Pós-Graduação em Ciência da Computação

"Guidelines para implantação de processos híbridos de software: Uma proposta para a definição de um ciclo de vida com um padrão ISO e metodologia ágeis"

Por

Hugo Vieira Lucena de Souza

Dissertação de Mestrado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, AGOSTO/2012

Universidade Federal de Pernambuco
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

HUGO VIEIRA LUCENA DE SOUZA

"Guidelines para implantação de processos híbridos de software: Uma proposta para a definição de um ciclo de vida com um padrão ISO e metodologias ágeis"

ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO.

.

ORIENTADOR(A): PROF. DR. SILVIO ROMERO DE LEMOS MEIRA

RECIFE, AGOSTO/2012

Catalogação na fonte
Bibliotecária Jane Souto Maior, CRB4-571

Souza, Hugo Vieira Lucena de
Guidelines para implantação de processos híbridos de software: uma proposta para a definição de um ciclo de vida com um padrão ISO e metodologia ágeis. / Hugo Vieira Lucena de Souza. - Recife: O Autor, 2012.
140 folhas: il., fig., tab., gráf.

Orientador: Silvio Romero de Lemos Meira.
Dissertação (mestrado) - Universidade Federal de Pernambuco. CIn, Ciência da Computação, 2012.

Inclui bibliografia e apêndice.

1. Engenharia de software. 2. Processos de software. I. Meira, Silvio Romero Lemos de (orientador). II. Título.

005.1 CDD (23. ed.) MEI2012 – 189

Dissertação de Mestrado apresentada por **Hugo Vieira Lucena de Souza** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Guidelines para Implantação de Processos Híbridos de software: Uma Proposta para a Definição de um Ciclo de Vida com um Padrão ISO e Metodologias Ágeis"** orientada pelo Prof. Silvio Romero de Lemos Meira e aprovada pela Banca Examinadora formada pelos professores:

_____

Prof. Alexandre Marcos Lins de Vasconcelos
Centro de Informática / UFPE

_____

Profa. Maria da Conceição Moraes Batista
Departamento de Estatística e Informática / UFRPE

_____

Prof. Silvio Romero de Lemos Meira
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 20 de agosto de 2012

_____

Prof. Nelson Souto Rosa
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

“Precisei cair, para aprender a voar...”
Marcella Thebaldi.

“Dedico este trabalho aos meus pais, Clodomiro de Souza Lima e Maria Estela Vieira de Lucena Souza, e à minha irmã Amanda Vieira Lucena de Souza, os quais amarei durante todos os dias de minha vida. . . .”

Agradecimentos

Em primeiro lugar, agradeço ao Grande Arquiteto do Universo que iluminou todos os meus passos nesta árdua e empolgante batalha que formula um crescimento intelectual e espiritual inerente às conquistas obtidas com a conclusão de um programa de Pós-graduação com reconhecimento comprovado.

Agradeço a minha família, fonte de minha existência, em especial aos meus pais Clodomiro de Souza Lima e Maria Estela Vieira de Lucena Souza, além de minha irmã Amanda Vieira Lucena de Souza, alicerces essenciais que sempre me apoiam nas horas fáceis e difíceis onde quer que eu esteja.

Agradeço ao professor Silvio Meira pela oportunidade concedida de ser seu orientando, como também ao amigo Felipe Furtado, por assumir a função de ser meu co-orientador disponibilizando seu precioso tempo com correções, propostas de pesquisa e contatos com as empresas para que este trabalho pudesse ser finalizado.

Agradeço também à professora Teresa Maciel pelos conselhos e dicas para a formulação desta pesquisa durante os dois anos de convivência no Centro de Informática da Universidade Federal de Pernambuco.

Agradeço aos colegas de trabalho da FACOL por sempre disponibilizarem seu tempo para eventuais sugestões sobre a pesquisa, e pelo convívio diário na arte de compartilhar o nosso conhecimento e experiências para muitas pessoas a cada semestre letivo.

Agradeço a CAPES pela parceria com a Universidade Federal de Pernambuco para conceder, através da biblioteca digital do IEEE, a versão original e completa da norma pela qual tive oportunidade de acessá-la para congratular a realização deste trabalho.

Agradeço a todas as empresas e profissionais que contribuíram durante a aplicação dos questionários e pesquisas para possibilitar o levantamento dos dados. A cada um expresso os mais sinceros votos de sucesso em seus respectivos mercados.

Agradeço aos colegas de Mestrado e amigos do Recife pela convivência curta, porém proveitosa, sem citar nomes para não cometer injustiças de esquecer alguns deles.

Agradeço as amigas Armanda Maria e Lídia Melo, e aos amigos Emanuell Faustino e Leandro Caetano, pelos bons momentos vividos durante este curto tempo, mas proveitosos pela amizade que foi construída ao longo da convivência diária e que se fortalece a cada novo dia de nossas vidas.

Agradeço a todos aqueles que contribuíram direta ou indiretamente para a realização deste trabalho de dissertação.

Resumo

Ao longo dos anos a Engenharia de Software tem proporcionado o surgimento de novos modelos, processos e metodologias para o desenvolvimento de software. Algumas metodologias ágeis, como é o caso do *Scrum*, têm por objetivo tornar mais simples o gerenciamento dos processos de software ao longo do projeto. Outras, como o XP, recomendam a inserção de boas práticas durante a produção do software como uma estratégia indispensável para obter um produto com maior qualidade. Já o FDD, por exemplo, equipara os fatores de gerência e engenharia, formando processos mais completos com atividades bem definidas. As diferenças existentes dentre essas metodologias têm despertado um problema vivenciado por muitas empresas, que é de como selecionar o método mais adequado ao cenário configurado para o desenvolvimento? Quando a escolha do método é equivocada, as consequências podem afetar diretamente nos resultados dos projetos. Desta forma, uma alternativa que vem sendo adotada para contornar esse tipo de situação consiste em combinar as atividades de um ou mais processos diferentes ao processo já ambientado, ficando esta temática conhecida como *processos híbridos de software*. Entretanto, esta perspectiva adquire certa resistência pelo fato de não existirem os meios tangíveis que orientam os profissionais a realizarem as devidas adaptações. Portanto, este trabalho tem como propósito apresentar, através do alinhamento das metodologias ágeis *Scrum*, XP, e FDD em aderência aos requisitos da norma ISO/IEC e IEEE 12207, dez guias contendo as boas práticas que habilitem os profissionais a formarem um ciclo de vida com processos híbridos de software.

Palavras-chave: Processos híbridos de software, ISO 12207, metodologias ágeis, *Scrum*, XP, FDD, guidelines de processos de software, engenharia de software.

Abstract

Over the years the software engineering has provided the emergence of new models, processes and methodologies for the software development. Some agile methodologies, such as Scrum, aims to simplify the software management processes throughout the project. Others, such as XP, recommend the insertion of good practices during the software production as a strategy to obtain a product with higher quality. Already the FDD, for example, balances management and engineering factors, forming most complete processes with well-defined activities. The differences among these methods have aroused a problem experienced by many companies, which is how to select the most appropriate method to set up the development scenario? When the method choosing is wrong, the consequences can directly affect the outcome of the projects. Thus, an alternative that has been adopted to avoid this situation type is to combine the activities of one or more different processes to the process already acclimated, getting this theme known as hybrid software processes. However, this view gains some resistance because there are not tangible means to guide professionals to carry out the necessary adaptations. Therefore, this paper presents, through the alignment of Agile Scrum, XP and FDD in compliance with the requirements of ISO / IEC and IEEE 12207 standard, ten tabs containing best practices that enable professionals to form a life cycle with hybrid software processes.

Keywords: Hybrid processes software, ISO 12207, Agile, *Scrum*, XP, FDD, guidelines for software process, software engineering.

.

Lista de Figuras

Figura 1.1: Protocolo para a síntese da proposta de guias 20
Figura 2.1: Constructo de processos em padrões ISO 27
Figura 2.2: Grupos de processos da ISO/IEC e IEEE 12207 28
Figura 3.1: Ciclo de vida do *Scrum* 44
Figura 3.2: Ciclo de vida do XP 49
Figura 3.3: Processos do ciclo de vida do FDD 52

Lista de Tabelas

Tabela 2.1: Categorias dos processos do contexto de sistemas 29
Tabela 3.1: Práticas propostas pelo XP 47
Tabela 3.2: Práticas fundamentais do FDD 51
Tabela 3.3: Comparativo entre *Scrum*, XP, e FDD 56
Tabela 4.1: Mapeamento do processo híbrido de implementação de software 60
Tabela 4.2: Mapeamento do processo híbrido de análise de requisitos de software 61
Tabela 4.3: Mapeamento do processo híbrido de construção de software 62
Tabela 4.4: Mapeamento do processo híbrido de integração de software 63
Tabela 4.5: Mapeamento do processo híbrido de qualificação e testes de software 64
Tabela 4.6: Mapeamento do processo híbrido de gerência da documentação de software 65
Tabela 4.7: Mapeamento do processo híbrido de gerência da configuração de software 66
Tabela 4.8: Mapeamento do processo híbrido de garantia da qualidade de software 67
Tabela 4.9: Mapeamento do processo híbrido de verificação de software 68
Tabela 4.10: Mapeamento do processo híbrido de validação de software 69
Tabela 6.1: Distribuição das respostas por estados 100
Tabela 6.2: Cargos e funções dos entrevistados 101

Lista de Gráficos

Gráfico 6.1: Quantidade de profissionais no ambiente de trabalho dos entrevistados 102
Gráfico 6.2: Tempo de mercado do ambiente de trabalho dos entrevistados 103
Gráfico 6.3: Opinião relativa a gerência como causa de problemas nos processos 104
Gráfico 6.4: Opinião relativa a engenharia como causa de problemas nos processos 104
Gráfico 6.5: Opinião relativa a análise do tema de processos adaptados 105
Gráfico 6.6: Opinião relativa a análise do planejamento para o uso de processos adaptados 105
Gráfico 6.7: Opinião relativa aos benefícios do uso de processos adaptados 105
Gráfico 6.8: Opinião sobre o uso de guias para a implantação de processos adaptados 106
Gráfico 6.9: Opinião sobre a importância da metodologia nos guias de implantação 106
Gráfico 6.10: Opinião sobre os benefícios de utilizar os guias instrucionais para adaptar os processos 106
Gráfico 6.11: Aderência ao guia do processo híbrido de implementação de software 107
Gráfico 6.12: Aderência ao guia do processo híbrido de análise de requisitos de software .. 108
Gráfico 6.13: Aderência ao guia do processo híbrido de construção de software 109
Gráfico 6.14: Aderência ao guia do processo híbrido de integração de software 110
Gráfico 6.15: Aderência ao guia do processo híbrido de qualificação e testes de software ... 111
Gráfico 6.16: Aderência ao guia do processo híbrido de gerência da documentação de software 112
Gráfico 6.17: Aderência ao guia do processo híbrido de gerência da configuração de software 113
Gráfico 6.18: Aderência ao guia do processo híbrido de garantia da qualidade de software. 114
Gráfico 6.19: Aderência ao guia do processo híbrido de verificação de software 115
Gráfico 6.20: Aderência ao guia do processo híbrido de validação de software 116

Lista de Abreviaturas e Siglas

ABNT – Associação Brasileira de Normas Técnicas

CASE - *Computer-Aided Software Engineering*

CLF – Construir Lista de Funcionalidades

CPF – Construir Por Funcionalidade

CR – *Change Request*

DMA – Desenvolvimento do Modelo Abrangente

DFD – Diagrama de Fluxo de Dados

DPF – Detalhar por Funcionalidade

FAQ – *Frequently Asked Questions*

FDD – *Feature Driven Development*

GED – Gerenciamento Eletrônico de Documentos

HTML – *Hypertext Markup Language*

IDE – *Integrated Development Environment*

IEC – *International Eletrotechnical Commission*

IEEE – *Institute of Electrical and Electronic Engineers*

IP – *Internet Protocol*

ISO – *International Organization for Standardization*

OS – Ordem de Serviço

PPF – Planejar Por Funcionalidade

RF – Requisito Funcional

RNF – Requisito Não Funcional

URL – *Uniform Resource Locator*

XML – *eXtensible Markup Language*

XP – *eXtreme Programming*

Sumário

1 Introdução 17
1.1 Motivação 17
1.1.1 Contextualização do problema 19
1.1.2 Justificativa de pesquisa 19
1.2 Objetivos 21
1.2.1 Objetivo geral 21
1.2.2 Objetivos específicos 21
1.3 Trabalhos relacionados 22
1.4 Metodologia da pesquisa 23
1.5 Estrutura do trabalho 24
2 Processsos de software e a norma ISO/IEC e IEEE 12207 25
2.1 Contextualização sobre processos de software 25
2.1.1 Processos de software e a normatização 26
2.1.2 Constructos de atividades e tarefas 26
2.2 Escopo dos processos da norma 27
2.2.1 Processos do contexto de sistemas 29
2.2.2 Processos específicos de software 30
2.3 Processos de implementação de software 30
2.3.1 Processo de implementação de software 31
2.3.2 Processo de análise de requisitos de software 32
2.3.3 Processo de construção de software 33
2.3.4 Processo de integração de software 33
2.3.5 Processo de qualificação e testes de software 34
2.4 Processos de suporte de software 35
2.4.1 Processo de gerência da documentação de software 35
2.4.2 Processo de gerência da configuração de software 36
2.4.3 Processo de garantia da qualidade de software 37
2.4.4 Processo de verificação de software 38
2.4.5 Processo de validação de software 39

2.5 Considerações finais 40
3 Metodologias ágeis 41
3.1 Contextualização sobre as metodologias ágeis 41
3.2 Scrum 42
3.2.1 Papéis 43
3.2.2 Fases e ciclo de vida 44
3.2.3 Artefatos 45
3.3 eXtreme Programming 46
3.3.1 Valores, princípios e práticas 46
3.3.2 Equipes XP 48
3.3.3 Ciclo de vida 48
3.4 Feature Driven Development 50
3.4.1 Disciplinas fundamentais 50
3.4.2 Papéis 51
3.4.3 Processos 52
3.5 Resumo das metodologias ágeis 56
3.6 Considerações finais 57
4 Mapeamento das metodologias ágeis em adesão aos processos de software da norma ISO/IEC e IEEE 12207 58
4.1 Contextualização do mapeamento 58
4.2 Metodologia do mapeamento 59
4.3 Mapeamento do processo híbrido de implementação de software 60
4.4 Mapeamento do processo híbrido de análise de requisitos 61
4.5 Mapeamento do processo híbrido de construção de software 62
4.6 Mapeamento do processo híbrido de integração de software 63
4.7 Mapeamento do processo híbrido de qualificação e testes de software 64
4.8 Mapeamento do processo híbrido de gerência da documentação de software 65
4.9 Mapeamento do processo híbrido de gerência da configuração de software 66
4.10 Mapeamento do processo híbrido de garantia da qualidade de software 67
4.11 Mapeamento do processo híbrido de verificação de software 68
4.12 Mapeamento do processo híbrido de validação de software 69
4.13 Considerações finais 70

5 Guidelines para a implantação de um ciclo de vida com processos híbridos de software 71
5.1 Contextualização dos guias 71
5.1.1 Metodologia de implantação híbrida proposta 72
5.1.2 Escopo dos guias 73
5.2 Guia do processo híbrido de implementação de software 74
5.3 Guia do processo híbrido de análise de requisitos de software 76
5.4 Guia do processo híbrido de construção de software 78
5.5 Guia do processo híbrido de integração de software 80
5.6 Guia do processo híbrido de qualificação e testes de software 82
5.7 Guia do processo híbrido de gerência da documentação de software 84
5.8 Guia do processo híbrido de gerência da configuração de software 86
5.9 Guia do processo híbrido de garantia da qualidade de software 88
5.10 Guia do processo híbrido de verificação de software 90
5.11 Guia do processo híbrido de validação de software 92
5.12 Considerações finais 94
6 Aplicação e resultados da pesquisa de campo 95
6.1 Classificação da pesquisa de campo 95
6.2 Instrumentação, definição da amostra e tratamento dos dados 97
6.3 Resultados da pesquisa de campo 99
6.3.1 Apresentação dos resultados relativos à amostra 100
6.3.2 Apresentação dos resultados quantitativos do questionário 102
6.4 Considerações finais 117
7 Conclusões 118
7.1 Contribuição da pesquisa 118
7.2 Limitações da pesquisa 119
7.3 Trabalhos futuros 120
Referências 122
Apêndice A 129
Apêndice B 137
Apêndice C 139

Capítulo

1

Introdução

Neste capítulo são apresentados os aspectos que contextualizam a motivação, com a contextualização do problema e a justificativa de pesquisa. Também estão listados os objetivos geral e específico, os trabalhos relacionados e a metodologia de pesquisa utilizada. Por fim, é apresentada a estrutura que compõe os capítulos da dissertação.

1.1 Motivação

Ao passar dos anos, a Engenharia de Software se tornou uma área promissora com o surgimento de vários modelos, processos e metodologias para o desenvolvimento de software (SOMMERVILLE, 2011). Estas ferramentas lançadas no mercado buscam até hoje solucionar os problemas apresentados pelos clientes, através da organização de etapas bem definidas a serem inseridas nos projetos (PRESSMAN, 2006).

Highsmith (2010) explica que o surgimento das metodologias ágeis em 2001 foi um marco que contribuiu para realizar a produção de um software de maneira objetiva. Algumas focam tornar menos rígido o gerenciamento dos processos, como é o caso do *Scrum*, simplificando as ações de acompanhamento das iterações. Outras, entretanto, disponibilizam um conjunto de práticas que tornem mais eficientes a análise, a codificação e outras atividades típicas da construção do software como é o caso do *eXtreme Programming* (BECK, 1999).

Há ainda algumas metodologias que idealizam o balanceamento de ambos os fatores (gerência e engenharia) para alcançar processos mais completos, como é o caso do *Feature Driven Development*. Esta metodologia é característica por prover um conjunto de atividades

que caracterizam um gerenciamento mais rígido, mas sem descartar o uso das práticas simples que facilitam a execução de seus processos (COAD *et al.*, 1999).

Com as peculiaridades existentes em cada metodologia, a aplicação de alguma delas pode não ser suficiente para suprir a demanda de operações exigida em um cenário de software (BOEHM e TURNER, 2005). Siakas *et al.*(2005) citam que em muitos casos esses modelos não possuem as atividades compatíveis para suprir certas necessidades, ou as que possuem não surtem efeitos devido à superficialidade pela qual as mesmas estão compostas.

Apesar das metodologias estarem susceptíveis às mudanças, nenhuma delas sugere como contornar as dificuldades que possam surgir. Meira *et al.* (2008) afirmam que uma estratégia que vem sendo utilizada por algumas empresas, e tem rendido bons resultados para esses casos, é realizar a fusão do processo existente com as atividades de outras metodologias, caracterizando o que se tornaria uma "adaptação" como solução rápida e eficiente.

A partir do momento em que um processo passa a ser adotado desconsiderando seus padrões originais para promover a combinação de dois ou mais processos diferentes em sua estrutura funcional, ele é denominado como *processo híbrido de software* (RODRÍGUEZ *et al.*, 2005). Peter Lakey (2003) explica que o processo híbrido é um processo comum, mas que a diferença básica é que a constituição das atividades de seu arcabouço não é pré-estabelecida anteriormente, mas sim, combinada ao longo de sua execução (LAKEY, 2003).

Smojver e Car (2009) explicam que os processos híbridos podem ser criados de várias formas. A primeira, segundo Donaires (2008), é a combinação de processos *tradicionais* como o RUP[1] (IBM, 1998), SSM[2] (DONAIRES, 2008), *Open*[3] (GRAHAN *et al.*,1997) etc. Outra forma, segundo Campelo (2003), é o alinhamento dos processos *tradicionais e ágeis* com as combinações do CMMI (SEI, 2001) com o XP (BECK, 1999), RUP (IBM, 1998) com o *Scrum* (SCHWABER, 2004), além de outras possíveis de serem realizadas.

A terceira forma, sendo uma das mais requisitadas, é a combinação dos *processos ou metodologias ágeis* (AgilCoop, 2011). Lycett (2003) cita que esse alinhamento exige que sejam considerados vários fatores como políticas, estratégias e prioridades, para verificar se a inserção das boas práticas em um ambiente ágil poderá ser viável para arcar bons resultados, ao invés de tornar mais críticos os problemas existentes por conta de fusões mal realizadas.

[1] RUP é a abreviatura de *Rational Unified Process*, marca registrada da IBM®.
[2] SSM é a abreviatura de *Soft System Methodology*, marca registrada de Peter Chekland®.
[3] *Open* é a abreviatura de *Open Process*, uma marca registrada de Brian Sellers®.

1.1.1 Contextualização do problema

Diante da dificuldade em identificar as diferenças existentes dentre as metodologias e combinar as boas práticas que cada uma pode oferecer para formar um novo processo, muitos engenheiros desistem do uso dessa temática (ZAKI e MOAWAD, 2010). Ramakrishnan (2010) cita que a resistência é criada pelo fato de que o processo existente pode inviabilizar a prática das adaptações, ou por não haver os meios que possibilitem adotar tais medidas.

Desta forma, um problema que vem sendo discutido há algum tempo pela comunidade ágil (Visão Ágil, 2007) e por grandes empresas (IBM, 2009) é o de como poderia ser estipulado um meio de tornar os processos eficientes utilizando como alternativa a prática de adaptações. As dificuldades podem estar relacionadas à falta do equilíbrio que precisa ser estabelecido dentre os fatores de gerência e engenharia que configuram os processos.

1.1.2 Justificativa de pesquisa

A teoria de que as equipes de desenvolvimento sigam as recomendações de apenas uma metodologia em seus projetos é completamente diferente da realidade. Abrahamsson *et al.* (2003) publicaram um comparativo em que apresentam as principais características dos ciclos de vida das metodologias ágeis *Scrum* (SCHWABER, 2004), XP (BECK, 1999) e FDD (PALMER e FELSING, 2002), apontando evidências de que os focos de gestão e engenharia dessas metodologias se complementam para viabilizar uma produção eficiente.

Uma pesquisa que contribuiu com resultados satisfatórios na tentativa de propor a integração dos escopos das metodologias ágeis foi publicada por Stephen Cohen e William Money em 2008 (COHEN e MONEY, 2008). Eles propuseram um método de alinhamento para os processos de design, operação e validação, na tentativa de dispor um ciclo de vida genérico para esses processos, utilizando como referência a norma ISO/IEC e IEEE 12207.

Pelo fato da norma ISO/IEC e IEEE 12207 disponibilizar um vocabulário comum para a síntese de processos bem definidos em um ciclo de vida de software, seu uso é viabilizado em diversas oportunidades (KOSCIANSKI e SOARES, 2007). Sua aplicação é realizada com o intuito de estipular os processos compatíveis através de recomendações acerca do que cada processo deve alcançar como objetivo, segundo citam Souza e Oliveira (2006), que a utilizaram entre os anos de 2000 e 2004 em três empresas no estado de Goiás.

Sabe-se que a maturidade é inserida nos processos de software à medida que a qualidade do produto evolui, em função do retrabalho contínuo que deve ser realizado ao longo do projeto. Porém, nos casos em que as empresas não dispõem dos recursos necessários para investir em programas de melhorias para os processos, uma alternativa comum a ser adotada é utilizar novas técnicas que tornem o acompanhamento mais flexível (CHO, 2009).

Portanto, a justificativa que direciona este trabalho é utilizar a norma ISO/IEC e IEEE 12207[4] como guia de referência para definir um ciclo de vida comum de software através do alinhamento dos ciclos de vida das metodologias ágeis *Scrum* (SCHWABER, 2004), XP (BECK, 1999) e FDD (PALMER e FELSING, 2002). Para isso, é verificada a adesão das metodologias em relação ao atendimento dos requisitos dos processos específicos de software da norma para identificar as características requeridas para a formação de novos processos.

O alinhamento é realizado através de um mapeamento que efetua o comparativo de cada processo e suas atividades, com o objetivo de identificar as compatibilidades e incompatibilidades existentes. A partir do resultado dos mapeamentos obtidos, poderá ser observado o que cada metodologia sugere como boas práticas que podem ser inseridas em cada fase de um projeto que esteja em execução.

O trabalho tem como contribuição principal reunir essas boas práticas mapeadas e distribuí-las organizadamente. O intuito da pesquisa não foi o de formar processos próprios, mas sim, sintetizar dez guias fundamentados, seguindo-se o protocolo representado na Figura 1.1, de modo a disponibilizar as sugestões para que as empresas criem um novo ciclo de vida com seus próprios processos híbridos de software.

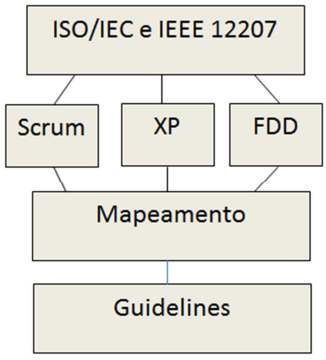

Figura 1.1: Protocolo para a síntese da proposta dos guias

[4] ISO® é uma marca registrada da *International Organization for Standardization.*
IEC® é uma marca registrada da International Eletrotechnical Comission.
IEEE ® é uma marca registrada do *Institute of Electrical and Electronics Engineers.*

1.2 Objetivos

Os objetivos deste trabalho estão divididos em objetivo geral e objetivos específicos.

1.2.1 Objetivo geral

O objetivo geral deste trabalho é desenvolver um conjunto de dez guias instrucionais baseados no alinhamento das boas práticas das metodologias ágeis *Scrum* (SCHWABER, 2004), XP (BECK, 1999) e FDD (PALMER e FELSING, 2002), em atendimento aos requisitos da norma ISO/IEC e IEEE 12207, para capacitar os engenheiros de software a criarem seus próprios processos híbridos de software.

