo executado e, em seguida, avaliar o quão aderente é para com o modelo de maturidade. Essa foi a maior deficiência encontrada nos estudos analisados, pois os questionários são criados com base em critérios objetivos (o que é esperado na execução do processo). Esses critérios estão disponíveis nos guias de avaliação de modelos de qualidade. Uma das questões da indústria em relação aos modelos de qualidade é não descreverem *como* algo deve ser feito, mas apenas *o que* deve ser feito. No entanto, isso pode ser considerado como o mérito desses modelos, que agrupam os principais conceitos relacionados à qualidade de processos de software e os organiza de uma forma estrutural e evolutiva. Dessa forma, a avaliação feita estritamente por uma ferramenta, fazendo uso de critérios objetivos, pode produzir um resultado parcial para determinar que alguma prática cumpriu os seus critérios de conformidade. Na verdade, o objetivo principal de uma prática definida por um modelo de maturidade poderia ser de alguma forma satisfeita, mas seria difícil avaliá-la utilizando apenas critérios objetivos fornecidos pelas metodologias identificadas.

Metodologias que auxiliem a execução de iniciativas de SPI e SPA são um fator chave para a implementação de modelos de maturidade de processos em organizações de software. O principal objetivo é conquistar melhor nível de qualidade em seus produtos e processos que também funcionam como um diferencial de mercado.

Realizar a implementação de um programa de SPI sem uma metodologia de apoio ao processo de avaliação e de acompanhamento do ciclo de vida dos processos utilizados no desenvolvimento dos projetos de software pode causar falhas na documentação e, conseqüentemente, na gestão do conhecimento dos produtos da organização, acarretando consumo de recursos humanos e financeiros muito além dos previstos pelas organizações (GARCÍA *et al.*, 2012). Dessa forma, é possível observar a existência de diversas iniciativas para a criação de apoio computacional para a realização de SPA e SPI (HOMCHUENCHOM *et al.*, 2012; MONTONI

et al., 2006), para a harmonização de modelos de referência (MUSAT *et al.*, 2010; JENERS; LICHTER; ROSENKRANZ, 2013) e para a criação de repositórios de ativos de engenharia de software e modelos de capacidade e maturidade de processos de software (PARK; KIM; LEE, 2008; WANGENHEIM *et al.*, 2011), o que caracteriza a necessidade de pesquisas na área.

No próximo capítulo, é apresentada uma consulta realizada à comunidade de engenheiros de software, com o intuito de identificar o perfil de profissionais e de empresas que tenham experiência com definição de processos e de implantação de modelos de maturidade, conhecer as principais dificuldades e identificar interessados em participar da avaliação da abordagem descrita nesta dissertação de mestrado.

SURVEY - INDÚSTRIA DE SOFTWARE E PROCESSOS

4.1 Considerações Iniciais

Conforme apresentado no Capítulo 3, muitos trabalhos identificados no mapeamento sistemático realizado foram efetuados em conjunto com empresas e profissionais da área de engenharia de software, no intuito de avaliar a metodologia proposta e auxiliar os interessados em obter certificações e melhorar seus processos. Portanto, como parte da metodologia adotada para a elaboração desta dissertação, foi elaborado um *survey* e disponibilizado para a comunidade de engenharia de software. Uma pesquisa do tipo *survey* pode ser descrita como a obtenção de dados ou informações sobre características, ações ou opiniões de determinado grupo de pessoas, indicado como representante de uma população-alvo, por meio de um instrumento de pesquisa, geralmente em forma de questionário (FREITAS *et al.*, 2000).

Como resultado era esperado obter os seguintes dados: (i) traçar um perfil das empresas que utilizam modelos de maturidade ou investem na definição e na melhoria de seus processos de desenvolvimento; (ii) traçar o perfil dos profissionais envolvidos nesse processo dentro das organizações; (iii) identificar as experiências obtidas no processo de implementação de modelos de maturidade; (iv) entender quais as expectativas da indústria em relação a parcerias com a academia em prol do desenvolvimento de seus processos de desenvolvimento; e (v) recrutar possíveis interessados em participar do processo de avaliação da abordagem proposta.

4.2 Planejamento

Para execução do *survey*, foi realizado um planejamento, que consistiu das seguintes etapas:

1. **Elaboração:** nessa etapa, foram identificadas as informações de interesse da pesquisa em andamento. Portanto, foi elaborado um questionário, dividido em quatro grupos de questões, cada grupo continha quantidade diferente de questões de múltipla escolha ou subjetivas para serem utilizadas para captar o contexto em que o grupo de questões estava inserido. Os grupos de questões criados foram:
 - a) **Perfil da Empresa:** nesse grupo, foram incluídas questões que permitissem avaliar se a empresa trabalha com definição de processos, se implementou ou pretende implementar algum modelo de maturidade;
 - b) **Perfil do Participante:** esse grupo foi criado para coletar dados dos participantes, contatos, sua formação e quais experiências possui com processos, qualidade e maturidade de software;
 - c) **Experiência com Implementação de Modelos de Maturidade e Processo de Software:** esse grupo pode ser considerado o principal elemento do *survey*, pois concentra as questões relacionadas às dificuldades encontradas na implementação de modelos de maturidade e quais abordagens a empresa utilizou para auxiliar nesse processo;
 - d) **Expectativas em Relação a Ferramentas de Avaliação e Melhoria de Processos de Software:** esse grupo foi inserido com o intuito de identificar o grau de interesse das empresas em realizar parcerias com a academia, quais as suas expectativas e o que elas esperam que a academia possa lhes proporcionar;
2. **Revisão:** algumas revisões foram realizadas no questionário pelo autor em parceria com a orientadora, o que resultou em ajustes de opções de respostas em questões, remoção e inserção de questões em alguns grupos.
3. **Disponibilização:** o questionário foi desenvolvido utilizando a ferramenta *Google Forms*, uma plataforma gratuita disponibilizada pela Google para a criação de questionários de pesquisa para serem disponibilizados pela *web*. No Apêndice A, é apresentado o questionário disponibilizado para preenchimento;
4. **Divulgação:** o questionário final foi divulgado em listas de *e-mails*, contatos profissionais e acadêmicos;
5. **Encerramento:** o período de disponibilidade do questionário abrangiu o período entre Abril a Agosto de 2014;
6. **Análise:** entre Setembro e Outubro de 2014, foi realizada a análise dos dados obtidos e sintetizados para inclusão nesta dissertação.

4.3 Análise dos Dados

Ao todo, 16 participantes foram registrados de 14 empresas. Duas empresas tiveram dois participantes registrados no questionário. Iniciando a avaliação dos dados coletados, o primeiro aspecto identificado foi o grupo de questões referente ao perfil das empresas de desenvolvimento de software, que consistiu de 9 questões. Nesse sentido, foi constatado que as empresas participantes possuem as seguintes características:

1. Duas empresas atuam no mercado de TI por um período entre 2 e 5 anos, as outras 14 possuem mais de 5 anos de atuação no mercado;
2. O segmento de atuação das empresas foi bastante variado, a maioria é do segmento de desenvolvimento de software para o varejo (5 empresas), 1 empresa trabalha com sistemas de e-Commerce, 1 empresa com gestão de funerárias, 2 empresas com gestão hospitalar, 3 empresas com sistemas ERP, 2 empresas com gestão pública, 1 empresa com comunicação visual e 1 empresa com RH e gestão de pessoas;
3. Com relação ao porte das empresas, 2 empresas declararam possuir um quadro de colaboradores relacionados a TI de até 10 funcionários, 4 empresas possuem entre 11 e 30, 6 empresas possuem entre 31 e 50, 1 empresa possui entre 51 e 70 e somente 1 com mais de 71 colaboradores;
4. A maioria das empresas, 13 no total, identificaram-se como fábrica de software e uma declarou-se como atuante no segmento de aquisição de software;
5. Das empresas que participaram, 9 declararam que possuem algum tipo de certificação em modelos de maturidade de software e 5 declararam não possuir;
 - a) Dentre 9 empresas que declararam possuir uma certificação, 8 possuem certificação do modelo MPS.Br e 1 declarou possuir certificação CMMI;
 - b) Em relação ao modelo MPS.Br, 5 empresas declararam estar certificadas no nível G de maturidade do modelo, 3 empresas declararam estar nível E e a empresa declarou possuir certificação CMMI nível 3.

Para traçar o perfil dos participantes do *survey*, o quadro de questões correspondente possuía 13 questões e os principais aspectos foram sintetizados e detalhados em:

1. Em relação a formação dos participantes, a maioria possui formação em áreas relacionadas a computação (cursos como Ciência da Computação, Análise de Sistemas e Sistemas da Informação). Somente 1 participante declarou não possuir graduação concluída, outro possui formação em Administração de Empresas e outros 2 participantes declararam ter formação em Ciências da Informação e da Documentação. Dentre os participantes, da área de computação, 3 possuem título de mestre;

2. Dos 16 participantes, 9 declararam possuir mais de 5 anos de experiência com qualidade e processos de software, 6 declararam trabalhar com o assunto em um período entre 2 e 5 anos. Somente 1 participante se declarou sem experiência no assunto;
3. No contexto de desenvolvimento de software, 9 declararam possuir mais de 5 anos de experiência, 2 declararam possuir experiência entre 2 e 5 anos e 5 declararam possuir menos de 1 ano de experiência com desenvolvimento;
4. Dentro do contexto de garantia da qualidade, os participantes declararam possuir experiência com atividades como: gestão de processos (3), auditorias de garantia de qualidade (1), implantação de modelos de maturidade (1), gerência de projetos (1), revisões formais (9) e verificação e validação (12);
5. No contexto de processos de desenvolvimento de software, 15 participantes declararam participar de atividades com objetivo de definição dos processos da empresa, 14 declararam possuir envolvimento com análise de melhoria dos processos internos, 6 declararam desenvolver atividades de consultoria em certificação de modelos de maturidade e somente 1 declarou ser o responsável pela coordenação e acompanhamento da definição e implantação dos processos de software na empresa em que trabalha;
6. Sobre métodos ágeis, 14 participantes declararam possuir experiência com *Scrum*, 1 com *XP* e 2 não possuem experiência com métodos ágeis;
7. Dentre os participantes, 11 declararam ter participado de comitês criados com pessoas envolvidas nos diferentes processos da empresa. Esses comitês têm a função de discutir os processos usados pela empresa, trazer sugestões das áreas em que atuam, propor melhorias e auxiliar na implantação. O principal foco de discussão dos comitês em que os participantes participaram são melhoria dos processos, gestão dos requisitos, gestão de projetos, gerência de configuração e metodologias de desenvolvimento de software. Em relação ao desenvolvimento de software, os comitês discutiam assuntos relacionados ao processo utilizado no desenvolvimento, novos padrões, tecnologias e metodologias de desenvolvimento;
8. Ao final desse grupo de questões, foi disponibilizada uma questão subjetiva, para que o participante pudesse acrescentar alguma informação adicional relacionada ao seu perfil. Nas respostas coletadas, foi identificado um participante que declarou ter participado de uma equipe de avaliação CMMI e outro em que a empresa estava passando por análise de implantação da norma ISO 29.110.

Após definidos os perfis das empresas e profissionais de engenharia de software que participaram do *survey*, foi iniciada a análise do quadro de questões “Experiência com Implementação de Modelos de Maturidade e Processo de Software” que possui 14 questões. Esse é o principal

componente do *survey* e, por meio dele, foi possível extrair os dados sintetizados a seguir. Isso possibilitou identificar quais as dificuldades enfrentadas ao adotar modelos de maturidade pelas empresas participantes, quais as suas impressões pré e pós definição dos processos, quais foram os prós e contras da decisão. Em algumas questões foi observado que o número de resposta é maior que o número de empresas que declararam serem certificadas. Isso ocorre devido ao fato de algumas empresas iniciaram o processo de adoção de um modelo de maturidade mas não finalizou o processo com a certificação, entretanto, algumas práticas foram mantidas nos processos da empresa.

1. Dentre os modelos de maturidade utilizados, mesmo pelas empresas sem certificação, foram utilizados como referência para criar seus processos os modelos CMMI (2), MPS.Br (13), ISO/IEC 15.504 (1), ITIL (3), COBIT (1) e ISO 29.110 (1);
2. Para as empresas que realizaram o processo de certificação em um modelo de maturidade, foram apontadas como principais problemas e dificuldades no processo de implementação os seguintes aspectos: (i) problemas na priorização nas atividades envolvidas no processo (1); (ii) cultura empresarial dos clientes (1); (iii) implantar todas as áreas de processo exigidas pelo nível de maturidade pretendido ao mesmo tempo (1); (iv) dificuldade em entender os requisitos do modelo (3); (v) problemas com consultorias externas (4); e (vi) resistência das equipes internas (10). Mesmo as empresas que não possuem certificação em modelo de maturidade reportaram resistência das equipes internas na mudança de processos internos, sendo que 6 das 9 empresas certificadas reportaram o problema;
3. Em uma das questões, foi solicitado ao participante apontar quais foram as áreas de processo mais difíceis de adequar do modelo adotado mediante o *status* inicial dos processos organizacionais da empresa. No geral, foram reportadas dificuldades na implementação de todas as áreas de processo identificadas nos modelos MPS.Br e CMMI (ver Capítulo 2). Entretanto, as que mais se destacaram foram Medição, Gerência de Riscos, Gerência de Reutilização, Gerência de Requisitos, Gerência de Projetos, Gerência de Decisões, Gerência de Configuração e Desenvolvimento de Requisitos;
4. As ferramentas de apoio mais utilizadas pelas empresas na execução e na definição de seus processos foram Project Builder, Google Sites, SharePoint, Pacote Office, Tortoise SVN, Bizagi Modeler e Eclipse Enterprise Process Framework. Também foram reportadas empresas que desenvolveram ferramentas próprias (6 empresas); nesse sentido, as ferramentas desenvolvidas possuíam funções, principalmente, para a gestão dos projetos, geração de indicadores e gestão dos requisitos;
5. Muitas vezes, o uso de consultorias externas em qualidade de software são necessárias para realizar as etapas preliminares à avaliação do modelo. Dessa forma, foi solicitado ao participante descrever como foi o processo de apoio da consultoria, se foi satisfatório e se

- houveram problemas. Nessa questão, todos os participantes que passaram por certificação reportaram problemas. No geral, foi avaliada como satisfatória a atuação das consultorias; entretanto, alguns problemas foram identificados pelas empresas. Dentre eles, foram citados problemas de comunicação e problemas de definições divergentes entre diferentes consultores. Na opinião dos participantes, isso ocorre pois por vezes, foram recebidas instruções diferentes ou que no momento da avaliação oficial foram identificadas não conformidades em requisitos definidos junto com a consultoria. Em alguns casos, diferentes orientações entre consultores e sugestões de implementação de requisitos fornecidas não condiziam com o que o avaliador esperava;
6. O tempo médio reportado para implementação dos processos informado pelas empresas variou entre 1 a 3 meses (2), 10 a 12 meses (3), 4 a 6 meses (2) e mais de 12 meses (8);
 7. Em termos de custo médio, foi reportado pelas empresas as seguintes faixas de valores: até R\$ 10.000,00 (2), entre R\$ 10.000,00 e R\$ 30.000,00 (9), entre R\$ 30.000,00 e R\$ 50.000,00 (1), entre R\$ 50.000,00 e R\$ 75.000,00 (1), entre R\$ 150.000,00 e R\$ 200.000,00 (1) e entre R\$ 300.000,00 e R\$ 350.000,00 (1);
 8. Após a implantação dos processos, os ganhos notados pela empresa/equipe de desenvolvimento foram: (i) uniformidade no desenvolvimento de sistemas entre os vários produtos (1); (ii) melhoria no entendimento do que deve ser desenvolvido (12); (iii) melhoria no desempenho dos projetos (5); (iv) melhor comunicação entre as equipes de desenvolvimento (11); (v) melhor comunicação com o cliente (6); e (vi) a diminuição das falhas dos produtos (5). Somente 1 empresa reportou não ter notado melhoria após a implantação dos processos;
 9. Outro aspecto avaliado, após a implantação dos processos, foi as perdas notadas pela empresa/equipe de desenvolvimento. Nessa questão, foram reportados os seguintes itens: (i) necessidade de recursos adicionais para a manutenção do processo implementado (5); (ii) excesso de documentação (10); (iii) padronização dos processos (1); (iv) sensação de “engessamento” do processo (6); e (v) a criação de “gargalos” operacionais (2). Somente 1 empresa reportou não ter notado perdas após a implantação dos processos;
 10. O fator de ganho de desempenho na execução dos projetos também foi questionado nesse grupo de questões e foram identificadas pelas empresas ganhos de até 15% (2), até 40% (8), até 50% (3), até 75% (2) e até 100% (1);
 11. Dentre as empresas com certificação em modelo de maturidade, foi questionado se havia motivação para prosseguir a escala de melhoria dos processos dentro do modelo de qualidade adotado, ou seja, se possuía interesse em continuar nos níveis de maturidade mais elevados. Das 9 empresas certificadas, 8 informaram possuir interesse em continuar incrementando o nível de maturidade de seus processos. Dentre essas 8 empresas, 7 empresas já haviam iniciado processo de implementação de mudança de nível de maturidade;

12. Assim como nos grupos de questões anteriores, foi disponibilizada uma questão subjetiva, para o participante poder acrescentar alguma informação que considerasse relevante. Nas respostas coletadas, foi constatado que, mesmo sem possuir certificação ou não possuir motivação para continuar a escala de evolução proposta pelos modelos, as empresas investem em melhorar processos que consideram cruciais, são eles: (i) Gerência de Configuração; (ii) Garantia da Qualidade; (iii) Gerência de Projetos; (iv) Gerência de Requisitos; e (v) Medição.

Por fim, foi iniciada a análise do quadro de questões “Expectativas em Relação a Ferramentas de Avaliação e Melhoria de Processos de Software” que possui 8 questões e as respostas coletadas foram sintetizadas a seguir:

1. Dentro do contexto de processo de software, foi questionado qual funcionalidade a empresa esperava de uma ferramenta de apoio à gestão e à melhoria dos processos de software. Os participantes reportaram as funções: gestão das áreas de processo (3), gestão dos artefatos gerados na execução do processo (7), identificação de pontos de melhoria no processo (4) e pré-análise de aderência a modelos de maturidade (2);
2. Dentre as questões desse quadro, foi questionado se a empresa utilizaria uma abordagem baseada em serviços para realizar a gestão e a melhoria dos processos de software da empresa. A abordagem descrita na questão proveria serviços que auxiliariam a empresa a identificar ativos de reúso para seus processos e projetos. Além disso, seria possível representar o processo executado da empresa e comparar com um modelo de maturidade com o intuito de indicar os pontos em que não há aderência entre eles. O objetivo da abordagem seria facilitar o processo de adequação para uma possível certificação no modelo de maturidade a ser adotado, independente de qual seja (MPS.Br, CMMI, normas ISO/IEC). Todas se mostraram dispostas a utilizar uma abordagem com as características descritas;
3. A partir da questão anterior, foi questionado se a empresa compartilharia suas experiências no contexto de engenharia de software em geral em um ambiente similar ao de uma rede social voltada para reúso, avaliação e melhoria de processos de software para apoiar e/ou receber apoio de outros profissionais. Somente 2 das 14 empresas reportaram que não compartilhariam suas experiências;
4. Ainda no contexto de reúso, foi questionado se a empresa faria uso de um ambiente de reúso em engenharia de software no qual seria possível disponibilizar e/ou ter acesso a ativos de reúso relacionados a projetos, requisitos e linhas de produto de software. Novamente, somente 2 das 14 empresas reportaram que não compartilhariam suas experiências;
5. Quando questionado se as empresas teriam interesse em participar de um projeto piloto relacionado à melhoria e à avaliação de processos de software, 11 empresas reportaram

que teriam interesse em participar. Por outro lado, ao questionar se a empresa aceitaria disponibilizar os relatórios técnicos de avaliação preliminar do processo de certificação de modelos de maturidade, com o intuito de contribuir para a criação de cenários de validação da metodologia, somente 7 das 11 empresas concordaram em disponibilizar informações, mesmo utilizando um termo de confidencialidade sobre a identidade da empresa e do conteúdo dos relatórios técnicos.