1.2.2 Objetivos específicos

Os objetivos específicos deste trabalho são os seguintes:

- Apresentar as recomendações que a norma ISO/IEC e IEEE 12207 descreve em seu escopo, como requisitos necessários para que as empresas definam processos padrões e bem definidos de software.
- Realizar uma revisão literária das principais características das metodologias ágeis *Scrum* (SCHWABER, 2004), XP (BECK, 1999) e FDD (PALMER e FELSING, 2002) para identificar as atividades de gerência e engenharia que cada uma propõe nos processos e fases dos seus ciclos de vida.
- Realizar o mapeamento das metodologias para diagnosticar as principais diferenças existentes dentre os seus ciclos de vida, bem como a adesão das mesmas em relação aos requisitos definidos pelos processos específicos de software da norma.
- Conceber dez guias instrucionais, sendo cinco com os processos de implementação, e os outros cinco com os processos de suporte de software, contendo as práticas que facilitam a adaptação nos processos de software.
- Avaliar a proposta de pesquisa pela comunidade de engenheiros de software, com a realização de uma pesquisa de campo quantitativa que apresente como resultado as porcentagens de aceitação dos guias e de rejeição das práticas.

1.3 Trabalhos relacionados

O avanço nas pesquisas sobre a concepção dos processos híbridos de software tem crescido expressivamente ao longo dos últimos anos. Vários trabalhos têm destacado a combinação das metodologias ágeis como uma nova forma de propor soluções eficientes que proporcionem melhorar a qualidade dos processos e, consequentemente, alcançar bons resultados (COHEN e MONEY, 2008). A seguir, são descritos três exemplos que estão relacionados com esta pesquisa.

O primeiro, de autoria de Magdaleno *et al.*(2009), é um relatório técnico publicado pela COPPE/UFRJ que destaca as demandas que o mercado vem exigindo atualmente. Através da realização de uma revisão sistemática, os autores apontam que a maior quantidade de propostas disponíveis para a combinação de processos de software ainda é voltada para apenas dois modelos diferentes, mas que este cenário está mudando gradativamente.

O relatório revela que uma tendência que vem se lançando aos poucos é a procura de novos formatos de soluções, com a combinação de três ou mais modelos diferentes. A demanda crescente, justificada pelas empresas, é regida pelo fato de que as fusões podem gerar alternativas mais práticas que proporcionem balancear fatores essenciais como a disciplina, flexibilidade e a praticidade (MAGDALENO *et al.*, 2009).

O segundo trabalho, uma pesquisa de Irrazabal *et al.* (2011), apresenta uma análise comparativa do *Scrum* com os requisitos de dois processos (planejamento e avaliação de controle do projeto) da norma ISO/IEC e IEEE 12207. O mapeamento realizado possibilitou verificar que os percentuais de aderência das práticas de gerência da metodologia aos processos da norma foram altamente satisfatórios, comprovando que é possível identificar possíveis recomendações através da comparação de modelos diferentes.

Na pesquisa, é afirmado que a norma habilita definir as prioridades (*goals*) que cada processo possui para escolher as boas práticas sugeridas por ambos os modelos. O alinhamento com outras metodologias também se torna uma medida válida a ser considerada, visto que a mesma abrange recomendações para as diversas áreas abordadas pelas metodologias no ciclo de vida, tais como engenharia, gerência e reuso de software (IRRAZABAL *et al.*, 2011).

O terceiro e último trabalho é uma pesquisa publicada em 2011 que sugere reunir as melhores práticas de processos diferentes, como uma alternativa para inserir a melhoria

almejada nos processos de software. Ramachandran (2012) apresenta um processo que tem como objetivo instruir os engenheiros de software a organizarem as práticas selecionadas utilizando como recurso a síntese de guias instrucionais estruturados.

O autor propõe que a concepção de guias de processos se baseie em uma metodologia que fundamente a orientação de seus leitores. Para isso, é importante que sejam identificadas claramente as práticas, os meios de aplicação das mesmas, exemplos de como podem ser viabilizadas, além de definir uma estrutura que reporte as saídas esperadas como resultados a serem obtidos.

Ainda segundo o estudo, o modelo de guias é extremamente funcional, no sentido de prover a informação direta, conduzindo o usuário a refletir sobre sua operação (RAMACHANDRAN, 2012). As propostas que utilizem essa abordagem podem ajudar na redução dos custos através do compartilhamento das ideias que são descritas, semelhantes aos propósitos que são almejados com a aplicação dos guias propostos neste trabalho.

1.4 Metodologia da pesquisa

Esta pesquisa foi realizada em cinco fases. A primeira é relativa à revisão bibliográfica exercida sobre os processos específicos de software propostos pela norma. Também foram executadas as revisões literárias das metodologias, além da coleta das referências de pesquisa através de trabalhos, processos, *frameworks*, *guidelines* etc.

Na segunda fase, foi realizado um estudo detalhado das três metodologias ágeis *Scrum*, XP e FDD. Foram identificadas as atividades de engenharia e gerência que cada uma propõe, sendo realizada uma breve análise de compatibilidade com os processos específicos de software, para definir quais seriam selecionados para integrarem os guias da pesquisa.

A terceira fase foi característica pela concepção dos guias. Foi definida a estrutura de um cabeçalho com três campos: nome, objetivo do processo e o tipo do guia. Também foi arquitetada uma metodologia para a implantação dos processos híbridos através de quatro passos: definir as entradas do processo, realizar a implementação híbrida do processo, avaliar os benefícios, impactos e problemas da implantação e definir as saídas do processo.

Na quarta fase foi estruturado um questionário dividido em três partes para avaliar: o perfil dos profissionais, aceitação do tema e avaliação dos guias pelos engenheiros de

software. Foram escolhidos o tipo da pesquisa (pesquisa de campo), o modo de aplicação (online), a ferramenta (*Google® form*), e por fim o público-alvo.

Na quinta e última fase, foram escritas as conclusões. Também foram realizadas as correções, revisões e melhorias propostas pelo orientador e o co-orientador.

1.5 Estrutura do trabalho

Este trabalho, além de apresentar a motivação, objetivos, contribuição de pesquisa e trabalhos relacionados descritos neste capítulo, desenvolverá o seguinte escopo:

O capítulo 2 apresenta a norma ISO/IEC e IEEE 12207 com a contextualização sobre sua estrutura. São descritos os conceitos, terminologias e recomendações características de cada um dos dez processos específicos de software utilizados na pesquisa.

No capítulo 3, é esboçada uma breve contextualização do manifesto ágil e das metodologias ágeis. O intuito deste capítulo é apresentar os principais aspectos das metodologias *Scrum, eXtreme Programming, Feature Driven Development* apresentando os ciclos de desenvolvimento e práticas que cada uma delas propõe.

O capítulo 4 formaliza o alinhamento das atividades das metodologias citadas de acordo com os processos normativos sumarizados no trabalho. Estão ilustradas as diferenças entre as metodologias para cada fase proveniente do ciclo composto pelos processos específicos de software através de uma série de diagnósticos.

O capítulo 5 descreve a proposta desta pesquisa. São apresentadas as melhores práticas distribuídas em dez guias de processos, de implementação e suporte de software, que auxiliam os profissionais a realizarem corretamente a implantação dos processos híbridos baseados em uma série de requisitos que deverão ser atendidos.

O capítulo 6 enquadra o tipo da pesquisa quantitativa utilizada nesta pesquisa, além da caracterização das variáveis de avaliação, a delimitação da amostra utilizada, os métodos de coleta e tratamento de dados e os resultados de aplicação.

Por fim, no capítulo 7, são expostas as conclusões obtidas com a pesquisa, além das perspectivas dos trabalhos futuros que serão realizados.

Capítulo

2

Processos de software e a norma ISO/IEC e IEEE 12207

Este capítulo apresenta a revisão literária sobre a norma ISO/IEC e IEEE 12207 com uma breve contextualização sobre processos de software, dentre os quais se destacam os processos do contexto de sistema e os processos específicos de software. São detalhadas as atividades dos cinco processos de implementação de software e dos cinco processos de suporte de software utilizados na pesquisa para realizar o alinhamento das metodologias ágeis. Todos os termos contidos nas próximas seções são de exclusiva propriedade da ISO®.

2.1 Contextualização sobre processos de software

O desenvolvimento de software tem se tornado uma atividade complexa de ser realizada ao longo dos últimos anos. A Engenharia de Software tem lançado no mercado uma gama de ferramentas que tem ajudado as empresas a estipularem corretamente a organização dos recursos que serão consumidos nos projetos, através da definição de etapas bem definidas. Quando se planeja realizar algo que define uma sucessão de estados sobre uma perspectiva constante de mudanças, fala-se em processos (BUENO, 2000).

Roger S. Pressman explica que os processos são uma coletânea de atividades estruturadas através de uma organização hierárquica denominada de *arcabouço de processo*, que visa a alcançar produtos de alta qualidade (PRESSMAN, 2006). Ian Sommerville detém a ideia de que os processos agregam ações autogerenciáveis que contêm um fluxo de todas as informações reunidas para solucionar os problemas, associando a busca de resultados eficazes como meta principal a ser estabelecida (SOMMERVILLE, 2011).

Ainda, segundo Leon Osterweil, a essência do processo de software é tida como fator básico para sintetizar soluções para problemas individuais e coletivos (OSTERWEIL, 1987). Cada processo, conforme Watts S. Humphrey torna-se ideal quando contenha os insumos suficientes para desencadear todo o controle das atividades que são requeridas ao longo de um projeto, através das especificações que orientam as pessoas a estabelecerem a divisão correta do trabalho que será realizado (HUMPHREY, 1988).

2.1.1 Processos de software e a normatização

A estrutura dos processos de software sempre varia, não havendo um modelo que reflita uma arquitetura ideal que deve ser seguida pelas empresas (COX, 1990). Entretanto, algumas instituições responsáveis pela normatização de conceitos relativos à Tecnologia da Informação, dentre elas, a ISO®[5], viabilizam o desenvolvimento de pesquisas que avaliam quais processos reportam uma aplicabilidade confiável, e os publicam como documentos de referência, ficando conhecidos como *normas técnicas* (FUGGETTA, 2000).

As normas técnicas, segundo a própria ISO (2007), visam a validar conhecimentos, processos e meios de produção conexos que proporcionam obter resultados eficazes quando forem utilizados. Já o IEEE (2012), afirma que as normas são certificados de segurança que garantem reportar um padrão de qualidade aceitável sobre algo que se pretende tratar como objetivo, sendo os resultados obtidos de sua aplicação, iguais ou melhores aos que foram planejados.

2.1.2 Constructos de atividades e tarefas

Quando as instituições normativas adotam os processos de software como um formato de normas técnicas, elas têm como objetivo publicar uma arquitetura padrão que sirva como modelo de referência no mercado. Para que isso ocorra, a ISO (2008b) explica que é necessário definir uma terminologia, também padrão, que possibilite orientar os profissionais a identificarem como deverá estar organizada a decomposição atual de seus processos, além

[5] ISO® - *International Organization for Standardization* é uma organização não governamental, funda em 1947, com sede em Genebra, Suíça. Ela é responsável regulamentar mais de vinte mil padrões (ISO, 2012).

de distinguir detalhadamente quantas, e quais são, as atividades importantes que estarão contidas nesta estrutura (ISO, 2008b).

Nas normas da ISO, a terminologia que padroniza essa organização é denominada de *constructo* (ISO, 2008b). Os constructos têm como função delimitar as relações existentes dentre os processos de software através da identificação do fluxo das operações que são realizadas por cada processo, como pode ser observado na Figura 2.1, para tentar garantir que o desenvolvimento de novos produtos seja coordenado durante a formação do *ciclo de vida do software*.

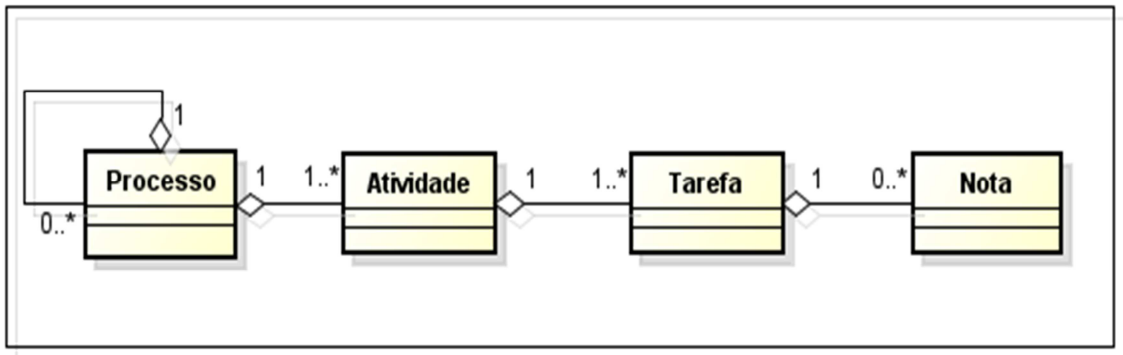

Figura 2.1: Constructo de processos em padrões ISO (ISO, 2008b)

Patel *et al.* (2007) explicam que o ciclo de vida de um software é a cadeia de processos que se comunicam utilizando a base arquitetural apresentada na Figura 2.1. Cada processo pode conter outros processos, denominados de *subprocessos*, sendo ambos formados por *atividades*. As atividades são os procedimentos técnicos relativos à concepção do software, subdivididos em partes menores conhecidas como *tarefas*. Nas tarefas são inseridas as *notas* que contêm sugestões acerca do cumprimento da sequência fixada pelos processos.

2.2 Escopo dos processos da norma

A criação do projeto intitulado 12207 ocorreu em meados de 1988 a partir de um acordo de cooperação realizado entre a ISO e o IEC. O propósito da época era "*disponibilizar um framework comum com processos de software baseado em um ciclo de vida, e uma terminologia bem definida, que pudesse ser referenciado pela indústria de software*" (ISO, 2008a, p.1).

A ABNT (2009) cita que a norma tem como objetivo fornecer uma terminologia de fácil compreensão, com requisitos claros e objetivos. Diferentemente de outras normas da ISO, a ISO/IEC e IEEE 12207 2008, versão que vigora atualmente, não impõe um padrão criterioso de avaliação qualitativa visando à certificação. Ela *"contém um conjunto de processos, atividades e tarefas projetados para serem adaptados de acordo com cada projeto de software"* (ABNT, 1998, p.2).

A aplicação do *framework* pode ser realizada de duas maneiras: conformidade total ou conformidade adaptada. No primeiro caso, "*a conformidade requer que o atendimento dos requisitos de todos os processos veiculados pela norma tenham sido totalmente satisfeitos no projeto*" (ISO, 2008a, p.2). Já para o segundo caso, "*a conformidade parcial é obtida comprovando-se que os requisitos dos processos selecionados para a formação de um novo ciclo de vida tenham sido totalmente satisfeitos no projeto*" (ISO, 2008a, p.2).

O escopo da ISO/IEC e IEEE 12207 é dividido em dois grandes grupos de processos: os processos do contexto de sistema e os processos específicos de software. Cada grupo contém categorias que abrangem especificidades que variam de acordo com a etapa do ciclo de vida, como pode ser observado na Figura 2.2.

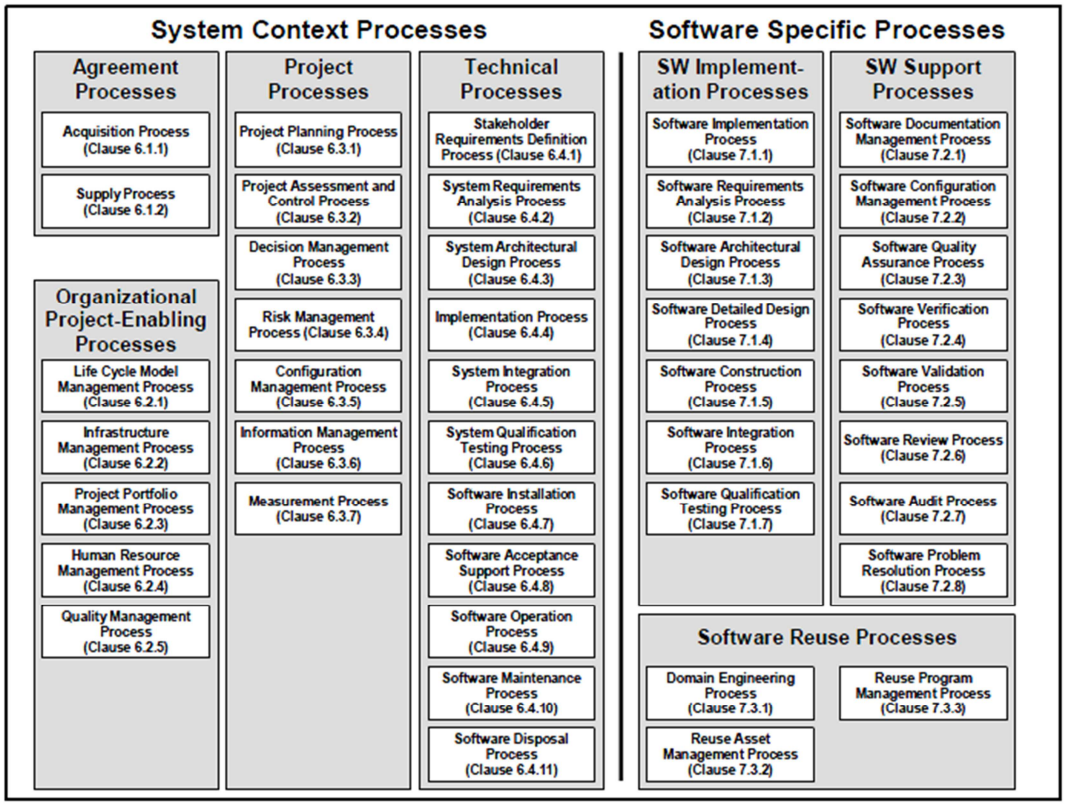

Figura 2.2: Grupos de processos da ISO/IEC e IEEE 12207 (ISO, 2008a)

2.2.1 Processos do contexto de sistemas

A partir do momento em que as empresas de desenvolvimento passam a comercializar o software como um produto para grandes corporações, suas ações comerciais precisam se adequar aos respectivos mercados de atuação. Pennoti *et al.* (2009) explicam que para garantir o atendimento da demanda nos projetos em um modelo específico de negócio, é indispensável desenvolver uma infraestrutura organizacional que integre os processos de negócios aos processos de desenvolvimento no ciclo de vida do software.

Visando a facilitar a integração nesses dois aspectos, a ISO/IEC e IEEE 12207 apresenta, nos processos do contexto de sistema, uma terminologia que especifica os requisitos necessários para prover as relações dos chamados *adquirente* e *fornecedor* (ABNT, 2009). O propósito é firmar as atividades que facilitem a aquisição de produtos de software evitando que surjam eventuais problemas nos contratos existentes. Os processos são divididos em quatro categorias, sendo retratadas na Tabela 2.1.

Tabela 2.1: Categorias dos processos do contexto de sistemas (ISO, 2008a)

Processos	Características
Processos de acordo	Correspondem às atividades voltadas para a definição de contratos, trâmites, na validação da aquisição e fornecimento do software.
Processos organizacionais de projeto	Correspondem às atividades que abrangem os processos de gerenciamento dos recursos externos agregados ao projeto, como contratos com terceiros, gestão de portfólio e gestão de pessoas.
Processos de projeto	Correspondem às atividades relacionadas para a gerência de projeto, tomada de decisões e gerência de riscos.
Processos técnicos	Correspondem às atividades técnicas de acompanhamento da evolução do produto em relação a integração, validação e aceitação pela organização em seu modelo de negócios.

Os processos de cada categoria abrangem aspectos de outras áreas relacionadas ao gerenciamento de sistemas de informação, tais como gestão de projetos, configuração, pessoas, recursos humanos e portfólio. A ISO (2008a) alerta que as parcerias realizadas pelas empresas sejam viabilizadas considerando vários fatores não técnicos, como os riscos, a

qualificação pessoal, a prática da terceirização dos contratos e serviços, além de outros pontos importantes que impactam na eficiência do ciclo de vida que estiver definido (ISO, 2008b).

2.2.2 Processos específicos de software

Diferentemente do grupo anterior, os processos específicos de software não abordam aspectos gerenciais de negócios entre o *adquirente* e o *fornecedor*. Este grupo tem como objetivo reportar para os engenheiros de software[6] quais são os processos, atividades e tarefas essenciais para a formação de novos processos de engenharia, gerência e reuso a serem inseridos no ciclo de vida do software (ISO, 2008a).

As recomendações ocorrem basicamente em três níveis. No primeiro, são apresentados os *processos de implementação*, responsáveis por garantir a análise, codificação, testes, e outras funções relativas à concepção do produto. No segundo nível, são descritos os *processos de suporte*, responsáveis por proverem o apoio necessário aos processos de concepção, com o gerenciamento da documentação, configuração e outras atividades que são paralelas ao desenvolvimento. O último nível abrange os *processos de reuso*, indicados para as empresas que possuam programas de reaproveitamento de software (ISO, 2008a).

Desta forma, assim como nos processos do contexto de sistema, os processos específicos de software também estão divididos em categorias. A primeira delas contém os processos de implementação de software, a segunda categoria possui os processos de suporte de software, e a terceira e última categoria possui os processos que viabilizam o reuso de software (ISO, 2008a).

2.3 Processos de implementação de software

Quando se planeja implementar um software, vários aspectos precisam ser analisados para viabilizar o projeto (NASIR *et al.*, 2008). É indispensável que sejam verificados, por exemplo, os recursos que serão consumidos, a infraestrutura que estará disponível, as tecnologias que os profissionais terão acesso, além de estabelecer todas as atividades que

[6] Engenheiro de software é profissional competente para desenvolver software com o uso de técnicas, modelos e tecnologias avançadas da Engenharia de Software (PRESSMAN, 2006).

serão desempenhadas para que o produto ganhe forma e atenda os requisitos que foram elencados (XIAOGUANG *et al.*, 2008).

Os processos de implementação de software colaboram nesse sentido ao proporem que o desenvolvimento ocorra de maneira direcionada. Em cada um deles são designadas várias saídas (*outcomes*) que referenciam os resultados que podem ser obtidos com a aplicação das atividades recomendadas. Também é sugerida a prática de documentar todos os insumos produzidos, denominados pela norma como *itens de software*[7], como uma estratégia válida para facilitar o controle das operações realizadas ao longo do ciclo (ISO, 2008a).

Ao todo, a norma ISO/IEC e IEEE 12207 dispõe de sete processos de implementação de software. Nas próximas seções, são apresentadas as atividades de apenas cinco desses processos, compatíveis para realizar o alinhamento dos ciclos de vida das metodologias ágeis abordadas nesta pesquisa.

2.3.1 Processo de implementação de software

O primeiro processo é denominado de implementação de software, sendo responsável por orientar os engenheiros de software a traçarem as metas para o desenvolvimento das aplicações (ISO, 2008a). Nesta primeira fase, são abordados os aspectos que possibilitam identificar qual será a escala o projeto, as prioridades definidas pelo cliente, além dos planos que serão necessários para desenvolver o design arquitetural do produto.

A norma sugere que *"se não estiverem fixadas limitações nos contratos, os desenvolvedores têm total liberdade para definir um ciclo próprio de desenvolvimento baseado em estágios relativos à complexidade do projeto"* (ISO, 2008a, p.58). Ela não pré-estabelece uma ordem de como os processos devem estar alinhados, mas recomenda que essa questão seja abordada utilizando-se as seguintes atividades (ABNT, 2009):

- **Definir estratégias, métodos e tecnologias**: a atividade consiste na escolha dos métodos de desenvolvimento, a quantidade de fases, o cronograma, além da alocação das estratégias de projeto que serão utilizadas.

[7] Itens de software são todos os artefatos gerados por cada processo do ciclo de vida da ISO/IEC e IEEE 12207.

- **Definir itens de configuração na implementação:** a atividade consiste em identificar quais serão os fatores de configuração a serem abordados no projeto, tais como ferramentas, tipos de arquitetura e portabilidade.
- **Definir prioridades:** a atividade consiste em identificar e classificar as prioridades de projeto, mediante o comum acordo prestado entre o adquirente e o fornecedor.
- **Documentar e relatar problemas de não conformidade:** a atividade consiste em registrar e reportar os eventuais problemas que possam acontecer durante a execução das atividades de planejamento do projeto.

2.3.2 Processo de análise de requisitos de software

O segundo processo é denominado de *análise de requisitos de software*, sendo responsável por captar as necessidades dos usuários para que posteriormente sejam elencados os requisitos funcionais e não funcionais do sistema (ISO, 2008a). A norma cita que esta fase requer uma atenção especial das empresas no intuito de que sejam selecionados os procedimentos adequados que facilitem a coleta dos requisitos com a participação ativa do cliente.

Segundo a ISO (2008a), o processo deve prover garantias de rastreabilidade em relação ao planejamento efetuado e o controle das especificações. Para que isso ocorra, as empresas devem adotar os meios viáveis de gerir as informações provenientes dos requisitos, evitando que ocorram problemas posteriores durante o processo de construção. As atividades recomendadas para serem inseridas em processo eficiente, segundo a ABNT (2009), são:

- **Especificar os requisitos funcionais de projeto:** a atividade consiste em especificar os requisitos técnicos, baseados em *estórias*, e o levantamento das funcionalidades que serão documentadas como requisitos do usuário.
- **Especificar os requisitos não funcionais de projeto:** a atividade consiste em especificar os requisitos não funcionais que também são definidos durante o planejamento das funcionalidades. Dentre alguns citados pela norma, estão os recursos humanos, ambientação de trabalho e o treinamento.
- **Qualificar os requisitos:** a atividade consiste em padronizar as informações, a nomenclatura e todos os termos abordados pelos requisitos para que não sejam gerados problemas no entendimento dos objetivos.