Para finalizar o questionário, foi disponibilizado um espaço em que o participante poderia compartilhar opiniões ou sugestões que julgasse relevantes. Dentre as contribuições disponibilizadas, foi possível notar que existe um consenso entre os participantes. É do conhecimento dos participantes que o número de atividades, que antes não eram realizadas, cresce, gerando na opinião dos participantes apenas mais burocracia. Esse é um ponto considerado pela indústria como um fator de impedimento para a adoção dos modelos de maturidade. O número de documentação gerada para executar todos os requisitos solicitados pelos modelos é demasiada, aumentando o número de recursos humanos necessários para garantir a execução de tudo que é requerido e, consequentemente, os custos operacionais das empresas.

Dessa forma, entende-se que uma ferramenta de apoio deveria possuir funções que facilitassem a inclusão e a recuperação de informações referentes ao processo, bem como funções que permitissem consultar o histórico a fim de servir como apoio para a melhoria do processo. Seria também útil que a ferramenta apoiasse na extração do desempenho de projetos, permitindo comparar o seu desempenho à medida que melhorias são implementadas.

Outras opiniões e experiências compartilhadas pelos participantes, obtidas por meio da experiência em que se submeteram no processo de certificação, foram compartilhadas, tais como: (i) o ganho de experiência e conhecimento ao se submeter pelo processo de certificação de modelos de maturidade; (ii) obtenção de redução de custos com retrabalhos por meio de investimento em processos; (iii) melhoria na transparência em relação aos processos internos da empresa a todos colaboradores; (iv) melhoria da qualidade dos produtos; (v) falta de companheirismo em certos momentos do processo de implantação dos processos; e (vi) a geração de indicadores e dados estatísticos para tomada de decisão.

Em um relato de um dos participantes, foi destacada a necessidade de uma espécie de rede social realmente ativa, na qual haja troca de ideias e discussões reais sobre as dificuldades e as melhorias da área de qualidade de software. Na opinião do participante, as redes disponíveis atualmente possuem um papel meramente comercial, abusando de espaço para propaganda de cursos/treinamentos ou debates e não evoluem nem colaboram com as necessidades dos profissionais.

4.4 Considerações Finais

A definição de processos de desenvolvimento de software, por si só, pode ser considerada uma tarefa complexa. Adicionalmente, existe a necessidade de criar um processo que atenda aos interesses da organização e seja aderente a um modelo de maturidade. Fatalmente, haverá aumento complexidade nessa tarefa. Isso pode ser comprovado a partir das experiências compartilhadas pelos participantes do *survey* apresentado neste capítulo.

Os resultados apontados no Capítulo 3, referente ao mapeamento sistemático elaborado, juntamente com os dados analisados no *survey* apresentado confirmam o interesse da indústria em investir em seus processos de desenvolvimento. É reconhecido que, por meio de processos bem definidos, vários problemas podem ser superados nos projetos de desenvolvimento de software, tais como: (i) falta de qualidade no produto entregue, o que para produtos de software consiste em produtos que não atendem às necessidades dos usuários; (ii) retrabalho por parte das equipes de desenvolvimento, o que pode ser causado pela falta de entendimento ou de levantamento adequado dos requisitos de usuário; e (iii) problemas com comunicação entre as equipes e com os fornecedores de requisitos, levando à identificação problemas na fase de implantação dos produtos. Outro ponto confirmado é a necessidade de metodologias que apoiem as empresas na execução de atividades de SPA e SPI. Um fator interessante levantado por um dos participantes é a falta de um ambiente em que os profissionais realmente possam compartilhar experiências e contribuam mutuamente em seus projetos de definição e de melhoria de processos.

Em relação às metodologias disponíveis para modelagem de processos, o *framework* SPEM, apresentado no Capítulo 2, não foi identificado como uma ferramenta largamente utilizada para a documentação de processos pelos participantes. Somente uma empresa declarou utilizar a ferramenta *Eclipse Enterprise Process Framework*, que permite a modelagem de processos utilizando um conjunto de metamodelos e classes baseados no modelo SPEM. A maioria das empresas utiliza métodos mais simples para representar seus processos, por exemplo, ferramentas de modelagem baseadas em *BPMN - Business Process Model And Notation* (BPMN, 2015) como o *Bizagi Process Modeler* (BIZAGI, 2015). Apesar de disponibilizar uma estrutura rica para a modelagem e documentação de processos, o excesso de classes, de atributos, de relacionamentos e de documentação definidos pelo SPEM pode afastar usuários menos experientes, que necessitam de formas mais objetivas para descrever seus processos, muitas vezes com escopo demasiado reduzido para usar toda a estrutura disponibilizada pelo SPEM.

Dessa forma, a proposta de metodologia baseada em metamodelos e transformações de modelos descrita no Capítulo 5, visa disponibilizar um conjunto de metamodelos que possibilite a representação de processos e de modelos de maturidade de forma simples, porém permita que ambos sejam avaliados e as sugestões de melhoria sejam compartilhadas pela comunidade de engenharia de software.

ABORDAGEM PROPOSTA

Neste capítulo, é apresentada a abordagem proposta por esta dissertação de mestrado. Primeiramente são detalhadas algumas definições realizadas para a criação da abordagem, uma visão geral da proposta, seus principais componentes e suas relações. Em seguida, são apresentados e detalhados os metamodelos e suas respectivas transformações criadas para abstrair os conceitos envolvidos em modelos de maturidade e processos, seguidos de alguns exemplos de aplicação. Ao final, são levantados alguns pontos de observação sobre a abordagem nas considerações finais.

5.1 Considerações Iniciais

No Capítulo 3, foram apresentados alguns *gaps* na área de *SPA/SPI* que serviram de base para a proposta deste projeto. Entre eles, os principais a serem destacados são: (i) dependência do modelo de maturidade e do nível de maturidade específico para realizar algum tipo de avaliação ou recomendação de melhoria; (ii) falta de ferramentas que promovam o reúso de processos de software e/ou artefatos de engenharia de software utilizados na execução de processos; e (iii) pouco envolvimento da comunidade de engenharia de software, pois a maioria dos trabalhos selecionados desenvolveram suas ferramentas e métodos utilizando uma única empresa como estudo de caso e isso pode ter causado certo direcionamento no sentido de solucionar problemas específicos de um ambiente de desenvolvimento (aspectos não funcionais de processos de desenvolvimento de software).

A abordagem proposta e detalhada neste Capítulo visa minimizar as limitações mencionadas fazendo uso de conceitos como: (i) abstração dos modelos de maturidade, por meio da criação de metamodelos que representem de forma simplificada os principais elementos encontrados em modelos de maturidade e em um processo de desenvolvimento de software em si; (ii) repositório de processos que possam ser compartilhados em sua totalidade ou em partes de acordo com o interesse da comunidade de engenheiros de software que utilizarem a abordagem; e (iii) incentivo

à participação da comunidade de engenharia de software a compartilhar experiências e ajudar engenheiros iniciantes a elaborar e gerir seus próprios processos.

Como resultado da aplicação da abordagem proposta espera-se obter: (i) uma representação visual de um ou mais processos de software de uma organização; (ii) a realização de avaliações entre um processo de software e um modelo de maturidade com o intuito de identificar não conformidades que devem ser corrigidas para atender plenamente ao modelo se desejado; (iii) um repositório de processos que possa ser constantemente alimentado e melhorado por engenheiros de software que trabalhem com qualidade de processos de software; e (iv) o compartilhamento de experiências entre os membros da comunidade.

Dessa forma, a abordagem proposta tem como elemento central a identificação/classificação dos principais elementos encontrados em um processo de software e qual a sua relação com um requisito de um modelo de maturidade. Para tanto, algumas definições foram realizadas, tais como:

1. Os principais elementos considerados para a definição de um processo de software são atividades, papéis e os produtos de trabalho (artefatos) produzidos e consumidos, organizados em áreas de processo que definem requisitos específicos ou genéricos;
2. Um requisito específico de um modelo de maturidade corresponde a alguma prática que deve ser aplicada a uma área de processo específica e que não possui efeito sobre outras áreas de processo de um modelo de maturidade;
3. Um requisito genérico de um modelo de maturidade corresponde a alguma prática que deve ser aplicada a todas as áreas de processo definidas no modelo de maturidade para um nível em específico:
 - A maioria dos modelos de maturidade é estruturado em níveis de maturidade que agrupam áreas de processo (áreas de conhecimento em engenharia de software) e definem para cada um suas práticas específicas e genéricas para guiar a avaliação de conformidade entre o processo definido pela organização e o modelo escolhido;
4. Na abordagem é utilizado do conceito de *taxonomia* para a classificação desses elementos com o intuito de facilitar a relação entre os elementos de um modelo de maturidade e um processo de software a ser avaliado:
 - Uma taxonomia pode ser considerada como um indexador utilizado para agrupar/classificar elementos de um mesmo domínio, porém não há como garantir uma semântica entre os elementos que a compõem (CORREIA; AGUIAR, 2011). Por exemplo, na abordagem proposta, a taxonomia é usada para indexar as atividades, os papéis e os produtos de trabalho utilizados para exemplificar possíveis implementações de modelos de maturidade e instâncias de processos modelados por engenheiros de software. Na Seção 5.2, o uso da taxonomia é melhor detalhado;

5. A abordagem refere-se a um ambiente que poderá ser utilizado por dois tipos de usuários, sendo eles:

- Oráculo: pode ser considerado como um engenheiro de software experiente na área de qualidade e processos de software, seu diferencial é somente ele poderá: (i) criar taxonomias e modelos de maturidade no ambiente computacional que implemente a abordagem; e (ii) definir os métodos de avaliação de conformidade (os métodos de avaliação são detalhados na subseção 5.6). O termo *Oráculo* tem inspiração no teste de software na qual são definidos *Oráculos de Teste*, recursos humanos que detêm o conhecimento de aplicação de uma determinada área de conhecimento;
- Usuário: usuário comum do ambiente. Entre suas atividades podem ser citadas: (i) criar seus processos com base em taxonomias criadas por um Oráculo; (ii) realizar avaliações de conformidade; e (iii) fornecer *feedback* em relação aos ajustes realizados por ele em seus processos ou sugerir melhorias para outros usuários.

5.2 Visão Geral da abordagem

Antes de detalhar a abordagem criada, deve-se conhecer os objetivos que se pretende alcançar com sua criação, sendo eles:

1. Criar uma abordagem para avaliação de conformidade de processos em relação a modelos de maturidade que não possua dependência direta com ele ou com um nível de maturidade em específico;
2. Possibilitar o reuso dos processos e de seus elementos pela comunidade de engenharia de software, por meio da criação de um repositório de processos que poderá ser consultado por qualquer pessoa;
3. Compartilhar experiências e lições aprendidas pela comunidade de engenharia de software no domínio de qualidade e processos;
4. Possibilitar às organizações de software que não recorram de imediato a consultorias externas na área de qualidade de processos de software, onerando seu orçamento. Esse fator econômico muitas vezes é considerado como impeditivo para empresas de pequeno e médio porte que não têm os recursos financeiros e humanos necessários para definir, gerenciar/manter e evoluir seus processos organizacionais (ALI; IBRAHIM, 2011).

A abordagem proposta está estruturada em três componentes principais para a sua utilização, e está dividida em seis etapas para a sua execução completa. Os principais elementos da abordagem são: (i) definição da taxonomia, base para a criação dos modelos de maturidade e relacionamentos dos processos; (ii) criação de um modelo de maturidade de acordo com o seu

manual de referência; e (iii) modelagem de um processo para avaliar o nível de conformidade com o modelo de maturidade criado. A partir disso, poderá ser realizada uma avaliação de conformidade entre o modelo de maturidade e a instância de processo, que produzirá um relatório contendo o resultado da análise realizada. Com base no resultado da avaliação, o engenheiro de software responsável pelo processo (criador) pode disponibilizar o relatório para outros engenheiros de software, que podem contribuir com sugestões e lições aprendidas em seus próprios projetos.

Na Figura 15, são apresentadas as seis etapas a serem executadas para a aplicação da abordagem; nela, é possível notar que os três primeiros elementos representam os principais componentes citados. Na figura, é apresentada a sequência de passos executados na abordagem, sendo que ao final, após a execução do passo **Sugestão de Melhorias**, uma estrutura de decisão é apresentada, para permitir ao usuário escolher se encerra a execução da abordagem ou reinicia na etapa *Modelagem Usuário* aplicando as sugestões de melhoria coletadas na etapa **Sugestão de Melhorias**. As atividades executadas em cada uma das etapas são detalhadas nas Figuras 16, 17, 18, 19 e 20. A seguir, uma descrição de cada etapa é apresentada:

Figura 15 – Representação das etapas de execução da abordagem.

- **Modelagem Oráculo:** nessa etapa, o usuário Oráculo realiza a modelagem dos pacotes de metamodelos *Taxonomy*, *ProcessTemplate* e *MaturityModel*. Para os pacotes *ProcessTemplate* e *MaturityModel*, também são modeladas as suas respectivas estruturas de trabalho (papéis, atividades e produtos de trabalho produzidos/consumidos) com o pacote de metamodelo *StructureAndBehavior*;
- **Criação Avaliações de Conformidade:** em seguida, após a definição das informações base para a aplicação dos metamodelos em algoritmos de transformação, o usuário Oráculo inicia o processo de criação das avaliações de conformidade. As avaliações, conforme detalhado na Seção 5.7 deste Capítulo, podem variar entre métodos que utilizam questionários customizados ou *scripts* de transformação;
- **Modelagem Usuário:** nessa etapa, o usuário inicia a modelagem da sua instância de processo e, se desejado, cria uma instância dos projetos que usam o processo criado. Nessa etapa, o usuário inicia a criação da estrutura de comparação para uma possível avaliação pela taxonomia, conforme é apresentado em detalhes nas seções seguintes;

Figura 16 – Representação das atividades executadas na etapa **Modelagem Oráculo**.

Figura 17 – Representação das atividades executadas na etapa **Criação Avaliações de Conformidade**.

- **Execução Avaliação de Conformidade:** nesse momento, o usuário possui suas instâncias de processos devidamente modelados e poderá iniciar uma avaliação de conformidade a partir de uma avaliação disponibilizada pelo usuário Oráculo, assim como a seleção de um modelo de maturidade para a base de avaliação. Ao final dessa etapa, um relatório com a síntese da análise é apresentado;
- Na Figura 20, estão concentradas as atividades das etapas **Análise Relatório** e **Sugestão de Melhorias**:
 - Na etapa **Análise Relatório** o usuário pode avaliar os pontos de não conformidade

Figura 18 – Representação das atividades executadas na etapa **Modelagem Usuário**.

Figura 19 – Representação das atividades executadas na etapa **Execução Avaliação de Conformidade**.

identificados entre a sua instância de processo e o modelo de maturidade escolhido;

- Por fim, na etapa **Sugestão de Melhorias**, o usuário que criou as instâncias de processos pode coletar sugestões de melhorias fornecidas por usuários de um ambiente computacional que implemente a abordagem proposta.

Até o momento, foram apresentadas as etapas para aplicar a abordagem, seus principais elementos e objetivos. Entretanto, o alicerce da abordagem é os metamodelos criados para abstrair o modelo de maturidade e o processo de software utilizado pelo usuário. A ideia de criar metamodelos para se abstrair os conceitos envolvidos na definição de modelos de maturidade e processos de software tem por objetivo: (i) criar uma representação visual dos manuais de

Figura 20 – Representação das atividades executadas nas etapas **Análise Relatório** e **Sugestão de Melhorias**.

referência dos modelos de maturidade, com isso espera-se que seja mais fácil e intuitivo o entendimento desses manuais, representados de forma textual, formal e técnica em sua maioria, dificultando a aplicação por engenheiros de software com pouca experiência; e (ii) possibilidade da manipulação dos modelos por meio de transformações entre eles, por exemplo, a partir dos metamodelos de processo e modelo de maturidade é realizada uma transformação referente à avaliação de conformidade, que resultará no metamodelo de relatório de resultado da avaliação.

Como a estrutura de metamodelos é complexa e envolve várias entidades de diferentes contextos de aplicação, foi utilizado o conceito de empacotamento de orientação a objetos e algumas entidades foram agrupadas. Existem associações entre entidades de diferentes pacotes de metamodelos, descritas na medida em que aparecem ao longo deste capítulo. Os pacotes de metamodelos criados na abordagem proposta são *Taxonomy*, *Process Template*, *Process Instance*, *Project*, *Maturity Model*, *StructureAndBehavior*, *Assessment Evaluation*, *Script Based*, *Questionnaire Based*, *Report* e *Improvement Feedback*.

Na Figura 21, são apresentados os pacotes de metamodelos que compõem a abordagem e as suas relações. Na imagem, pode-se observar que, conforme descrito anteriormente, a taxonomia é a base para a definição do processo e do modelo de maturidade.

O pacote *Process Template* representa um modelo de processo mais conceitual, por exemplo, Cascata, Espiral, Incremental e Métodos Ágeis. O pacote *Process Instance*, como o nome indica, é a instância do processo utilizado pela organização em uso em seus projetos e poderá passar pela avaliação de conformidade. Ao criar um *Process Instance*, o usuário poderá utilizar um *Process Template* disponível que servirá de base para o que está sendo criado — aqui pode ser realizada uma analogia ao conceito de herança de orientação a objetos — e, em seguida, realizar os ajustes que julgar necessários. Ainda no contexto da composição do processo, o pacote *Project* representa uma instância com os dados do projeto que utiliza o processo em sua execução. O pacote *Maturity Model* representa o modelo de maturidade ligado ao processo que o implementa. Ainda no contexto de processos e modelos de maturidade, o

pacote de metamodelos *StructureAndBehavior* representa uma estrutura unificada para instanciar o elementos de processos como atividades, papéis e produtos de trabalho.

O pacote *Assessment Evaluation* é responsável por realizar a avaliação de conformidade entre o modelo de maturidade e a instância de processo que se quer avaliar. Para esse pacote, pode-se observar a existência de dois subpacotes, que consistem das duas formas de avaliação de conformidade disponibilizadas pela abordagem, ambas são descritas na subseção 5.6. O pacote *Report* contém entidades que permitem sintetizar o resultado obtido na execução da avaliação de conformidade. A partir disso, a comunidade de usuários do ambiente computacional que pretende implementar a abordagem poderá dar sugestões de melhoria, a fim de auxiliar o criador do processo a obter sucesso em uma nova avaliação, após realizar os ajustes sugeridos. Por fim, o pacote *Improvement Feedback* representa as sugestões de melhoria informadas pelos usuários do ambiente. Na Figura 21, os subpacotes presentes no pacote *Assesment Evaluation* possuem uma notação diferenciada (símbolo no canto superior direito). Esse símbolo denota um sistema, representando que os subpacotes *Script Based* e *Questionnaire Based* estão contidos no pacote *Assesment Evaluation*.

Figura 21 – Pacotes de metamodelos criados para dar suporte à aplicação da abordagem proposta.

Para ilustrar todos os componentes presentes na abordagem, na Figura 22 é apresentada uma síntese das etapas previstas na abordagem e dos atores envolvidos em cada etapa, bem como as relações de entrada e saída entre elas. Nela, temos a figura do Oráculo, indicando suas funções no ambiente, sendo elas: (i) criação da taxonomia; (ii) templates de processos; e (iii) definir os modelos de maturidade. Também está presente a figura do Usuário, que define as suas instâncias de processos e informações de seus projetos de desenvolvimento de software (se desejar). A partir disso, é possível executar uma avaliação de conformidade que gerará um relatório descrevendo o nível de aderência identificado entre o processo e o modelo de maturidade escolhido. Com base nos dados dos relatórios de conformidade, os membros da comunidade de engenharia de software podem sugerir melhorias aos “donos” dos processos para que possam alcançar o nível de maturidade desejado.

Figura 22 – Visão geral da abordagem proposta.