- **Ordenar os requisitos:** a atividade consiste em especificar as exigências de como os requisitos serão implementados, situando uma escala de importância para os que têm maior e menor prioridades.

2.3.3 Processo de construção de software

O terceiro processo de implementação de software é conhecido como *processo de construção de software*, característico por recomendar os procedimentos utilizados durante a codificação das chamadas *unidades de software*. Cada unidade representa uma classe que contém várias estruturas funcionais como métodos, variáveis e objetos, que possibilitam garantir que o desenvolvimento das aplicações seja dividido a partir de vários módulos (ABNT, 2009).

Para a ABNT (2009), a construção de um software deve abranger três fatores básicos: a realização contínua dos testes das unidades, a verificação de atendimento aos requisitos e a concepção da documentação do funcionamento do software. Apesar de ser um processo extremamente técnico, a ISO (2008a) recomenda como atividades:

- **Codificar módulos e procedimentos de software:** o implementador deve estabelecer toda a configuração necessária para promover a codificação do software, através da codificação das bases de dados, procedimentos, métodos, dentre outras estruturas.
- **Testar as unidades de códigos e bases de dados:** na construção os testes devem ser providos simultaneamente a codificação com o uso de estratégias a serem adotadas pelos profissionais.
- **Avaliar códigos em aceitação aos requisitos:** a compatibilidade e atendimento aos requisitos é uma prioridade para determinar a qualidade do produto, sendo uma prática que precisa ser adotada pelos profissionais ao longo da codificação do software.
- **Documentar/gerenciar documentos de códigos:** todas as informações precisam ser documentadas para facilitar o controle sobre a codificação.

2.3.4 Processo de integração de software

O quarto processo, conhecido como *processo de integração de software*, tem como objetivo garantir que os componentes codificados no processo de construção possam ser integrados corretamente (ISO, 2008a). Durante esta etapa, são escolhidos os procedimentos que corrigem

os problemas de incompatibilidade gerados na codificação, para que as funcionalidades não sejam afetadas quando forem geradas as futuras versões do software.

Alguns pontos destacados pela norma expressam a inserção maciça de testes de software. As empresas devem garantir que a uniformidade seja mantida quando for preciso alterar documentos, códigos ou até mesmo o próprio processo para viabilizar o acompanhamento em tempo real das equipes envolvidas. As atividades que a norma sugere para a formação deste processo são (ABNT, 2009):

- **Definir um plano de integração:** a atividade consiste em especificar um cronograma estratégico que ajude os profissionais a realizarem a integração bem sucedida, organizando a integração em relação ao design, a codificação e a usabilidade.
- **Definir os casos de testes agregados:** a atividade consiste em comparar duas ou mais unidades de software para identificar os problemas durante a integração e corrigi-los para que a funcionalidade não seja afetada.
- **Avaliar e reportar diferenças:** a avaliação consiste em detectar quais são os efeitos positivos e negativos que a integração pode ocasionar. Neste caso, são documentados os resultados obtidos para serem apresentados às equipes em seguida.
- **Propor mudanças:** depois de detectados e avaliados os problemas, é essencial propor mudanças a cada novo ciclo de integração.

2.3.5 Processo de qualificação e testes de software

O quinto processo da primeira categoria, selecionado para esta pesquisa, é o *processo de qualificação e testes de software* (ABNT, 2009). Considerado como um dos mais importantes, este processo tem como finalidade *"verificar se as funcionalidades do software integrado após a codificação correspondem às suas exigências definidas"* (ISO, 2008a, p. 65).

O termo *qualificação* nesta fase corresponde a uma minuciosa adequação do produto para a execução de re-testes sempre quando forem geradas novas versões. Os procedimentos dependem de como está customizada a sequência de testes, além da seleção de um conjunto de ferramentas que usem como protocolo as seguintes atividades (ABNT, 2009):

- **Especificar os testes unitários:** a atividade consiste em criar os cenários com várias situações diferentes para avaliação das funcionalidades e dos problemas encontrados com a aplicação dos testes.

- **Avaliar o processo e os testes:** a atividade consiste em avaliar a eficiência do processo de testes com a criação de diagnósticos e a realização de auditorias ao longo do processo.
- **Elaborar relatórios dos testes:** para cada teste, podem-se produzir relatórios que apresentam uma breve descrição dos diagnósticos obtidos. Os relatórios podem conter informações técnicas que variam de acordo com os procedimentos utilizados.
- **Propor inspeções:** a atividade consiste no acompanhamento dos engenheiros para garantir que todos os procedimentos estão sendo seguidos corretamente pelas equipes.

2.4 Processos de suporte de software

Grande parte dos modelos atuais de desenvolvimento provém condições suficientes para que seus processos de *engenharia* surtam efeitos positivos quando forem aplicados. Entretanto, esta possibilidade só se torna possível de ser alcançada, quando as atividades que os compõem são geridas corretamente pelos engenheiros ao longo do ciclo de vida do software (HOYER e CHROUST, 2006).

A norma ISO/IEC e IEEE 12207 recomenda que essa gestão seja adotada através dos *processos de suporte de software* (XU, 2005). Segundo a norma, a função dos processos é proporcionar as condições ideais para que a construção do software seja coordenada, abordando os fatores não funcionais como documentação, configuração e verificação, que também contribuem para que o produto obtenha índices aceitáveis de qualidade.

Ao todo, a norma dispõe de oito processos de suporte de software. Nas próximas seções são apresentadas as atividades de apenas cinco processos, selecionados por serem compatíveis para realizar o alinhamento dos ciclos de vida das metodologias ágeis abordadas nesta pesquisa.

2.4.1 Processo de gerência da documentação de software

O primeiro processo dessa categoria é denominado de *processo de gerência da documentação de software*, responsável por *"registrar e manter as informações colhidas em todos os processos durante a produção do software"* (ISO, 2008a, p. 66). Desde as entrevistas, reuniões entre equipes, aos acertos sobre as versões dos protótipos, é inevitável gerar um

volume de dados que precise ser gerenciado para orientar quais medidas serão adotadas em uma determinada etapa do ciclo de implementação.

A documentação, como muitos profissionais imaginam, não visa a arquivar um contingente alto de informações sobre uma determinada tarefa específica. Para a ISO (2008a), cada artefato gerado deve garantir que as informações representem uma fonte de entrada ou saída dos dados suficiente para adequar a execução de um ou mais processos sucessivos. As atividades recomendadas para a gerência das documentações segundo a norma são:

- **Especificar os documentos que serão produzidos:** a atividade consiste em identificar quais artefatos serão produzidos durante o ciclo de vida do software. A documentação pode ser simbolizada por guias, documentos técnicos, relatórios etc.
- **Especificar o conteúdo e formato de produção:** a atividade consiste em definir um padrão visual para a documentação, se a mesma será física ou digitalizada, qual o formato dos dados e o conteúdo que será abordado.
- **Especificar as fontes de produção:** a atividade consiste em identificar os autores da documentação, a versão rotulada no documento e as demais documentações que referenciam a mesma para serem utilizadas.
- **Editar, revisar e alterar documentos**: a atividade consiste em alocar os procedimentos que garantam realizar as devidas atualizações dos documentos.

2.4.2 Processo de gerência da configuração de software

O segundo processo de suporte, incluído na pesquisa, é conhecido como *processo de gerência da configuração de software*, responsável por alinhar, disponibilizar e manter a integridade dos artefatos, insumos e documentos gerados ao longo do projeto. O processo é executado paralelamente aos demais processos do ciclo de vida, instituindo uma infraestrutura acessível que possibilite gerenciar todas as modificações que serão realizadas.

No *framework* não são detalhadas as diretrizes e limitações sobre possíveis estratégias de configuração, mas é recomendado que as empresas contratem profissionais capacitados para realizarem este serviço em parceria com os *stakeholders*. A ISO (2008a) apresenta como atividades importantes para gerir as configurações as seguintes:

- **Definir um plano de gerenciamento:** a atividade consiste em elaborar um plano de gerenciamento das configurações com procedimentos, cronogramas de manutenções, atualizações e o controle de versões para facilitar eventuais backups.
- **Identificar os itens de software:** a atividade consiste em criar uma base de dados para que sejam armazenadas as versões dos documentos e todos os códigos-fontes gerados durante a codificação.
- **Gerenciar status/execução de configuração:** a atividade consiste em prover o gerenciamento adequado das mudanças ocorridas nos itens de software com o uso de ferramentas especializadas.
- **Avaliar e reportar mudanças:** a atividade consiste em acompanhar o processo avaliando os problemas encontrados e em seguida reportando as mudanças que podem ser viabilizadas para sua melhoria.

2.4.3 Processo de garantia da qualidade de software

O terceiro processo de suporte é denominado como *processo de garantia da qualidade de software*, possuindo como função *"padronizar um nível excelência no atendimento das exigências implícitas e explícitas estabelecida nos planos*". ISO (2008a, p. 69). Neste processo são destacados os subsídios necessários para que os engenheiros mantenham uma política de melhorias para tornar seus produtos e serviços reconhecidos (ABNT, 2009).

A norma não restringe o termo "qualidade" através de uma série de critérios semelhantes a outras normas da ISO, mas aconselha que as empresas sempre realizem uma avaliação crítica em seus processos. Desta forma, é possível buscar investimentos quando surgirem novas oportunidades para a aquisição de "maturidade" no desenvolvimento de novas tecnologias. As atividades-chave traçadas pela norma pela busca da qualidade são (2008a):

- **Definir um plano de qualidade:** a atividade consiste em desenvolver os planos da qualidade que descrevem os cronogramas de acompanhamento e avaliação dos processos de software.
- **Especificar os procedimentos de processo:** a atividade consiste em descrever os procedimentos relativos a análise, medição e melhoria para corrigir os problemas encontrados nos processos.

- **Especificar os procedimentos do produto:** a atividade é similar à anterior, consistindo na descrição dos procedimentos relativos a análise, medição e melhoria para corrigir os problemas encontrados no produto.
- **Avaliar e documentar índices da qualidade:** a atividade consiste em registrar e avaliar o ganho ou perda da qualidade nos processos e produtos através de procedimentos estatísticos.

2.4.4 Processo de verificação de software

O quarto processo de suporte é o *processo de verificação de software*, responsável por promover uma "confirmação" de que os resultados obtidos com a codificação correspondem aos requisitos elencados (ISO, 2008a). A verificação é realizada pelos os engenheiros, com o objetivo de encontrar possíveis problemas que tenham passados despercebidos durante a realização dos processos de construção e testes de software (BASILI *et al.*, 2009).

A norma cita que a verificação é uma espécie de segundo "pente fino" após os testes, em que as equipes avaliam a usabilidade do produto. Para que isso ocorra, é necessário realizar a verificação a cada nova iteração, à medida que o produto for evoluindo, ou caso sejam geradas novas versões, mesmo que contenham apenas pequenas mudanças. As atividades que regem uma verificação eficiente, segundo a ISO (2008a), são:

- **Definir um plano com requisitos de verificação:** a atividade consiste em elaborar um plano de verificação abordando os pontos que devem ser considerados como critérios de aceitação do software. A norma cita que a criação do plano não é obrigatória, mas orienta que sejam estipulados pelos menos os protocolos de inspeção para garantir que a verificação surta efeitos.
- **Verificar o código do software:** a atividade consiste avaliar os códigos-fonte, métodos, funções, bases de dados e demais estruturas funcionais do software que poderão ser movidas as alterações.
- **Verificar o design do software:** a atividade consiste em analisar se todas as funcionalidades do software estão devidamente acessíveis para o cliente de maneira intuitiva.
- **Verificar documentação:** a atividade consiste em atualizar os documentos mediante as mudanças realizadas no software ao longo das verificações.

2.4.5 Processo de validação de software

O último processo de suporte selecionado para a pesquisa é conhecido como *processo de validação de software*, responsável por dispor para o cliente versões de testes do software (ISO, 2008a). O processo é realizado geralmente após o processo de verificação, e consta também com a participação dos *stakeholders*, desenvolvedores e demais envolvidos diretamente no projeto (MATTIELO-FRANCISCO *et al.*, 2006).

Na norma, é recomendada a simulação do cenário em que o software irá desempenhar suas rotinas para a execução de testes. A validação é alcançada gradativamente, à medida que o cliente vai adquirindo um contingente de satisfação através dos resultados apresentados no seu dia a dia. Para que seja possível perceber os possíveis problemas durante o refinamento da aplicação, são recomendadas as seguintes atividades (ABNT, 2009):

- **Definir um plano de validação:** a atividade consiste em estipular os prazos, os meios, os recursos e todo o cronograma de validação que será adotado ao longo do processo de interação dos envolvidos.
- **Elaborar testes de simulação:** a atividade consiste em especificar os testes de simulação para identificar as falhas no software. Alguns dos testes recomendados pela norma são os testes de estresse, testes de entrada de dados e os testes de repetição.
- **Elaborar casos de testes do usuário:** a atividade consiste em planejar como serão realizados os testes individuais e quais serão as tarefas pré-estabelecidas a serem realizadas pelos clientes.
- **Avaliar e reportar mudanças:** a atividade consiste em analisar as mudanças identificadas pelo processo e comunicá-las aos desenvolvedores para que sejam corretamente implementadas.

2.5 Considerações finais

A utilização das normas ISO como solução para a melhoria dos processos de software ainda gera resistência por grande parte das empresas. Koscianski e Soares (2007) citam que muitos profissionais desconhecem que as instituições normativas também publicam suas normas em formatos de guias mais simples, sem que sejam criados vínculos obrigatórios que exijam a certificação (KOSCIANSKI e SOARES, 2007).

O padrão ISO/IEC e IEEE 12207 não é uma norma que “certifica” que as empresas que a utilizarem atingirão êxito em suas operações, mas “possibilita” que as mesmas promovam uma organização de seus processos plausível a sua infraestrutura. O *framework* pode ser considerado como um leque de atividades tidas como padrões na maioria dos modelos, processos e métodos de desenvolvimento, concedendo aos seus usuários uma alternativa para consulta, quando se planeje formatar ciclos de vida de várias complexidades (KOSCIANSKI e SOARES, 2007).

As duas categorias de processos específicos de software abordadas neste trabalho servem como norte para alinhar os dois fatores principais que regem o funcionamento ideal do ciclo de vida de software: a engenharia e a gestão dos processos. Percebe-se que os processos da primeira categoria focam abordar as atividades primárias de concepção do software, mas que as atividades de suporte promovidas pela segunda categoria provêm as condições necessárias para que as mesmas sejam realizadas corretamente.

O que se pode concluir acerca de uma análise detalhada deste padrão é que o mesmo é uma fonte indispensável para a criação de novos processos. As recomendações são objetivas acerca do objetivo principal que cada processo deve alcançar com a descrição das atividades apresentadas ao longo das seções. Além disso, ele não restringe a ordem, fluxo ou sequência em que os processos precisam estar organizados, concedendo total liberdade para que os engenheiros sintetizem livremente o ciclo de vida com seus próprios processos (ISO, 2008a; ABNT, 2009).

Capítulo

3

Metodologias ágeis

Este capítulo apresenta a revisão literária das metodologias ágeis, abordando seu surgimento, princípios e valores. São detalhadas as características do *Scrum*, que veicula o desenvolvimento através do gerenciamento de fases bem definidas, do *eXtreme Programming* que reúne práticas que melhoram a relação das equipes com o cliente, e por fim, do *Feature Driven Development*, que busca balancear a agilidade de acordo com as funcionalidades que serão concebidas.

3.1 Contextualização sobre as metodologias ágeis

A expressão "*metodologias ágeis"* surgiu no ano de 2001, a partir de um encontro que reuniu dezessete profissionais experientes de várias empresas, tornando-se conhecido como "Manifesto Ágil" (AGILE MANIFESTO, 2012). Este acontecimento ficou marcado para a Engenharia de Software como o divisor de uma nova era, com o surgimento de novos métodos, modelos e ferramentas para o desenvolvimento de software (FOWLER, 2001).

As metodologias ágeis foram lançadas no mercado como uma promessa de simplificar as atividades tradicionais de seus modelos antecessores (HIGHSMITH e COCKBURN, 2001). Soares (2004) explica que essa abordagem sugere métodos eficazes que descartam o uso de documentações extensas, a redução dos custos, e prezam pela adaptabilidade, visto que a concepção do software não depende apenas da definição de um processo qualificado, mas também das pessoas que o executam (SFETSOS E STAMELOS, 2010).

Para Ferreira e Lima (2006), as metodologias ágeis "*direcionam as condições favoráveis para fornecer as iterações e retroalimentações entre usuários e o sistema durante*

todo o projeto, uma vez que são as necessidades reais do usuário, e não o conceito do sistema ideal" (FERREIRA e LIMA, 2006, p. 6). Vasco *et al.* (2005) defendem que o que diferencia o modelo "pesado" do "leve" é a intensa participação dos membros envolvidos no projeto.

Nerur e Balipejally (2007) discutem que a "aquisição" da agilidade depende da condição de tornar flexíveis as políticas consideradas rígidas e legadas. A burocracia precisa estar extinta para que as decisões tomadas não gerem futuros problemas, ou principalmente dependências, mesmo que o processo adotado não seja totalmente eficiente.

Alguns autores (GHEORGE *et al.*, 2006; ABBAS *et al.*, 2008) concordam que grande parte das metodologias ágeis trata eficientemente situações classificadas como "imprevisíveis". A escolha da metodologia que melhor convém ao cenário deve ser definida através de um planejamento realizado previamente pelos engenheiros, para viabilizar o atendimento da demanda arcada pelas suas empresas (BORGES JÚNIOR *et al.*, 2011).

Baseado nisso, a comunidade ágil instituiu doze tópicos com os valores ágeis, denominados de *princípios*. O intuito é priorizar a inserção das práticas que valorizem a maturidade dos engenheiros, com um foco destinado para a constante melhoria do produto e a satisfação do cliente. Esses princípios ficaram conhecidos como: foco no cliente, requisitos sobre demanda, software funcionando, time unido, ambiente promissor, comunicação efetiva, software como ponto de evolução, desenvolvimento sustentável, nivelamento técnico, simplicidade, melhor arquitetura e reflexão (AGILE MANIFESTO, 2012).

Desta forma, várias tendências surgiram, diversificando aos poucos a abordagem do desenvolvimento ágil. Algumas metodologias propõem atividades e ciclos completos para gerir os projetos, enquanto outras apenas reúnem um conjunto de práticas que melhoram os processos já existentes. Nas próximas seções, serão apresentadas três dessas metodologias ágeis, referenciadas nos guias contidos nesta pesquisa.

3.2 Scrum

A primeira metodologia ágil, criada oficialmente em 1995, é conhecida como *Scrum* (SCHWABER, 2004; SCHWABER, 2007). Seu criador, *Ken Schwaber*, a define como um *framework* que tem como propósito garantir o gerenciamento ágil dos projetos em que os requisitos mudam rapidamente. Ela também é caracterizada como iterativa e incremental, mas

que não descreve as atividades que devem ser realizadas em cada fase, e sim a formação de um ciclo de vida progressivo de implementação.

O *Scrum* se baseia nos princípios ágeis, sendo recomendado para equipes que possuam um número reduzido de componentes (KNIBERG, 2007). A adaptação da metodologia é realizada no processo de software, buscando adequar uma infraestrutura de desenvolvimento que esteja compatível à execução das práticas que a mesma propõe. Dentre uma dessas práticas, está a inserção do cliente em algumas atividades que serão desempenhadas, para que o mesmo sirva como referência para a evolução do produto.

Kniberg (2007) explica ainda que o *Scrum* não possui restrições quanto ao seu uso em relação a diferentes portes de projeto. A flexibilidade depende de como a metodologia é utilizada pela empresa, podendo abranger vários times trabalhando simultaneamente, desde que todos sigam o mesmo processo de maneira sincronizada. Outro ponto de destaque é a objetividade, quando a documentação é parcialmente descartada, visando a tornar mais simples o conteúdo de seus artefatos.

3.2.1 Papéis

No *Scrum*, a divisão do trabalho é explícita, sendo as responsabilidades estabelecidas de acordo com o perfil de atuação de cada profissional (SCHWABER, 2004). Os papéis são atribuídos analisando-se alguns fatores, tais como a quantidade de tarefas, o tempo de projeto, a experiência dos participantes, a demanda de operações nas iterações, e até mesmo o custo arcado pela contratação de cada um. Os principais papéis e suas respectivas características, segundo *Schwaber* (2004), são:

- ***Product Owner:*** é o profissional responsável por contatar os clientes e representar os mesmos durante o projeto, visto que sua função exige o conhecimento dos negócios.
- ***Scrum Master:*** é o profissional responsável por remover todos os impedimentos existentes no processo durante o desenvolvimento do software. Também é conhecido como “facilitador do time”, pois busca as soluções mais rápidas até mesmo nas situações consideradas mais difíceis.
- ***Scrum Team:*** é o time de profissionais que desenvolve o software, contando com os papéis de analistas, desenvolvedores e testadores, variando em média dentre cinco a nove componentes.

3.2.2 Fases e ciclo de vida

A agilidade é idealizada pelo *Scrum* desde o planejamento inicial até a entrega do software funcionando em sua versão final. O ciclo de vida é composto por fases curtas que têm como objetivo facilitar o acompanhamento da concepção do produto ao longo do projeto. Na metodologia, é mantida uma sequência contínua de execução, representada na Figura 3.1, que é repetida pelos times até que os resultados das primeiras versões satisfaçam as necessidades dos clientes, ou, em segundo caso, quando o projeto estiver totalmente finalizado.

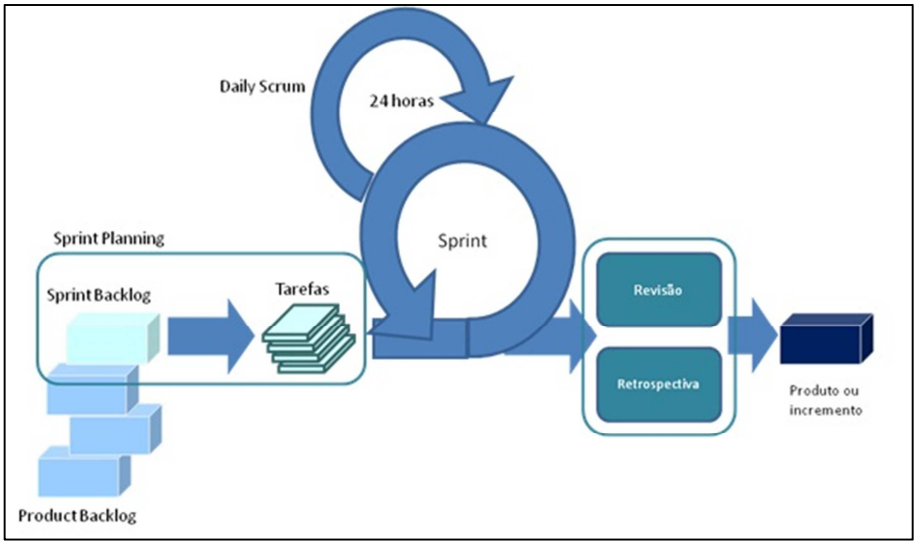

Figura 3.1: Ciclo de vida do *Scrum* (RORIZ FILHO, 2011)

A metodologia contribui para o desenvolvimento sequencial sem estipular atividades fixas, entretanto, recomenda o gerenciamento semelhante ao modelo PDCA[8] (*Plan-Do-Check-Act*). Sempre que houver *"um incremento de software pronto, o esforço de cada iteração está em se entregar os itens com o maior valor de negócio (e prioridade) ao cliente, dando a ele algo real e de valor para o negócio"* (SALGADO *et al.*, 2010, p. 3).

O fluxo de desenvolvimento é baseado em iterações denominadas pela metodologia como *Sprints*, possuindo um prazo médio de duração de duas a quatro semanas. O projeto se inicia na fase de *Planejamento* quando há a formação de uma lista com os primeiros requisitos do cliente, chamada de *Product Backlog*, onde são definidos os *itens de backlog*. Cada item de *backlog* é dividido em atividades menores, "quebradas" em tarefas, sendo

[8] PDCA é um modelo de quatro fases que ordena as atividades de gestão de um projeto (VILAÇA *et al.*, 2011).

posteriormente lançados para o time, com um tempo de entrega estipulado através de uma técnica conhecida como *Planning Poker* (SCHWABER, 2004).

Na segunda fase, a *Stagging*, a *Sprint* se inicia com uma reunião de planejamento dividida em duas partes, conhecida como *Sprint Planning Meeting*. Na primeira parte da reunião, a *Sprint Planning I*, o *Product Owner*, em parceria com time, estipula o que será implementado em cada uma das *Sprints*. Na segunda parte, a *Spring Planning 2*, o time planeja como organizará seu trabalho com a formação do *Sprint Backlog*, que facilitará a ordenação e a seleção das tarefas pelos profissionais (MARÇAL *et al.*, 2007).

A fase de *Desenvolvimento* dura até trinta dias, dependendo do tempo afixado do projeto e da quantidade de times envolvidos. Diariamente, é recomendada a prática do *Daily Scrum Meeting*, que são reuniões diárias que não excedem o tempo máximo de quinze minutos, onde cada time discute basicamente uma atualização sobre o estado em que se encontra suas atividades. Na pauta deve ser perguntado o que foi feito, o que está sendo feito, o que será feito, além de reportar os impedimentos ao *Scrum Master*.

Um recurso importante que facilita o gerenciamento das atividades ao longo da *Sprint*, segundo explicam Salgado *et al.* (2010), é o uso de um quadro típico das metodologias ágeis, conhecido como *Task Board*. Nele são afixadas as tarefas com as nomenclaturas *To Do* (para fazer), *In Progress* (em andamento), e *Done* (concluída), separadas através de linhas e colunas que dispensam a adoção de ferramentas mais avançadas.

Ao término da *Sprint*, inicia-se a fase de *Releasing*, com a realização da *Sprint Review* e da *Sprint Retrospective*. A primeira *release* consiste de uma reunião do time com o *Product Owner* para verificar os resultados da *Sprint* e a demonstração das funcionalidades finalizadas. Após o término da revisão, é realizada a retrospectiva do time com a participação do *Scrum Master*, onde são avaliadas as lições aprendidas e as mudanças da próxima *Sprint*.

3.2.3 Artefatos

As atividades realizadas ao longo do ciclo de vida do *Scrum* produzem poucos, mas alguns artefatos importantes. O fato de a iteração ser curta não dispensa totalmente a prática de sintetizar pequenas documentações que servem como um auxílio para guiar os profissionais a desenvolverem novas soluções (SCHWABER, 2007).

O primeiro artefato, mencionado na seção anterior, é o ***Product Backlog***. Este artefato, segundo explicam Oliveira Filho e Alves (2010), tem o formato de uma lista convencional de requisitos funcionais, mas com um pequeno diferencial ao atribuir as estimativas de cumprimento, as dependências, e um breve resumo (*estória*) dos objetivos de cada um deles.

Outro fundamental, segundo os autores, é o ***Sprint Backlog***. Seu formato é semelhante ao *Product Backlog*, mas como o próprio nome sugere, o conteúdo é formado com as tarefas específicas traçadas para cada *Sprint*. O terceiro artefato é o ***Burndown Chart***, composto por um gráfico que compara a escala de horas trabalhadas ao tempo percorrido de projeto.

3.3 eXtreme Programming

A programação extrema é uma metodologia ágil criada por Kent Beck em 1996 que recorre aos princípios do Manifesto Ágil. Ela é caracterizada como um processo iterativo e incremental que possibilita o software ser implementado desde as primeiras fases iniciais do projeto, ganhando forma e novas funcionalidades à medida que a satisfação do cliente for sendo correspondida (BECK, 1999).

O XP, segundo Teles (2006), é voltado para projetos curtos, cujos requisitos são classificados como vagos e mudam constantemente. Uma peculiaridade levantada pelo autor em seu livro é que essa metodologia é aconselhada para o desenvolvimento de sistemas que possuam uma arquitetura orientada a objetos e equipes preferencialmente pequenas, com o intuito de facilitar a integração do cliente como participante das atividades.

Layman *et al.* (2006) relatam que uma das regras do XP é empregar uma perspectiva em esforços pessoais para almejar o ganho de desempenho e possibilitar o desenvolvimento rápido de software. As metas são traçadas através de um jogo de planejamento realizado em cada fase, para evitar que surjam problemas inesperados ou futuras redundâncias.

3.3.1 Valores, princípios, e práticas

Os mecanismos que habilitam a condução de um processo de software detém certa dependência de como atuam os profissionais. O *feedback* constante concede o aprendizado obtido em diversas oportunidades retratadas pelas pessoas, através de um cenário de

reavaliação das necessidades, das tarefas, e de um direcionamento de mudanças que ocorre gradativamente com o contato direto entre a empresa, a equipe e os clientes.

O que fomenta a atenção e a agilidade está diretamente interligado ao valor permanente da ***comunicação***, para encaminhar as informações certas para as pessoas certas. É importante sempre garantir o ***feedback*** entre a equipe, e que as atividades transcorram com ***simplicidade*** preocupando-se apenas com o que realmente interessa (BECK, 1999).

Beck (1999) relembra que os profissionais devem ser guiados por esses valores com o uso de *doze práticas* sugeridas pelo XP. Campelo (2003) explica que os procedimentos adotados ao pouco tempo de projeto, apresentados na Tabela 3.1, são de fundamental importância para alinhar a criação de ciclos curtos a uma produção realmente efetiva.

Tabela 3.1: Práticas propostas pelo XP (CAMPELO, 2003)

Prática	**Característica**
Jogo de Planejamento	Esta prática sugere que seja criada uma estreita relação entre o cliente e a equipe para definir as prioridades momentâneas de projeto.
Releases curtos	A prática consiste em fixar fases curtas e sempre disponibilizar o produto com pequenas atualizações.
Metáfora	Consiste em adotar termos comuns para a comunicação com o cliente.
Design simples	A arquitetura e o código devem ser os mais objetivos possíveis.
Desenvolvimento guiado por testes	Todas as funcionalidades implementadas devem ser testadas para eliminar o máximo de erros e problemas possíveis.
Refatoramento	Consiste em simplificar o código para viabilizar o reaproveitamento.
Programação em pares	A prática consiste em se ter dois programadores trabalhando simultaneamente no mesmo computador.
Código coletivo	Consiste em compartilhar o código para identificar novos problemas.
Código padronizado	A prática é recomendada para facilitar a organização das estruturas lógicas e o entendimento do código por todos os programadores.
Ritmo sustentável	Nenhum componente da equipe deve suportar uma carga de trabalho extensiva ou acumular muitas funções ao mesmo tempo.
Integração contínua	Depois que uma nova versão é implementada, ela pode acarretar alterações que precisam ser revistas para evitar conflitos.
Cliente presente	Consiste em inserir o cliente como um participante ativo na equipe.

3.3.2 Equipes XP

No XP, os times são denominados de “equipes”, obtendo-se papéis fixos. O intuito é estimular que os membros se aperfeiçoem em suas respectivas áreas dando o melhor de si diariamente, mas sem limitar o relacionamento e a preocupação nas demais responsabilidades para ajudar os colegas (TELES, 2006).

O ***gerente de projeto*** é o encarregado de tratar os assuntos pertinentes à administração organizacional e os assuntos externos para viabilizar a participação ativa do cliente como protagonista no fornecimento das informações. Ele também é responsável por garantir o cálculo das estimativas, as parcerias fechadas com as outras empresas, a alocação de novos profissionais e a aplicação dos recursos ao longo do projeto (TELES, 2006).

O líder da equipe é chamado de ***coach***, sendo o encarregado de orientar todos para que as instruções do gerente de projetos sejam repassadas corretamente. O responsável por averiguar os problemas nas funcionalidades, o ***analista de testes***, exerce um papel fundamental ao planejar os cenários e executar os testes de aceitação, além de verificar os resultados alcançados por cada versão do software em parceria com o cliente (TELES, 2006).

O ***redator técnico*** é o profissional competente para documentar e homologar os artefatos durante o projeto, dos quais se podem destacar as *estórias*, modelagens, o próprio código e os manuais do software e do usuário. Já o ***desenvolvedor*** é o profissional credenciado para analisar, codificar, testar e prestar o suporte técnico nas unidades de software através de um ambiente integrado de desenvolvimento (TELES, 2006).

3.3.3 Ciclo de vida

No ciclo de vida do XP, as atividades de desenvolvimento são realizadas todas ao mesmo tempo. As iterações envolvem compartilhar um conjunto de pequenos recursos disponibilizando os subsídios para que as equipes façam um pouco de planejamento, codificação, testes etc. As fases estão divididas através de um fluxo sequencial, representado na Figura 3.2, que estabelece o que deve ser feito em cada etapa à medida que as prioridades vão sendo alinhadas e o produto vai reportando os resultados esperados (BECK, 1999).

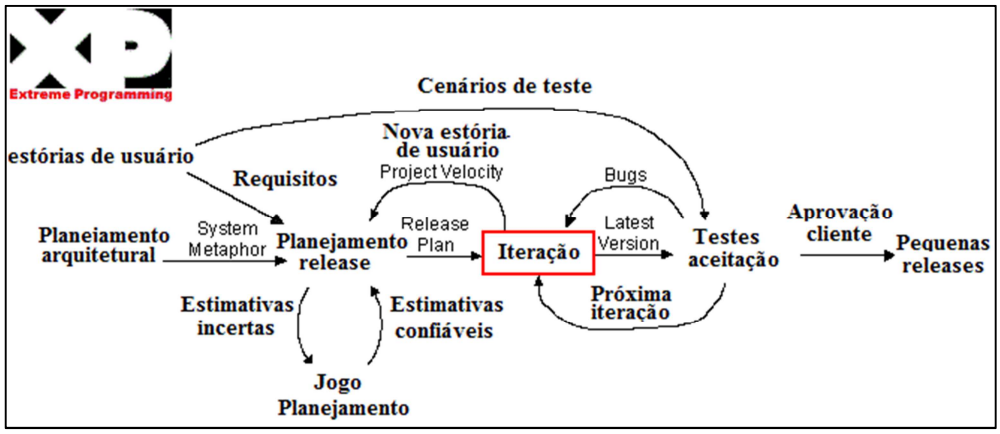

Figura 3.2: Ciclo de vida do XP (Adaptado de GONZÁLEZ, 2004)

O ***planejamento*** é baseado nas estimativas de tempo, custo e esforço pessoal para que seja gerada uma espécie de "plano" da versão que será desenvolvida. O jogo estratégico, neste caso, visa a gerenciar as decisões que serão tomadas através de uma análise dos riscos e das dificuldades que serão encontradas ao longo da *release*, para garantir o retorno de investimento do cliente, sem que nenhum contratempo impacte diretamente nas próximas iterações (AMBLER, 2002).

Na ***análise***, não há um tempo fixo para a coleta e listagem dos requisitos. O processo ocorre a todo o momento, sempre que o cliente se reúne com a equipe e apresenta suas mais novas necessidades, dando ênfase a uma rotatividade alta de informações que é tratada pela *modelagem ágil* (AMBLER, 2002). O design e a usabilidade também detêm certa atenção para serem implementados aos poucos com base em uma vistoria feita ao longo do processo.

Já para a ***codificação***, Tian *et al.* (2010) explicam que uma característica comum nesta parte da produção é o desenvolvimento guiado por testes. "*O desenvolvedor escreve os testes para uma unidade de código, antes de escrever o código em si, para passar no teste de aceitação e compatibilizar os padrões de refatoramento.*" (TIAN *et al.*, 2010, p. 3).

Complementando a codificação, nos ***testes*** são agregadas mais ferramentas que contribuem para uma avaliação mais detalhada do software, buscando eliminar os defeitos. Os testadores integram-se com os programadores para analisar falhas, erros e bugs, elaborando uma sequência de testes que submetem o software a situações extremas. No XP, isso se torna possível graças à inserção das práticas do *Test Driven Development*[9] (BECK, 1999).

[9] *Test Driven Development* (TDD) é uma técnica que agrega um laço de repetições das operações de teste. Os principais são os testes de unidade, testes dirigidos e os testes de aceitação (BECK, 1999).

Ao término do ciclo, a iteração encerra-se com a ***implementação*** de uma versão parcial de testes para a avaliação prática do cliente. Neste meio tempo, são discutidas as mudanças, a manutenção e a instalação da próxima versão para garantir que o software esteja funcionando perfeitamente até o término do projeto.

3.4 Feature Driven Development

O *Feature Driven Development* (Desenvolvimento Guiado por Funcionalidades) é uma metodologia que foi criada em 1997 por Peter Coad e Jeff DeLuca, que combinava algumas técnicas de gerenciamento ágil com atividades da engenharia para o desenvolvimento iterativo e incremental de software (ANDERSON, 2004).

O diferencial do FDD é que sua filosofia de projeto não segue diretamente os princípios ágeis, mas também não é considerada um processo prescritivo tradicional. Ela reúne um equilíbrio entre ambos os extremos, no sentido de inserir uma postura de gerenciamento mais firme nos processos, nivelada pelos gerentes e desenvolvedores, em seu ciclo de vida aliada à agilidade de suas atividades para sintetizar uma engenharia adequada ao seu lema "resultados frequentes, tangíveis e funcionais" (GIUFFRA e VILAIN, 2010).

Um dos motivos da metodologia ser caracterizada como intermediária é a constante adaptação de seus processos visto que o FDD *"é adequado para projetos grandes que possuam equipes grandes, e iterações curtas, para que seja eficaz no desenvolvimento das funcionalidades relativas ao tipo do negócio"* (KANWAL *et al.*, 2010, p. 2). A documentação não é dispensada totalmente, por exemplo, com a concepção de modelagens através do uso de UML[10] para facilitar dividir as tarefas durante o desenvolvimento.

3.4.1 Disciplinas fundamentais

O FDD sugere que o desenvolvimento de software priorize aspectos importantes, como o planejamento, a gerência dos recursos e as atividades de programação. A metodologia propõe um conjunto de boas práticas que auxiliam os profissionais a conduzirem melhor seus processos como uma estratégia válida para alcançar bons resultados. Palmer e Felsing (2002) apresentam as práticas e suas respectivas características na Tabela 3.2:

[10] UML é a Linguagem de Modelagem Unificada para documentar artefatos com diagramas (LARMAN, 2001).

Tabela 3.2: Práticas fundamentais do FDD (PALMER e FELSING, 2002)

Prática	Característica
Modelagem dos Objetos de Domínio	O domínio deve ser representado através de diagramas UML com as dependências, generalizações e especializações.
Desenvolvimento por funcionalidades	O desenvolvimento se dá com a codificação das pequenas funcionalidades contidas nas versões do software.
Propriedade de classe (código) individual	Cada programador deve ser responsável por uma ou mais classes, provendo o desenvolvimento e os extensivos testes.
Formar equipe de recursos	Deve ser formada uma equipe que forneça os subsídios e recursos necessários para garantir a construção do software.
Inspeções	As inspeções asseguram a melhoria nos processos e verificam a qualidade do produto que foi concebido.
Intervalos regulares de compilação	As equipes mensuram o tempo de entrega do produto proporcionalmente a complexidade do projeto.
Gerência de configuração	As equipes devem utilizar o controle de versão nos códigos e modelagens para identificar as mudanças.
Reportar visibilidade dos resultados	Os programadores precisam sempre verificar o status do projeto para atualizar os demais membros das equipes.

3.4.2 Papéis

A divisão de trabalho no FDD é semelhante às metodologias ágeis citadas anteriormente. Para cada profissional, é estabelecido um papel fixo, com responsabilidades bem descritas, que facilitam estipular uma carga de trabalho adequada para cada tipo diferente de cenário. De acordo com Rychlý e Tichá (2007), a metodologia propõe papéis pouco convencionais, mas que são importantes quando estão distribuídos corretamente.

O ***gerente de projeto***, conhecido como líder da equipe, é o responsável por administrar todos os recursos funcionais e não funcionais que serão empregados. O propósito maior deste profissional é remover os problemas que impeçam o trabalho dos demais fluir, além de despertar a motivação, cobrando dedicação dos colegas no dia a dia (FIGUEIREDO, 2007).

O segundo papel, conforme Figueiredo (2007), é o ***arquiteto-chefe***, responsável por promover o desenvolvimento do design global do sistema. Ele trabalha em parceria com o

gerente de projetos, detendo conhecimento técnico em configuração, e outros aspectos para dar o suporte às equipes e aos seus líderes. Outro papel fundamental para auxiliar as equipes é o ***gerente de desenvolvimento***, que atua diariamente liderando uma ou mais equipes simultaneamente para acompanhar o rendimento dos programadores.

Abaixo na hierarquia do gerente de desenvolvimento encontram-se os programadores experientes, denominados de ***programadores-chefes***. Eles são responsáveis por orientarem seus colegas, além de servir como referência para a consulta de dúvidas dos programadores plenos, reconhecidos pela metodologia como ***proprietários de classes*** (FIGUEIREDO, 2007).

Além desses cincos papéis, o FDD incorpora outras funções recomendadas para serem inseridas em projetos maiores. Palmer e Felsing (2002) explicam que a metodologia atribui duas categorias complementares conhecidas como ***papéis de suporte*** e ***papéis adicionais***, que são utilizados nos casos em que os profissionais precisam obter um conhecimento mais avançado sobre a área de negócios em que o software irá atuar.

3.4.3 Processos

O ciclo de vida do FDD é o mais completo das três metodologias ágeis abordadas neste trabalho. A metodologia propõe que o desenvolvimento do software ocorra em duas fases que facilitem o *feedback* dos profissionais e a rápida mudanças dos planos para a síntese de novas funcionalidades. A primeira delas é a fase de “Concepção e Planejamento”, que ocorre em um prazo que varia de uma a duas semanas, e a segunda fase, “Construção”, que varia de duas a três semanas, conforme pode ser representado na Figura 3.3.

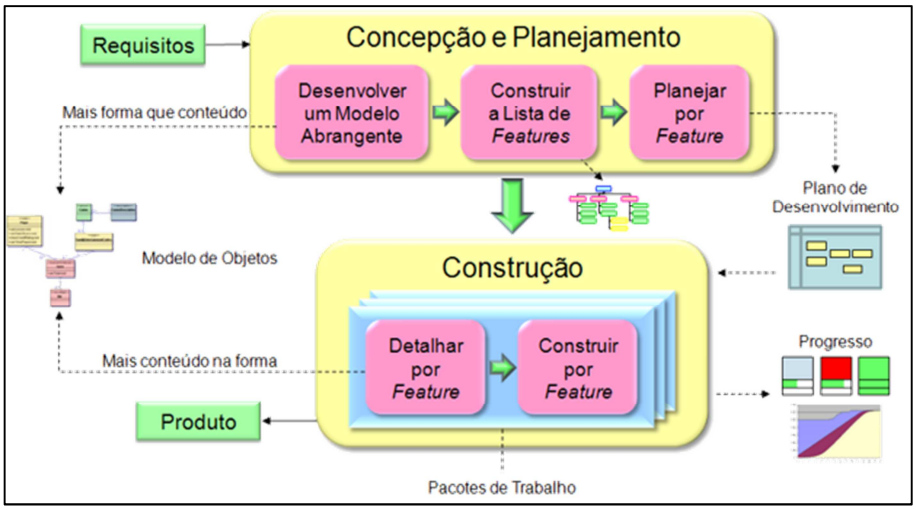

Figura 3.3: Processos do ciclo de vida do FDD (HOLL, 2010)

A primeira fase é iniciada com o ***Processo de Desenvolvimento do Modelo Abrangente*** **(DMA)**, onde é realizado um estudo sobre a área de negócios do cliente. Nesta etapa é criado o domínio do negócio[11], que é dividido em subáreas, para que os profissionais definam um design mais preciso do escopo do sistema. Como critérios de entrada, a equipe precisa estar formada e os requisitos definidos para atender as atividades (RETAMAL, 2005):

- **Formar a equipe de modelagem:** a atividade consiste em selecionar os especialistas no negócio que irão trabalhar em parceria com os programadores.
- **Fixar estudo prescritivo sobre o domínio de negócio:** a atividade consiste em planejar um cronograma de estudos referente à área de negócios do projeto.
- **Estudar documentação:** a atividade consiste em analisar os dados técnicos do domínio de negócios para facilitar a modelagem pelos programadores.
- **Desenvolver o modelo em equipes:** a atividade consiste em desenvolver os modelos de objetos das subáreas do domínio com métodos, atributos e funções de programação.
- **Desenvolver o modelo de objetos abrangente:** a atividade consiste em integrar os modelos de equipes para promover um modelo abrangente de design do sistema.

Como critérios de saída, devem ser gerados os diagramas de classes, os diagramas de sequência e os diagramas de estados. Ao término do processo, é realizada uma avaliação interna de sua qualidade com uma verificação efetuada pela equipe (DeLuca, 1998; 2005).

Concluída a criação do domínio de negócios, entra em cena o segundo processo do ciclo FDD, conhecido como ***Processo de Construção da Lista de Funcionalidades*** **(CLF)**. Como critérios de entrada são exigidos que ambos os modelos de objetos do processo DMA estejam finalizados. As atividades para concepção da lista, segundo Coad *et al.* (1999), são:

- **Formar a equipe da lista de funcionalidades:** a atividade consiste em compor a equipe que irá formar a lista de requisitos em parceria com a equipe do DMA.
- **Construir a lista de funcionalidades:** a atividade consiste em classificar os requisitos do cliente para que sejam implementados de acordo com as prioridades definidas.

Como critério de saída é exigido que pelo menos a lista parcial de requisitos esteja formada. Ao término do processo, é realizada uma verificação de sua qualidade pela equipe, ou caso seja necessário, uma avaliação externa por outras empresas (DeLuca,1998).

[11] Domínio de negócios significa reunir todas as características, termos e palavras sobre o nicho de atuação de uma determinada área de negócios (NUNES *et al.*, 2010).

Concluídos o modelo abrangente e a lista provisória de requisitos, o terceiro passo é executar e manter uma carga contínua de planejamento com o ***Processo de Planejar por Funcionalidade*** **(PPF)**. O critério de entrada exigido é que todos os requisitos de saída dos processos antecessores tenham sido atendidos. O planejamento é realizado paralelamente ao desenvolvimento, executando, segundo Ritchie (2009), as seguintes atividades:

- **Formar a equipe de planejamento:** a atividade consiste em selecionar os profissionais para executar o planejamento de construção das funcionalidades.
- **Determinar sequência no desenvolvimento:** a atividade consiste em definir um calendário com as datas de implementação das funcionalidades, considerando vários fatores como tempo, esforço do trabalho e complexidade.
- **Atribuir classes aos desenvolvedores:** a atividade consiste em dividir as tarefas de codificação para os programadores da equipe.
- **Atribuir atividades de negócios aos programadores-líderes:** a atividade consiste em distribuir as prioridades do negócio para serem quebradas em tarefas de projeto.

Como requisitos de saída do processo são exigidos que estejam datados os prazos das iterações, o tempo e os responsáveis pela avaliação do projeto. A verificação do processo deve ser realizada com o intuito de avaliar o índice de produtividade das equipes para detectar os impedimentos que estejam ocorrendo (BUGLIONE e ABRAN, 2007).

A fase de construção do ciclo FDD se inicia com o ***Processo de Detalhar por Funcionalidade*** **(DPF)**, que tem por responsabilidade promover o mapeamento das classes, métodos, pacotes, e todos os recursos funcionais necessários para a concepção da funcionalidade. Neste processo, o requisito de entrada exigido é o de que todos os processos de planejamento tenham sido finalizados. As atividades, segundo DeLuca (1998), são:

- **Formar equipe de funcionalidades:** a atividade consiste em formar uma equipe que descreva os requisitos exigidos para a construção da funcionalidade.
- **Promover o estudo prescritivo do domínio:** a atividade consiste em consultar as documentações do domínio para retirar eventuais dúvidas que possam surgir.
- **Promover o estudo da documentação de referência:** a atividade consiste em consultar os artefatos técnicos, como a lista de funcionalidades e os diagramas de modelagem, para retirar eventuais dúvidas durante a implementação.

- **Desenvolver diagramas de sequência:** a atividade consiste em desenvolver os diagramas que representam o fluxo de operações executado pelas funcionalidades.
- **Escrever prefácio de classes e métodos:** a atividade consiste em definir os tipos de parâmetros, retornos, exceções e as assinaturas dos métodos e classes.
- **Refinar o modelo de objetos:** a atividade consiste em avaliar e reportar os problemas encontrados nas classes, métodos e toda a estrutura de relacionamentos dos objetos.

O que é requerido como resultado para este processo é que os pacotes contendo os códigos-fontes estejam revisados, e o "escopo" das funcionalidades, terminado. Isso exige a verificação no projeto (design), como também uma avaliação das equipes para constatar a execução correta das atividades e o andamento do projeto (BUGLIONE e ABRAN, 2007).

O ciclo encerra-se com o ***Processo de Construir por Funcionalidade*** **(CPF)**, responsável por fornecer os subsídios para gerar as versões do software. O processo exige como entrada a inspeção nos modelos de objetos refinados. "*A equipe inspeciona o código-fonte, antes do teste de unidade, em seguida aplicando as verificações pelo proprietário nas classes para o gerenciamento das configurações*" (COAD *et al.*, 1999, p. 195). As atividades recomendadas para este processo, segundo DeLuca (1998), são:

- **Codificar classes e métodos:** a atividade consiste em codificar as funcionalidades utilizando linguagens de programação orientadas a objeto.
- **Realizar inspeção no código-fonte:** a atividade consiste em verificar a legibilidade nos códigos-fonte para identificar falhas que possam gerar problemas.
- **Testar as unidades de software:** a atividade consiste em verificar a semântica e a sintaxe do código-fonte para garantir que não haja empecilhos durante a integração das classes.
- **Checar e promover versões:** a atividade consiste em promover as versões parcial e final de um ciclo de funcionalidades (*build*) a cada nova funcionalidade integrada.

Ao término do processo, é realizada uma avaliação da eficiência das funcionalidades com a liberação de uma versão de testes para o cliente. A verificação é realizada pela equipe com a inspeção do software (código) e a validação em seguida, com reuniões com cliente, caso haja mudanças a serem implementadas que não tenham sido agendadas para que seja iniciada uma nova iteração (COAD *et al.*, 1999).

3.5 Resumo das metodologias ágeis

As metodologias ágeis descritas ao longo das seções estão direcionadas para suprirem necessidades específicas nos projetos de software. É possível perceber que existem diferenças simples, por exemplo, em seus objetivos, ciclos de vida, processos, fases, papéis e artefatos produzidos. Para facilitar a visualização dessas diferenças, é apresentado na Tabela 3.3 um breve comparativo que resume as principais características que compõem cada uma.