5.3 Modelos Conceituais

Para cada pacote de metamodelos criado na abordagem proposta (conforme ilustrado na Figura 21), existem várias entidades de negócio envolvidas. Até o momento, foram apresentados os objetivos pretendidos com a abordagem e a sua estruturação. Nesta seção, é apresentado o detalhamento de cada um dos pacotes de metamodelos apresentados anteriormente, suas entidades, atributos e relacionamentos. Na Figura 23, é ilustrada a modelagem conceitual com as entidades e seus relacionamentos, porém, para facilitar a visualização do diagrama, nesse momento, ele é apresentado sem os atributos de cada entidade mapeada. Na Figura 21, as linhas tracejadas representam as relações de dependência entre cada pacote de metamodelos detalhados a seguir.

- *implements*: indica que um modelo de maturidade é implementado por uma instância de processo. Essa instância representa um processo candidato a ser certificado seguindo as premissas e os requisitos definidos pelo modelo de maturidade;
- *follows*: um projeto de desenvolvimento de software deve possuir a sua estrutura de trabalho clara e bem definida para que seus objetivos sejam satisfeitos. Dessa forma, esse relacionamento indica qual o processo seguido pelo projeto para guiar a sua execução e monitoramento até o seu encerramento e entrega;
- *inherits*: representa a possibilidade de um *ProcessInstance* de herdar as definições de processo a partir de um *ProcessTemplate*;
- *elementsDefinedBy*: representa a dependência entre os elementos definidos na modelagem dos processos e dos modelos de maturidade e a estrutura de classificação criada;
- *basedOn*: define a base necessária para a realização de uma avaliação de conformidade entre processos e modelos de maturidade;
- *synthetizes*: a origem dos dados dos resultados obtidos na avaliação de conformidade serão sintetizados e descritos por meio deste relacionamento;
- *provides*: uma referência para o item de não conformidade encontrado e uma sugestão de melhoria registrada por algum usuário da abordagem;
- *suggestsImprovements*: indica a uma instância de processo quais as possíveis melhorias a serem realizadas para aumentar o grau de conformidade com o modelo de maturidade desejado;
- *defineExecutedElements*, *defineTemplateElements* e *defineSuggestedElements*: representam o relacionamento entre os elementos (atividades, papéis e produtos de trabalho) definidos na modelagem de processos e nos modelos de maturidade e seu vínculo com uma instância de processo, um *template* de processo ou modelo de maturidade, respectivamente.

Figura 23 – Modelagem conceitual da abordagem proposta sem atributos.

5.3.1 Taxonomy

O pacote de metamodelo *Taxonomy* representa a estrutura de classificação dos elementos utilizados como base na concepção dos modelos de maturidade e processos dos usuários da abordagem. Conforme ilustrado na Figura 24, esse pacote é composto pelas entidades *Taxonomy*, *ElementType*, *TaxonomyElement* e *Synonym*. Todas as entidades são compostas por dois atributos: nome e descrição. A seguir, detalham-se cada uma das entidades:

1. *Taxonomy*: entidade que representa a raiz da taxonomia. Por exemplo, pode-se criar uma taxonomia específica para um modelo de maturidade — MPS.Br por exemplo — na qual são mapeados os elementos específicos do modelo para que seja representado um processo que se aplique especificamente ao MPS.Br. Nesse ponto, destaca-se a escolha da taxonomia para ser o alicerce da metodologia. Da mesma forma que se pode criar uma estrutura específica para um modelo de maturidade, pode-se criar outras que se adequem a mais de um ou que represente somente aspectos semelhantes de modelos distintos (conceito de harmonização de modelos de maturidade);
2. *ElementType*: representa uma estrutura de classificação para os elementos da taxonomia, possibilitando a seleção de elementos específicos por determinadas entidades dos demais pacotes em seus relacionamentos. Por exemplo, aqui podem ser classificados elementos da taxonomia como áreas de processo, atividades, papéis, produtos de trabalho e ferramentas de apoio;
3. *TaxonomyElement*: representa o elemento da taxonomia. Nesse pacote são nomeados e descritos os elementos que fazem parte da taxonomia criada. Essa entidade é relacionada à entidade relacionada *Element*, do pacote *StructureAndBehavior*. Esse relacionamento visa vincular os elementos da taxonomia com os elementos utilizados na definição do processo e modelo de maturidade. Dessa forma, é possível classificar os elementos de ambos os lados, processo e modelo de maturidade, facilitando a identificação de elementos semelhantes nas avaliações de conformidade;
4. *Synonym*: essa entidade foi adicionada com o intuito de facilitar uma forma de classificação muito utilizada em modelos de maturidade, acrônimo. Por exemplo, as áreas de processo no modelo MPS.Br são comumente representadas por acrônimos: Gerência de Requisitos como *GRE*, Gerência de Projetos como *GPR*, Gerência de Configuração como *GCO*, Gerência de Reúso como *GRU*, entre outros. Outra utilidade para essa entidade é mapear sinônimos utilizados para descrever elementos de diferentes formas. Por exemplo, um documento de requisitos pode possuir diferentes nomes em diferentes organizações: *Elicitação de requisitos*, *Requisitos do Projeto*, *Solicitações de Mudança*, são exemplos de variações que podem ser encontradas na indústria para um mesmo documento de engenharia utilizado em seu processo de desenvolvimento.

Figura 24 – Modelagem conceitual do pacote de metamodelo *Taxonomy*.

5.3.2 MaturityModel

O pacote de metamodelo *MaturityModel* representa a estrutura de um modelo de maturidade/qualidade de processo de software. Suas entidades visam representar a estrutura dos modelos de maturidade, sua divisão em níveis de maturidade, requisitos específicos e genéricos e áreas de processo ou de negócio. Dessa forma, conforme ilustrado na Figura 25, esse pacote é composto pelas entidades *MaturityModel*, *ProcessArea*, *SpecificPractice*, *GenericPractice*, *MaturityLevel* e *GPSubPractice*, detalhadas a seguir:

1. *MaturityModel*: representa o modelo de maturidade em si. Nele, estão presentes atributos que definam o modelo, tais como, nome, versão utilizada como referência para a modelagem, data de liberação da versão utilizada, autor (grupo/entidade responsável pelo modelo), descrição do modelo, acrônimo e uma *URL* onde podem ser encontradas mais informações do modelo, caso desejado;
2. *ProcessArea*: representa as áreas de processo ou negócio em que são divididas as disciplinas de engenharia de software utilizadas pelo modelo. Como exemplo, pode-se citar Gerência de Requisitos, Gerência de Configuração e Gerência de Reúso, todas presentes no modelo MPS.Br;
3. *SpecificPractice*: a partir do momento que o modelo define uma área de processo ou de negócio, para que se adeque à ela, é necessário estar em conformidade com o que é definido como sendo práticas requeridas pelo modelo. Nesse ponto, observa-se que os modelos definem *o quê* deve ser feito para estar em conformidade com o que o modelo pede, mas o *como* não é fornecido pelos modelos. Dessa forma, pode-se afirmar que os modelos de maturidade são metamodelos de processos, em que é fornecida uma estrutura base para criar um processo que, seguindo as diretrizes do modelo, poderá ser um processo de qualidade e, conseqüentemente, resultará em produtos de software de melhor qualidade;
4. *MaturityLevel*: os modelos de maturidade agrupam diversas áreas de processo e diversas otimizações em cada uma delas que, se aplicadas todas de uma vez, se tornariam muito “pesadas” para as empresas aderirem de uma vez. Dessa forma, alguns modelos são organizados em níveis de maturidade, sendo que a cada nível áreas de processo ou negócio são evoluídas e outras mais refinadas são introduzidas e refinadas nos níveis posteriores. O

modelo MPS.Br, por exemplo, está dividido em 7 níveis de maturidade, iniciando no nível G até o nível A;

5. *GenericPractice*: seguindo o exemplo do modelo MPS.Br, para os níveis de maturidade definidos, também são definidas algumas práticas aplicadas não somente a uma área de processo, mas a todas as áreas que fazem parte do nível de maturidade em questão. Essas práticas são conhecidas como *AP* — Atributos de Processo — e definem diretrizes amplas que servirão de guia para a execução dos processos que fazem parte de um nível de maturidade em questão. Analogamente aos atributos de processo, o modelo CMMI define as *Generic Goals* e *Generic Practices*. O MPS.Br disponibiliza um manual que descreve a equivalência entre os modelos, indicando para cada prática específica do modelo qual a sua equivalente no CMMI ([SOFTEX, 2012](#));
6. *GPSubPractice*: ainda no contexto das práticas genéricas aplicadas ao nível de maturidade, elas podem ser subdivididas em subpráticas representadas por esta entidade na abordagem.

Figura 25 – Modelagem conceitual do pacote de metamodelo *MaturityModel*.

5.3.3 Process

O pacote de metamodelo *Process* representa a estrutura base de um processo de software. Nesse caso, conforme ilustrado na Figura 21, este pacote é usado para representar os elementos *ProcessTemplate* e *ProcessInstance*. Conforme ilustrado na Figura 26, este pacote é composto pelas entidades *Process* e *ProcessPhase* detalhadas a seguir:

1. *Process*: define um *template* ou instância de processo, tendo os atributos: nome, descrição, propósito do processo e um tipo (*ProcessInstance* ou *ProcessTemplate*). O atributo *background* é utilizado para definir quais as origens do processo, por exemplo, se ele é

inspirado em processos ágeis (como Scrum e/ou XP) ou processos tradicionais (como Cascata, Espiral e/ou Prototipação);

2. *Phase*: entidade utilizada para definir as fases do processo, por exemplo, seguindo o *RUP* pode-se definir as fases de *iniciação*, *elaboração*, *construção* e *transição*. Possui os atributos nome, descrição e um atributo inteiro para determinar a ordem de execução das fases do processo, iniciando em 1 a primeira fase.

Figura 26 – Modelagem conceitual do pacote de metamodelo *Process*.

5.3.4 Project

O pacote de metamodelo *Project* representa algumas informações relativas ao projeto de software que seguiu o processo definido pelo usuário. Não é escopo deste projeto a gestão do projeto; sendo assim, nesse pacote não são definidas estruturas que permitam manipular informações que podem ser utilizadas para o monitoramento do projeto. Portanto, aspectos como custos, estimativas e previsões de início e entrega são armazenados nas instâncias do metamodelo e, futuramente, após a criação de um repositório, com os dados de projetos da indústria, é possível traçar perfis de execução de projetos e identificar dificuldades e propor soluções para os problemas identificados por meio de análises estatísticas dos dados coletados.

Conforme ilustrado na Figura 27, esse pacote é composto pelas entidades *Project*, *ProcessElementExecution*, *ProjectFeedback* e *Goal*. As entidades desse pacote consistem de um repositório e podem ser utilizadas pela academia para análises estatísticas. Por exemplo, após coletar dados de vários projetos mapeados com a abordagem, pode-se extrair indicadores do tipo: (i) quantos projetos atingem seus objetivos estabelecidos; (ii) qual a distorção de estimativas de esforço e custos desses projetos; e (iii) quantos são iniciados e entregues nos prazos estabelecidos. A seguir, as entidades são detalhadas:

1. *Project*: representa os dados estimados e reais do projeto, como custo, cronograma e esforço (em horas). Essa entidade que se relaciona-se com a entidade *Process*, indicando qual o processo utilizado pelo projeto;

2. *Goal*: representa os objetivos pretendidos com o projeto;
3. *ProjectFeedback*: representa o *feedback* obtido com a execução do projeto, tais como pontos positivos, negativos e lições aprendidas;
4. *ProcessElementExecution*: é um elemento adicional a este pacote. Sua estrutura permite um possível mapeamento das atividades previstas no processo e as atividades realmente executadas. Esse tipo de análise pode ser utilizada para verificar a conformidade do projeto ao processo, indicando o quanto o processo é executado, quais atividades não foram executadas e porque, gerando dados para possíveis ajustes antes de uma avaliação de conformidade. Entretanto, conforme dito anteriormente, a gestão dos projetos não é escopo da abordagem. Essa *feature* presente nesse pacote é melhor aproveitada por análises de conformidade baseadas em *scripts*, detalhada na Seção 5.6.

Figura 27 – Modelagem conceitual do pacote de metamodelo *Project*.

5.3.5 StructureAndBehavior

Esse pacote representa a base para a modelagem dos modelos de maturidade e processos e foi inspirado em uma parte da estrutura disponibilizada pelo SPEM (SPEM, 2015). O pacote de metamodelo *StructureAndBehavior* representa os elementos base para a definição de um processo, que consiste em atividades executadas, os papéis que as executam e os produtos de trabalho produzidos e consumidos ao longo da execução do processo. Conforme ilustrado na Figura 28, esse pacote é composto pelas entidades *Element*, *Activity*, *Task*, *Role*, *WorkProduct* e *Tool*, detalhadas a seguir:

1. *Element*: é base do pacote e é “herdada” pelas entidades *Activity*, *Role* e *WorkProduct*, pois, a partir dela, é realizado o relacionamento com um item da taxonomia correspondente. Dentre seus atributos, é definido um nome, uma descrição, um objetivo e seu tipo de instância, que poderá ser definida como: (i) elemento *sugerido*, utilizado como sugestões

- de atividades, papéis e produtos de trabalho na modelagem de um modelo de maturidade; (ii) elemento *template*, utilizado na modelagem de um *ProcessTemplate*; e (iii) elemento *executado* utilizado na modelagem de um *ProcessInstance*;
2. *Activity*: descreve as atividades desempenhadas pelos papéis definidos. Como há uma herança da entidade *Element*, seu único atributo específico é um tipo, que definirá se a execução da atividade é mandatória ou opcional. Essa entidade possui relacionamentos com quase todas as demais entidades do pacote e elas são detalhados na listagem a seguir:
 - *subExecutes*: define um relacionamento entre atividades que possuem dependência com outras atividades, por exemplo, ordem de prioridade de execução;
 - *produceAndConsume*: determina quais são os produtos de trabalho consumidos (entradas) e produzidos (saídas) ao executar a atividade;
 - *performedBy*: define quais os papéis responsáveis pela execução da atividade;
 - *executes*: relacionamento com a entidade que define os passos executados na atividade;
 3. *Task*: utilizada para descrever as “tarefas” ou passos a serem executados na atividade definida. Nesse pacote, também há elementos do SPEM, que definem atividades de forma macro e as refina enumerando passos para exemplificar melhor a sua execução;
 4. *Role*: define os papéis desempenhados pelos recursos humanos da organização na execução do processo. Por exemplo, engenheiro de software, programador, analista de requisitos e gerente de projetos. Nesse pacote, existe uma herança e são definidos os atributos de requisitos, que contêm uma descrição das habilidades necessárias para desempenhar o papel e um tipo, que pode ser definido como: (i) primário, para o papel preferencial para a execução da atividade; ou (ii) secundário, para um papel de apoio ao primário ou substituto;
 5. *WorkProduct*: são os artefatos de engenharia de software produzidos e consumidos ao longo da execução do processo de software. Nessa entidade, são definidos, além dos atributos da herança, os atributos: (i) *URL* com o caminho onde se pode encontrar o artefato, o padrão de nomenclatura do artefato no projeto (padrões de nomenclatura de artefatos são utilizados em modelos de maturidade para padronizar o acesso aos artefatos e otimizar a busca por artefatos em repositórios de projetos); (ii) um arquivo *template* de modelo do artefato; e (iii) um tipo. O atributo tipo é utilizado para definir se o artefato poderá ser utilizado como uma evidência direta, indireta ou opcional em uma possível avaliação de conformidade com modelo de maturidade e deverá ser indicada, preferencialmente, por um Oráculo;
 6. *Tool*: entidade complementar, que indicará qual ferramenta de apoio poderá ser utilizada para consultar e editar os produtos de trabalho indicados. Dentre seus atributos, estão

definidos: (i) a versão da ferramenta utilizada; (ii) *URL* para *download*/aquisição; (iii) descrição da licença de uso; e (iv) tipo, definido como *proprietário* ou *open source*.

Figura 28 – Modelagem conceitual do pacote de metamodelo *StructureAndBehavior*.

5.3.6 Assessment

O pacote de metamodelo *Assessment* representa as estruturas de avaliação de conformidade disponibilizadas pela abordagem proposta. Existem dois métodos de avaliação disponíveis: (i) baseado em *script*, identificada na modelagem conceitual como *ObjectiveAssessment*; e (ii) baseado em questionário, identificada na modelagem conceitual como *SubjectiveAssessment*. Ambos os métodos são mais detalhados na Seção 5.6. Vale ressaltar que o Oráculo é responsável por definir os métodos de avaliação de conformidade, uma vez que é necessário certo grau de *expertise* no assunto para criação das estruturas de simulação de avaliação.

Por enquanto, são detalhadas somente a modelagem conceitual das estruturas criadas para a sua representação. Conforme ilustrado na Figura 29, esse pacote é composto pelas entidades *ExecutedAssessment*, *SubjectiveAssessment*, *Questionnaire*, *ComplianceItem*, *Question*, *AnswerOption*, *Rule*, *ComplianceType*, *ObjectiveAssessment* e *ScriptTemplate*, detalhadas a seguir:

1. *ExecutedAssessment*: representa uma avaliação de conformidade executada. Essa entidade relaciona-se com uma instância de processo e um modelo de maturidade para que seja realizada a análise de conformidade;
2. *SubjectiveAssessment*: representa um dos dois tipos de avaliação disponíveis na abordagem proposta, em que é realizada uma forma de avaliação baseada em questionários. Dessa forma, são definidas questões que se refiram a uma prática específica ou genérica do modelo de maturidade e, dentre as possíveis respostas, é realizada uma média que determinará o

grau de conformidade com o que é pedido pelo modelo. As entidades que representam a estrutura do questionário são:

- a) *Questionnaire*: representa o questionário criado. Uma mesma avaliação pode ter vários questionários, cada um avaliando um item em específico, que pode ser uma área de processo, um nível de maturidade ou uma prática específica. Essa é mais uma das vantagens identificadas na abordagem proposta, a avaliação pode ser realizada em diferentes instâncias da estrutura dos modelos de maturidade, oferecendo níveis distintos de especificidade nas avaliações realizadas. Uma limitação identificada no mapeamento sistemático é a necessidade de criar extensões das metodologias propostas para avaliar diferentes modelos de maturidade e níveis diferentes. Na abordagem proposta, a única limitação é a motivação dos Oráculos em criar diferentes modelos de avaliação;
 - b) *ComplianceItem*: representa um elemento do modelo de maturidade avaliado por meio das respostas providas para a questão. Um item de conformidade pode possuir mais de uma questão para poder determinar a sua aderência;
 - c) *Question*: representa uma questão, utilizada para determinar o quão aderente é o processo ao modelo escolhido. Para cada, questão pode ser definido um peso, que pode ser utilizado para dar uma nota ao item respondido;
 - d) *AnswerOption*: representa as opções de resposta, cada opção pode ser utilizada para filtrar níveis de aderência ao item de conformidade sendo avaliado. Cada resposta possui uma nota, que pode ser utilizada para calcular uma média;
 - e) *ComplianceTypeRule*: utilizada para determinar faixas de conformidade de acordo com a nota obtida no item de conformidade sendo avaliado. Por exemplo, entre 0 e 0,25, o item pode ser avaliado como *não aderente*; entre 0,26 e 0,50, pode ser avaliado como *pouco aderente*; entre 0,51 e 0,75, pode ser avaliado como *aderente com ressalvas*; e entre 0,76 e 1, pode ser avaliado como *aderente*;
3. *ObjectiveAssessment*: representa o outro tipo de avaliação disponibilizado pela abordagem proposta, em que é feita uma avaliação objetiva do processo. Essa avaliação é baseada em *scripts* de transformações de modelos. Os Oráculos poderão gerar *scripts* customizados para realizarem avaliações de conformidade com base nas informações disponibilizadas pelos modelos criados a partir da estrutura de pacotes de metamodelos disponibilizados pela abordagem. Um exemplo é disponibilizado na Seção 5.6 para demonstrar melhor esse tipo de avaliação;
 4. *ScriptTemplate*: armazena o *script* customizado criado pelo Oráculo. Os resultados obtidos a partir dessa avaliação podem ser gerados a partir do próprio *script* criado ou armazenado no pacote *Report*. Outra possível aplicação desse método de avaliação é complementar algum ponto avaliado por um questionário. Por exemplo, uma possível customização de

avaliação que pode ser realizada é a comparação entre os dados modelados pela entidade *ProcessElementExecution* em detrimento com as atividades definidas na entidade *ProcessInstance*. Com isso, é possível apontar quais atividades previstas não foram executadas, identificar os motivos e realizar ajustes. Essa primeira avaliação entre o que é previsto e o que é de fato realizado eliminaria vários ajustes que seriam identificados somente em simulações de avaliações realizadas por consultorias externas em qualidade de processos de software. Cada *script* criado pode ser classificado como: (i) *model-to-model*, ou seja, uma transformação de modelo para modelo, pois o modelo de destino pode ser um modelo customizado pelo Oráculo, na qual algum aspecto do processo ou modelo de maturidade foi estendido para melhor retratar algum problema que não tenha sido mapeado pela abordagem proposta; ou (ii) *model-to-text*, ou seja, uma transformação de modelo para texto; nesse caso, é um relatório com as conclusões da avaliação.