Tabela 3.3: Comparativo entre o *Scrum*, XP e FDD

Característica	***Scrum***	**XP**	**FDD**
Objetivo da Metodologia Ágil	Fornecer um *framework* para o desenvolvimento de software.	Fornecer um processo para o desenvolvimento de software.	Fornecer processos sequenciais para o desenvolvimento de software.
Ciclo de vida	Quatro fases definidas.	Cinco fases definidas.	Duas fases com cinco processos definidos.
Práticas, princípios e valores.	Práticas para o gerenciamento dos processos	Práticas para o desenvolvimento do software	Práticas para o gerenciamento dos processos/desenvolvimento do software.
Papéis	Três papéis fixos	Cinco papéis fixos	Seis papéis fixos, papéis de suporte e adicionais.
Artefatos	Possui artefatos fixos	Não possui artefatos fixos	Possui artefatos fixos

Observando a Tabela 3.3, percebe-se que o *Scrum* tem como foco o gerenciamento das fases do projeto, mas sem prescrever quais serão os processos de desenvolvimento do software. O XP, entretanto, possui uma característica inversa ao formar um processo ágil bem definido com o uso de práticas simples que tornam a produção mais objetiva. Por fim, o FDD disponibiliza atividades fixas em seus processos, que abordam ambos os fatores de gerência e engenharia, para garantir que os profissionais possam formar um ciclo de vida ágil mais completo (ANDERSON, 2004).

3.6 Considerações finais

A evolução das metodologias ágeis ao longo dos últimos anos tem contribuído expressivamente para a ascensão do mercado de software. Várias empresas têm adotado a estratégia de eliminar a burocracia existente em suas documentações, como uma alternativa de diminuir as dificuldades encontradas para fechar parcerias com outras empresas, além de tornar mais simples a criação de novos projetos (FERREIRA e LIMA, 2006).

Um fator que tem sido decisivo para que isso aconteça, segundo aponta um estudo da AgilCoop (2011), é o ganho de flexibilidade proporcionado pelas metodologias. Em cada uma são descritas recomendações objetivas, que orientam os profissionais a formarem um ciclo de vida ágil do software, através de práticas que facilitam a gerência de processos curtos e garantem que as equipes mantenham um ritmo sustentável durante a produção.

Outra característica diferenciada, conforme mostra a pesquisa, é a possibilidade de adaptação rápida dos ciclos de vida. No *Scrum*, são traçados os cronogramas das *Sprints*, mas não são sugeridas as atividades exercidas por cada papel, algo típico, por exemplo, no XP e do FDD, despertando uma perspectiva na comunidade ágil de que a combinação das práticas de diferentes metodologias vem se tornando uma tendência crescente a cada dia.

Uma pesquisa recente, publicada pela VersionOne (2010), comprova que essa tendência tem ganhado espaço, e mudado o cenário atual dos processos ágeis de software. Entre as cinco metodologias mais utilizadas em noventa e um países no ano de 2010, o primeiro lugar geral era ocupado por um “combinado híbrido entre *Scrum* e XP”, dando ênfase de que a busca pela agilidade acontece como um processo de renovação contínua.

Desta forma, pode-se concluir que as metodologias ágeis são ferramentas importantes, que reportam bons resultados sem exigir grandes investimentos, tempo e custos excessivos no projeto. É possível obter um processo eficiente de desenvolvimento quando as atividades corretas forem selecionadas para suprirem as necessidades de momento dos clientes, e um produto de boa qualidade, quando forem alocados profissionais competentes que dividam sem muitas dificuldades o esforço de trabalho (SCHWABER, 2004; 2007).

Capítulo

4

Mapeamento das metodologias ágeis em adesão aos processos de software da norma ISO/IEC e IEEE 12207

Este capítulo apresenta o alinhamento dos ciclos de vida das metodologias ágeis *Scrum*, XP, e FDD em aderência aos processos específicos de software da norma ISO/IEC e IEEE 12207. É realizado o mapeamento das atividades dos processos de implementação e suporte de software, descritos no capítulo dois, com as atividades das metodologias ágeis abordadas no capítulo três. Ao término de cada mapeamento, é apresentado um diagnóstico contendo as conclusões sobre a compatibilidade existente entre as atividades.

4.1 Contextualização do mapeamento

A pesquisa descrita neste trabalho tem como objetivo desenvolver os guias para a implantação de processos híbridos de software. Para que isso se torne possível de ser realizado, é necessário efetuar primeiramente um comparativo entre as fases e os processos das metodologias ágeis, para identificar quais são as melhores práticas a serem utilizadas pelos profissionais em cada uma das etapas do projeto.

O propósito do mapeamento foi alinhar os ciclos de vida das metodologias ágeis para verificar a aderência aos requisitos exigidos pela norma para a criação de novos processos. Através de uma análise realizada em cada um dos processos de implementação e suporte de software, é possível verificar o que *Scrum*, o XP e o FDD recomendam como atividades válidas de planejamento, análise, codificação, além de outras importantes que possam ser adotadas e, em seguida, adaptadas para a formação de um ciclo de vida híbrido de software.

Os processos da norma apesentados na revisão literária, e consequentemente neste mapeamento, foram selecionados seguindo-se como critério de escolha a compatibilidade de suas atividades com pelo menos uma atividade de alguma das três metodologias ágeis citadas na pesquisa. Foi realizado um estudo prévio acerca de quais processos se identificariam com as características ágeis para tornar possível alinhá-los sem que existissem maiores conflitos.

Dentre as duas categorias de processos específicos de software citadas, destacaram-se os cinco processos descritos na primeira categoria, implementação de software, e os cinco processos descritos na segunda categoria, suporte de software. Para facilitar a organização do alinhamento, e garantir que o comparativo estivesse objetivo, foi desenvolvida uma metodologia própria de mapeamento, como poderá ser observado na próxima seção.

4.2 Metodologia do mapeamento

A metodologia adotada para efetuar o mapeamento consiste em identificar cada um dos processos, obedecendo-se uma decomposição adaptada do modelo de constructo da norma, mencionada na seção 2.1 do capítulo dois para identificar os *processos ou fases equivalentes* dos paradigmas e dispô-los conjuntamente. É sintetizado um esboço situando as *atividades* ou *práticas equivalentes*, que emparelhadas, podem formar um conjunto de *referências híbridas geradas* sobre “o que fazer” de acordo com um cenário específico.

As tarefas e notas que compõem a hierarquia do constructo foram inviabilizadas de serem mapeadas devido às mesmas reportarem o “como fazer”, simbolizando a síntese de procedimentos técnicos variáveis. Cada procedimento sempre é modificado mediante as regras adotadas pelas empresas, de acordo com a visão, missão e o modelo de negócios, na formação de ciclos de vida, não tendo como contabilizá-las sem que descaracterize a formatação híbrida que fornece flexibilidade na composição de processos diferentes.

A representação estipulada para formular o esboço foi realizada através de um comparativo com o uso de tabelas que alinham a norma ISO/IEC e IEEE 12207, e logo abaixo as metodologias ágeis *Scrum*, XP e FDD. As tabelas estão divididas em três colunas contendo em cada uma delas um título referente ao processo híbrido e uma marcação abreviada em um formato *metodologia_processo* com o intuito de rastrear as atividades mapeadas no capítulo quatro aos guias propostos do capítulo cinco. Em cada processo é descrito um *diagnóstico* que explica as compatibilidades e incompatibilidades ao longo de cada etapa do ciclo de vida.

4.3 Mapeamento do processo híbrido de implementação de software

A primeira categoria dos processos específicos de software a ser mapeada é a de processos de implementação de software. O primeiro processo, de mesmo nome da referida categoria, é o *processo de implementação de software*, que pode obter um novo formato através do comparativo apresentado na Tabela 4.1.

Tabela 4.1: Mapeamento do processo híbrido de implementação de software

Processo híbrido de implementação de software		
Framework/ Metodologia	**Processo(s)/Fase Equivalente**	**Atividades/Práticas Equivalentes** **Referências híbridas de processo geradas**
ISO/IEC IEEE 12207	Implementação de software	ISO_IMP: Definir estratégias e métodos. ISO_IMP: Definir itens de configuração na implementação. ISO_IMP: Definir prioridades. ISO_IMP: Documentar não conformidades.
Scrum	Planejamento	SCR_IMP: Estabelecer a visão do negócio. SCR_IMP: Especificar recursos de projeto. SCR_IMP: Definir plano de projeto e *releases*.
XP	Planejamento	XP_IMP: Realizar jogo de planejamento. XP_IMP: Definir possíveis riscos. XP_IMP: Planificar estimativas de tempo.
FDD	DMA	FDD_IMP: Formar a equipe de modelagem. FDD_IMP: Fixar estudo sobre a área de negócios. FDD_IMP: Desenvolver o modelo em equipes. FDD_IMP: Desenvolver o modelo abrangente.

Diagnóstico: a norma é parcialmente compatível com o *Scrum*, XP, e FDD. No processo é requerido que sejam definidas as estratégias, métodos de desenvolvimento, tecnologias, e os itens de configuração através de prioridades de projeto. O *Scrum* atende estes requisitos, quando na fase de *Planejamento*, são traçados a visão do negócio, as prioridades do cliente, e o planejamento das etapas do projeto. O XP atende parcialmente os requisitos ao definir as estratégias de projeto na fase de *Planejamento,* recomendando que sejam traçados os riscos do projeto e que o desenvolvimento do software também seja guiado por prioridades. O FDD atende apenas parte dos requisitos, quando na fase do *DMA*, estabelece que o desenvolvimento seja guiado pelas prioridades do negócio, mas sem definir os itens de configuração, etapas do projeto e documentar as não conformidades encontradas.

4.4 Mapeamento do processo híbrido de análise de requisitos de software

O segundo processo de implementação de software mapeado é o *processo de análise de requisitos de software*, responsável por subsidiar as atividades de coleta dos requisitos. Sua representação híbrida pode ser disposta através do comparativo apresentado na Tabela 4.2.

Tabela 4.2: Mapeamento do processo híbrido de análise de requisitos de software

Processo híbrido de análise de requisitos de software		
Framework/ Metodologia	**Processo(s)/Fase Equivalente**	**Atividades/Práticas Equivalentes** **Referências híbridas de processo geradas**
ISO/IEC IEEE 12207	Análise de requisitos	ISO_REQ: Especificar requisitos funcionais. ISO_REQ: Especificar requisitos não funcionais. ISO_REQ: Firmar qualificação de requisitos. ISO_REQ: Especificar a ordenação da informação.
Scrum	*Stagging*	SCR_REQ: Definir requisitos (*Product Backlog*). SCR_REQ: Planejar *Sprints* (*Sprint Meeting*). SCR_REQ: Planejar codificação (*Sprint Planning*).
XP	Análise	XP_REQ: Definir estórias do cliente (*user stories*). XP_REQ: Desenvolver modelagem ágil do requisito.
FDD	CLF	FDD_REQ: Formar a equipe de análise de requisitos. FDD_REQ: Construir a lista de funcionalidades.
	PPF	FDD_REQ: Formar a equipe de planejamento. FDD_REQ: Fixar sequencia de desenvolvimento. FDD_REQ: Atribuir atividades aos líderes. FDD_REQ: Formar equipe de funcionalidades.
	DPF	FDD_REQ: Desenvolver diagramas de sequencia. FDD_REQ: Escrever prefácio das classes e métodos.

Diagnóstico: A norma é parcialmente compatível com o *Scrum*, XP e FDD. No processo é requerido que sejam listados os requisitos funcionais e não funcionais, e que os mesmos estejam ordenados e bem descritos. O *Scrum* adere parcialmente ao processo quando define os requisitos dos clientes, e os subdivide em atividades e tarefas menores na fase de *Stagging*. O XP também é compatível em parte, quando define que os requisitos tenham uma descrição objetiva (*estórias*) na fase de *Análise*. O FDD é parcialmente compatível ao ordenar e qualificar os requisitos funcionais através das atividades dos processos *CLF*, *PPF* e *DPF*.

4.5 Mapeamento do processo híbrido de construção de software

O terceiro processo de implementação de software a ser mapeado é o *processo de construção de software*, responsável por disponibilizar as atividades de codificação do produto. Sua representação híbrida pode ser disposta através de um comparativo apresentado na Tabela 4.3.

Tabela 4.3: Mapeamento do processo híbrido de construção de software

Processo híbrido de construção de software		
Framework/ Metodologia	**Processo(s)/Fase Equivalente**	**Atividades/Práticas Equivalentes** **Referências híbridas de processo geradas**
ISO/IEC IEEE 12207	Construção de software	ISO_COD: Codificar módulos e procedimentos. ISO_COD: Testar as unidades de códigos-fontes. ISO_COD: Avaliar aceitação códigos/requisitos. ISO_COD: Documentar códigos-fontes.
Scrum	Desenvolvimento	SCR_COD: Codificar componentes do software. SCR_COD: Reportar impedimentos da codificação.
XP	Desenvolvimento	XP_COD: Escrever os testes para os códigos-fontes. XP_COD: Padronizar o código-fonte. XP_COD: Compartilhar o código-fonte. XP_COD: Codificar em pares.
FDD	CPF	FDD_COD: Codificar classes e métodos.

Diagnóstico: a norma é parcialmente compatível com o *Scrum*, XP e FDD. O processo recomenda que sejam codificados os procedimentos, funções, métodos e demais estruturas lógicas do software. Também é sugerida a inserção de testes durante a codificação, a documentação do código-fonte e a verificação contínua de atendimento aos requisitos do cliente. O *Scrum* atende a estes requisitos, quando na fase de *Desenvolvimento*, as equipes estabelecem as atividades de gerenciamento da codificação executadas ao longo da *Sprint*. O XP também adere aos requisitos, ao definir que a codificação do software seja guiada simultaneamente por uma intensa carga de testes realizada na fase de *Desenvolvimento*. O FDD adere em parte ao processo, quando no processo *CPF*, determina que as equipes codifiquem as classes, métodos, atributos e configurem todas as propriedades existentes dos modelos de objetos.

4.6 Mapeamento do processo híbrido de integração de software

O quarto processo de implementação de software é o *processo de integração de software*, responsável por recomendar as atividades de integração das unidades funcionais do produto. A representação híbrida pode ser disposta através do comparativo apresentado na Tabela 4.4

Tabela 4.4: Mapeamento do processo híbrido de integração de software

Processo híbrido de integração de software		
Framework/ Metodologia	**Processo(s)/Fase Equivalente**	**Atividades/Práticas Equivalentes** **Referências híbridas de processo geradas**
ISO/IEC IEEE 12207	Integração de software	ISO_INT: Definir um plano de integração. ISO_INT: Definir casos de testes agregados. ISO_INT: Avaliar e reportar diferenças. ISO_INT: Propor mudanças.
Scrum		
XP	Desenvolvimento	XP_INT: Integrar continuamente o código-fonte. XP_INT: Promover pequenas versões integradas.
FDD	CPF	FDD_INT: Checar e promover versões (*build*).

Diagnóstico: a norma é parcialmente compatível com o XP e FDD, e incompatível com o *Scrum*. O processo recomenda que seja definido um plano de integração, os testes de integração entre as unidades do software, e uma avaliação do processo para verificar a eficiência dos seus resultados. O *Scrum* é incompatível com os requisitos, pelo fato de não especificar as atividades que devem ser adotadas pelas equipes para realizar a integração do software. O XP, entretanto, adere parcialmente ao processo, quando adota a prática de integrar continuamente o código-fonte como uma estratégia válida para promover pequenas versões do software ao longo da fase de *Desenvolvimento*. Já o FDD tem uma compatibilidade parcial aos requisitos da norma, quando no processo *CPF*, determina uma atividade fixa que orienta os profissionais a checarem e promoverem (*build*) versões integradas do software.

4.7 Mapeamento do processo híbrido de qualificação e testes de software

O quinto processo de implementação de software a ser mapeado é o *processo de qualificação e testes de software*, responsável por recomendar as atividades de testes e qualificação das unidades funcionais do software. Sua representação híbrida está representada na Tabela 4.5.

Tabela 4.5: Mapeamento do processo híbrido de qualificação e testes de software

Processo híbrido de qualificação e testes de software		
Framework/ Metodologia	**Processo(s)/Fase Equivalente**	**Atividades/Práticas Equivalentes** **Referências híbridas de processo geradas**
ISO/IEC IEEE 12207	Qualificação e testes de software	ISO_TST: Especificar testes unitários. ISO_TST: Avaliar o processo, códigos e testes. ISO_TST: Elaborar relatórios dos testes. ISO_TST: Propor inspeções.
Scrum		
XP	Testes	XP_TST: Desenvolver métodos de testes. XP_TST: Realizar testes unitários. XP_TST: Realizar testes dirigidos. XP_TST: Realizar testes de aceitação.
FDD	CPF	FDD_TST: Testes de unidade.

Diagnóstico: a norma é parcialmente compatível com o XP e FDD e incompatível com o *Scrum*. No processo é requerido que sejam especificados os testes unitários de qualificação do software, a inspeção nos códigos-fontes, os relatórios com os resultados da aplicação dos testes, e uma avaliação para verificar a eficiência dos seus resultados. O *Scrum* é incompatível aos requisitos, pelo fato de não traçar as atividades que orientam os profissionais a conduzirem a qualificação e os testes do software. O XP, entretanto, adere parcialmente ao processo, sugerindo na fase de *Testes*, que a qualificação ocorra basicamente com a aplicação dos testes unitários, dirigidos e de aceitação. Já o FDD, é aderente em parte aos requisitos, quando recomenda que a qualificação aconteça no processo *CPF* com a realização dos testes de unidade do software.

4.8 Mapeamento do processo híbrido de gerência da documentação de software

A segunda categoria dos processos específicos de software a ser mapeada é a de *processos de suporte de software*. O primeiro processo é o *processo de gerência da documentação de software*, responsável por recomendar as atividades utilizadas para gerenciar os artefatos gerados no projeto. Sua representação híbrida pode ser apresentada na Tabela 4.6.

Tabela 4.6: Mapeamento do processo híbrido de gerência da documentação de software

Processo híbrido de gerência da documentação de software		
Framework/ Metodologia	**Processo(s)/Fase Equivalente**	**Atividades/Práticas Equivalentes** **Referências híbridas de processo geradas**
ISO/IEC IEEE 12207	Gerenciamento da Documentação	ISO_DOC: Especificar os documentos produzidos. ISO_DOC: Especificar conteúdo e formato. ISO_DOC: Especificar as fontes de produção. ISO_DOC: Editar, revisar e alterar documentos.
Scrum	Desenvolvimento	SCR_DOC: Gerir *Product Backlog* e *Sprint Backlog*. SCR_DOC: Gerenciar o *Burndown Chart*.
XP	Desenvolvimento	XP_DOC: Homologar novos artefatos produzidos. XP_DOC: Desenvolver manuais de usuário/software.
FDD	DPF	FDD_DOC Fixar estudo dirigido do domínio. FDD_DOC: Fixar estudo da documentação gerada. FDD_DOC: Refinar o modelo de objetos.

Diagnóstico: a norma é parcialmente compatível com as metodologias *Scrum*, XP e FDD. No processo é requerido que sejam especificados os documentos que serão produzidos no projeto, o formato e o conteúdo de produção, as fontes de produção, e que sejam definidos os protocolos de alteração dos documentos. O *Scrum* atende parcialmente aos requisitos, a partir do momento que orienta os profissionais a desenvolverem e manterem seus artefatos *Product Backlog*, *Sprint Backlog*, e *Burndown Chart* durante a fase de *Desenvolvimento*. O XP também é parcialmente compatível, quando estipula que o redator técnico desenvolva os manuais de usuário e do software, e homologue os novos artefatos durante a etapa de *Desenvolvimento*. O FDD adere parcialmente a norma, quando no processo *DPF*, orienta aos profissionais a estudarem e melhorarem continuamente a documentação de referência dos seus modelos de objetos.

4.9 Mapeamento do processo híbrido de gerência da configuração de software

O segundo processo da categoria de processos de suporte de software a ser mapeado é o *processo de gerência da configuração de software*, responsável por subsidiar as atividades de configuração e manutenção dos artefatos de projeto gerados. Sua representação híbrida pode ser apresentada na Tabela 4.7.

Tabela 4.7: Mapeamento do processo híbrido de gerência da configuração de software

Processo híbrido de gerência da configuração de software		
Framework/ Metodologia	**Processo(s)/Fase Equivalente**	**Atividades/Práticas Equivalentes** **Referências híbridas de processo geradas**
ISO/IEC IEEE 12207	Gerenciamento da Configuração	ISO_CNF: Definir um plano de gerenciamento. ISO_CNF: Identificar itens de software. ISO_CNF: Gerenciar status de configuração. ISO_CNF: Avaliar e reportar mudanças.
Scrum		
XP	Desenvolvimento	XP_CNF: Realizar integração contínua.
FDD	CPF	FDD_CNF: Fixar controle de versões no software. FDD_CNF: Fixar controle de versões nos artefatos.

Diagnóstico: a norma é parcialmente compatível com o XP e FDD, e incompatível com o *Scrum*. No processo é requerido que seja definido um plano de gerenciamento da configuração, a identificação dos itens de software gerados, o protocolo de gerenciamento das mudanças, e a avaliação do processo para verificar a eficiência dos seus resultados. O *Scrum* é incompatível aos requisitos descritos do processo, pelo fato de não especificar nenhuma atividade que recomende explicitamente o gerenciamento da configuração do software. O XP tem uma aderência parcial, ao sugerir que os desenvolvedores realizem a integração contínua das unidades funcionais do software durante a fase de *Desenvolvimento*. Já o FDD, tem uma compatibilidade parcial com o processo, ao estabelecer como práticas do processo *CPF*, o controle de versões das modelagens e das unidades funcionais do software.

4.10 Mapeamento do processo híbrido de garantia da qualidade de software

O terceiro processo da segunda categoria mapeada é o *processo de garantia da qualidade de software*, característico por recomendar as atividades que atribuam um nível de excelência aos processos e ao produto de software. A representação híbrida pode ser disposta através do comparativo apresentado na Tabela 4.8.

Tabela 4.8: Mapeamento do processo híbrido de garantia da qualidade de software

Processo híbrido de garantia da qualidade de software		
Framework/ Metodologia	**Processo(s)/Fase Equivalente**	**Atividades/Práticas Equivalentes** **Referências híbridas de processo geradas**
ISO/IEC IEEE 12207	Garantia da qualidade de software	ISO_QLD: Definir um plano de qualidade. ISO_QLD: Definir procedimentos de processo. ISO_QLD: Definir procedimentos de produto. ISO_QLD: Documentar índice da qualidade.
Scrum		
XP		
FDD	DMA CLF PPF	FDD_QLD: Autoavaliação interna dos processos. FDD_QLD: Autoavaliação externa dos processos. FDD_QLD: Autoavaliação das equipes. FDD_QLD: Inspecionar o código.

Diagnóstico: a norma é parcialmente compatível com o FDD e incompatível com o *Scrum* e o XP. O processo sugere que seja criado um plano de gerenciamento da qualidade e que sejam traçados os procedimentos que agregam melhorias tanto no processo, quanto no produto. Também é requerido que os profissionais desenvolvam os métodos adequados de avaliação e a documentação para registrar os índices de qualidade obtidos. O *Scrum* é incompatível com os requisitos, pois não especifica em suas fases atividades direcionadas para a garantia da qualidade do software. O XP é incompatível com a norma, pois não recomenda práticas no seu ciclo de vida que também estejam voltadas para manter a garantia da qualidade do software. Já o FDD, adere parcialmente ao processo, ao exigir como requisito de saída dos processos *DMA* e *CLF*, uma avaliação interna ou externa de sua qualidade, e no processo PPF, uma avaliação da produtividade desempenhada pelas equipes.

4.11 Mapeamento do processo híbrido de verificação de software

O quarto processo da segunda categoria mapeada é o *processo de verificação de software*, característico por orientar os profissionais a realizarem a inspeção das funcionalidades após o os testes do software. O mapeamento híbrido deste processo pode ser apresentado através da Tabela 4.9.

Tabela 4.9: Mapeamento do processo híbrido de verificação de software

Processo híbrido de verificação de software		
Framework/ Metodologia	**Processo(s)/Fase Equivalente**	**Atividades/Práticas Equivalentes** **Referências híbridas de processo**
ISO/IEC IEEE 12207	Verificação de software	ISO_VER: Definir plano de verificação. ISO_VER: Verificar o código do software. ISO_VER: Verificar o design do software. ISO_VER: Verificar documentação.
Scrum	*Releasing*	SCR_VER: Verificar o produto (*Sprint Review*). SCR_VER: Verificar/Analisar status do projeto.
XP	Desenvolvimento	XP_VER: Realizar verificações em pares (refatorar). XP_VER: Demonstrar novas funcionalidades.
FDD	DPF	FDD_VER: Inspecionar o projeto (design).

Diagnóstico: a norma é parcialmente compatível com o *Scrum*, o XP e o FDD. O processo sugere que seja definido um plano de verificação e que os procedimentos de análise do código-fonte, do design, e da documentação do software estejam especificados. O *Scrum* adere parcialmente aos requisitos, a partir do momento em que inicia a fase de *Releasing*, recomendando que os desenvolvedores se reúnam com o *Product Owner* para demonstrarem as novas funcionalidades e verificarem o status em que se encontra atualmente o projeto. O XP também possui a mesma compatibilidade, quando adota na fase de *Desenvolvimento*, a prática da inspeção em pares do código-fonte, e a verificação das funcionalidades com a participação ativa do cliente. O FDD, entretanto, não torna explícita nenhuma atividade de verificação do software, mas adere parcialmente a norma, ao requisitar como saída do processo *DPF* a inspeção de andamento do projeto e a avaliação do design do software.

4.12 Mapeamento do processo híbrido de validação de software

O quinto, e último processo da categoria de suporte a ser mapeado nesta pesquisa, é o *processo de validação de software*, responsável por consolidar as mudanças implementadas após a verificação das funcionalidades. O mapeamento híbrido do processo pode ser representado através da Tabela 4.10.