Figura 29 – Modelagem conceitual do pacote de metamodelo *Assessment*.

Outra vantagem identificada na estrutura de avaliação sugerida pela abordagem proposta é a flexibilidade proporcionada. Para avaliar a aderência a modelos de maturidade diferentes para uma mesma instância de processo, basta trocar o modelo de maturidade de referência e executar a avaliação. Outro cenário possibilitado pela abordagem é a utilização de diferentes questionários para avaliar um mesmo processo, por exemplo, vários usuários Oráculo poderiam disponibilizar formulários que avaliem especificamente o processo Gerência de Requisitos do

nível G do modelo MPS.Br, fornecendo aos usuários uma vasta gama de possibilidades e opiniões diferentes que o auxilie a avaliar seus processos.

5.3.7 Report

O pacote de metamodelo *Report* representa a síntese das conclusões obtidas após a execução de uma avaliação de conformidade. Conforme ilustrado na Figura 30, esse pacote é composto pelas entidades *Report* e *ReportItem*, detalhadas a seguir.

1. *Report*: representa as informações da avaliação realizada, ou seja, a data da avaliação, o tipo (*script based* ou *questionnaire based*), o modelo de maturidade adotado como referência, o nível de maturidade utilizado (opcional), a área de processo ou negócio avaliada e a instância de processo executado em avaliação. A entidade *Report* é relacionada com a entidade *AssessmentExecuted* do pacote de metamodelos *Assessment*;
2. *ReportItem*: para essa entidade, é criado um item relativo a cada item de conformidade avaliado, no caso de uma avaliação baseada em questionário. Para avaliações baseadas em *scripts*, os atributos *itemDescription*, *evaluationDescription* e *evaluationResult* poderão ser utilizados para descrever o que foi avaliado, de que forma e qual o resultado obtido, respectivamente.

Figura 30 – Modelagem conceitual do pacote de metamodelo *Report*.

5.3.8 Improvement

O pacote de metamodelo *Improvement* representa a estrutura de sugestões de melhoria para os itens de conformidade avaliados e sintetizados pelo pacote de metamodelos *Report*. As entidades que compõem esse pacote permitem agrupar as sugestões de melhoria indicadas pelos demais usuários da ferramenta computacional que implemente a abordagem proposta. Os usuários acessam as avaliações realizadas no ambiente computacional e poderiam sugerir melhorias (como em um fórum de discussões) e, a partir disso, receber *feedback* de suas sugestões, ganhando notas e construindo uma reputação dentro do ambiente social proporcionado pela ferramenta. Conforme ilustrado na Figura 31, esse pacote é composto pelas entidades *ImprovementHint*, e *HintComments*, detalhadas a seguir:

1. *ImprovementHint*: é relacionada com a entidade *ReportItem* e é utilizada para o usuário fornecer uma sugestão/dicas de melhoria. Os atributos disponíveis são: descrição da dica, um exemplo de aplicação textual e opcionalmente, algum tipo de artefato de exemplo como anexo:
2. *HintComments*: representa o retorno fornecido pelo usuário que recebeu a sugestão. A partir dos *hints* fornecidos pela comunidade, pode ser gerada uma discussão sobre o item, sendo registrada por comentários adicionais ao *hint*. A partir disso, um usuário pode criar uma reputação a partir do tipo do *feedback*. Os usuários que recebem a sugestão podem dar *feedback*, classificando-o como positivo, negativo ou neutro, caso a sugestão não tenha realmente ajudado na melhoria do item em discussão.

Figura 31 – Modelagem conceitual do pacote de metamodelo *Improvement*.

5.4 Metamodelos

Após a etapa de modelagem conceitual do problema e da solução proposta por esta dissertação, foi iniciada a fase de criação dos metamodelos, que serviriam de base para a modelagem das entidades de negócio manipuladas pela abordagem nas avaliações de conformidade. Para cada entidade representada nos modelos conceituais dos pacotes de metamodelos, foi criada uma classe no nível de metamodelo.

Para a criação dos modelos utilizados na modelagem das entidades de negócio da abordagem, foi utilizado o *framework* de metamodelagem *Ecore* conforme descrito na Seção 2.6 do Capítulo 2. Após representadas as entidades do modelo conceitual em classes no diagrama *Ecore*, é gerado o arquivo de geração de modelos. A partir dele, são exportados os pacotes de código fonte para a criação de um *plug-in* com a capacidade de utilizar um editor de modelos para a modelagem desejada. O *plug-in* gerado deve estar disponível na pasta de instalação da distribuição do Eclipse utilizada para ser possível criar arquivos de modelagem com a extensão do metamodelo desejado.

Na Figura 32, é apresentada a representação do diagrama pelo editor do *Ecore*. Adicionalmente, na Figura 33, é apresentada a modelagem visual realizada do pacote de metamodelos *MaturityModel* (ver Seção 5.3.2). Para que leitores e possíveis interessados possam consultar os metamodelos e os modelos criados na elaboração deste estudo de caso, apresentado nas seções

a seguir, foi disponibilizado o *workspace* do Eclipse no *GitHub* para consulta. A *URL* para consulta ao repositório criado é <<https://github.com/dfeloni/DissertacaoMestrado.git>>.

Figura 32 – Arquivo do modelo Ecore criado para o pacote de metamodelos *MaturityModel*.

Conforme descrito na Seção 2.6, os diagramas Ecore são descritos por meio de um arquivo *XMI*. Na listagem de código fonte 1, é apresentado o arquivo gerado pelo Ecore para o pacote de metamodelos *MaturityModel*. Na listagem, é possível identificar nas linhas 5, 18, 27, 34, 41 e 50 a declaração de cada uma das entidades presentes na modelagem conceitual. As linhas seguintes a cada uma das citadas anteriormente descrevem os atributos (e seus respectivos tipos de dados), relacionamentos e referências entre as metaclasses.

Figura 33 – Modelagem no *framework* Ecore do pacote de metamodelo *MaturityModel* pelo editor visual (*plugin Ecore Diagram*).

Código-fonte 1: Arquivo em *XMI* do diagrama Ecore referente ao pacote de metamodelos *MaturityModel*.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="
  MaturityModel" nsURI="platform:/resource/MaturityModelModel/model"
4   nsPrefix="matQlt">
5 <eClassifiers xsi:type="ecore:EClass" name="MaturityModel">
6   <eStructuralFeatures xsi:type="ecore:EAttribute" name="name"
  eType="ecore:EDatatype http://www.eclipse.org/emf/2002/Ecore#//
  EString"/>
7   <eStructuralFeatures xsi:type="ecore:EAttribute" name="version"
  eType="ecore:EDatatype http://www.eclipse.org/emf/2002/Ecore#//
  EString"/>
8   <eStructuralFeatures xsi:type="ecore:EAttribute" name="
  releaseDate" eType="ecore:EDatatype http://www.eclipse.org/emf
  /2002/Ecore#//EDate"/>
9   <eStructuralFeatures xsi:type="ecore:EAttribute" name="author"
  eType="ecore:EDatatype http://www.eclipse.org/emf/2002/Ecore#//
  EString"/>
10  <eStructuralFeatures xsi:type="ecore:EAttribute" name="
  description" eType="ecore:EDatatype http://www.eclipse.org/emf
  /2002/Ecore#//EString"/>
11  <eStructuralFeatures xsi:type="ecore:EAttribute" name="acronym"
  eType="ecore:EDatatype http://www.eclipse.org/emf/2002/Ecore#//

```

```

    EString"/>
12    <eStructuralFeatures xsi:type="ecore:EAttribute" name="url" eType
    ="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"
    />
13    <eStructuralFeatures xsi:type="ecore:EReference" name="organizes"
    upperBound="-1"
14        eType="#//ProcessArea" containment="true"/>
15    <eStructuralFeatures xsi:type="ecore:EReference" name="
    evolvesInto" upperBound="-1"
16        eType="#//MaturityLevel" containment="true"/>
17 </eClassifiers>
18 <eClassifiers xsi:type="ecore:EClass" name="ProcessArea">
19    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//
    EString"/>
20    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
    shortDescription" eType="ecore:EDataType http://www.eclipse.org/
    emf/2002/Ecore#//EString"/>
21    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
    mainDescription" eType="ecore:EDataType http://www.eclipse.org/emf
    /2002/Ecore#//EString"/>
22    <eStructuralFeatures xsi:type="ecore:EAttribute" name="acronym"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//
    EString"/>
23    <eStructuralFeatures xsi:type="ecore:EReference" name="defines"
    upperBound="-1"
24        eType="#//SpecificPractice" containment="true"/>
25    <eStructuralFeatures xsi:type="ecore:EReference" name="implements
    " eType="#//MaturityLevel"/>
26 </eClassifiers>
27 <eClassifiers xsi:type="ecore:EClass" name="SpecificPractice">
28    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//
    EString"/>
29    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
    description" eType="ecore:EDataType http://www.eclipse.org/emf
    /2002/Ecore#//EString"/>
30    <eStructuralFeatures xsi:type="ecore:EAttribute" name="acronym"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//
    EString"/>
31    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
    complementaryDescription"
32        eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore
    #//EString"/>
33 </eClassifiers>
34 <eClassifiers xsi:type="ecore:EClass" name="MaturityLevel">
35    <eStructuralFeatures xsi:type="ecore:EAttribute" name="name"

```

```

    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//
    EString"/>
36    <eStructuralFeatures xsi:type="ecore:EAttribute" name="
    description" eType="ecore:EDataType http://www.eclipse.org/emf
    /2002/Ecore#//EString"/>
37    <eStructuralFeatures xsi:type="ecore:EAttribute" name="acronym"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//
    EString"/>
38    <eStructuralFeatures xsi:type="ecore:EReference" name="
    evolvesInto" upperBound="-1"
39        eType="#//GenericPractice" containment="true"/>
40 </eClassifiers>
41 <eClassifiers xsi:type="ecore:EClass" name="GenericPractice">
42     <eStructuralFeatures xsi:type="ecore:EAttribute" name="name"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//
    EString"/>
43     <eStructuralFeatures xsi:type="ecore:EAttribute" name="
    description" eType="ecore:EDataType http://www.eclipse.org/emf
    /2002/Ecore#//EString"/>
44     <eStructuralFeatures xsi:type="ecore:EAttribute" name="acronym"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//
    EString"/>
45     <eStructuralFeatures xsi:type="ecore:EAttribute" name="
    complementaryDescription"
46         eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore
    #//EString"/>
47     <eStructuralFeatures xsi:type="ecore:EReference" name="divided"
    upperBound="-1"
48         eType="#//GPSubPractice" containment="true"/>
49 </eClassifiers>
50 <eClassifiers xsi:type="ecore:EClass" name="GPSubPractice">
51     <eStructuralFeatures xsi:type="ecore:EAttribute" name="name"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//
    EString"/>
52     <eStructuralFeatures xsi:type="ecore:EAttribute" name="
    description" eType="ecore:EDataType http://www.eclipse.org/emf
    /2002/Ecore#//EString"/>
53     <eStructuralFeatures xsi:type="ecore:EAttribute" name="acronym"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//
    EString"/>
54 </eClassifiers>
55 </ecore:EPackage>

```

5.5 Exemplo de Modelagem

Para ilustrar uma instância de modelos criados utilizando os pacotes de metamodelos, foram criados alguns exemplos. Nas Figuras 34 e 35 é apresentada a instância de uma modelagem realizada para um modelo de maturidade. Nelas, estão representadas as práticas específicas e genéricas da área de processo *Gerência de Requisitos* aderente ao nível de *G* de maturidade do modelo MPS.Br. Na Figura 34, é apresentada a visualização do modelo pelo Eclipse, mostrando a estrutura de objetos criados de acordo com a estrutura do metamodelo. Na Figura 35, são visualizadas as propriedades/atributos de cada objeto. Como o Eclipse não permite visualizar os atributos dos objetos no editor de modelos, foi necessário dividir a visualização em duas figuras.

Figura 34 – Arquivo de instância do modelo *MaturityModel* no Eclipse.

Figura 35 – Aba de propriedades de instância do modelo *MaturityModel* no Eclipse.

5.6 Métodos de Avaliação

Na abordagem criada, são disponibilizados dois métodos de avaliação de conformidade, conforme citado anteriormente. O primeiro deles é um método baseado em questionários, ou seja, utilizando a estrutura definida pelo subpacote *SubjectiveAssessment* do pacote *Assessment*, os usuários poderão criar formulários customizados para avaliar itens específicos de um modelo de maturidade de acordo com as necessidades de cada usuário.

A utilização de formulários para avaliar conformidade entre processos e modelos de maturidade foi uma das abordagens identificadas na literatura (HOMCHUENCHOM *et al.*, 2012). O uso de questionários é uma abordagem barata computacionalmente, uma vez que não

há necessidade de processamentos complexos para chegar a um resultado. Entretanto, os artigos avaliados no mapeamento sistemático que disponibilizaram uma abordagem utilizando esse método possuem uma limitação que pretende ser sanada com a abordagem proposta. A evolução dos formulários dependiam de novas iterações dos trabalhos, ficando em muitos casos indicados como trabalhos futuros. Utilizando metamodelagem, foi criada uma estrutura em que usuários experientes podem criar seus próprios questionários e até combinar itens entre formulários de profissionais diferentes, flexibilizando a sua utilização e removendo a dependência de uma iniciativa da academia em fornecer os formulários.

No outro método disponibilizado, a flexibilidade é maior, pois utiliza algoritmos de transformações, que, quando aplicados aos modelos criados na abordagem, permitem realizar algumas análises adicionais. Extensões da estrutura criada nesta dissertação também podem ser realizadas com o intuito de agregar informação útil para a realização de diferentes abordagens de avaliar a conformidade entre processos e modelos de maturidade.

Como exemplos de análises que podem ser realizadas por meio da aplicação de transformações customizáveis, pode-se citar os itens detalhados a seguir:

1. Avaliação dos termos presentes na taxonomia

- O pacote de metamodelo *Taxonomy* tem como objetivo concentrar termos comuns utilizados na descrição de processos e modelos de maturidade e, com isso, fornecer um mecanismo para identificar conceitos presentes no modelo de maturidade e que façam parte da descrição dos elementos presentes na instância de processo utilizada pela empresa. Por exemplo, em relação a área de processo Gerência de Requisitos do modelo MPS.Br, os termos que facilmente são utilizados para descrever suas atividades são requisitos, análise, elicitação, documentação de usuário, entrevistas, requisitos de usuários, entre outros. Uma simples abordagem para avaliar a instância do processo que implementa essa área de processo do modelo seria avaliar se as atividades descritas na instância do processo possuem em sua descrição algum dos termos identificados na taxonomia. Caso forem identificados, pode deduzir que existe uma correlação entre ambos; caso contrário, uma análise mais detalhada (utilizando questionários, por exemplo) deverá ser realizada;

2. Avaliação da execução de atividades

- Outra abordagem seria analisar se todas as atividades descritas na instância do processo foram executadas. Apesar de não ser escopo desta proposta fornecer mecanismos para planejamento, monitoramento e execução de projetos (para isso, os usuários devem utilizar ferramentas próprias para o gerenciamento de seus projetos, como o Microsoft *Project*), a entidade *ProcessElementExecution* pode ser utilizada

para indicar quais atividades foram executadas e, a partir dela, verificar quais atividades ficaram fora da execução do projeto. Esse tipo de análise é importante pois no processo de avaliação de conformidade, um dos itens inspecionados pelos avaliadores dos modelos de maturidade é a execução das atividades previstas no processo. A não execução de uma atividade descrita em processo pode caracterizar uma não conformidade, comprometendo o atendimento a alguma prática específica ou genérica do modelo.

5.7 Algoritmos de Transformações

As transformações de modelos são uma ferramenta utilizada pela metodologia MDE para refinar os níveis de abstração dos metamodelos em diferentes contextos. Dessa forma, a modelagem pode ser iniciada por um modelo mais abrangente, como um diagrama de classes UML, a partir do qual são realizados refinamentos (transformações) em que seja possível derivar modelos que representem, por exemplo, um diagrama de entidades relacionamento (BRAMBILLA; CABOT; WIMMER, 2012).

Existem dois tipos de transformações que podem ser realizadas em MDE: (i) *model-to-model*, que consiste em algoritmos aplicados a um modelo de entrada que resultará em um modelo de saída, ou seja, entrada e saída desse tipo de transformação requerem um modelo para ser manipulado. Para esse tipo de transformação, o EMF Eclipse disponibiliza o *plug-in* ATL descrito na Seção 2.6; e (ii) *model-to-text*, em que são realizadas as transformações de modelo para texto, ou seja, um modelo que represente um domínio de uma arquitetura de desenvolvimento de software que pode ser convertido em código fonte para a construção da aplicação. Outra possibilidade para a aplicação de transformações *model-to-text* é a utilização de um modelo para a geração de relatórios em texto, com o intuito de sintetizar informações contidas nos atributos dos objetos dos modelos.

Alguns dos algoritmos propostos pela abordagem, no intuito de avaliar as informações contidas nos modelos criados com a estrutura da abordagem proposta, são detalhados a seguir:

1. Algoritmos de transformações *model-to-model*

- Avaliação do questionário (pseudo-código disponível no Algoritmo 1)
 - Modelo de entrada: *SubjectiveAssessment*
 - Modelo de saída: *Report*
 - Objetivo: Avaliar as respostas fornecidas pelo usuário ao responder o formulário de avaliação de conformidade com modelos de maturidade. Ao final, um modelo *Report* será gerado com os resultados da avaliação.
 - Processamento: Para cada objeto da classe *ComplianceItem*, serão avaliadas as respostas fornecidas pelo usuários. A cada resposta coletada, uma média será

calculada a partir do atributo *score* da classe *AnswerOption* e o seu peso no item de conformidade, definido pelo atributo *weight* da classe *Question*. Após calculada a média das respostas às questões do item de conformidade, o *score* obtido será avaliado de acordo com as faixas de conformidade estabelecidas nos objetos da classe *ComplianceTypeRule*. Ao iniciar o processamento do questionário, um objeto da classe *Report* deverá ser criado com as informações da avaliação em andamento. Após processado o item de conformidade, um objeto da classe *ReportItem* será criado com as informações coletadas. Esse processo se repete para todos os objetos criados da classe *ComplianceItem* até que o questionário seja processado totalmente;

- Avaliação de termos da taxonomia (pseudo-código disponível no Algoritmo 2)
 - Modelo de entrada: *Taxonomy* e *StructureAndBehavior*
 - Modelo de saída: *Report*
 - Objetivo: Avaliar se existem termos contidos na taxonomia definida a partir de um modelo de maturidade na instância de processo sendo avaliada.
 - Processamento: Ao iniciar o processamento dos elementos da instância de processo, um objeto da classe *Report* deverá ser criado com as informações da avaliação em andamento. Para cada objeto das classes *Activity*, *Role* e *WorkProduct*, será analisado se seus atributos *name* e *description* possuem algum termo definido na taxonomia. Se nenhum dos termos for encontrado em seus respectivos sinônimos, um objeto da classe *ReportItem* será criado indicando por meio de seus atributos qual atividade da instância de processo não possui relação com os conceitos chave definidos para o modelo de maturidade na taxonomia. Caso todos os objetos das classes *Activity*, *Role* e *WorkProduct* possuam termos presentes na taxonomia utilizada na análise, pode-se concluir que todos possuem relação com os conceitos chave definidos para o modelo de maturidade na taxonomia;

2. Algoritmos de transformações *model-to-text*

- Síntese dos resultados
 - Modelo de entrada: *Report*
 - Texto de saída: Relatório em texto sintetizando os resultados da avaliação.
 - Objetivo: Criar um relatório em texto dos resultados obtidos na avaliação de conformidade entre um modelo de maturidade e uma instância de processo.
 - Processamento: Um cabeçalho com os dados da avaliação será formatado com os atributos presentes no objeto da classe *Report*. Em seguida, para cada objeto da classe *ReportItem* vinculado ao objeto da classe *Report* em processamento,

são formatados os dados presentes nos atributos do objeto em processamento. Se não houver objetos da classe *ReportItem* instanciados, assume-se que não foram identificadas não conformidades na avaliação e somente os dados de cabeçalho serão formatados.

Algoritmo 1: Algoritmo de transformação *model-to-model* **Avaliação do questionário.**

```

Input: subjectiveAssessment SubjectiveAssessment
Output: report Report
  // Os atributos do objeto de saída são inicializados
1 report.date ← Date()
2 report.evaluationType ← 'Avaliação baseada em questionário'
3 report.maturityModel ← subjectiveAssessment.maturityModel
4 report.maturityLevel ← subjectiveAssessment.questionnaire.maturityLevel
5 report.processArea ← subjectiveAssessment.questionnaire.processArea
6 report.processInstance ← subjectiveAssessment.process
7
  // Uma lista com os itens de conformidade é instanciada
8 complianceItens ← subjectiveAssessment.questionnaires.complianceItens
9
10 for i = 0; i <= complianceItens.Size() - 1; i++ do
11   score ← 0
12
13   // Para cada questão presente no item de conformidade
14   // Será calculado o score de acordo com a nota e peso da questão
15   for j = 0; j <= complianceItens[i].question.Size() - 1; j++ do
16     score ← complianceItens[i].question[j].userAnswer.score *
17     complianceItens[i].question[j].weight
18
19   // Após calculado o score do item de conformidade
20   // É avaliado em qual range de tipos de conformidade se adequa
21   for j = 0; j <= complianceItens[i].complianceTypeRule.Size() - 1; j++ do
22     if complianceItens[i].complianceTypeRule[j].minValue <= score OR
23     complianceItens[i].complianceTypeRule[j].maxValue >= score then
24       complianceItens[i].complianceType ←
25       complianceItens[i].complianceTypeRule[j]
26
27   // Por fim são criados os itens do relatório com os dados
28   // avaliados
29   for j = 0; j <= complianceItens[i].question.Size() - 1; j++ do
30     reportItem.complianceItemEvaluated ← complianceItens[i].description
31     reportItem.question ← complianceItens[i].question[j].question
32     reportItem.userAnswer ← complianceItens[i].question[j].userAnswer.option
33     reportItem.complianceLevel ← complianceItens[i].complianceType.name
34     report.addReportItem(reportItem)
35
36 return report

```

Algoritmo 2: Algoritmo de transformação *model-to-model* **Avaliação de termos da taxonomia.**

```

Input: taxonomy Taxonomy
Input: structureAndBehavior StructureAndBehavior
Output: report Report
  // Os atributos do objeto de saída são inicializados
1  report.date ← Date()
2  report.evaluationType ← 'Avaliação baseada em script de transformação customizado'
3  report.maturityModel ← subjectiveAssessment.maturityModel
4  report.maturityLevel ← subjectiveAssessment.questionnaire.maturityLevel
5  report.processArea ← subjectiveAssessment.questionnaire.processArea
6  report.processInstance ← subjectiveAssessment.process
  // Uma lista com os objetos das classes TaxonomyElement, Activity,
  // Role e WorkProduct são instanciados
7  roles ← structureAndBehavior.activities.roles
8  activities ← structureAndBehavior.activities
9  workProducts ← structureAndBehavior.activities.workProducts
10 taxonomyElements ← taxonomy.taxonomyElements
  // Para cada objeto da classe Activity é avaliado se existe um termo
  // correspondente nos elementos ou sinônimos definidos na taxonomia
11 for i = 0; i <= activities.Size() - 1; i++ do
12   references ← []
13   for j = 0; j <= taxonomyElements.Size() - 1; j++ do
14     if activities[i].name.contains(taxonomyElements[j].name) OR
15     activities[i].description.contains(taxonomyElements[j].name) then
16       references.add(taxonomyElements[j].name)
17     else
18       for k = 0; k <= taxonomyElements.synonyms.Size() - 1; k++ do
19         if activities[i].name.contains(taxonomyElements[j].synonyms[k].name)
20         OR
21         activities[i].description.contains(taxonomyElements[j].synonyms[k].name)
22         then
23           references.add(taxonomyElements[j].synonyms[k].name)
24   // Se não foi encontrado nenhum termo relacionado na taxonomia
  // um item do relatório é criado
  if references.isNull then
    reportItem.itemDescription ← activities[i].name
    reportItem.evaluationDescription ← 'Análise de termos relacionados'
    reportItem.evaluationResult ← 'Termos relacionados identificados: ' +
      references.toString()
    report.addReportItem(reportItem)