Tabela 4.10: Mapeamento do processo híbrido de validação de software

Processo híbrido de validação de software		
Framework/ Metodologia	**Processo(s)/Fase Equivalente**	**Atividades/Práticas Equivalentes** **Referências híbridas de processo**
ISO/IEC IEEE 12207	Validação de software	ISO_VAL: Definir plano de validação. ISO_VAL: Elaborar cenários de simulação. ISO_VAL: Elaborar casos de testes do usuário. ISO_VAL: Avaliar e reportar mudanças.
Scrum	*Releasing*	SCR_VAL: Validar ações (*Sprint Retrospective*) SCR_VAL: Definir novas ações para as *Sprints*.
XP	Implementação	XP_VAL: Distribuir versões de testes para o cliente.
FDD	CPF	FDD_VAL: Realizar reuniões com o cliente. FDD_VAL: Testes de integração com o cliente.

Diagnóstico: a norma é parcialmente compatível com o *Scrum*, o XP e o FDD. O processo sugere que seja elaborado um plano de validação, os cenários para a simulação das funcionalidades, os testes de usabilidade, e os procedimentos que reportem as mudanças corretivas no software. O *Scrum* adere parcialmente aos requisitos, quando propõe a segunda reunião na fase de *Releasing*, instruindo que os desenvolvedores e o *Scrum Master* viabilizem as mudanças exigidas pelo cliente para serem aplicadas nas próximas *Sprints*. O XP também possui uma aderência parcial a norma, quando recomenda na fase de *Implementação*, que sejam disponibilizadas pequenas versões de demonstração para que cliente validade gradativamente as funcionalidades do software. Por fim, o FDD não especifica atividades de validação no processo *CPF*, mas determina como requisitos de saída que os desenvolvedores se reúnam para acertar os detalhes dos testes de integração do cliente com o software.

4.13 Considerações finais

O alinhamento realizado ao longo das seções possui uma contribuição fundamental para a concepção dos processos híbridos de software. Através dos diagnósticos obtidos com o comparativo dos ciclos de vida das metodologias ágeis *Scrum*, XP, e FDD, é possível identificar quais são as atividades que podem ser adaptadas ao longo das etapas que compõem um projeto de software.

O *Scrum* é a metodologia ágil que possui a menor aderência aos requisitos exigidos pela norma. O *framework* é parcialmente compatível com seis, dos dez processos mapeados, estabelecendo em suas fases que o gerenciamento do projeto seja uma responsabilidade adquirida por todos os integrantes da equipe. Apesar de a estratégia ser válida, a metodologia se torna limitada, quando recomenda as práticas para o gerenciamento da análise e da construção, mas sem reportar nenhuma para os testes, integração e configuração do software.

No XP, a aderência é parcialmente compatível com nove, dos dez processos de mapeados. A metodologia recomenda que os profissionais adotem práticas simples de análise, programação, testes, e validação, buscando interagir sempre com o cliente para alcançar resultados mais precisos e um software com índices aceitáveis de qualidade. Entretanto, ela não especifica como deve ser promovido o suporte nas reuniões ao longo do projeto, tornando o andamento do processo algo extremamente dependente da participação ativa do cliente.

A metodologia ágil FDD atende parcialmente os requisitos de todos os processos de implementação e suporte mapeados. Diferente das demais, ela é a mais abrangente quando recomenda práticas, atividades, e requisitos, que abordam os aspectos que o *Scrum* e o XP consideram rígidos, como a avaliação da qualidade dos processos e do produto. Mesmo que esta alternativa seja importante para atribuir melhorias no ciclo, a metodologia não determina como devem ser conduzidas as verificações realizadas pelas equipes, e nem delimita os fatores-chave da qualidade que precisam ser avaliados para garantir as mudanças corretivas.

Portanto, o que se pode concluir é que as propriedades oferecidas por cada metodologia se complementam para formar um ciclo de vida ágil mais eficiente. Isso requer que os profissionais desenvolvam inicialmente, um comparativo entre as principais práticas de projeto que serão adotadas, para garantir que a aplicação dos processos combinados proporcione resultados que obtenham pleno êxito.

Capítulo

5

Guidelines para a implantação de um ciclo de vida com processos híbridos de software

Este capítulo apresenta os guias para a implantação dos processos híbridos de software. Ao longo das seções são formatados dez modelos estruturados que tem como objetivo reunir as práticas ágeis mapeadas e propor as sugestões de como elas podem ser combinadas. Também são descritas as recomendações, benefícios e limitações obtidos como resultados de aplicação dos guias.

5.1 Contextualização dos guias

Com a realização do mapeamento no capítulo quatro, pôde-se perceber que a fusão dos ciclos de vida das metodologias ágeis é possível de ser veiculada. Abordou-se na seção 4.1 do referido capítulo que o alinhamento serve como uma base para reformular a organização funcional das fases, processos e atividades, no sentido de promover um ciclo de vida mais eficiente que diminua os problemas encontrados para o desenvolvimento de novas soluções.

A proposta desenvolvida neste trabalho foi a de conceber dez guias instrucionais que reunissem as práticas mapeadas no capítulo quatro para orientar os profissionais a criarem seus próprios processos. Os guias são formados por uma série de recomendações que tem como objetivo apresentar novas ideias, exemplos e sugestões de como as práticas podem ser combinadas durante as etapas realizadas em um projeto de software.

Cada guia é composto por uma metodologia de implantação híbrida, que organiza a distribuição das práticas através de um passo-passo para facilitar a adaptação. Também é

proposto um escopo que padroniza o formato, a ordem e a enumeração direcionando os profissionais a selecionarem as melhores práticas para que os mesmos possam criar seus próprios processos híbridos de software.

É importante ressaltar que os guias não assumem o formato de processos de software, nem muito menos fixam uma sequencia obrigatória de execução para o ciclo de vida. Em cada modelo são especificados apenas os requisitos de como as práticas devem ser alocadas, deixando que a hierarquia e os relacionamentos existentes entre os processos seja uma responsabilidade a ser planejada pelos profissionais.

5.1.1 Metodologia de implantação híbrida proposta

A metodologia organiza a adaptação através de quatro passos. As recomendações são apresentadas seguindo-se o modelo de guias publicado por Sommerville (1997), utilizando expressões como "*deve*" ou "*pode*", no sentido de simplificar a explicação do uso das práticas e tornar a leitura objetiva. Ao término da descrição de cada prática é adicionada a marcação mencionada no capítulo quatro para identificar de qual metodologia a mesma é proveniente.

O **primeiro passo** a ser seguido para formular a combinação das atividades, é **definir os requisitos de entrada do processo**. Nesta primeira etapa os profissionais precisam descrever quais serão as pré-condições iniciais para que o processo seja adaptado, identificando as demandas que devem ser traçadas pelas empresas para que as primeiras decisões relativas às mudanças possam ser tomadas.

Após definidos tais requisitos, o **segundo passo** é **realizar a implementação híbrida do processo**. A segunda etapa consiste em propor a combinação dos novos constructos dos processos, seguindo-se o modelo mencionado na seção 2.1.2 do capítulo dois, alinhando as práticas sugeridas pelos guias de acordo com a hierarquia do ciclo de vida definida pela empresa. Cada constructo deve conter ainda os procedimentos técnicos (tarefas) que devem ser escolhidos pelos profissionais mediante as necessidades encontradas em cada projeto.

O **terceiro passo** proposto na metodologia é **avaliar os benefícios, impactos e problemas** provenientes da criação ou modificação do processo original para um novo formato. Para os benefícios devem ser justificadas as mudanças, apontando as perspectivas de melhoria almejadas, além, claro, dos impactos gerados para os fatores importantes, tais como tempo, custos, esforço, além das consequências geradas. Os problemas previstos também

devem ser avaliados, para diminuir as dificuldades ao longo da implantação, mesmo que a adaptação demore certo tempo para ser devidamente efetuada.

O **quarto passo** consiste em **definir os requisitos de saída do processo**. A última é etapa tem o objetivo semelhante à primeira, entretanto, requerendo que sejam estabelecidos os resultados que devem ser alcançados com a aplicação do processo adaptado. Neste último passo são mediadas as chamadas pós-condições, em que se tornam comuns a geração de artefatos que sirvam como referência de entrada para o próximo processo do ciclo de vida.

5.1.2 Escopo dos guias

O escopo proposto torna-se semelhante ao formato do mapeamento do quarto capítulo, alterando-se apenas a sua estrutura para adequar corretamente a metodologia híbrida de implantação. Para facilitar a organização das informações, adotou-se como base de referência o modelo de guias de Sommerville (1997), adaptando-se apenas a quantidade de passos que devem ser executados.

Cada guia contém um *cabeçalho*, onde deverá ser exposta a seção subsequente a uma ordem no ciclo de vida (seções do capítulo), adjunto ao título do guia do processo híbrido mencionado. Nestes cabeçalhos são inclusos o *objetivo-chave do processo* e o *tipo do guia de implantação*, conforme pode ser observado no exemplo a seguir.

Guia do Processo Híbrido de [nome do processo]

Objetivo-chave do processo: [informar objetivo-base do processo]

Tipo de guia: [informar o tipo guia/processo de implantação]

- **Definir as entradas do processo:**

- **Realizar a implementação híbrida do processo:**

- **Avaliar os benefícios, impactos, e problemas da implantação:**

- **Definir as saídas do processo:**

5.2 Guia do processo híbrido de implementação de software

Guia do Processo Híbrido de Implementação de Software

Objetivo-chave do processo: Realizar o macro/micro planejamento de projeto

Tipo de guia: Implementação de software

Definir as entradas do processo:

1. As empresas devem firmar como primeiro critério de entrada a identificação do cliente para que possa ser possível traçar um perfil do projeto pela qual será desenvolvido. Esta informação é de fundamental importância, principalmente para que sejam compatibilizadas as perspectivas de atuação na área de negócios (FDD_IMP).
2. O segundo critério de entrada é a identificação de um problema que precisa ser resolvido pelo referido cliente, com o objetivo de avaliar a complexidade do projeto e verificar a viabilidade de realização do mesmo (SCR_IMP).
3. O terceiro critério de entrada recomendado é a descrição do cenário estabelecido pelo cliente, identificando suas condições atuais de trabalho e as possibilidades de mudança para realizar as adaptações (SCR_IMP; XP_IMP).

Realizar a implementação híbrida do processo:

4. Deve-se sintetizar um esboço com a descrição do problema mencionado pelo cliente, de maneira que seja possível representar didaticamente sua contextualização, a motivação para resolvê-lo e possíveis dependências que possam existir (FDD_IMP).
5. Pode-se, neste caso, produzir insumos como planos de projeto ou domínios de negócios com objetos para representar didaticamente as propostas de soluções mais viáveis identificadas pelos profissionais em parceria com os clientes (FDD_IMP).
6. Deve-se delimitar uma classificação com valores, de 1 a 5, por exemplo, para atribuir as prioridades de projeto, visando solucionar os problemas elencados através de uma análise das perspectivas de negócio, esforço de trabalho, tempo e riscos (SCR_IMP).
7. Deve-se obrigatoriamente estipular quantos e quais são os processos, fases e etapas que melhor se adequam a estrutura do cenário do projeto que será realizado. É

inevitável associar uma quantidade mínima de atividades, descrevê-las as quais processos as mesmas pertencem, e como deverão ser realizadas (SCR_IMP).

8. Devem ser definidos os critérios para selecionar os profissionais competentes para conduzir este, e os demais processos do ciclo de vida do software. São sugeridos como fatores de escolha: o tempo de experiência, nível de conhecimento da área, participação em projetos similares e especialidade de atuação (XP_IMP; FDD_IMP).

Avaliar os benefícios, impactos e problemas da implantação:

- O benefício de tornar o processo com um formato híbrido é o de poder adaptá-lo de acordo com a área específica de negócios. Caso as empresas precisem abordar um foco mais específico, basta formularem um escopo que insira como complemento das atividades convencionais de planejamento a criação de um domínio de negócios.
- A facilidade de simplificar o processo é outra vantagem que pode ser destacada. O mesmo escopo pode utilizado em projetos de diferentes complexidades, neste caso, sendo de grande, médio ou pequeno porte.
- A adaptação gera impactos sobre como o problema deve ser analisado. Em projetos que exigem um maior nível de detalhamento, as atividades precisam ser mais rígidas para realizar o levantamento mais preciso dos dados, diferente, por exemplo, de cenários mais simples, que demandam apenas uma avaliação em pequena escala.
- O problema de adaptar um processo de planejamento é o de alterar a hierarquia original fixada pela empresa. As mudanças podem acarretar instabilidades ou podem não ser suficientes para atender as políticas exigidas pelos clientes no mercado.

Definir as saídas do processo:

9. As empresas devem gerar seus planos de negócios ou planos de projeto contendo um levantamento dos dados que apontem as soluções encontradas. É importante que sejam listados o cronograma, quantidade de fases, ferramentas e demais aspectos abordados pelos profissionais para que o projeto possa ser realizado (SCR_IMP; XP_IMP).
10. Recomenda-se também que sejam gerados os artefatos que reportem uma avaliação dos recursos não funcionais que serão consumidos no projeto. Enquadram-se nesta análise um levantamento dos custos que serão exigidos ao longo das fases, os recursos materiais que serão necessários e o capital humano contratado (SCR_IMP; XP_IMP).

5.3 Guia do processo híbrido de análise de requisitos

Guia do Processo Híbrido de Análise de Requisitos

Objetivo-chave do processo: Realizar a análise/coleta dos requisitos de usuário

Tipo de guia: Implementação de software

Definir as entradas do processo:

1. As empresas devem firmar como primeiro critério de entrada a seleção de quais artefatos produzidos durante o planejamento serão utilizados para tornar mais simples o processo de identificação e coleta dos requisitos do cliente (FDD_REQ).
2. As empresas devem fixar como segundo critério de entrada a escolha de um meio adequado de comunicação e participação do cliente (presencial ou virtual), ou representante do mesmo durante as atividades requeridas pelo processo (SCR_REQ).

Realizar a implementação híbrida do processo:

3. Deve-se fixar um escopo de definição dos requisitos através de um método de coleta dos dados. O método precisa ser contextualizado de acordo com o público-alvo, os meios de contato adotados e as técnicas de identificação utilizadas (FDD_REQ).
4. Pode-se utilizar como estratégia de elicitação dos requisitos, a aplicação de questionários e a realização de entrevistas. Recomenda-se utilizar expressões que tonem o diálogo mais simples e a coleta mais objetiva (SCR_REQ; XP_REQ).
5. Deve-se utilizar técnicas de representação dos requisitos coletados, recomendando-se a produção de diagramas UML de casos de uso, classes, sequencia e estados. É importante que seja criado um cenário com os atores, suas responsabilidades, e o comportamento do fluxo de dados que será desempenhado pelo software (FDD_REQ).
6. Deve-se padronizar a descrição dos requisitos funcionais codificados nas iterações do projeto. Sugere-se identificá-los com a atribuição de um código em ordem crescente, por exemplo, “RF-XY”, para facilitar a leitura e futuras atualizações (FDD_REQ).
7. Pode-se complementar a padronização dos requisitos funcionais utilizando-se a estratégia de classificá-los de acordo com as prioridades de negócio. Cada RF pode

receber um valor simbólico, pertencente a uma escala (1-5), que represente o grau de importância que ele precisa ser codificado durante as etapas do projeto (SCR_REQ).

8. Devem ser criadas as categorias que facilitam o alinhamento dos requisitos não funcionais (RNFS) do software. É recomendado que sejam definidas pelo menos duas, sendo a primeira composta pelos RNFS relativos às demandas do cliente, e a segunda com os RNFS relativos às demandas de construção do software (XP_REQ).

Avaliar os benefícios, impactos e problemas da implantação:

- A adaptação acarreta benefícios no contexto de tornar mais dinâmicos os métodos de coleta de requisitos. Em projetos que o cliente participa ativamente, os requisitos podem estar mais claros e simplificados, com o uso das estratégias que foram citadas.
- O escopo de representação dos requisitos e a quantidade de documentos tornam-se flexíveis com o tipo de projeto. Em projetos menores, de mudanças constantes, a documentação pode ser mínima, em contraponto aos projetos de maior complexidade.
- Os impactos causados com a formação de um novo processo interferem diretamente na qualidade do produto. A prerrogativa de que quanto mais requisitos a qualidade do software será superior está errada, entretanto, quanto melhores descritos, então podem contribuir com melhor segurança no desenvolvimento para atender todas as demandas.
- Como problema mais evidente destaca-se a imprevisibilidade em saber se os documentos concebidos serão suficientes para realizar a construção do software. Em alguns casos, a documentação pode garantir esta situação, mas em outros, a mesma estará defasada em virtude do desuso dos artefatos originais modificados no processo.

Definir as saídas do processo:

9. Ao término da adaptação do processo a lista de requisitos funcionais do cliente precisa estar parcialmente finalizada. É sugerido que o artefato esteja classificado através de uma versão ou algo que identifique a última atualização realizada (FDD_REQ).
10. Os profissionais devem gerar como artefatos de saída do processo os rascunhos do fluxo dos requisitos funcionais ou os diagramas resultantes das modelagens UML. Cada artefato precisa ser identificado quanto ao tipo, importância e versão para facilitar o armazenamento e as possíveis atualizações que possam surgir (FDD_REQ).

5.4 Guia do processo híbrido de construção do software

Guia do Processo Híbrido de Construção de Software

Objetivo-chave do processo: Realizar a codificação dos requisitos do cliente

Tipo de guia: Implementação de software

Definir as entradas do processo:

1. As empresas devem fixar como primeiro critério de entrada a seleção de quais artefatos produzidos durante a coleta de requisitos funcionais do cliente serão utilizados como fonte de referência para a codificação do software (FDD_COD).
2. Pode-se adotar como segundo requisito de entrada a definição de um tempo padrão de execução das atividades relativas ao desenvolvimento, com o propósito de garantir a codificação coordenada entre os profissionais envolvidos (SCR_COD).

Realizar a implementação híbrida do processo:

3. As equipes devem fixar as atividades de gerenciamento das iterações. Para isso é necessário que seja estabelecido um cronograma que estabeleça os prazos de codificação, testes, integração e entrega dos protótipos, além de definir a frequência de repasse dos impedimentos e dos demais problemas encontrados (SCR_COD).
4. Devem ser definidas as regras de distribuição/escolha das atividades. Recomenda-se que os programadores mais experientes determinem como será a divisão do trabalho, atribuindo as tarefas de acordo com a capacidade dos profissionais (SCR_COD).
5. Devem ser definidas as regras de padronização do código-fonte. É importante, por exemplo, atribuir uma nomenclatura nas variáveis, métodos, funções e demais estruturas lógicas com termos específicos da área de negócios, facilitando a legibilidade e a integração dos módulos que compõem o software (XP_COD).
6. Recomenda-se que acompanhamento da codificação utilize a prática de reuniões curtas e constantes. Caso o contato seja virtual, aconselha-se o uso de mensageiros instantâneos, além de outras ferramentas que ofereçam portabilidade (SCR_COD).

7. Deve-se melhorar continuamente o processo de codificação com o compartilhamento do código-fonte. Cada programador pode ser proprietário, em parceria com outros colegas, por um pacote de classes ao longo das iterações (XP_COD; FDD_COD).
8. Pode-se documentar o código-fonte como uma estratégia para mapear a infraestrutura do software. Uma boa alternativa seria utilizar marcações HTML para descrever os relacionamentos entre os componentes, bibliotecas e as bases de dados (XP_COD).

Avaliar os benefícios, impactos e problemas da implantação:

- O benefício de tornar a codificação uma atividade híbrida é poder alinhar boas práticas de gerência e engenharia para obter um processo eficiente. É possível que os profissionais acompanhem o cumprimento dos prazos de entrega do processo ao mesmo tempo em que verificam se as atividades alinhadas estão surtindo bons efeitos.
- O tempo e o esforço de codificação podem ser reduzidos com novas práticas alinhadas ao processo. Caso o projeto tenha menor porte, serão alocadas práticas mais simples de gerência e engenharia, diferente da demanda sempre exigida em projetos maiores.
- O impacto influencia na qualidade do produto, não dependendo apenas dos testes, mas no modo como o processo é conduzido. Os testes influenciam com plena a certeza, mas antes disso, necessitam de um escopo de atividades que estejam bem organizadas.
- Como pontos negativos, a codificação quando aborda um formato híbrido pode não atingir os requisitos da qualidade exigidos para possíveis certificações, modelos de maturidade, ou tornar-se ineficiente em alguns casos, como nos projetos onde o controle excessivo das atividades pode interferir diretamente na perda de tempo.

Definir as saídas do processo:

9. Ao término da codificação as equipes podem reunir os artefatos (arquivos-fonte, arquivos binários, arquivos de configuração, bibliotecas etc.) e compactá-los em pacotes de dados para facilitar em seguida a integração dos módulos (FDD_COD).
10. Podem ser gerados também artefatos característicos como relatórios, em que sejam registrados os impedimentos, problemas de codificação e as sugestões de mudanças para serem aplicadas na próxima iteração (SCR_COD).

5.5 Guia do processo híbrido de integração de software

Guia do Processo Híbrido de Integração de Software

Objetivo-chave do processo: Realizar a integração das unidades funcionais

Tipo de guia: Implementação de software

Definir as entradas do processo:

1. As empresas devem fixar como primeiro critério de entrada a verificação dos artefatos de códigos-fontes necessários, resultantes da codificação, para garantir que a integração dos pacotes do software ocorra sem nenhum problema (FDD_INT).
2. Pode-se firmar como segundo critério de entrada a definição de um tempo padrão de execução das atividades relativas à integração do software, com o propósito de facilitar a resolução dos problemas encontrados pelos profissionais (XP_INT).

Realizar a implementação híbrida do processo:

3. Os profissionais devem fixar um escopo de gerenciamento da integração durante as iterações. É indispensável estabelecer um cronograma com as datas das atividades, a rastreabilidade das unidades que serão integradas, a identificação das funcionalidades afetadas e a alocação dos responsáveis pelo processo (SCR_INT; XP_INT).
4. Deve-se analisar/checar continuamente os erros de escrita, padronização e legibilidade dos códigos-fonte. Recomenda-se que esta atividade seja realizada paralelamente a integração, em grupos compostos por dois ou três programadores (XP_INT).
5. Pode-se utilizar como uma alternativa para gerenciar os testes de integração, o mapeamento dos arquivos e *paths* de configuração. Nestes casos, sugere-se que sejam utilizados arquivos de marcação XML, responsáveis por facilitar a padronização, manipulação e o compartilhamento de vários tipos de dados (XP_INT; FDD_INT).
6. As equipes devem estabelecer uma ordem de integração entre as unidades funcionais do software. É importante sintetizar um protocolo que contenha a descrição sobre os testes realizados para que o processo formado seja mais eficiente (XP_INT).

7. Pode-se realizar a integração das unidades funcionais do software através de dois ou três grupos. A estratégia é aplicar os testes através de um rodízio de vários programadores para que os erros encontrados não passem despercebidos (XP_INT).
8. As equipes devem promover o controle de mudanças na integração. Sugere-se que a cada nova versão, a identificação seja atribuída com um rótulo seguindo-se, por exemplo, o modelo: [nome da aplicação] + número atual da versão (FDD_INT).

Avaliar os benefícios, impactos e problemas da implantação:

- A adaptação do processo de integração reporta como benefício a possibilidade de gerar novas versões do software em paralelo ao desenvolvimento. A vantagem, neste sentido, é devido ao fato de que o processo não precisa ser executado obrigatoriamente após a codificação, atribuindo flexibilidade para os profissionais avaliarem e melhorarem os resultados das funcionalidades ao longo das iterações.
- A adaptação do processo de integração também contribui para o ganho de tempo de trabalho dos profissionais. Em projetos menores as versões do software serão produzidas continuamente, adiantando a verificação de atendimento dos requisitos do cliente para que as mudanças corretivas sejam aplicadas cada vez mais rapidamente.
- Os impactos afetam os custos do projeto, caso a integração for realizada seguindo-se um fluxo pós-codificação. O tempo de projeto poderia aumentar e a continuação das atividades ser inviabilizada em virtude da aquisição de novos recursos financeiros.
- Um ponto negativo que pode ser destacado com a adaptação do processo é o de que algumas ferramentas, modelos e *frameworks* disponíveis no mercado podem ter seu uso inviabilizado pelo fato de exigirem que o software tenha sido finalizado.

Definir as saídas do processo:

9. Ao término da integração, uma estratégia que pode ser adotada é a criação de *mocks* de objetos. As estruturas lógicas são copiadas, sendo em seguida aplicadas simulações nas funcionalidades, possibilitando que as falhas sejam corrigidas mais rapidamente sem que código-fonte original seja danificado (FDD_INT).
10. Devem ser sintetizados os artefatos que representam a estrutura raiz (web) ou os instaladores executáveis para possibilitar a verificação dos resultados das funcionalidades e posteriormente a qualificação e os testes do software (XP_INT).

5.6 Guia do processo híbrido de qualificação e testes de software

Guia do Processo Híbrido de Qualificação e Testes de Software

Objetivo-chave do processo: Realizar os testes nas unidades funcionais

Tipo de guia: Implementação de software

Definir as entradas do processo:

1. Deve-se fixar como primeiro critério de entrada do processo a verificação dos arquivos de códigos-fonte, configuração e executáveis que são necessários para garantir a qualificação e os testes das unidades funcionais do software (FDD_TST).
2. Pode-se definir como segundo critério de entrada a identificação dos critérios de qualificação que serão adotados, além de listar todos os testes de software que foram selecionados para que torne possível executar o processo corretamente (XP_TST).