  // Processamento é repetido para os objetos activities.roles
  // e para os objetos activities.workProducts...
25 return report

```

5.8 Modelagem de Casos de Uso

Com o intuito de disponibilizar uma modelagem de referência para possíveis implementações da abordagem proposta neste capítulo, foram criados diagramas de casos de uso *UML*. Os diagramas descrevem o comportamento esperado entre os atores definidos na abordagem (Oráculo e usuário comum) e as funções propostas para cada pacote de metamodelos disponível.

Na Figura 36, é apresentado um diagrama de casos de uso sugerindo as funções envolvidas entre os pacotes *Taxonomy*, *Process Template*, *ProcessInstance*, *MaturityModel* e os respectivos *StructureAndBehavior* de cada pacote relacionado. Nele, foram representados principalmente os casos de uso relacionados à criação dos modelos com as informações dos modelos de maturidade, processos (instância e *template*) e taxonomia. Um fator que pode ser destacado no diagrama é a separação de responsabilidades entre os atores *Oracle* e *User*. Para o primeiro ator, são acessíveis os casos de uso relacionados à gestão (funções de *CRUD*, ou seja, criação, recuperação, atualização e remoção) dos modelos *MaturityModel*, *ProcessTemplate* e *Taxonomy* e, para o segundo, somente a definição do *ProcessInstance* está disponível nesse momento. Outro ponto importante de ser notado no diagrama é o caso de uso *Define Structure And Behavior Elements*, incluído pelos casos de uso *Manage Process Template*, *Manage Process Instance*, *Define Generic Practices* e *Define Specific Practices*, em que será definido, a partir do caso de uso que o inclui, qual o tipo dos elementos criados, ou seja, se é um elemento **sugerido**, **template** ou **executado**. Por fim, o caso de uso *Manage Taxonomy* indica o momento em que o ator *Oracle* realiza a criação dos termos e sinônimos da taxonomia, ou seja, após definida a estrutura sugerida para um modelo de maturidade ou *template* de processo.

Figura 36 – Diagrama de casos de uso representando a etapa de definição dos modelos *Taxonomy*, *Process Template*, *ProcessInstance*, *MaturityModel* e os respectivos *StructureAndBehavior* da abordagem proposta.

Vale ressaltar que um usuário comum, em uma implementação da abordagem, poderia sim

criar modelos dos pacotes *MaturityModel*, *ProcessTemplate* e *Taxonomy*, porém seria desejável que os modelos criados fossem revisados pelo ator *Oracle*. Isso possibilitaria mais uma maneira de troca de experiências entre os profissionais em um ambiente social de compartilhamento de conhecimento em engenharia de software. Após definidos os modelos básicos para a execução de uma avaliação de conformidade, o usuário do ambiente está apto a executar os casos de uso ilustrados na Figura 37.

Figura 37 – Diagrama de casos de uso representando a etapa da avaliação e sugestão de melhorias dos modelos criados.

Nele, foram descritos os comportamentos esperados entre os pacotes de metamodelos *Assessment*, *Report* e *Improvement*. No Apêndice B, é disponibilizado um glossário com a descrição resumida de cada caso de uso apresentado. Alguns pontos importantes nesse diagrama devem ser ressaltados e estão detalhados a seguir.

1. Ao criar uma avaliação (caso de uso *Manage Assessment*), existem duas possibilidades de caso de uso a serem acessados, na qual pode ser criada uma avaliação baseada em questionário (em que são definidos os itens de conformidade, questões e tipos de conformidade) e uma baseada em *script* de transformação customizado. Para a segunda opção, deverá ser definido no ambiente computacional alguns vínculos não previstos na especificação dos pacote de metamodelos, uma vez que algoritmos customizados podem utilizar modelos diferentes como entrada ou até mesmo extensões dos modelos criados;
2. Ao executar uma avaliação, o modelo de maturidade e a instância de processo são definidos;

3. O ator *Oracle* herda as responsabilidades do ator *User*, indicando que ambos podem acessar os casos de uso definidos no diagrama;
4. Ao criar uma sugestão de melhoria, o ambiente deve enviar uma notificação ao usuário que criou a instância de processo avaliada. Na sequência, ao avaliar a sugestão recebida, o usuário pode dar o seu *feedback*, avaliando a sugestão e iniciando uma discussão, respondendo a notificação recebida que, consequentemente, receberá uma notificação sobre o retorno de sua sugestão;
5. Ao consultar as sugestões de melhoria disponíveis para a avaliação de conformidade executada, o usuário poderá iniciar modificações na sua instância de processo (extensão do caso de uso *Manage Process Instance*), iniciando em seguida o caso de uso *Execute Assessment*.

5.9 Considerações Finais

A criação de abordagens para a realização de atividades de SPA e SPI pode ser considerada uma tarefa desafiadora. Durante o processo de desenvolvimento da proposta descrita neste Capítulo, alguns aspectos foram levados em consideração, baseados nos trabalhos relacionados descritos no Capítulo 3. Muitos trabalhos baseados em ferramentas de apoio para SPA e SPI oferecem um ambiente estático para realizar avaliações de conformidade, muitas vezes limitados a um modelo e níveis específicos de maturidade. Esse fato ocorre porque as pesquisas são geralmente direcionadas a atender necessidades de empresas específicas, sem que sejam extrapolados para ambientes mais amplos, em que qualquer empresa possa usufruir da solução proposta (MONTONI *et al.*, 2006). É possível que muitas das propostas tenham a capacidade de serem aplicadas em ambientes diferentes dos que foram avaliados, mas isso não foi observado nas conclusões dos trabalhos; pelo contrário, esse fato era indicado como limitação ou motivação para trabalhos futuros (HOMCHUENCHOM *et al.*, 2012).

A partir disso, um conceito interessante identificado no mapeamento sistemático foi harmonização dos modelos, fornecendo uma maneira diferente de representar modelos de maturidade e avaliar itens de conformidade presentes em modelos diferentes (MUSAT *et al.*, 2010) (JENERS; LICHTER; ROSENKRANZ, 2013). Dessa forma, o uso de MDE foi escolhido para agregar na abordagem uma forma de harmonizar os modelos de maturidade. A taxonomia, apesar de simples, ofereceu um mecanismo de classificação dos termos e possibilitou a criação de um repositório de sinônimos. Com isso, possibilitou-se a identificação de conceitos semelhantes em diferentes modelos de maturidade, bastando modelar uma taxonomia que preveja conceitos existentes presentes em diferentes modelos e associá-los por meio de sinônimos. Outra abordagem seria criar uma transformação em que fosse possível comparar uma taxonomia com outra, cada taxonomia especializada nos conceitos de um modelo de maturidade específico, e identificar conceitos semelhantes pelos sinônimos definidos.

Uma das contribuições da abordagem proposta é a flexibilidade oferecida com o uso dos metamodelos. Ao longo deste Capítulo foram descritos exemplos de aplicações para diferentes finalidades, possibilitando variedade de análises que podem ser criadas a partir da estrutura fornecida. Esse fator se destaca quando comparado com as limitações apresentadas nos trabalhos relacionados no Capítulo 3, pois na maioria dos casos analisados, as metodologias criadas focavam apoiar uma empresa ou grupo de interesse específico, gerando uma lacuna de abordagens que propunham uma cooperação entre a comunidade de engenharia de software de forma independente.

Adicionalmente à proposta de pacotes de metamodelos, foi apresentada uma modelagem de diagramas de casos de uso, com o objetivo de fornecer uma visão geral sobre o comportamento esperado em um ambiente computacional que implemente a abordagem proposta. O detalhamento resumido dos casos de uso está disponível no Apêndice B. Por fim, como não foi possível identificar uma empresa para o processo de avaliação da abordagem, no Capítulo 6, é apresentado um estudo de caso elaborado com o intuito de exemplificar um cenário de avaliação com alguns requisitos de modelos de maturidade.

ESTUDO DE CASO

6.1 Considerações Iniciais

Neste Capítulo, é apresentado um estudo de caso elaborado para simular a aplicação dos modelos criados na abordagem proposta em um ambiente de avaliação de modelos de maturidade e processos de desenvolvimento de software. Nas Seções a seguir, são apresentados exemplos de modelagem utilizados para ilustrar as instâncias dos modelos criados a partir dos pacotes de metamodelos definidos.

Novamente, como a forma com que o Eclipse disponibiliza a visualização dos modelos (propriedades são visualizadas em uma aba específica na IDE), em que não é possível visualizar os atributos dos objetos no editor de modelos diretamente (num estrutura em árvore, por exemplo), para o estudo de caso apresentado, os modelos são disponibilizados na forma de figuras de suas instâncias visuais no Eclipse. Para que o leitor possa consultar os pacotes de metamodelos e os modelos de exemplo criados na elaboração do estudo de caso, apresentado nas seções a seguir, o *workspace* do Eclipse foi disponibilizado no *GitHub* para consulta. A URL para consulta ao repositório criado é <<https://github.com/dfeloni/DissertacaoMestrado.git>>.

6.2 Planejamento

Embora algumas empresas que participaram do survey apresentado no Capítulo 4 terem indicado disponibilidade para participar de avaliação da abordagem, por diversos motivos isso não foi possível. Dessa forma, a avaliação da abordagem foi realizada por meio de um estudo de caso apresentado nesta seção. O estudo de caso foi planejado seguindo os passos detalhados a seguir.

1. Definição do modelo de maturidade

- Para o estudo de caso conduzido, foi escolhido o modelo MPS.Br, em razão de seu desenvolvimento nacional e por ser um modelo bastante aceito no mercado brasileiro. Sua estrutura possui mais níveis de maturidade que o CMMI, permitindo uma evolução em estágios de esforço mais flexível para as empresas, principalmente as de pequeno e médio porte (SOFTEX, 2016).

2. Escolha dos elementos a serem avaliados

- Para o estudo de caso apresentado neste Capítulo foi selecionada a área de processo Gerência de Requisitos em sua implementação do nível G de maturidade. Nesse estudo de caso, é avaliada a prática específica (SOFTEX, 2013) GRE1, na qual é avaliado pelo modelo se o entendimento dos requisitos é obtido junto aos fornecedores de requisitos. Adicionalmente, foi realizada a modelagem do requisito equivalente do modelo CMMI. Com esse exemplo, é ilustrada a flexibilidade de modelagem proporcionada pelo metamodelo *MaturityModel* e exemplificada a utilidade do pacote *Taxonomy* em se destacar conceitos definidos semelhantes existentes nos dois modelos, mas descritos utilizando termos distintos pelos modelos;

3. Criação do questionário de avaliação

- Um questionário foi elaborado para avaliar cada um dos itens selecionados na etapa anterior;

4. Definição da estrutura de taxonomia

- Uma modelagem de taxonomia foi elaborada para exemplificar a identificação de termos chave nos itens selecionados para avaliação;

5. Simulação dos cenários possíveis para as respostas ao questionário

- Com base nas questões elaboradas para o estudo de caso alguns cenários são apresentados, ilustrando situações que podem ser encontradas em ambientes reais de avaliação;

6. Execução da avaliação

- Os algoritmos de transformação são executados;

7. Apresentação dos resultados

- Um relatório sintetizando os resultados obtidos é apresentado, criado a partir da execução do algoritmo de transformação definido na Seção 5.7.

6.3 Modelos *MaturityModel*, *StructureAndBehavior* e *Taxonomy*

O exemplo de modelagem do modelo de maturidade foi apresentado na Figura 34 do Capítulo 5. A partir disso, foi criado um modelo *StructureAndBehavior* que represente atividades executadas na fase de iniciação de um processo de Gerência de Requisitos. A divisão de processos em fases é comum, pois divide a estrutura de trabalho nas diferentes etapas de execução de um projeto. Para a prática específica GRE1, que define se os requisitos de sistema foram obtidos com apoio dos fornecedores de requisitos, foi criado um modelo em que é sugerida uma atividade para ser executada na iniciação do projeto com o intuito de agendar, de executar, de documentar e de obter aprovação dos requisitos definidos. Para isso, foram definidos os seguintes objetos: (i) um *Role*, definindo o papel de Analista de Requisitos; (ii) dois *WorkProduct*, definindo um documento de entrada e um documento de saída para a atividade; e (iii) três *Task*, descrevendo os passos definidos para a execução da atividade. Na Figura 38, é apresentada a instância do modelo no Eclipse.

Figura 38 – Arquivo de instância do modelo *StructureAndBehavior* criado para a fase de iniciação do processo de GRE no Eclipse.

Seguindo o processo de modelagem definido pela abordagem (ver Seção 5.2, o próximo modelo criado foi o *Taxonomy*. Nele, foram definidos os termos **requisito** e **fornecedor** e sugeridos sinônimos que podem ser empregados para descrição do conceito que o termo representa nas modelagens de instâncias de processos. Na Figura 39 é apresentado o modelo descrito.

Para mostrar um exemplo de modelagem de outro modelo de maturidade, é apresentado na Figura 40 um exemplo de modelagem utilizando o modelo CMMI. Nele, são apresentadas as práticas específicas e genéricas equivalentes à prática GRE1 utilizada neste estudo de caso. Na Tabela 12, é apresentada a correlação entre os modelos para a prática GRE1. Uma definição necessária que deve ser destacada é a diferença de nomenclatura entre os dois modelos. Para o MPS.Br, são definidas **práticas** específicas e genéricas, enquanto no CMMI são definidos **objetivos** específicos e genéricos, conforme pode ser notado na nomenclatura utilizada na Tabela 12 e no manual disponibilizado pela Softex para a realização de avaliações MPS.Br em conjunto com um *appraisal* CMMI (SOFTTEX, 2012). Outro fator interessante que pode ser destacado neste exemplo é a utilização de termos diferentes para definir um mesmo conceito. Na descrição da

Figura 39 – Arquivo de instância do modelo *Taxonomy* criado para o processo de GRE no Eclipse.

prática específica GRE1 do MPS.Br, é utilizado o termo **fornecedores de requisitos**, para indicar o papel da pessoa responsável por prover os requisitos. No modelo CMMI, o mesmo conceito é definido como **provedores de requisitos**. Esse é um exemplo claro da falta de compatibilidade entre os termos utilizados por diferentes modelos e como a taxonomia, utilizando sinônimos, poderia ajudar a minimizar esse problema.

Figura 40 – Arquivo de instância do modelo *MaturityModel* criado para exemplificar uma modelagem do CMMI.

Tabela 12 – Práticas equivalentes entre MPS.Br e CMMI para o resultado de processo GRE1.

Prática genérica MPS.Br		Objetivo genérico - CMMI	
AP 1.1	Este atributo evidencia o quanto o processo atinge o seu propósito.	SG 1	Os requisitos são gerenciados e as inconsistências são identificadas em relação aos planos de projeto e produtos de trabalho.
Prática específica MPS.Br		Objetivo específico - CMMI	
GRE1	O entendimento dos requisitos é obtido junto aos fornecedores de requisitos.	SP 1.1	Trabalhar com os provedores de requisitos para obter um melhor entendimento do significado dos requisitos.

6.4 Modelos *Project*, *Process* e *StructureAndBehavior*

Nas Figuras 41 e 42, é exemplificada a utilização dos modelos *Project*, *Process* e *StructureAndBehavior*. Na Figura 41, é possível notar a presença de dois modelos no mesmo

arquivo. Isso ocorre pela necessidade de carregar o modelo *Project* como um *resource* do modelo *ProcessInstance*, uma vez que existe uma relação de dependência entre eles. Após indicado o modelo do projeto, ele deve ser vinculado ao objeto do processo por meio de seu atributo de relacionamento. Na Figura 42, estão presentes os elementos de uma sugestão de atividade para atender ao requisito GRE1 do modelo MPS.Br. Nela, é sugerida uma atividade para ser executada na fase de iniciação do processo de GRE para que seja garantida a realização da coleta dos requisitos do projeto com os fornecedores dos requisitos. Há também outro detalhe importante, enquanto na estrutura sugerida de atividade descrita na Seção 6.3, o fornecedor de requisitos está representado de forma implícita na descrição da atividade e tarefas definidas, na instância de processo, ele é representado por um objeto *Role* e é definido como *stakeholder* ao invés de fornecedor. Essa particularidade foi adicionada a esse exemplo para mostrar uma possível indicação do uso da taxonomia. Enquanto o Oráculo pode ter criado um processo referenciando seu fornecedor de requisitos de forma textual, um usuário pode modelar a presença desse papel utilizando outro termo, apontado na taxonomia criada pelo Oráculo como sinônimo. Outro detalhe oculto na representação do Eclipse é o atributo *instanceType* ser definido no modelo de maturidade como *SUGGESTED* e na instância de processo como *EXECUTED*.

Figura 41 – Arquivo de instância do modelo *Project* e *ProcessInstance* criado para o processo de GRE no Eclipse.

Figura 42 – Arquivo de instância do modelo *StructureAndBehavior* criado para o processo de GRE no Eclipse.

6.5 Modelo Assessment, Report e Improvement

Nesta Seção, é apresentada uma modelagem do pacote *Assessment* utilizando o subpacote de metamodelos *SubjectiveAssessment* para criar um formulário que avalie a prática específica

GRE1 do processo de GRE do modelo MPS.Br no nível G de maturidade. A saber, o requisito GRE1 visa avaliar se a organização coleta o escopo do projeto com o fornecedor de requisitos, comumente identificado como patrocinador do projeto ou *stakeholder*.

Em um processo de avaliação real de um modelo de maturidade, o trabalho dos avaliadores consiste em coletar evidências de que as atividades, papéis e produtos de trabalho são realmente executados nos projetos da empresa e se estão de acordo com o que é definido nos resultados esperados de processo do modelo (SOFTEX, 2015). Ou seja, se existem evidências que comprovem a execução de um elemento previsto na documentação do processo da empresa, ele é considerado como executado, faltando analisar se está de acordo com o que é esperado pelo modelo. De nada adianta definir atividades no processo que não se encaixem no contexto do que é exigido pelo modelo, porque fatalmente a avaliação não obterá um resultado positivo. Na Figura 43, é apresentado o modelo *Assessment* criado para ilustrar a modelagem do formulário de avaliação para o requisito GRE1.

Figura 43 – Arquivo de instância do modelo *Assessment* criado para o processo de GRE no Eclipse.

Para esse exemplo, foram criadas três questões com o intuito de avaliar os seguintes fatores: (i) se os fornecedores de requisitos participam do processo de entendimento dos requisitos; (ii) se após realizado o levantamento dos requisitos, eles são documentados os pontos definidos; e (iii) se após a elaboração do documento de escopo do projeto, ele é enviado para aprovação por parte dos fornecedores de requisitos. Para a primeira questão, foi definido peso de 0.5 ponto, pois ela representa a maior parte do que se quer avaliar na GRE1. As segunda e terceira questões receberam peso de 0.25 ponto cada, pois representam um refinamento do que foi questionado na primeira questão. Na Tabela 13, são apresentados os possíveis cenários identificados e os resultados apresentados foram baseados na execução do algoritmo de transformação apresentado na Seção 5.7.

A estrutura criada para a elaboração dinâmica de questionários para avaliação de modelos de maturidade proporciona flexibilidade não identificada nos trabalhos relacionados apresentados

Tabela 13 – Cenários identificados ao responder as questões definidas no modelo *SubjectiveAssessment*.

#	Descrição	Resultado
1	Não é realizado levantamento de requisitos junto ao fornecedor de requisitos.	Nesse caso, o relacionamento <i>result</i> será vinculado ao tipo de conformidade <i>Não atendido</i> (objeto da classe <i>ComplianceType</i>).
2	O levantamento de requisitos é realizado junto ao fornecedor de requisitos, porém de maneira informal e sem muita documentação. Essa conclusão é obtida caso o usuário utilize a segunda opção de respostas das questões 2 e 3 do formulário.	Nesse caso, o relacionamento <i>result</i> será vinculado ao tipo de conformidade <i>Atendido parcialmente</i> (objeto da classe <i>ComplianceType</i>).
3	O levantamento de requisitos é realizado junto ao fornecedor de requisitos, de maneira formal e mas não há evidência de aprovação do escopo definido em documentação do projeto. Essa conclusão é obtida caso o usuário utilize a terceira opção de resposta da questão 2 e a segunda opção de resposta para a questão 3 do formulário.	Nesse caso, o relacionamento <i>result</i> será vinculado ao tipo de conformidade <i>Atendido parcialmente</i> (objeto da classe <i>ComplianceType</i>).
4	O levantamento de requisitos é realizado junto ao fornecedor de requisitos, de maneira formal e com registro em documentação do projeto. Essa conclusão é obtida caso o usuário utilize a terceira opção de respostas das questões 2 e 3 do formulário.	Nesse caso, o relacionamento <i>result</i> será vinculado ao tipo de conformidade <i>Totalmente atendido</i> (objeto da classe <i>ComplianceType</i>).

no Capítulo 3. Entretanto, sua aplicação é dependente de profissionais de engenharia de software experientes e que estejam dispostos a compartilhar seus conhecimentos com a comunidade. Um fator positivo que pode ser considerado é, a partir do momento em que um usuário começar a construir uma reputação positiva dentro de um ambiente computacional social que implemente a abordagem proposta neste trabalho, ele pode receber ofertas de trabalho.

Após realizada a avaliação das respostas do questionário criado, os modelos *Report* e *Improvement* são instanciados para representar, respectivamente, o resultado da avaliação e as sugestões de melhoria fornecidas pelos usuários da abordagem. A formação do relatório pode possuir diferentes *layouts* de acordo com algum algoritmo de transformação definido (ver Seção 5.7). Na Tabela 14, são fornecidas algumas sugestões de melhoria para o processo modelado e avaliado de acordo com as situações previstas na Tabela 13.

Tabela 14 – Sugestões para os cenários identificados ao responder as questões definidas no modelo *SubjectiveAssessment*.

#	Sugestão de melhoria
1	Deverá ser criada no processo executado pela empresa uma atividade em que seja realizado o entendimento dos requisitos do projeto com os fornecedores de requisitos. Isso deve ser realizado de maneira formal, de preferência com atas que comprovem a execução e os participantes do levantamento. Após isso, um documento de definição de escopo do projeto deverá ser elaborado e enviado para aprovação junto aos <i>stakeholders</i> .
2	A parte mais importante do GRE1 é realizada pelo processo definido pela empresa. Adicionalmente, a empresa deverá elaborar um documento de definição de escopo do projeto que deverá ser enviado para aprovação dos <i>stakeholders</i> .

Tabela 15 – Sugestões para os cenários identificados ao responder as questões definidas no modelo *SubjectiveAssessment*. (Continuação)

#	Sugestão de melhoria
3	Adicionar, ao final da fase de iniciação do processo de GRE, o envio do documento de definição de escopo para aprovação dos <i>stakeholders</i> e adicioná-lo à documentação do projeto para ter uma evidência que comprove execução da atividade.
4	O item GRE1 é atendido pelo processo definido pela empresa. Entretanto, a empresa poderia adicionar ao processo uma atividade em que é realizado o encerramento da fase de iniciação do processo de GRE. Isso seria utilizado para comprovar que um marco do projeto foi atingido e poderia ser utilizado como evidência de monitoramento do projeto pelo processo de Gerência de Projetos (GPR), também definido no nível G de maturidade do MPS.Br.

6.6 Considerações Finais