Realizar a implementação híbrida do processo:

3. As equipes devem elaborar um roteiro que especifique como serão realizadas as atividades de cada teste. É indispensável que estejam descritos os objetivos das atividades, as condições de implementação necessárias, as limitações criadas, os impactos causados nas funcionalidades e os resultados que estão previstos (XP_TST).
4. Devem ser especificados os protocolos (passo-a-passo) de montagem do ambiente e a identificação das ferramentas para realizar os testes. É importante analisar quais recursos serão necessários, além de alocar os profissionais responsáveis (XP_TST).
5. Devem ser definidos os métodos dos testes. No primeiro caso, a avaliação precisa ser interna, baseada em uma série de testes dirigidos, com inspeções no código-fonte sem a interferência de terceiros. No segundo caso, a verificação é voltada para os clientes com a aplicação dos testes de aceitação e usabilidade (XP_TST; FDD_TST).
6. Pode-se controlar a sequencia de testes internos promovendo versões. Para isso, sugere-se que a identificação seja atribuída através de um rótulo, seguindo-se o modelo: [nome da aplicação] + número atual da versão + *Alpha* (FDD_TST).

7. Pode-se utilizar a mesma estratégia para controlar a sequencia dos testes externos. Sugere-se que a identificação seja atribuída com o mesmo rótulo, alterando-se apenas a nomenclatura: [nome da aplicação] + número atual da versão + *Beta* (FDD_TST).
8. As equipes devem avaliar o índice de eficiência dos testes. Recomenda-se que os erros sejam identificados e catalogados, atribuindo-se a nomenclatura [código de referência + nome do erro] para viabilizar os meios de quantificação dos dados (FDD_TST).

Avaliar os benefícios, impactos e problemas da implantação:

- A adaptação do processo reporta como principal benefício a possibilidade de alinhar as mesmas práticas de gerência para os métodos internos e externos de testes do software. Desta forma, as estratégias podem ser reaproveitadas proporcionando a diminuição do trabalho e o ganho de tempo dos profissionais ao longo das iterações.
- Outro benefício obtido é a integração de vários tipos de testes de diferentes metodologias. Os profissionais podem formar um processo que forneça uma avaliação mais detalhada, com os subsídios para construir um código-fonte de melhor qualidade, e verificar com maior precisão se o software obtém os resultados que são esperados.
- O impacto é relevante na qualidade do software, que de certa fora, depende do processo. A prioridade, assim como nos padrões normativos ou metodologias ágeis, é conscientizar os profissionais a manterem medidas preventivas que impulsionem a busca pela confiabilidade, segurança e garantias de funcionamento do software.
- Os principais problemas são interligados a ineficiência de alguns escopos formados dos testes. As atividades selecionadas podem ser limitadas, desperdiçando tempo, recursos e a confiança do cliente acerca da capacidade operacional do software.

Definir as saídas do processo:

9. Ao término do processo podem ser gerados os artefatos que contenham os dados técnicos de configuração do software. O formato recomendando é o de uma espécie de "ficha técnica do software", padronizando os requisitos de armazenamento, instalação, execução e manutenção para orientar a manipulação realizada pelo cliente (XP_TST).
10. Uma prática fundamental a ser adotada é a criação de um cronograma que viabilize a otimização contínua do software. Neste caso, deve ser definido um calendário de visita ao cliente para verificar os problemas encontrados e propor as soluções (XP_TST).

5.7 Guia do processo híbrido de gerência da documentação de software

Guia do Processo Híbrido de Gerência da Documentação de Software

Objetivo-chave do processo: Realizar o gerenciamento dos artefatos gerados

Tipo de guia: Suporte de software

Definir as entradas do processo:

1. Deve-se firmar como primeiro requisito de entrada a identificação dos processos do ciclo de vida que deverão gerar documentos, reportando esta abordagem aos profissionais para que seja providenciado o gerenciamento compatível (FDD_DOC).
2. Pode-se fixar como segundo critério de entrada uma avaliação que defina qual será o porte da documentação que será gerada, caracterizando-a como extensiva, intermediária, ou ágil para facilitar a escolha das atividades de gerência (SCR_DOC).

Realizar a implementação híbrida do processo:

3. As equipes devem padronizar a estrutura e o formato das documentações ao longo do projeto. Recomenda-se que os profissionais confeccionem *templates* com uma identificação externa, versão e o conteúdo dividido através de seções (FDD_DOC).
4. Também é importante que sejam identificadas as ferramentas para a produção de documentos. Em projetos menores, poderão ser utilizados materiais de papelaria para viabilizar a modelagem ágil como papéis, lápis e canetas, ou caso a demanda seja superior, ferramentas gráficas ou suítes de escritórios (SCR_DOC; XP_DOC).
5. Pode-se utilizar como estratégia de homologação a síntese de redações técnicas. Sugere-se que o conteúdo interno de cada seção seja formado por campos, lacunas, ou textos corridos onde estão transcritas as dúvidas, críticas e sugestões (XP_DOC).
6. Devem ser elaborados os manuais de funcionamento do software. O primeiro precisa abordar as especificações técnicas, tais como políticas de uso, licença do usuário, instalação, manutenção e remoção do software. O segundo tem o objetivo de capacitar os usuários a operarem todas as funcionalidades que estão disponíveis (XP_DOC).

7. Devem ser fixadas as regras de manipulação e configuração dos documentos. Se os artefatos forem lógicos, recomenda-se o uso de ferramentas de gerenciamento eletrônico de documentos (GED) que oferecem uma maior praticidade (FDD_DOC).
8. Caso a documentação seja extensa, e com o formato físico, recomenda-se gerenciá-la através do controle de versões, ou caso seja necessário, com o uso de técnicas arquivistas que tornam mais seguras as atividades de armazenamento (FDD_DOC).

Avaliar os benefícios, impactos e problemas da implantação:

- O benefício em adaptar um processo de documentação é a possibilidade das empresas reduzirem a burocracia de seus projetos. Em projetos menores, a especificação de muitos artefatos pode retardar a produção, além de exigir a necessidade de contratar profissionais especializados, responsáveis apenas por gerenciar as documentações.
- Outra vantagem é eliminar a dependência das operações das documentações, visto que a concepção de documentos não se torna uma atividade obrigatória. A quantidade dos artefatos pode ser proporcional à complexidade e ao porte característico do projeto.
- O impacto é direto na riqueza dos detalhes e na qualidade dos documentos que são gerados. Independente das atividades selecionadas para formarem o processo, o resultado que deve ser esperado é a síntese de uma documentação que seja consistente.
- A desvantagem de adaptar um processo são os efeitos de incompatibilidade nos documentos que podem ser gerados entre os associados e parceiros. O descarte de modelos aceitos pelo mercado pode gerar inconveniências no fechamento de parcerias.

Definir as saídas do processo:

9. Ao término do processo, é importante que seja gerado um artefato que mapeie todos os documentos gerados pelos demais processos. Sugere-se que para cada documento seja atribuído um índice que sirva de referência para eventuais consultas (FDD_DOC).
10. Podem ser gerados também “diários de relatos” que reportem os problemas encontrados na documentação. Neste caso, os profissionais precisam reportar o título, versão, seções e demais erros identificados nas iterações para que os autores sejam comunicados para providenciarem rapidamente as mudanças (SCR_DOC).

5.8 Guia do processo híbrido de gerência da configuração do software

Guia do Processo Híbrido de Gerência da Configuração do Software

Objetivo-chave do processo: Gerenciar as mudanças dos artefatos do software

Tipo de guia: Suporte de software

Definir as entradas do processo:

1. Deve-se fixar como primeiro critério de entrada a verificação de que todos os artefatos gerados pelos processos ao longo do ciclo de vida estão identificados, além de coletar as informações técnicas necessárias para facilitar o gerenciamento (FDD_CNF).
2. Pode-se fixar como segundo critério de entrada uma verificação que constate se a infraestrutura de configuração é compatível com a demanda do projeto, para garantir a obtenção de condições suficientes para a formação do processo (FDD_CNF).

Realizar a implementação híbrida do processo:

3. As equipes devem especificar um roteiro que facilite o controle das mudanças. Recomenda-se que o formato adotado seja similar ao de um protocolo, que gere um código para cada mudança, contenha os nomes dos artefatos relacionados, as prioridades e os tipos de operação para *upload* e *download* dos arquivos (FDD_CNF).
4. Pode-se complementar o gerenciamento da configuração dos artefatos atribuindo uma escala de processamento das mudanças. A escala poderia conter três fases que indicassem o status: “iniciada, em andamento, finalizada” (FDD_CNF).
5. Devem ser identificados os tipos de arquivos para armazená-los organizadamente através de diretórios. É importante que o repositório contenha uma estrutura-raiz para garantir a submissão (*commit*), modificação (*merge*), e versionamento (FDD_CNF).
6. Pode-se gerenciar as mudanças no repositório de armazenamento através de solicitações de acesso (CRs). Neste caso, deverão ser providenciados: o cadastramento dos profissionais que terão acesso, as regras de manipulação e as estratégias de monitoramento que portarão a segurança dos arquivos existentes (FDD_CNF).

7. Devem ser criadas diferentes cópias para os arquivos contidos no repositório de armazenamento. A primeira pode ser uma cópia de segurança (*backup*), a segunda uma cópia de resgate (*recovery*) para a edição local do arquivo (FDD_CNF).
8. Pode-se adotar a alternativa de gerenciar as atualizações das cópias dos arquivos, mapeando-se as mudanças realizadas nas iterações. Aconselha-se criar uma matriz de rastreabilidade contendo: o número da iteração + o número da mudança (FDD_CNF).

Avaliar os benefícios, impactos e problemas da implantação:

- O principal benefício de adaptar um processo de configuração é a possibilidade das empresas diminuírem o esforço de trabalho, a partir do momento que criam ambientes expansíveis de armazenamento, manipulação e acesso dos arquivos nas iterações.
- Outro benefício é a possibilidade de extinguir o uso de ferramentas pagas no mercado. Os profissionais podem recorrer às aplicações gratuitas, adaptando as funcionalidades mais simples para serem utilizadas de acordo com cada ambiente de configuração.
- O impacto está relacionado ao planejamento do processo e a gerência dos artefatos. Caso as atividades não estejam estabelecidas corretamente, a falta de organização dos artefatos pode gerar consequências desagradáveis ao projeto como os atrasos no cronograma, a perda de documentos importantes e a desconfiança por parte do cliente.
- As desvantagens em atribuir configurações híbridas condizem às incompatibilidades geradas quando as empresas contratam serviços no mercado que auxiliam a gerência dos artefatos, como ferramentas de *backups* online, plataforma empresariais etc.

Definir as saídas do processo:

9. As empresas podem gerar um artefato que relate as prioridades que não foram atendidas, as pendências alocadas para a próxima iteração e as propostas de mudanças para corrigir os problemas. Recomenda-se que seja elaborado um cronograma com as novas prioridades estabelecidas e a ordem de aplicação das atividades (FDD_CNF).
10. Pode-se gerar um artefato que descreva os mecanismos ideais de configuração para determinados tipos de projeto. O documento precisa conter os requisitos técnicos do ambiente, a descrição da infraestrutura do cenário, além das recomendações de como as ferramentas possam ser reaproveitadas para projetos futuros (FDD_CNF).

5.9 Guia do processo híbrido de garantia da qualidade do software

Guia do Processo Híbrido de Garantia da Qualidade do Software

Objetivo-chave do processo: Garantir/inserir melhorias nos processos de software.

Tipo de guia: Suporte de software

Definir as entradas do processo:

1. Deve-se adotar como primeiro critério de entrada a criação de uma lista onde estejam especificados os requisitos da qualidade nos processos do ciclo de vida para tornar possível identificar as atividades que irão promover as melhorias (FDD_QLD).
2. Pode-se adotar como segundo critério de entrada a criação de uma lista onde estejam especificados os requisitos da qualidade analisados nos produtos gerados, para facilitar a identificação de novas atividades que complementarão o processo (FDD_QLD).

Realizar a implementação híbrida do processo:

3. Deve-se estabelecer um método de avaliação para a qualidade dos processos de software. Recomenda-se a elaboração de um protocolo, dividido através de fases, onde a verificação possa ser realizada gradativamente com o *diagnóstico* do processo, *avaliação dos resultados*, a *aplicação* e o *monitoramento das mudanças* (FDD_QLD).
4. Devem ser definidas as variáveis de análise da qualidade dos processos. Neste caso, sugere-se que os profissionais adotem critérios simples, como o cálculo do tempo de horas trabalhadas, atrasos, tempo médio de execução das atividades etc. (FDD_QLD).
5. Pode-se avaliar os processos de acordo com a sua importância. Uma solução prática seria estabelecer uma escala onde fossem atribuídas classificações pela demanda de trabalho, sendo o processo: *crítico*, *importante* ou *complementar* (FDD_QLD).
6. Deve-se especificar um cronograma para a avaliação da qualidade do código-fonte do software. É indispensável que as equipes adotem boas práticas de verificação, como a leitura do código, busca de erros e a padronização do design (FDD_QLD).
7. Devem ser apresentados os resultados de verificação da qualidade dos processos. Em projetos de menor porte, aconselha-se que sejam atribuídos valores simbólicos (1-5)

que representem um *ranking* da necessidade de inserção de melhoria, alertando os profissionais a planejarem as medidas corretivas aplicadas nas iterações (FDD_QLD).

8. Deve-se avaliar as condições de trabalho e o índice de produtividade dos profissionais. É importante que seja atribuída uma cota (semanal/mensal/trimestral) de tarefas individuais e que os responsáveis pela condução do processo avaliem o tempo médio de execução elaborando gráficos e outros meios de acompanhamento (FDD_QLD).

Avaliar os benefícios, impactos e problemas da implantação:

- A principal vantagem de formatar um processo híbrido de garantia da qualidade do software é a de que as empresas podem definir quais são seus requisitos da qualidade, e não depender do atendimento dos requisitos obrigatórios impostos pelos modelos de maturidade de processo e produto disponíveis atualmente no mercado.
- Outra vantagem é a possibilidade de desenvolver um plano da qualidade totalmente compatível, que disponha todas as medidas que ofereçam melhorias, descartando-se aquelas que porventura não surtam efeitos por serem incompatíveis com os processos.
- Os impactos são sem sombra de dúvidas são colaborativos em todos os aspectos, sejam funcionais e não funcionais. A garantia da qualidade é alcançada em um escopo organizacional com o ganho de maturidade para com todos os processos e pessoas.
- A desvantagem em adaptar o processo de avaliação de outros processos, é a falta de nivelamento exigido por padrões de certificação. Quando a infraestrutura organizacional é modificada, não atendendo os critérios impostos pelos órgãos de normatização, o reconhecimento da qualidade torna-se inexistente para o mercado.

Definir as saídas do processo:

9. As empresas podem gerar artefatos como “guias de melhorias”, mapeando os problemas encontrados nas iterações e as respectivas soluções. Esta estratégia é fundamental para evitar a repetição dos erros de projetos anteriores, além de orientar sobre como os profissionais devem agir em determinadas situações (FDD_QLD).
10. As empresas podem adotar cartilhas ou informativos que estabeleçam as instruções para execução dos processos. Essa alternativa torna-se importante para combater o surgimento de possíveis problemas quando forem realizadas eventuais substituições ou o ingresso de novos profissionais durante o projeto (FDD_QLD).

5.10 Guia do processo híbrido de verificação de software

Guia do Processo Híbrido de Verificação de Software

Objetivo-chave do processo: Verificar as especificações construídas do software

Tipo de guia: Suporte de software

Definir as entradas do processo:

1. Os profissionais devem firmar como primeiro critério de entrada a escolha de um ambiente propício para interagir com o cliente com o objetivo de demonstrar o funcionamento do software (SCR_VER; XP_VER; FDD_VER).
2. Pode-se fixar como segundo critério de entrada a escolha dos recursos utilizados para que sejam articuladas as dinâmicas de demonstração das versões parcial ou final do software, tais como Datashow, slides, apresentações etc. (SCR_VER; XP_VER).

Realizar a implementação híbrida do processo:

3. As equipes devem desenvolver um roteiro que facilite o processo de verificação do software. É importante que seja elaborada uma pauta de discussão onde estejam listados os tópicos-chave com os assuntos discutidos na reunião, tais como a avaliação do projeto, funcionalidades e o suporte de dúvidas do cliente (SCR_VER).
4. Pode-se realizar a verificação dos manuais do software através de duas etapas. Na primeira precisa ser analisada a ortografia e semântica para que o conteúdo textual não gere dúvidas de operação. Na segunda, a verificação deve ser no conteúdo técnico, garantindo que as especificações do software estão todas corretas (XP_VER).
5. A avaliação da usabilidade (design) do software pode ser realizada com testes ergonômicos. Para cada funcionalidade pode ser padronizada a quantidade de passos a serem seguidos, verificando se o cliente obtém dificuldades de acesso (FDD_VER).
6. Deve-se realizar a verificação do software com a presença de profissionais experientes. Neste caso, recomenda-se que sejam montados sistemas de rodízio em pares para demonstrar com maior eficiência as funcionalidades ao cliente (XP_VER).

7. Pode-se dividir a reunião de verificação em duas partes. Na primeira parte devem-se reunir apenas os profissionais, buscando inspecionar o código-fonte e o design do software. A segunda parte deve contar com a presença do cliente, onde são apresentados os resultados e coletadas as críticas, dúvidas e sugestões (SCR_VER).
8. É recomendável que a reunião de verificação do software seja documentada. Em projetos menores, pode adotar a estratégia de produzir pequenas atas que registrem os assuntos abordados, os problemas e as decisões que serão tomadas (SCR_VER).

Avaliar os benefícios, impactos e problemas da implantação:

- O principal benefício de formatar um processo híbrido de verificação é o de definir o nível de formalidade, ambientação, tempo de duração e a riqueza de detalhes de acordo com conveniência, interesse, e satisfação do cliente para que o mesmo adquira totais condições de habituar-se em que situação atual encontra-se seu investimento.
- Um segundo benefício de manter a prática de adaptação no processo, é o de alterar a escala de verificações de uma ou mais iterações sempre que necessário, diferentemente de um processo fixo que determina a quantidade máxima por iteração.
- Os impactos são relevantes quando se tem uma criticidade das funcionalidades do software e a dependência extrema quanto a isso. Desta forma, a satisfação do cliente dependerá, em parte, diretamente do grau de eficiência que é provido pela verificação.
- Como efeito negativo, a não definição de um número fixo de verificações pode ocasionar mudanças no tempo de projeto, aumento da demanda de recursos e outros aspectos que prejudiquem a evolução do software. Caso a empresa não planeje um número-base, as consequências negativas podem ser difíceis de serem previstas.

Definir as saídas do processo:

9. Ao término da verificação os profissionais podem gerar um artefato que liste os itens que foram verificados com êxito e os que ficaram pendentes. Uma prática simples é gerar uma tabela que contenha os tópicos que tenha sido descritos (SCR_VER).
10. Outra prática indispensável a ser adotada ao término do processo é a de manter a comunicação regularmente com o cliente, para agendar aos poucos todos os detalhes necessários que serão abordados na próxima verificação (SCR_VER; XP_VER).

5.11 Guia do processo híbrido de validação de software

Guia do Processo Híbrido de Validação de Software

Objetivo-chave do processo: Validar as especificações construídas do software

Tipo de guia: Suporte de software

Definir as entradas do processo:

1. Deve-se firmar como primeiro critério de entrada a identificação do perfil do cliente (leigo, intermediário, experiente) em relação ao conhecimento de informática, para instruí-lo a reportar corretamente os problemas encontrados no software (XP_VAL).
2. Pode-se firmar como segundo critério de entrada a escolha de um método de acompanhamento e suporte ao cliente, além de definir qual será a periodicidade e o tempo de duração das reuniões realizadas no processo (XP_VAL; FDD_VAL).

Realizar a implementação híbrida do processo:

3. As equipes devem especificar um roteiro com as instruções para que o software seja validado. Recomenda-se elaborar casos de uso sobre as funcionalidades, repassando-se para o cliente uma cartilha com atividades pré-estabelecidas para verificar se o mesmo poderá realizar as operações sem maiores problemas (SCR_VAL; XP_VAL).
4. Pode-se utilizar a estratégia de elaborar os testes de validação para serem realizados individualmente ou em grupos. Neste caso, são indicados três ou quatro usuários propondo-se uma avaliação integrada e em maior escala (FDD_VAL).
5. Devem ser especificadas simulações na validação. Sugere-se que os profissionais testem as funcionalidades, comparando seu tempo de operação ao tempo obtido pelo cliente, para identificar possíveis problemas que tenham acontecido (XP_VAL).
6. Quando a demanda do cliente pelo software for alta, é importante que a validação seja contínua. Uma estratégia que auxilia o suporte é a distribuição de FAQs durante as iterações proporcionando ao cliente o ganho de tempo em suas operações (XP_VAL).
7. Pode-se utilizar a prática de realizar pesquisas de satisfação como uma forma de obter um *feedback* sobre os resultados obtidos com a validação. Uma alternativa interessante

seria disponibilizar algumas caixas de sugestões no ambiente de trabalho do cliente para coletar opiniões de melhorias que possam ser viabilizadas (XP_VAL).

8. Devem ser disponibilizadas pequenas cópias para que o cliente valide o software gradativamente. Recomenda-se que os profissionais mapeiem as versões e as respectivas funcionalidades para tornar mais ágil a aplicação de mudanças (XP_VAL).

Avaliar os benefícios, impactos e problemas da implantação:

- O principal benefício de formatar um processo híbrido de validação do software é o de tornar mais flexíveis os requisitos de aceitação a serem acatados em comum acordo pelos clientes e pela empresa. Ambos podem combinar quais são as metas a serem atingidas, possibilitando obter resultados tangíveis e mais objetivos com o projeto.
- Outro benefício da adaptação é a possibilidade de alterar a linha do tempo (*timeline*) de execução do processo ao longo do ciclo de vida, podendo o mesmo ser executado paralelamente a construção do software com reuniões conjuntas e simultâneas.
- O impacto de um processo híbrido neste contexto está diretamente associado à relação desempenhada pela empresa para fornecer o suporte devido ao cliente. Isso implica dizer que não basta obter o cliente presente nas atividades, mas sim, implantar e manter as estratégias de retaliação dos problemas encontrados ao longo do processo.
- Uma desvantagem da validação com caráter híbrido é a de que o planejamento das atividades pode gerar perda de tempo. Sempre que o calendário for alterado, esse efeito pode acarretar problemas nos cronogramas e prazos dos demais processos.

Definir as saídas do processo:

9. As empresas podem gerar quando necessário algum artefato que documente as solicitações de mudanças requeridas pelo cliente. Esta prática é indispensável, no sentido de descrever as justificativas das novas demandas (SCR_VAL; XP_VAL).
10. Podem ser gerados “relatórios de ocorrências” com um formato de um livro, ata ou algo similar em que sejam registrados os problemas. O objetivo é identificar as causas e propor as melhores soluções corretivas (SCR_VAL).

5.12 Considerações finais

A Engenharia de Software possibilitou o surgimento de diversos modelos que se complementam para formar uma infraestrutura de desenvolvimento mais organizada. Ao longo das últimas seções, pôde-se observar que as metodologias ágeis *Scrum*, XP e FDD oferecem práticas eficientes que colaboram para que este objetivo seja alcançado, quando a estratégia de melhoria escolhida for a formação de processos híbridos de software.

De certa forma, o que muitos profissionais almejam é alocar as melhores ideias, fornecer as condições ideais para seu condicionamento, obter uma gestão de processos e pessoas adequada, e garantir que o software seja entregue no tempo hábil para que o cliente fique satisfeito. O modo certo de fazer, e como se deve fazer, é o que se torna uma grande incógnita, pois ainda não se sabe qual é o modelo, método ou processo de desenvolvimento que reporta as garantias confiáveis de que os bons resultados serão obtidos.

O que se buscou neste capítulo foi apresentar dez guias com as melhores práticas ágeis de gerência e engenharia para que os profissionais formem seus próprios processos de software. A tendência desta temática, segundo as experiências vivenciadas pelo corrente autor, é algo que vem crescendo no mercado através da busca de novas soluções que surtem resultados positivos, mas sem a necessidade de que sejam desembolsados altos investimentos.

É importante concluir que nenhum processo, metodologia ou *framework* provêm uma escala completa de atividades em seus ciclos de vida. Por menores que sejam, sendo programadas, ou não, sempre haverá mudanças que precisarão de alternativas paralelas que supram as necessidades da infraestrutura do escopo original, agregando valores funcionais ao meio e aos personagens que as integram.

Portanto, cabe a cada profissional definir os valores, meios, justificativas e as propostas para classificar quão impactantes serão as atenções destinadas aos processos do projeto. A composição de um ciclo de vida híbrido de software depende extremamente do planejamento fixado, e das condições proporcionadas pelos profissionais, para que todas as metas traçadas possam ser alcançadas.

Capítulo

6

Aplicação e resultados da pesquisa de campo

Este capítulo apresenta a pesquisa de campo aplicada para validar as práticas sugeridas nos guias de implantação do ciclo de vida híbrido de software. É disposta a classificação da pesquisa quanto a seu tipo, formas, objetos e a instrumentação selecionada para a coleta, análise e interpretação dos dados. Também são apresentados os resultados obtidos com a aplicação do questionário desenvolvido através de uma análise quantitativa das respostas e, por fim, as conclusões sobre a validade dos guias propostos.

6.1 Classificação da pesquisa de campo

Considerando primeiramente a classificação estipulada por Lima (2010), quanto aos **tipos**, esta pesquisa está classificada como:

- **Pesquisa de campo e opinião:** também conhecida como ***Survey***, a pesquisa de opinião é um método aplicado de adentrar as diferenças sociais, filosofais e teológicas para captar a informação, comportamentos e ações através da interação direta ou indireta da representação do conhecimento (KERLINGER, 1980).
- **Pesquisa experimental:** este tipo de pesquisa almeja realizar a inserção de novos conhecimentos através da proposição de experiências, processos, análises, explicações, conjecturas e outros fenômenos da natureza (KERLINGER, 1980).
- **Pesquisa bibliográfica:** a pesquisa bibliográfica é caracterizada pela formatação de novos conhecimentos através da produção literária dos autores em suas pesquisas. Este

tipo de pesquisa é considerado como a base de formação para o processo de transição e propagação do conhecimento validado (LIMA, 2010).