Neste Capítulo, foi apresentado um estudo de caso para exemplificar as etapas de modelagem e de avaliação de conformidade entre uma instância de processo e um modelo de maturidade. Outros fatores apresentados no estudo de caso foram: (i) exemplificar a modelagem de um segundo modelo de maturidade, o modelo CMMI, em que foi ilustrado utilizando o objetivo específico SP 1.1, equivalente ao resultado esperado de processo GRE1 do MPS.Br; (ii) indicar diferentes definições de conceitos semelhantes para uma mesma prática existente nos dois modelos, exemplificando a utilização do metamodelo *Taxonomy*; e (iii) mostrar a ordem de execução das etapas definidas pela abordagem para sua utilização.

No exemplo utilizado, foi selecionado o resultado esperado de processo GRE1 contido no processo Gerência de Requisitos do modelo MPS.Br para o nível G de maturidade. Conforme exemplificado, caso fosse desejado avaliar outros resultados esperados de processo bastaria adicionar ao modelo *MaturityModel* mais estruturas de processos, utilizando o modelo *StructureAndBehavior* para indicar outras atividades que atendam ao resultado de processo desejado.

CONCLUSÕES E TRABALHOS FUTUROS

7.1 Contribuição

A definição, a implantação, a validação e o acompanhamento da execução de processos de desenvolvimento de software são tarefas complexas e exigem o envolvimento de pessoas hábeis e experientes no assunto. Conforme apresentado no Capítulo 3, a literatura tem concentrado esforços em propor metodologias que auxiliem as empresas, principalmente as de pequeno e médio porte, com suas limitações orçamentais, a conseguirem definir bons processos de desenvolvimento e torná-los aderentes às boas práticas estabelecidas por modelos de maturidade, seja ou não com intuito de certificação.

Na Seção 1.3, foram definidos os seguintes objetivos: (i) definir um conjunto de modelos de abstração (metamodelos) de modelos de maturidade para apoiar uma metodologia de avaliação/melhoria de processo de software com o objetivo de apoio à certificação; e (ii) permitir avaliar os processos de uma organização em comparação com um modelo de maturidade por meio de transformações desses metamodelos. Na avaliação do autor, ambos foram atingidos, conforme detalhado nas Seções presentes no Capítulo 5. Entretanto, alguns objetivos adicionais podem ser considerados como alcançados, se forem levados em consideração os *gaps* de pesquisa identificados na área de SPA e SPI, conforme apresentado na Seção 3.5 e detalhados a seguir.

1. Dependência com Modelo de Maturidade

- A definição do pacote de metamodelos *MaturityModel* permite representar modelos de maturidade compatíveis com a estrutura definida pelo MPS.Br. Tanto o CMMI quanto o MPS.Br possuem a mesma base técnica utilizada para a definição de sua estrutura, práticas específicas e genéricas e suas áreas de processo, sendo eles considerados compatíveis, uma vez que sua estrutura foi derivada de normas ISO/IEC como as normas ISO/IEC 15.504 e ISO/IEC 12.207 (SOFTEX, 2016). Ambos são

amplamente aceitos pela indústria de software, CMMI em nível internacional e o MPS.Br em nível nacional e na América Latina. Mesmo assim, existe a possibilidade de existirem modelos de maturidade com estruturas incompatíveis com a modelagem criada e não serem aderentes ao pacote *MaturityModel*. Nesse sentido, um dos itens identificados nos *gaps* de pesquisa apresentados no mapeamento sistemático (ver Seção 3.5) indica a falta de um manual de referência que padronize uma estrutura de trabalho para modelos de maturidade de processos de software. Nesse manual, seria definido um modelo de referência para os existentes e os que podem ser criados futuramente, evitando problemas de incompatibilidade entre modelos e dificultando a adoção de modelos variados de acordo com necessidades específicas (harmonização). Dessa forma, um dos *gaps* identificados, está relacionado à dependência das abordagens propostas pela literatura de um modelo de maturidade em específico, utilizado em sua elaboração. Adicionalmente, os pacotes de metamodelos *ProcessInstance* e *StructureAndBehavior* fornecem uma estrutura reutilizável para a modelagem de diversas possibilidades de implementação de um mesmo modelo de maturidade, proporcionando aos usuários da abordagem o poder da escolha, uma vez que haja um repositório em que se concentrem os modelos instanciados;

2. Reúso

- Por ser uma estrutura relativamente simples de se implementar e interpretar, os pacotes de metamodelos criados na abordagem permitem fácil distribuição desses modelos, em que, a partir de um ambiente computacional que a implemente, os modelos podem ser compartilhados livremente pelos usuários;

3. Envolvimento da Comunidade de Engenharia de Software

- A proposta fez uso de uma consulta à comunidade de engenharia de software (ver Capítulo 4) para conhecer suas experiências, quais as suas expectativas em relação à academia, quais expectativas possuem em relação a ambientes computacionais para execução de avaliação e melhoria de processos de software, além de identificar seus principais problemas ao propor novas abordagens. Um fator fundamental para a implementação da abordagem proposta é o ambiente computacional que a implemente possuir funções de compartilhamento e de socialização entre os usuários, possibilitando a troca de experiência no contexto de qualidade de software. Esse fator social, conforme descrito por um participante do *survey*, é precário no contexto de processos e modelos de maturidade, algo bem diferente quando comparado ao desenvolvimento e à codificação, em que fóruns espalhados pela *internet* possuem uma comunidade ativa e com iniciativas ousadas para o compartilhamento de código e de experiências;

4. Uso de MDE

- O uso de MDE foi o elemento central da abordagem proposta, pois ela fornece diversas possibilidades de implementação de metodologias para SPA e SPI. Não obstante, foi notável que nas metodologias propostas com o intuito de harmonizar modelos de maturidade, MDE foi empregada como elemento chave para a proposta elaborada pelos autores. A liberdade provida por essa ferramenta permite criar abstrações com objetivos específicos, aumentando a gama de possibilidades de sua aplicação;

5. Integração de Metodologias

- Esta dissertação faz parte de um projeto maior, em que está sendo desenvolvida uma arquitetura integrada para reúso de software (*AIRES*, Ambiente Integrado para Reúso de Engenharia de Software) (BRAGA *et al.*, 2016). Nesse projeto, o objetivo é fornecer um ambiente desenvolvido utilizando uma arquitetura orientada a serviços, um estilo de arquitetura de software cujo princípio fundamental prega que as funções implementadas pelas aplicações devem ser disponibilizadas na forma de serviços (SOA, 2015). Frequentemente, esses serviços são conectados utilizando um barramento de serviço (*Enterprise Service Bus*) que disponibiliza interfaces ou contratos acessíveis na Web Services ou outra forma de comunicação entre aplicações. A arquitetura SOA (*Service Oriented Architecture*) é baseada nos princípios da computação distribuída e utiliza o paradigma *request/reply* para estabelecer a comunicação entre os sistemas clientes e os sistemas que implementam os serviços disponibilizados. Uma proposta de integração entre a abordagem descrita nesta dissertação e a *AIRES* é detalhada na Seção 7.3 deste Capítulo;

6. Facilidade em utilizar modelos

- O uso de formalismos visuais para representar entidades de negócio é uma prática comum no dia-a-dia dos engenheiros de software. Como exemplo, pode-se citar o uso de *UML* para a modelagem de produtos de software. A metamodelagem não é uma tarefa de alta complexidade, pois pode ser comparada à criação de diagramas *UML*. Isso pode facilitar a adoção do uso da abordagem proposta, uma vez que a atividade de modelagem faz parte do cotidiano dos profissionais de software.

7.2 Limitações e Riscos à Validação

Inicialmente, o escopo desta dissertação abrangia a implementação de um ambiente computacional baseado em serviços *web*, em que seria desenvolvida uma camada pública de serviços disponíveis na nuvem para que usuários interessados pudessem fazer uso da abordagem

proposta. Essa utilização poderia ser dada pelo uso da ferramenta computacional disponibilizada ou os usuários poderiam criar suas próprias camadas de interface de usuário e utilizar os serviços *web* para modelar, avaliar e colher sugestões para seus processos. Após a fase de qualificação dessa dissertação, esses objetivos foram removidos, conforme sugerido pelos membros da banca, pois a criação da estrutura de representação de modelos de maturidade, processos, estrutura de classificação, estrutura de avaliação, síntese do resultado da avaliação e algoritmos de transformação seriam escopo suficiente para período de desenvolvimento de uma dissertação de mestrado. Portanto, uma primeira limitação do trabalho é a não existência de um ambiente computacional de apoio à análise de conformidade. Outras limitações identificadas na abordagem são detalhadas a seguir.

1. Não foi possível avaliar em ambiente de certificação de empresa

- Mesmo tendo realizado o *survey* e identificado possíveis interessados na avaliação da abordagem nenhuma empresa realmente participou do processo. As empresas contatadas justificaram a impossibilidade de participar da avaliação da abordagem por: (i) não ter aprovação da diretoria da empresa; (ii) a empresa suspender os projetos relacionados à melhoria de seus processos em razão de cortes no orçamento; e (iii) não haver recursos humanos disponíveis para participar do processo de validação. Portanto, uma possível continuação do trabalho seria a utilização da abordagem usando um ambiente real de definição de processos por uma empresa de desenvolvimento de software;

2. Dependência de usuários com experiência no assunto

- A definição das estruturas dos pacotes de metamodelos *Taxonomy*, *MaturityModel* e *StructureAndBehavior* pode ser considerada de complexidade alta. Na própria definição da abordagem, é descrita a necessidade de usuários com conhecimentos especiais no assunto (Oráculos). Dessa forma, existe a dependência de que usuários experientes estejam dispostos a compartilhar suas experiências e conhecimentos com os profissionais iniciantes no assunto, criando um ambiente de cooperação como é encontrado em fóruns de desenvolvimento;

3. A utilização dos metamodelos requer conhecimento técnico de ferramentas específicas e da metodologia MDSE

- O uso de MDE depende de ferramentas específicas, o que pode ser um problema para usuários que não conheçam o assunto e tenham receio em utilizar a abordagem por não conhecerem o aparato técnico necessário para utilizá-lo;

4. Estrutura de classificação de elementos para harmonização

- Para a abordagem proposta, foi utilizada uma estrutura simples de classificação de termos, o pacote de metamodelos *Taxonomy*, para prover uma iniciativa primitiva de harmonização de modelos a fim de mostrar o conceito de harmonização e as possibilidades que ela pode proporcionar. Entretanto, existem métodos de classificação mais eficazes, por exemplo o uso de *Ontologias*. Diferentemente da taxonomia, a ontologia fornece meios de dar sentido aos termos classificados em sua estrutura e uma vez que se tem o valor semântico de um termo fica mais simples correlacioná-lo com termos similares. O pacote de metamodelos *Taxonomy* possui a entidade *Synonym* com o intuito de fornecer uma forma de referenciar sinônimos que podem ser encontrados em diferentes modelos em um mesmo contexto de aplicação.

7.3 Trabalhos Futuros

A abordagem proposta nesta dissertação possui possibilidades de extensão que podem ser utilizadas por outros pesquisadores para avaliar ou melhorar a sua capacidade, ajudando a superar as limitações discutidas na Seção 7.2. Algumas sugestões identificadas para trabalhos futuros são listadas a seguir:

1. Implementação de ferramenta de apoio

- A implementação de ferramenta de apoio computacional se faz necessária para abstrair o uso dos metamodelos de forma manual no ambiente do *EMF Eclipse*. A ferramenta implementaria uma interface de usuário que fosse amigável aos usuários e que em seu *back-end* manipulasse os metamodelos e executasse os algoritmos de transformações descritos na Seção 5.7;

2. Integração com a *AIRES*

- Conforme descrito na Seção 7.1, esta dissertação faz parte de um projeto maior que está sendo desenvolvido pela orientadora desta dissertação. O objetivo é criar um ambiente computacional disponível na nuvem, que forneça um ambiente centralizado para concentrar ativos de reúso de engenharia de software dos mais diversos contextos de aplicação, sejam modelos de requisitos, componentes, processos de software, linhas de produto de software, entre outros (BRAGA *et al.*, 2016). Na Figura 44 o elemento “*Cert. Manager UI*” presente na camada *GUI* representa uma possível implementação de uma interface de usuário para a abordagem proposta nesta dissertação. A disponibilização de serviços *web* para a execução da abordagem está inerente a proposta da *AIRES*, conforme pode ser observado na camada *Public Services* que possui um vínculo com o módulo “*Cert. Manager UI*”;

3. Disponibilização de ambiente de compartilhamento social

- A *AIRES* prevê a disponibilização de seus ativos de reuso para todos os usuários do ambiente. No caso desta dissertação, serão persistidos no repositório de ativos da *AIRES* as instâncias de processos, modelos de maturidade e sugestões de melhoria fornecidas pelos usuários e que estariam disponíveis para consulta para os membros registrados do ambiente;

4. Avaliação em ambiente real de certificação

- Uma avaliação da abordagem proposta em um cenário de avaliação de modelos de maturidade em uma empresa de desenvolvimento de software é uma das possibilidades de extensão deste trabalho, pois em sua elaboração, não foi possível realizá-la. Com isso, poderiam ser avaliados aspectos como: (i) a receptividade por parte dos usuários em relação ao uso dos metamodelos; (ii) criação de novos algoritmos de transformações dos modelos; e (iii) possíveis ajustes para melhor aderência a cenários que podem não ter sido considerados no estudo de caso apresentado no Capítulo 6.

Figura 44 – Arquitetura do ambiente *AIRES*, retirado de (BRAGA *et al.*, 2016).

REFERÊNCIAS

- ALI, R. Z. R. M.; IBRAHIM, S. An application tool to support the implementation of integrated software process improvement for malaysia's sme. In: **2011 5th Malaysian Conference in Software Engineering, MySEC 2011**. [S.l.: s.n.], 2011. p. 177–182. Citado 3 vezes nas páginas 32, 79 e 95.
- BEYDEDA, S.; BOOK, M.; GRUHN, V. **Model-Driven Software Development**. [S.l.]: Springer-Verlag, 2005. ISBN 354025613X. Citado na página 48.
- BIZAGI. 2015. [On-line], *Bizagi - Business Process Management Solutions*. Disponível em <<http://www.bizagi.com/>> (Acessado em 20/03/2015). Citado na página 91.
- BPMN. 2015. [On-line], *BPMN - Business Process Model And Notation*. Disponível em <<http://www.omg.org/spec/BPMN/2.0/>> (Acessado em 20/03/2015). Citado na página 91.
- BRAGA, R. T. V.; FELONI, D.; PACINI, K.; FILHO, D. S.; GOTTARDI, T. Aires: an architecture to improve software reuse. In: **The 15th International Conference on Software Reuse (ICSR)**. [S.l.: s.n.], 2016. p. 231–246. Citado 4 vezes nas páginas 16, 139, 141 e 142.
- BRAMBILLA, M.; CABOT, J.; WIMMER, M. **Model-Driven Software Engineering in Practice**. 1. ed. ,USA: Morgan & Claypool, 2012. Citado 6 vezes nas páginas 15, 48, 49, 50, 51 e 121.
- CORREIA, F. F.; AGUIAR, A. Patterns of information classification. In: **Proceedings of the 18th Conference on Pattern Languages of Programs**. New York, NY, USA: ACM, 2011. (PLoP '11), p. 15:1–15:17. ISBN 978-1-4503-1283-7. Disponível em <<http://doi.acm.org/10.1145/2578903.2579153>>. Citado na página 94.
- ECLIPSE. 2015. [On-line], *The Eclipse Foundation*. Disponível em <<http://www.eclipse.org/>>, <<https://eclipse.org/modeling/emf/>>, <<http://www.eclipse.org/acceleo/>>, <<https://eclipse.org/at/>> (Acessado em 20/03/2015). Citado na página 53.
- FELONI, D. F. G.; BRAGA, R. T. V. Methodologies for evaluation and improvement of software processes in the context of quality and maturity models: A systematic mapping. In: **CIBSE 2015 - XVIII Ibero-American Conference on Software Engineering**. [S.l.: s.n.], 2015. p. 123–136. Citado na página 58.
- FREITAS, H.; OLIVEIRA, M.; SACCOL, A. Z.; MOSCAROLA, J. O método de pesquisa survey. **Revista de Administração**, v. 35, p. 105–112, 2000. Citado na página 83.
- GARCÍA, I.; PACHECO, C.; MENDOZA, E.; CALVO-MANZANO, J. A.; CUEVAS, G.; FELIU, T. S. Managing the software process with a software process improvement tool in a small enterprise. **Journal of software: Evolution and Process**, v. 24, n. 5, p. 481–491, 2012. Citado 6 vezes nas páginas 33, 57, 58, 77, 79 e 81.
- GAZEL, S.; SEZER, E. A.; TARHAN, A. An ontology based infrastructure to support CMMI-based software process assessment. **Gazi University Journal of Science**, v. 25, n. 1, p. 155–164, 2012. Citado na página 79.

HOMCHUENCHOM, D.; PIYABUNDITKUL, C.; LICHTER, H.; ANWAR, T. Spials: A light-weight software process improvement self-assessment tool. **Malaysian Conference in Software Engineering**, MySEC, p. 195–199, 2011. Citado 2 vezes nas páginas 34 e 79.

HOMCHUENCHOM, D.; PIYABUNDITKUL, C.; LICHTER, H.; ANWAR, T.; METHAWACHANANONT, A.; KROOTKAEW, C.; KRISANATHAMAKUL, T. Spials-ii: A light-weight software process improvement self assessment tool. **International Journal of Digital Content Technology and its Applications**, v. 6, n. 21, p. 16–26, 2012. Citado 10 vezes nas páginas 23, 62, 63, 65, 66, 79, 81, 82, 119 e 127.

ISO. 2013. [On-line], *ISO/IEC 15504*. Disponível em <<http://www.iso.org/iso/home.html>> (Acessado em 11/06/2013). Citado na página 32.

ISO/IEC. **ISO/IEC 9126. Software engineering – Product quality**. [S.l.]: ISO/IEC, 2001. Citado na página 41.

_____. **ISO/IEC 12207. Software engineering – Product quality**. [S.l.]: ISO/IEC, 2008. Citado na página 41.

JENERS, S.; LICHTER, H.; ROSENKRANZ, C. G. Efficient adoption and assessment of multiple process improvement reference models. **E-Informatica Software Engineering Journal**, v. 7, n. 1, p. 15–24, 2013. Citado 11 vezes nas páginas 15, 62, 63, 71, 72, 73, 77, 79, 80, 82 e 127.

JOHNSON, J. **My Life is Failure: 100 Things You Should Know to Be A Better Project Leader**, Standish Group International Publisher: Boston MA, 2006. Citado na página 57.

KASURINEN, J.; RUNESON, P.; RIUNGU, L.; SMOLANDER, K. **A self-assessment framework for finding improvement objectives with ISO/IEC 29119 test standard**. [S.l.: s.n.], 2011. v. 172. 25-36 p. (Communications in Computer and Information Science, v. 172). Citado na página 79.

KENT, S. Model driven engineering. In: **Proceedings - Integrated Formal Methods Lecture Notes in Computer Science**. [S.l.]: Springer Berlin Heidelberg, 2002. p. 286–298. Citado na página 48.

KITCHENHAM, B.; BRERETON, O. P.; BUDGEN, D.; TURNER, M.; BAILEY, J.; LINKMAN, S. Systematic literature reviews in software engineering - a systematic literature review. **Information and Software Technology**, v. 51, n. 1, p. 7–15, 2009. Citado na página 58.

LEPASAAR, M.; MAKINEN, T. Integrating software process assessment models using a process meta model. **IEEE Computer**, IEEE Computer Society, p. 224–229, 2002. Citado 2 vezes nas páginas 33 e 34.

MAS, A.; MESQUIDA, A. L. **A software tool to support the integrated management of software projects in mature SMEs**. [S.l.: s.n.], 2011. v. 172. 236-246 p. (Communications in Computer and Information Science, v. 172). Citado na página 79.

MONDRAGN, O. Addressing infrastructure issues in very small settings. **Proceedings of the First International Research Workshop for Process Improvement in Small Settings**, p. 5–10, 2006. Citado na página 57.

MONTONI, M.; SANTOS, G.; ROCHA, A. R.; FIGUEIREDO, S.; CABRAI, R.; BARCELLOS, R.; BARRETO, A.; SCARES, A.; CERDEIRAL, C.; LUPO, P. **Taba workstation: Supporting software process deployment based on CMMI and MR-MPS.BR**. [S.l.: s.n.], 2006. v. 4034 LNCS. 249-262 p. (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), v. 4034 LNCS). Citado 9 vezes nas páginas [62](#), [63](#), [67](#), [68](#), [76](#), [79](#), [81](#), [82](#) e [127](#).

MUSAT, D.; NO, V. C.; CALVO-MANZANO, J. A.; GARBAJOSA, J. **MATURE: A model driven based tool to automatically generate a language that supports CMMI process areas specification**. [S.l.]: Springer, 2010. v. 99 CCIS. 48-59 p. Citado 12 vezes nas páginas [15](#), [38](#), [48](#), [62](#), [63](#), [69](#), [70](#), [71](#), [79](#), [80](#), [82](#) e [127](#).

NIAZI, M.; WILSON, D.; ZOWGHI, D. A framework for assisting the design of effective software process improvement implementation strategies. **Journal of Systems and Software**, v. 78, n. 2, p. 204–222, 2005. Citado na página [79](#).

OKTABA, H.; PIATTINI, M. **Software Process Improvement for Small and Medium Enterprises: Techniques and Case Studies**, Information Science Reference Publisher: Hershey, New York, 2008. Citado na página [57](#).

PARDO, C.; PINO, F.; GARCÍA, F.; ROMERO, F. R.; PIATTINI, M.; BALDASSARRE, M. T. **HProcessTOOL: A support tool in the harmonization of multiple reference models**. [S.l.: s.n.], 2011. v. 6786 LNCS. 370-382 p. (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), PART 5). Citado na página [79](#).

PARK, E.; KIM, H.; LEE, R. Y. Software repository for software process improvement. In: . [S.l.: s.n.], 2008. (Studies in Computational Intelligence, v. 131), p. 51–64. Citado 6 vezes nas páginas [15](#), [63](#), [73](#), [74](#), [79](#) e [82](#).

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. Systematic mapping studies in software engineering. In: **Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering**. Swinton, UK, UK: British Computer Society, 2008. (EASE'08), p. 68–77. Citado na página [58](#).

PIYABUNDITKUL, C.; WONGCHINGCHAI, N.; METHAWACHANANONT, A. Step forward CMMI-project management by optimized Scrum, TGGS-RWTH Aachen-NECTEC. **Proceeding(s) of the 5th International Conference On Project Management (ProMAC 2010)**, 2010. Citado na página [34](#).

PRESSMAN, R. S. **Engenharia de Software**. 8. ed. New Jersey, EUA: Pearson Prentice Hall, 2014. Citado 6 vezes nas páginas [31](#), [33](#), [37](#), [38](#), [39](#) e [42](#).

ROUT, T. P.; TUFFLEY, A. Harmonizing iso-iec 15504 and cmmi. **Softw. Process**, John Wiley & Sons, Inc., New York, NY, USA, v. 12, n. 4, p. 361–371, 2007. ISSN 1077-4866. Citado na página [32](#).

RUNGRATRI, S.; USANAVASIN, S. Semantic based approach supporting CMMI gap analysis process. **Journal of Convergence Information Technology**, v. 7, n. 20, p. 127–137, 2012. Citado na página [79](#).

SCHMIDT, D. C. Model driven engineering. **Computer**, v. 39, n. 2, p. 25–31, 2006. Citado 2 vezes nas páginas [38](#) e [48](#).

- SEI. 2013. [On-line], *Portal CMMI*. Disponível em <<http://www.CMMIinstitute.com/>> (Acessado em 11/06/2013). Citado 3 vezes nas páginas 32, 42 e 43.
- SELIC, B. The pragmatics of model-driven development. **IEEE Software**, v. 20, n. 5, p. 19–25, 2003. Citado 2 vezes nas páginas 38 e 48.
- SOA. 2015. [On-line], *SOA: Service Oriented Architecture*. Disponível em <<http://www.omg.org/technology/readingroom/SOA.htm>> (Acessado em 20/03/2015). Citado na página 139.
- SOFTEX. 2011. [On-line], *Guia de Implementação - Parte 10: Implementação do MR-MPS em organizações do tipo Fábrica de Teste*. Disponível em <<http://www.softex.br/mpsbr/guias/>> (Acessado em 12/10/2013). Citado na página 44.
- _____. 2012. [On-line], *Guia de Implementação - Parte 11: Implementação e Avaliação do MR-MPS-SW:2012 em Conjunto com o CMMI-DEV v1.3*. Disponível em <<http://www.softex.br/mpsbr/guias/>> (Acessado em 12/10/2013). Citado 2 vezes nas páginas 106 e 131.
- _____. 2013. [On-line], *Guia de Implementação de Software – Parte I: Nível G*. Disponível em <<http://www.softex.br/mpsbr/guias/>> (Acessado em 12/10/2013). Citado na página 130.
- _____. 2015. [On-line], *Manual de Avaliação do MPS (MA-MPS)*. Disponível em <<http://www.softex.br/mpsbr/guias/>> (Acessado em 15/01/2016). Citado na página 134.
- _____. 2016. [On-line], *Guia Geral MPS de Software*. Disponível em <<http://www.softex.br/mpsbr/guias/>> (Acessado em 15/01/2016). Citado 13 vezes nas páginas 15, 23, 33, 41, 42, 44, 45, 46, 47, 48, 76, 130 e 137.
- SOMMERVILLE, I. **Engenharia de Software**. 9. ed. New Jersey, EUA: Pearson Prentice Hall, 2011. Citado 4 vezes nas páginas 37, 39, 40 e 48.
- SPEM. 2015. [On-line], *Object Management Group: Software & Systems Process Engineering Metamodel Specification (SPEM)*. Disponível em <<http://www.omg.org/spec/SPEM/2.0/>> (Acessado em 20/03/2015). Citado 4 vezes nas páginas 15, 51, 52 e 108.
- WANGENHEIM, C. G. V.; MCCAFFERY, F.; HAUCK, J. C. R.; LACERDA, T. C.; BUGLIONE, L.; CRUZ, R. F. V. D. Building a maturity & capability model repository. In: **ACM International Conference Proceeding Series**. [S.l.: s.n.], 2011. p. 2–5. Citado 6 vezes nas páginas 15, 63, 74, 75, 79 e 82.
- YEOM, H. .; HWANG, S. . A design of tool for software processes assessment and improvement. In: **Proceedings of the 2008 Advanced Software Engineering and its Applications, ASEA 2008**. [S.l.: s.n.], 2008. p. 245–248. Citado na página 79.

FORMULÁRIO APLICADO AO *SURVEY*

Neste apêndice, é apresentado o formulário utilizado para a realização do *survey* descrito no Capítulo 4. As questões que apresentam o caracter “*” são de preenchimento obrigatório. Algumas questões apresentam o campo “*Others*” em que era possível ao participante declarar um texto complementar a sua resposta ou informar uma opção não prevista nas opções de resposta disponibilizadas. O formulário pode ser consultado na *internet* pelo link <http://goo.gl/forms/NfRmC6Aios>.

A.1 Quadro de Questões: Perfil da Empresa

Nesse quadro, foram elaboradas questões para identificar o perfil das empresas em que os participantes trabalham ou tiveram alguma experiência profissional relacionada à definição de processos e adoção de modelos de maturidade. Além das questões apresentadas nas Figuras 45 a 50, foi questionado o nome da empresa e o *site* na *internet*.

Empresa atua no mercado de TI há quanto tempo? *

- ☐ 0 - 1 ano
- ☐ 2 - 5 anos
- ☐ mais que 5 anos

Figura 45 – Questão do quadro *Perfil da Empresa* do formulário aplicado ao *survey* realizado.

Qual o segmento dos produtos de software da empresa? *

- ☐ Varejo
- ☐ e-Commerce
- ☐ Segurança
- ☐ Other:

Figura 46 – Questão do quadro *Perfil da Empresa* do formulário aplicado ao *survey* realizado.

Quantos funcionários a empresa possui? *

- ☐ 0 - 10
☐ 11 - 30
☐ 31 - 50
☐ 51 - 70
☐ 71 - 100
☐ mais de 100.

Figura 47 – Questão do quadro *Perfil da Empresa* do formulário aplicado ao *survey* realizado.

Qual a área de atuação da empresa? *

- ☐ Fábrica de Software
☐ Aquisição
☐ Fábrica de Testes
☐ Other:

Figura 48 – Questão do quadro *Perfil da Empresa* do formulário aplicado ao *survey* realizado.

Somente se o participante escolhesse a opção “Sim” na questão ilustrada na Figura 49 as questões apresentadas na Figura 50 seriam visualizadas.

Possui certificação em modelos de qualidade? *

- ☐ Sim
☐ Não

Figura 49 – Questão do quadro *Perfil da Empresa* do formulário aplicado ao *survey* realizado.

Qual o modelo de qualidade em que possui certificação? *

- ☐ MPS.Br
☐ CMMI
☐ ISO/IEC 15504
☐ Other:

Qual o nível de maturidade certificado? *

Figura 50 – Questão do quadro *Perfil da Empresa* do formulário aplicado ao *survey* realizado.

A.2 Quadro de Questões: Perfil do Participante

Nesse quadro, foram elaboradas questões para identificar o perfil dos participantes que responderam às questões. Além das questões apresentadas nas Figuras 51 a 60, foi questionado o nome do participante, um *e-mail* para contato e, ao final, foi disponibilizada uma questão aberta para o participante complementar o quadro com informações que julgasse pertinentes.

Possui graduação ou está cursando um dos cursos abaixo? *

- ☐ Ciência da Computação
- ☐ Engenharia da Computação
- ☐ Análise de Sistemas
- ☐ Sistemas de Informação
- ☐ Não possui graduação e não cursa nenhum dos cursos listados acima
- ☐ Other:

Figura 51 – Questão do quadro *Perfil do Participante* do formulário aplicado ao *survey* realizado.

Caso graduado, qual a titulação? *

- ☐ Bacharel
- ☐ Mestre
- ☐ Doutor
- ☐ Pós-Doutor
- ☐ Other:

Figura 52 – Questão do quadro *Perfil do Participante* do formulário aplicado ao *survey* realizado.

Possui quantos anos de experiência em qualidade e processos de software? *

- ☐ 0 - 1
- ☐ 2 - 5
- ☐ mais que 5 anos

Figura 53 – Questão do quadro *Perfil do Participante* do formulário aplicado ao *survey* realizado.

Possui quantos anos de experiência em desenvolvimento de software? *

- ☐ 0 - 1
- ☐ 2 - 5
- ☐ mais de 5 anos

Figura 54 – Questão do quadro *Perfil do Participante* do formulário aplicado ao *survey* realizado.

Informe quais atividades desenvolvidas por você são relacionadas com qualidade de software. *

- ☐ Verificação e validação
- ☐ Inspeção
- ☐ Revisões Formais
- ☐ Code Review
- ☐ Other:

Figura 55 – Questão do quadro *Perfil do Participante* do formulário aplicado ao *survey* realizado.

Informe as atividades desenvolvidas relacionadas com processos de software. *

- ☐ Definição de processos
- ☐ Melhoria de Processos
- ☐ Consultoria para Certificação
- ☐ EPG
- ☐ Other:

Figura 56 – Questão do quadro *Perfil do Participante* do formulário aplicado ao *survey* realizado.**Possui experiência com quais metodologias ágeis? ***

- ☐ Scrum
- ☐ XP
- ☐ Não possui
- ☐ Other:

Figura 57 – Questão do quadro *Perfil do Participante* do formulário aplicado ao *survey* realizado.**Participou de algum comitê ou grupo de estudos relacionado a qualidade e processos de software? ***

- ☐ Sim
- ☐ Não

Figura 58 – Questão do quadro *Perfil do Participante* do formulário aplicado ao *survey* realizado.**Qual era o foco do grupo de estudos em que participou? ***

- ☐ Novas tecnologias de desenvolvimento
- ☐ Padrões de projeto
- ☐ Definição de processos
- ☐ Other:

Figura 59 – Questão do quadro *Perfil do Participante* do formulário aplicado ao *survey* realizado.**Qual era o foco do comitê de processos em que participou? ***

- ☐ Gerência de Projetos
- ☐ Gerência de Requisitos
- ☐ Melhoria de Processos
- ☐ Gerência de Configuração
- ☐ Other:

Figura 60 – Questão do quadro *Perfil do Participante* do formulário aplicado ao *survey* realizado.

A.3 Quadro de Questões: Experiência com Implementação de Modelos de Maturidade e Processo de Software

Nesse quadro, foram elaboradas questões para identificar as experiências das empresas e dos participantes no processo de adoção de modelos de maturidade. Além das questões

apresentadas nas Figuras 61 a 73, ao final foi disponibilizada uma questão aberta para que o participante complementar o quadro com informações que julgasse relevantes.

Possui experiência em implementação com algum modelo de qualidade de processos de software? *

- ☐ CMMI
- ☐ MPS.Br
- ☐ ISO/IEC 15504
- ☐ ISO/IEC 20000
- ☐ ITIL
- ☐ COBIT
- ☐ Other:

Figura 61 – Questão do quadro *Experiência com Implementação de Modelos de Maturidade e Processo de Software* do formulário aplicado ao *survey* realizado.

Selecione quais foram os principais problemas/dificuldades na implementação do modelo adotado. *

- ☐ Resistência das equipes internas
- ☐ Problemas com consultoria
- ☐ Dificuldade de entendimento do que o modelo pede
- ☐ Other:

Figura 62 – Questão do quadro *Experiência com Implementação de Modelos de Maturidade e Processo de Software* do formulário aplicado ao *survey* realizado.

Selecione quais foram as áreas de processo mais difíceis de adequar ao modelo adotado mediante o status inicial dos processos organizacionais da empresa. *

- ☐ Gerência de Projetos
- ☐ Gerência de Riscos
- ☐ Gerência de Decisões
- ☐ Verificação
- ☐ Validação
- ☐ Projeto e Construção de Produto
- ☐ Integração do Produto
- ☐ Desenvolvimento de Requisitos
- ☐ Gerência de Reutilização
- ☐ Gerência de Recursos Humanos
- ☐ Definição do Processo Organizacional
- ☐ Avaliação e Melhoria de Processo Organizacional
- ☐ Medição
- ☐ Garantia da Qualidade
- ☐ Gerência de Portfólio de Projetos
- ☐ Gerência de Configuração
- ☐ Aquisição
- ☐ Gerência de Requisitos
- ☐ Other:

Figura 63 – Questão do quadro *Experiência com Implementação de Modelos de Maturidade e Processo de Software* do formulário aplicado ao *survey* realizado.

Descreva quais foram as ferramentas utilizadas para apoiar a implementação dos modelos de qualidade. Indique se possível o nome, versão utilizada e URL para acesso em caso de ferramentas open-source ou proprietárias. *

Figura 64 – Questão do quadro *Experiência com Implementação de Modelos de Maturidade e Processo de Software* do formulário aplicado ao survey realizado.

Em caso de desenvolvimento interno de ferramentas de apoio a implementação do modelo de qualidade, indicar como foi feito o mapeamento entre o processo e as funcionalidades da ferramenta e quais as tecnologias utilizadas. *

Figura 65 – Questão do quadro *Experiência com Implementação de Modelos de Maturidade e Processo de Software* do formulário aplicado ao survey realizado.

Muitas vezes o uso de consultorias externas em qualidade de software são necessárias para se realizar as etapas preliminares a avaliação do modelo. Desta forma, descreva como foi o processo de apoio da consultoria, foi satisfatório, descreva se houveram problemas de comunicação ou de implementação de práticas dos modelos de qualidade. *

Figura 66 – Questão do quadro *Experiência com Implementação de Modelos de Maturidade e Processo de Software* do formulário aplicado ao survey realizado.

Qual o tempo médio para a implementação do modelo de qualidade? Considerando o período a partir da etapa de consultoria até a certificação seja com sucesso ou não. *

- ☐ 1 - 3 meses
- ☐ 4 - 6 meses
- ☐ 7 - 9 meses
- ☐ 10 - 12 meses
- ☐ mais que 12 meses.

Figura 67 – Questão do quadro *Experiência com Implementação de Modelos de Maturidade e Processo de Software* do formulário aplicado ao survey realizado.

Qual o custo médio para a implementação do modelo de qualidade (indique o nível do modelo que foi implementado)? *

- ☐ R\$ 1.000,00 - R\$ 10.000,00
- ☐ R\$ 10.000,00 - R\$ 30.000,00
- ☐ R\$ 30.000,00 - R\$ 50.000,00
- ☐ R\$ 50.000,00 - R\$ 75.000,00
- ☐ R\$ 75.000,00 - R\$ 100.000,00
- ☐ R\$ 100.000,00 - R\$ 150.000,00
- ☐ R\$ 150.000,00 - R\$ 200.000,00
- ☐ R\$ 200.000,00 - R\$ 250.000,00
- ☐ R\$ 250.000,00 - R\$ 300.000,00
- ☐ R\$ 300.000,00 - R\$ 350.000,00
- ☐ R\$ 350.000,00 - R\$ 400.000,00
- ☐ R\$ 400.000,00 - R\$ 450.000,00
- ☐ R\$ 450.000,00 - R\$ 500.000,00
- ☐ mais de R\$ 500.000,00

Figura 68 – Questão do quadro *Experiência com Implementação de Modelos de Maturidade e Processo de Software* do formulário aplicado ao survey realizado.

Quais foram os ganhos notados pela empresa/equipe de desenvolvimento após a adoção do modelo de qualidade? *

- ☐ Diminuição das falhas dos produtos
- ☐ Melhoria no entendimento do que deveria ser desenvolvido
- ☐ Melhor comunicação com o cliente
- ☐ Melhor comunicação entre as equipes de desenvolvimento
- ☐ Melhoria no desempenho dos projetos
- ☐ Other:

Figura 69 – Questão do quadro *Experiência com Implementação de Modelos de Maturidade e Processo de Software* do formulário aplicado ao *survey* realizado.

Quais foram as perdas notadas pela empresa/equipe de desenvolvimento após a adoção do modelo de qualidade? Houveram práticas que precisaram ser revistas? Houve algum tipo de 'engessamento' do processo? *

- ☐ "Engessamento" do processo
- ☐ Criação de gargalos operacionais
- ☐ Excesso de documentação
- ☐ Necessidade de recursos humanos adicionais para a manutenção do processo implementado
- ☐ Other:

Figura 70 – Questão do quadro *Experiência com Implementação de Modelos de Maturidade e Processo de Software* do formulário aplicado ao *survey* realizado.

Foi possível notar melhoria na performance de execução e/ou melhores resultados obtidos em projetos após a implementação do modelo? *

- ☐ 0 - 5%
- ☐ 5- 10%
- ☐ 10 - 15%
- ☐ 15 - 20%
- ☐ 20 - 25%
- ☐ 25 - 30%
- ☐ 30 - 40%
- ☐ 40 - 50%
- ☐ 50 - 75%
- ☐ 75 - 100%
- ☐ mais que 100%

Figura 71 – Questão do quadro *Experiência com Implementação de Modelos de Maturidade e Processo de Software* do formulário aplicado ao *survey* realizado.

A empresa possui motivação para prosseguir a escala de melhoria dos processos dentro do modelo de qualidade adotado, ou seja, possui interesse em continuar nos níveis de maturidade mais elevados? *

- ☐ Sim
- ☐ Não

Figura 72 – Questão do quadro *Experiência com Implementação de Modelos de Maturidade e Processo de Software* do formulário aplicado ao *survey* realizado.

Empresa já iniciou processo de implantação de mudança de nível de maturidade? Qual o nível de maturidade em implementação? *

- ☐ Sim
☐ Não

Figura 73 – Questão do quadro *Experiência com Implementação de Modelos de Maturidade e Processo de Software* do formulário aplicado ao survey realizado.

A.4 Quadro de Questões: Expectativas em Relação a Ferramentas de Avaliação e Melhoria de Processos de Software

Nesse quadro, foram elaboradas questões para identificar as experiências das empresas e dos participantes em relação a uma possível parceria com a academia e quais funções seriam esperadas em ambientes de apoio a avaliação e melhoria de processos de software. Além das questões apresentadas nas Figuras 74 a 80, ao final, foi disponibilizada uma questão aberta para o participante complementar o quadro com informações que julgasse importantes.

Dentro do contexto de processo de software, qual funcionalidade você e/ou a empresa espera de uma ferramenta de apoio a gestão e melhoria dos processo de software? *

- ☐ Gestão das áreas de processo
☐ Gestão dos artefatos gerados na execução do processo
☐ Identificação de pontos de melhoria no processo
☐ Pré-análise de aderência a modelos de maturidade
☐ Other:

Figura 74 – Questão do quadro *Experiência com Implementação de Modelos de Maturidade e Processo de Software* do formulário aplicado ao survey realizado.

Você ou a empresa utilizaria um ambiente baseado em serviços para realizar a gestão e melhoria dos processos de software da empresa? O ambiente proveria uma série de serviços que auxiliariam a empresa a identificar ativos de reuso para seus processos e projetos. Além disso, seria capaz de interpretar o processo executado da empresa e comparar com um modelo de qualidade com o intuito de indicar os pontos em que não há aderência entre eles. O objetivo seria facilitar o processo de adequação para uma possível certificação no modelo de qualidade a ser adotado, independente de qual seja (MPS.Br, CMMI, normas ISO/IEC). *

- ☐ Sim
☐ Não

Figura 75 – Questão do quadro *Experiência com Implementação de Modelos de Maturidade e Processo de Software* do formulário aplicado ao survey realizado.

Você ou a empresa compartilharia suas experiências no contexto de engenharia de software em geral em uma rede social voltada para avaliação e melhoria de processos de software para apoiar e/ou receber apoio de outros profissionais de engenharia de software? *

- ☐ Sim
☐ Não

Figura 76 – Questão do quadro *Experiência com Implementação de Modelos de Maturidade e Processo de Software* do formulário aplicado ao survey realizado.

Você ou a empresa faria uso de um ambiente de reúso em engenharia de software na qual seria possível disponibilizar e/ou ter acesso a ativos de reúso relacionados a projetos, processos, modelos de qualidade e linhas de produto de software? *

- ☐ Sim
☐ Não

Figura 77 – Questão do quadro *Experiência com Implementação de Modelos de Maturidade e Processo de Software* do formulário aplicado ao survey realizado.

A empresa aceitaria participar de um projeto piloto relaciona a melhoria e avaliação de processos de software? Seriam necessários o envolvimento com questionários, utilização de ferramentas e entrevistas. *

- ☐ Sim
☐ Não

Figura 78 – Questão do quadro *Experiência com Implementação de Modelos de Maturidade e Processo de Software* do formulário aplicado ao survey realizado.

A empresa aceitaria disponibilizar os relatórios técnicos de avaliação preliminar (PA) do processo de certificação de modelos de qualidade, com o intuito de contribuir para a criação de cenários de validação de metodologias de melhoria e avaliação de processos de software? *

- ☐ Sim
☐ Não

Figura 79 – Questão do quadro *Experiência com Implementação de Modelos de Maturidade e Processo de Software* do formulário aplicado ao survey realizado.

Descreva o que você esperaria de funcionalidades em um ambiente que contenha as características descritas nas questões anteriores. *

Figura 80 – Questão do quadro *Experiência com Implementação de Modelos de Maturidade e Processo de Software* do formulário aplicado ao survey realizado.

GLOSSÁRIO CASOS DE USO

Nesse apêndice, são detalhados os casos de uso disponibilizados na Seção 5.8. Nas Tabelas 16 e 17, é realizada a descrição resumida dos casos de uso presentes na Figura 36, que apresenta o diagrama de casos de uso representando a etapa de definição dos modelos *Taxonomy*, *Process Template*, *ProcessInstance*, *MaturityModel* e os respectivos *StructureAndBehavior* da abordagem proposta. Em seguida, nas Tabelas 18 e 19, é apresentada a descrição resumida dos casos de uso presentes na Figura 37, que contém o diagrama de casos de uso representando a etapa de avaliação e sugestão de melhorias dos modelos criados.

Tabela 16 – Descrição resumida dos casos de uso presentes na Figura 36.

Ator	Caso de Uso	Descrição
Oracle	Manage Process Template	Representa as funções de <i>CRUD</i> da entidade <i>ProcessTemplate</i> .
User	Manage Process Instance	Representa as funções de <i>CRUD</i> da entidade <i>ProcessInstance</i> .
User	Manage Project	Representa as funções de <i>CRUD</i> da entidade <i>Project</i> . Nesse caso de uso, também é realizado o vínculo de um projeto com a instância de processo utilizado na execução do projeto.
User	Execute Process Element Monitoring	Nesse caso de uso, será possível ao usuário indicar quais atividades previstas na execução do processo foram realmente executadas no projeto.
Oracle	Manage Maturity Model	Representa as funções de <i>CRUD</i> da entidade <i>MaturityModel</i> .
Oracle	Define Maturity Level	Representa as funções de <i>CRUD</i> da entidade <i>MaturityLevel</i> . Nesse caso de uso os níveis de maturidade definidos pelo modelo são modelados e vinculados ao modelo de maturidade.
Oracle	Define Process Area	Representa as funções de <i>CRUD</i> da entidade <i>ProcessArea</i> . Nesse caso de uso as áreas de processo definidas pelo modelo são modeladas e vinculadas ao modelo de maturidade.

Tabela 17 – Descrição resumida dos casos de uso presentes na Figura 36. (Continuação)

Ator	Caso de Uso	Descrição
Oracle	<i>Define Generic Practices</i>	Representa as funções de <i>CRUD</i> da entidade <i>GenericPractices</i> . Nesse caso de uso as práticas genéricas definidas para um nível de maturidade definidos pelo modelo são modeladas e vinculadas ao modelo de maturidade.
Oracle	<i>Define GP Sub Practices</i>	Representa as funções de <i>CRUD</i> da entidade <i>GP-SubPractice</i> . Nesse caso de uso, as práticas genéricas definidas para um nível de maturidade definidos pelo modelo são detalhadas em subitens sendo modeladas e vinculadas ao modelo de maturidade.
Oracle	<i>Define Specific Practices</i>	Representa as funções de <i>CRUD</i> da entidade <i>SpecificPractices</i> . Nesse caso de uso, as práticas específicas definidas para uma área de processo definidas pelo modelo são modeladas e vinculadas ao modelo de maturidade.
Oracle	<i>Define Structure And Behavior Elements</i>	Representa as funções de <i>CRUD</i> da entidade <i>StructureAndBehavior</i> . Nesse caso de uso, os elementos que descrevem a execução das práticas específicas e genéricas definidas pelos modelos de maturidade e elementos de processos (instâncias e <i>templates</i>) são definidos. Nela serão criadas as atividades, papéis e produtos de trabalho produzidos e consumidos na descrição dos processos e sugestões de implementações de modelos de maturidade. A partir desse caso de uso, também será possível ao usuário iniciar a definição dos termos e dos sinônimos que pretende incluir na taxonomia.
Oracle	<i>Manage Taxonomy</i>	Representa as funções de <i>CRUD</i> da entidade <i>Taxonomy</i> e suas entidades relacionadas no pacote de metamodelos que a define.

Tabela 18 – Descrição resumida dos casos de uso presentes na Figura 37.

Ator	Caso de Uso	Descrição
Oracle	<i>Manage Assessment</i>	Representa as funções de <i>CRUD</i> da entidade <i>ExecutedAssessment</i> .
Oracle	<i>Manage Questionnaire Based Assessment</i>	Representa as funções de <i>CRUD</i> das entidades envolvidas na definição do questionário de avaliação de conformidade entre modelos de maturidade e instâncias de processos. As entidades envolvidas são <i>SubjectiveAssessment</i> , <i>Questionnaire</i> , <i>ComplianceItem</i> , <i>Question</i> , <i>AnswerOption</i> e <i>ComplianceTypeRule</i> .
Oracle	<i>Manage Script Based Assessment</i>	Representa as funções de <i>CRUD</i> das entidades envolvidas na definição de avaliações de conformidade utilizando <i>scripts</i> customizados de transformações de modelos. Para isso o usuário deve indicar uma forma de execução do <i>script</i> criado nas entidades do modelo <i>ObjectiveAssessment</i> .

Tabela 19 – Descrição resumida dos casos de uso presentes na Figura 37. (Continuação)

Ator	Caso de Uso	Descrição
Oracle, User	<i>Execute Assessment</i>	Nesse caso de uso, é realizada a execução de uma avaliação de conformidade. A partir desse caso de uso, são inclusos os casos de uso <i>Search Maturity Models</i> , <i>Search Processes Available</i> e <i>Search Processes Available</i> , utilizados para dividir as etapas executadas e possibilitar o reúso de suas implementações por outros casos de uso.
Oracle, User	<i>Search Maturity Models</i>	Utilizado para consultar os modelos de maturidade disponíveis no ambiente computacional. No caso de execução de avaliação de conformidade, esse caso de uso será utilizado para buscar os modelos disponíveis para avaliação.
Oracle, User	<i>Search Processes Available</i>	Utilizado para consultar as instâncias de processo disponíveis no ambiente computacional. No caso de execução de avaliação de conformidade, esse caso de uso será utilizado para buscar os processos criados pelo usuário que acionou o caso de uso.
Oracle, User	<i>Generate Report</i>	Após executada a avaliação de conformidade, os resultados obtidos são persistidos na entidade <i>Report</i> e sua entidade relacionada <i>ReportItem</i> e, por fim, formatará os resultados para o usuário avaliar.
Oracle, User	<i>Manage Questionnaire</i>	Executado pelo usuário ao responder as questões definidas no questionário de avaliação de conformidade criado pelo <i>Oráculo</i> .
Oracle, User	<i>Search Assessment Reports</i>	Pesquisa os relatórios de avaliação de conformidade executadas para uma instância de processo.
Oracle, User	<i>Suggest Hint Improvement</i>	Utilizado para permitir ao usuário sugerir melhorias em um item de conformidade disponível em um relatório.
Oracle, User	<i>Send Improvement Hint Notification</i>	Ao realizar uma sugestão de melhoria para um item do relatório de conformidade, uma notificação ao usuário que executou a avaliação deverá ser enviada.
Oracle, User	<i>Manage Improvement Hint Notification</i>	Permite aos usuários visualizarem o <i>feedback</i> fornecido pelos usuários que receberam suas sugestões de melhoria de processos.
Oracle, User	<i>Manage Improvement Hint</i>	Representa as funções de <i>CRUD</i> da entidade <i>ImprovementHint</i> e sua entidade relacionada <i>HintComments</i> . Também é nesse caso de uso que os usuários podem avaliar as sugestões de melhoria recebidas, indicando se foram positivas, negativas, neutras ou iniciar nova discussão sobre o item.
Oracle, User	<i>Manage Process Instance</i>	Descrito na Tabela 16.