Por se tratar efetivamente de uma pesquisa aplicada de campo, Lima (2010) recomenda estipular uma classificação quanto às formas, abordando-se dois fatores importantes. O primeiro deles é identificar a **forma de orientação da pesquisa aos seus entrevistados**, assim como controlar a **forma de aplicação da pesquisa ao seu público-alvo**.

Considerando as formas de orientação aos entrevistados, proposta pelo autor, esta pesquisa está classificada como:

- **Não supervisionada**: é a forma de orientação pela qual são extintos quaisquer auxílios, instruções ou orientações diretas dos responsáveis pela aplicação aos entrevistados, neste caso, sendo comuns de serem realizadas individualmente ou em grupos (LIMA, 2010).

Considerando as formas de aplicação ao público-alvo, também recomendada por Lima (2010), esta pesquisa está classificada como:

- **Não probabilística:** não é selecionada uma amostragem controlada, classificada para representar a população de um modo geral. As taxas de amostragem são obtidas por critérios definidos nos objetivos da pesquisa, assim como no processo de montagem quando os entrevistados são de difícil acesso (LIMA, 2010).

Quando se discute os objetivos que caracterizam a pesquisa de campo, relevam-se alguns fatores que precisam ser analisados. Os objetivos não envolvem somente o propósito da realização, mas também a obtenção de subsídios (PINSONNEAULT e KRAMER, 1993).

De acordo com a classificação dos **objetivos** inerentes a uma pesquisa aplicada de campo, do trabalho de Pinsonneault e Kramer (1993), esta pesquisa está enquadrada como:

- **Exploratória:** tem como objetivo proporcionar a expansão do conhecimento na área abordada para o surgimento de teorias, baseando-se em uma investigação minuciosa de acordo com o interesse e a viabilidade da pesquisa, assim como da exploração de conceitos que podem ser medidos (PINSONNEAULT e KRAMER, 1993).
- **Descritiva:** busca reportar cenários, situações ou eventos que expressam a demanda e a importância dos acontecimentos descritos para a população. As hipóteses neste tipo de pesquisa visam a efetivar a comparação para averiguar se as premissas propostas estão aptas a serem utilizadas, ou se estão de acordo com a realidade (LIMA, 2010).

6.2 Instrumentação, definição da amostra e tratamento dos dados

A instrumentação de uma pesquisa se refere ao conjunto de estratégias adotadas para que os dados possam ser adquiridos direta ou indiretamente através de um processo de coleta bem definido (MARCONI e LAKATOS, 2009). O instrumento que oferta tal eficiência, utilizado nesta pesquisa, foi um questionário semiestruturado, elaborado estrategicamente para garantir uma validação efetiva.

O questionário, que pode ser observado no apêndice A, foi estruturado em três partes diferentes. Na "Parte I", foram disponibilizados aos entrevistados sete campos que precisariam ser preenchidos com seus dados pessoais e profissionais, com o objetivo de traçar os perfis relativos aos mercados de software em que os mesmos atuam.

A segunda parte buscou verificar a aderência dos entrevistados ao tema abordado no trabalho, a combinação das práticas como uma proposta válida para melhorar o ciclo de vida do software, e ao método (guias) proposto para subsidiar a adaptação. Nesta parte, foram disponibilizadas três questões, de múltipla escolha, para que pudessem expressar suas opiniões considerando as afirmações propostas por cada questão.

Para garantir a equivalência de oportunidades extremas aos entrevistados para expressar a concordância ou discordância de suas respostas, utilizou-se, na "Parte II", o balanceamento de Likert (1932), também conhecido como *Escala de Likert*[12]. Segundo o autor, a aplicação de questionários é um método extremamente eficiente, quando dispõe nas perguntas várias possibilidades de escolha que refletem com precisão o direcionamento das respostas obtidas através da seleção de uma das seguintes alternativas:

1. Concordo totalmente
2. Concordo parcialmente
3. Indiferente
4. Discordo parcialmente
5. Discordo totalmente

A "Parte III" objetivou validar a proposta do trabalho, com uma análise de aderência dos entrevistados acerca das práticas combinadas contidas nos guias. Nesta parte, foram aplicadas dez questões fechadas, também utilizando-se a escala de equivalência de *Likert.*

[12] A escala Likert é uma escala de avaliação, designada para questionários e meios afins de autoria de Rensis Likert que proporciona aos leitores opções incisivas sobre suas conclusões (LIKERT, 1932).

As práticas foram resumidas e enumeradas com valores de “1” a “10” para realizar a validação na Parte III. A estratégia inserida no questionário foi a de tornar implícita a metodologia que embasa a contextualização dos guias, denominando que os dois primeiros números (1 e 2) seriam padronizados como as entradas de processo (exceto o primeiro processo); os seis próximos números (3 a 8), as boas práticas da implementação híbrida, e os dois últimos números (9 e 10), as saídas dos processos.

O quarto passo, “avaliar os benefícios, impactos e problemas da implantação” não foi adicionado ao questionário aplicado na pesquisa. O motivo é justificado pelo fato de que este passo apenas alerta aos leitores acerca dos resultados que podem ser obtidos com a adaptação dos processos, variando com os fatores que não dependem da metodologia proposta, como recursos financeiros, tempo e o esforço em melhorias promovido pelas empresas.

Por fim, em cada questão da Parte III foi sugerido ao leitor que o mesmo descartasse as práticas consideradas inviáveis para a formação de um novo processo. Este campo se tornou opcional com a perspectiva de que, independente da aderência obtida, variar em um intervalo de 0 a 100%, os entrevistados poderiam efetivar os descartes estando todas as práticas susceptíveis de serem mencionadas no questionário.

Para obter uma maior praticidade, o questionário foi elaborado apenas na versão virtual, disponibilizada através de uma URL pela ferramenta *Google Form*[13]. A circulação foi realizada em várias listas de discussão escolhidas de acordo com os temas relativos ao ciclo de vida do software, como gerência, melhoria e qualidade de processos. Também foram enviados e-mails para diversos profissionais especializados, estando disponível para coleta de dados entre os dias 21 de maio e 13 de junho de 2012.

Dois fatores exerceram forte influência para que se utilizasse o questionário como uma forma de validação: a ausência de recursos disponíveis, dentre eles, o tempo, e os critérios para a escolha da amostra. No primeiro caso, foi considerado o receio de não alcançar a disposição dos profissionais para colaborarem na validação, caso fosse provido o acompanhamento nos cenários de desenvolvimento do software. As hipóteses embasaram-se nos possíveis riscos de atrasos em projetos e do não acatamento das práticas sugeridas.

O segundo fator analisado condiz aos critérios de escolha da amostra, local de aplicação e perfis dos entrevistados. Um estudo recém-publicado pela AgilCoop (2011)

[13] O *Google Form* é um serviço da empresa Google Inc.® que possibilita estruturar questionários na internet.

revelou que cada região do país desenvolve uma tendência para adequar o uso de uma metodologia ágil específica, tornando-se inviável que um questionário sobre processos híbridos fosse disponibilizado em apenas uma cidade ou estado da federação.

Desta forma, o presente autor definiu validar o trabalho com uma amostra obtida através do resultado de aplicação do questionário, considerando como perfil e público-alvo definidos a população de Engenheiros de Software do Brasil. Foram analisadas as respostas de diversos estados brasileiros, incluindo-se o Distrito Federal, descartando-se quaisquer restrições quanto à escolha dos cargos que atuam no ciclo de vida de software.

Também não foram atribuídas restrições na participação de empresas, instituições ou órgãos que promovem o desenvolvimento de software tanto em um âmbito público, quanto privado. O foco foi expandir a avaliação dos guias independente da infraestrutura de trabalho do participante, não sendo requerida a colaboração exclusiva apenas das fábricas de software.

Para isso, foi essencial que o tratamento dos dados fosse realizado com as variáveis quantitativas: frequência de descarte das práticas, porcentagens de aderência aos guias, porcentagens de aderência das práticas, além do uso de técnicas de estatística. Pelo fato de a amostra estar dispersa em várias localidades, tornou-se inviável a criação de questionários qualitativos típicos para serem utilizados em entrevistas ou reuniões.

A manipulação dos dados foi efetuada com as ferramentas *Google Form* e *Microsoft Excel*®. A primeira foi utilizada para calcular as porcentagens de aderência das questões relativas à “Parte II” e à “Parte III” do questionário. A segunda foi utilizada para manipular os dados em planilhas, tais como a frequência e a porcentagens de descarte das práticas, e gerar os gráficos com os resultados observados nas próximas seções.

6.3 Resultados da pesquisa de campo

Durante os vinte dias que se prosseguiram à circulação da pesquisa de campo, muitos empecilhos foram encontrados, dentre eles a resistência das pessoas para preencherem as perguntas do questionário. Outra dificuldade marcante foi o tempo de retorno dos entrevistados, atrasando-se em alguns casos o envio das respostas para que as mesmas fossem contabilizadas.

6.3.1 Apresentação dos resultados relativos à amostra

Diante da amostra obtida, foram coletadas ao todo 164 respostas. Desse valor, só foram consideradas válidas 152, pelo fato de que 5 respostas foram geradas como erro de repetição de envio, causados por problemas na ferramenta de coleta, e as demais 7, obtidas por profissionais que não se enquadravam no público-alvo da pesquisa.

Das respostas válidas, constatou-se a participação de profissionais que atuam em mais de 99 empresas espalhadas pelo Brasil. A amostra foi proveniente de 50 cidades distribuídas em 20 estados, incluindo-se o Distrito Federal, como pode ser observado na Tabela 6.1.

Tabela 6.1: Distribuição das respostas por estados

Estado	Frequência de entrevistados	Porcentagem
PE	51	33,6%
CE	7	4,6%
SC	6	3,9%
PB	29	19,1%
AL	3	2,0%
SE	1	0,7%
BA	3	2,0%
SP	11	7,2%
PR	5	3,3%
PA	1	0,7%
GO	3	2,0%
MT	1	0,7%
RS	7	4,6%
DF	6	3,9%
PI	1	0,7%
MA	1	0,7%
RJ	7	4,6%
AM	4	2,6%
RN	1	0,7%
MG	4	2,6%

Observando a distribuição, percebe-se que alguns estados obtiveram uma representatividade mais expressiva, como é o caso de Pernambuco, com 33,6%, a Paraíba, com 19,3%, e o estado de São Paulo, com 11,2%. Alguns estados obtiveram porcentagens inferiores, mas com uma mesma expressão na avaliação, como Ceará, Rio de Janeiro e Rio Grande do Sul, todos com 4,6%. Outros contabilizaram apenas uma resposta, como Piauí, Maranhão e Rio Grande do Norte, ficando com apenas 0,7% do valor total de participação.

Outro aspecto importante que foi analisado é referente à descrição dos cargos e funções dos entrevistados. Ao todo, foram citados 28 cargos diferentes que estão diretamente interligados com a produção do software, e 2 funções específicas, que contribuem para melhoria contínua dos processos de gerência e engenharia que formam o ciclo de vida, conforme é apresentado na Tabela 6.2.

Tabela 6.2: Cargos e funções dos entrevistados

Cargos e funções dos entrevistados da pesquisa		
Gerente de projetos	Programador	Analista da qualidade
Analista Desenvolvedor	Analista de Suporte	Analista de sistemas
Engenheiro de software	Engenheiro da qualidade	Consultor de projetos
Analista de negócios	Coordenador de T.I.	Líder de T.I.
Analista de Redes	Diretor de desenvolvimento	Administrador de sistemas
Gerente de Sistemas	Gerente de T.I.	Webmaster
Gestor de Processos	Scrum Master	Designer
Líder de Projetos	Analista de T.I.	*Coach* de treinamentos
Analista de Banco de Dados	Analista de requisitos	Pesquisador MPS[14]
Analista de Testes de software	Coordenador da qualidade	Implementador MR-MPS[15]

As duas funções específicas referidas na Tabela 6.2 são “Pesquisador MPS” e “Implementador MR-MPS”. Ambos obtiveram uma participação pequena na amostra, mas a contribuição de suas opiniões tanto no âmbito institucional, quanto para a avaliação dos modelos para processos de software adaptados, foi de grande valida, considerando a importância de que os nichos de pesquisa e mercado precisam promover uma sincronia efetiva para propor a síntese de soluções inovadoras.

[14] MPS é o acrônimo para Melhoria de Processos de Software (SOFTEX, 2011).
[15] MR-MPS é o acrônimo de “Modelo de Referência para Melhoria de Processos de Software” (SOFTEX, 2011).

6.3.2 Apresentação dos resultados quantitativos do questionário

Apresentados os dados da amostra, o próximo passo foi descrever os resultados quantitativos obtidos com a aplicação do questionário. Para facilitar esta apresentação, a ordem adotada nesta seção foi: descrever em primeiro plano as análises obtidas através da "Parte I", em seguida, as análises relativas à "Parte II", e por fim, as análises referentes à "Parte III" do questionário.

Para a "Parte I", não foram definidas questões. Os campos preenchidos tinham como objetivo traçar o perfil dos profissionais entrevistados com a obtenção das seguintes informações: **"Nome do profissional, Nome da empresa, instituição, ou órgão de atuação (opcional), Cargo, Quantidade de profissionais, Tempo de mercado, Cidade, Estado, e e-mail (opcional)**".

De posse destes dados, foi possível realizar duas análises sobre os perfis. A primeira, considerando a variável "**Quantidade de profissionais**", objetivou identificar em qual nível de infraestrutura organizacional (pequeno, médio ou grande porte) os entrevistados da pesquisa estariam inseridos. O resultado obtido pode ser verificado a seguir no Gráfico 6.1.

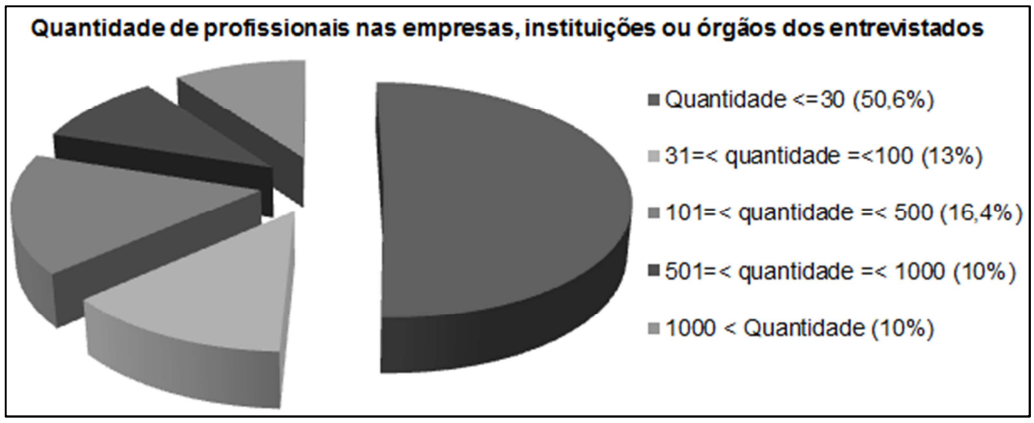

Gráfico 6.1: Quantidade de profissionais no ambiente de trabalho dos entrevistados

Para a segunda análise sobre os perfis, foi considerada a variável "**Tempo de Mercado**". O propósito foi identificar o tempo de consolidação no mercado das empresas, instituições ou órgãos em que atuam os entrevistados. O resultado obtido, conforme representado no Gráfico 6.2, foi:

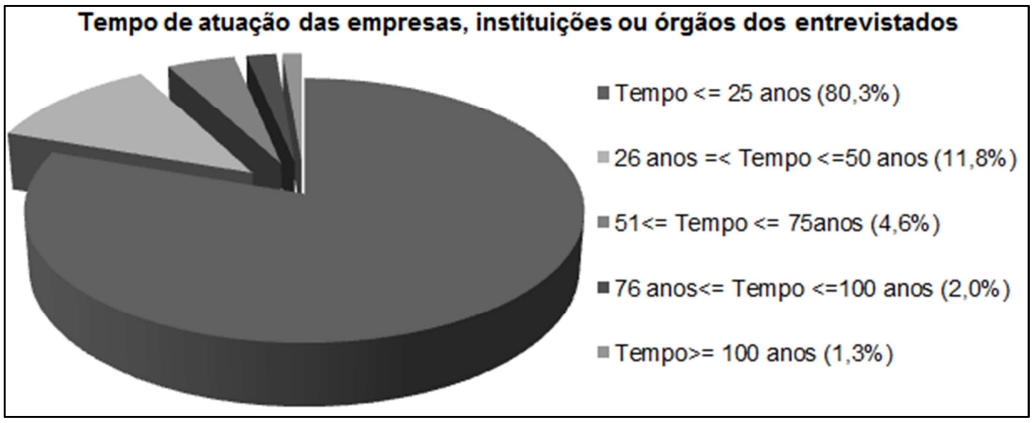

Gráfico 6.2: Tempo de mercado do ambiente de trabalho dos entrevistados

Observando os valores percentuais do Gráfico 6.1, nota-se que o cenário dominante em mais da metade dos entrevistados (50,6%), é característico por ser de pequeno porte, formado por no máximo 30 profissionais. No extremo desta parcela estão cenários típicos de grande porte, formados por mais de 1.000 integrantes, firmando-se com apenas 10% do valor total da amostra recolhida.

De acordo com o Gráfico 6.2, percebe-se que a maior parcela de profissionais interessados em adaptar os processos ágeis de software foi proveniente de entidades com até 25 anos de mercado. Os menos interessados em dispor suas opiniões estão engajados em ambientes que possuem mais de 100 anos de atuação, denominando-se um percentual de apenas 1,3% dos resultados.

Na segunda parte do questionário, foram disponibilizadas três questões. A **primeira questão da Parte II** buscou avaliar a opinião dos entrevistados acerca das principais causas, apontadas pelo autor nesta pesquisa, como responsáveis por gerarem os problemas nos processos que compõem o ciclo de vida de software. Nesta questão, foram determinadas duas variáveis para a análise, conforme pode ser verificado na seguinte descrição:

“Vários profissionais encontram-se despreparados atualmente para corrigir os problemas de adequação e execução nos seus processos de desenvolvimento. Pode-se afirmar que algumas das causas estão associadas à ineficiência de organização das atividades de gerência, assim como das atividades de engenharia de produção para o ciclo de vida do software. Você concorda que as ineficiências para o desenvolvimento do software provêm da:”

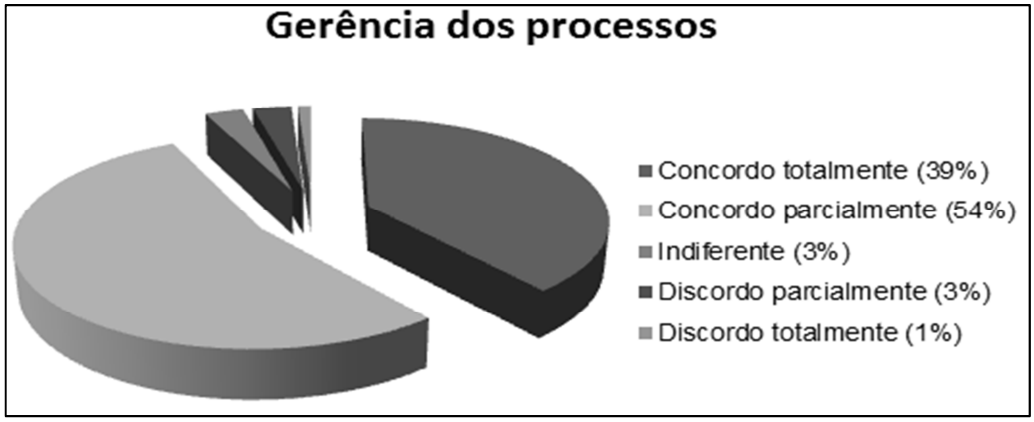

Gráfico 6.3: Opinião relativa à gerência como causa de problemas nos processos

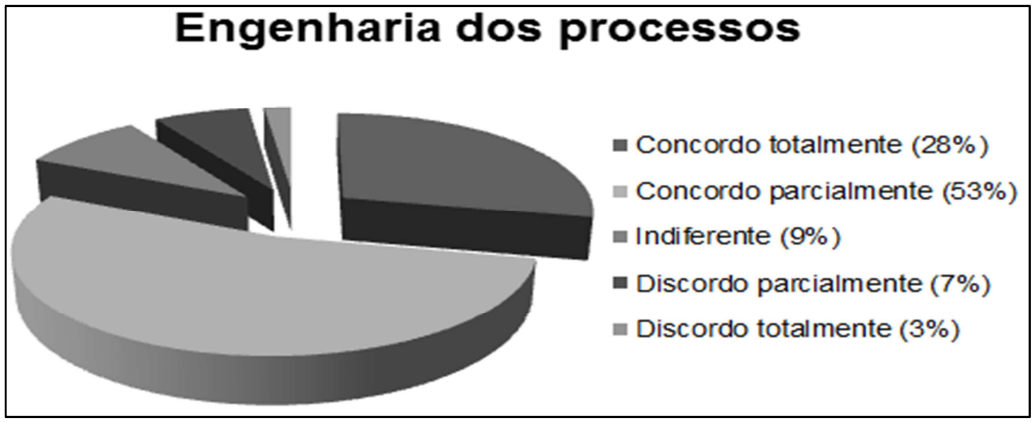

Gráfico 6.4: Opinião relativa à engenharia como causa de problemas nos processos

A **segunda questão da Parte II** buscou avaliar o percentual de aderência dos entrevistados acerca da prática da adaptação como uma solução estratégica para melhoria dos processos de software. Nesta questão, foram disponibilizadas três variáveis para análise, conforme pode ser verificado na sua descrição:

"Diante dos problemas citados na Questão 1, os profissionais recorrem à soluções alternativas que consistem em adaptar a arquitetura de seu processo principal com a inclusão de boas práticas provenientes de um segundo processo. Pode-se afirmar que é uma boa estratégia, desde que o processo suporte plenas condições para efetivar as mudanças. Você concorda coma a formação deste cenário que:"

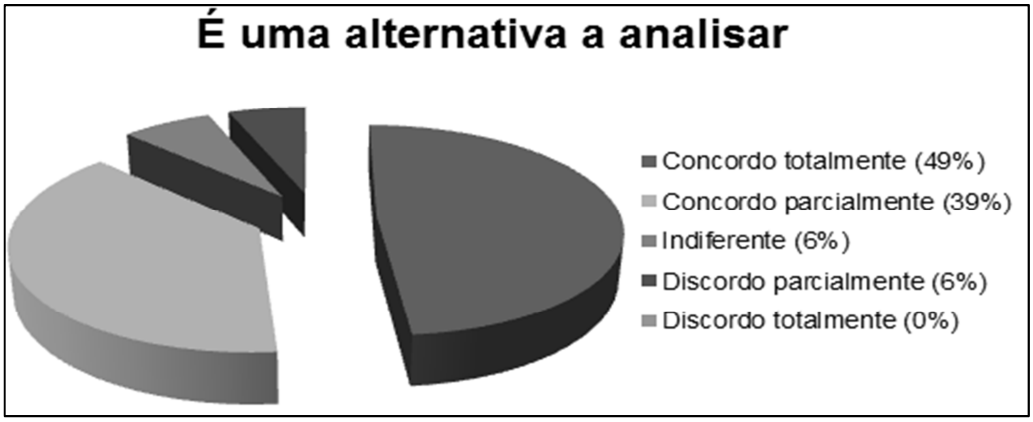

Gráfico 6.5: Opinião relativa à análise do tema de processos adaptados

Gráfico 6.6: Opinião relativa a analise do planejamento para o uso de processos adaptados

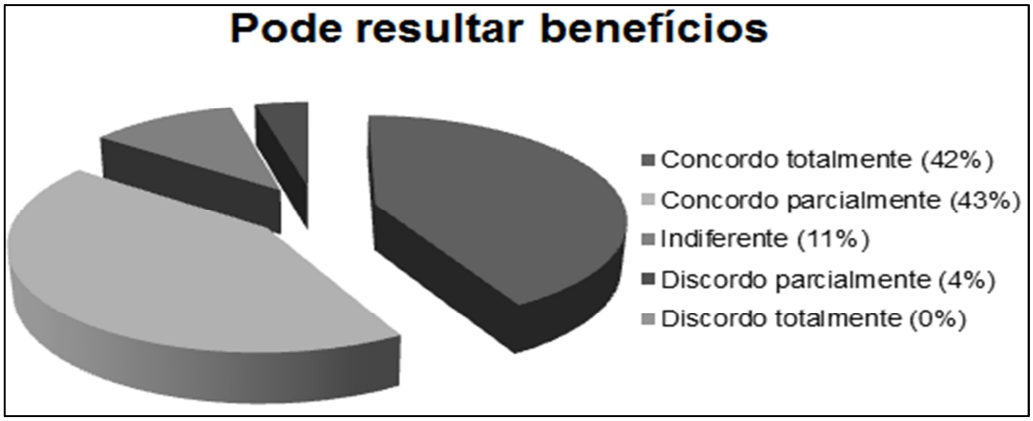

Gráfico 6.7: Opinião relativa aos benefícios do uso de processos adaptados

A **terceira questão da Parte II** buscou avaliar a aderência dos entrevistados acerca da proposta deste trabalho, que tem como objetivo dispor os guias para a implantação dos processos híbridos de software através de uma metodologia de implantação. Foram disponibilizadas três variáveis de análise, conforme pode ser verificado na sua descrição:

“A resistência dos profissionais relativa à prática da fusão dos processos de software é motivada pelo fato de não existirem meios tangíveis que norteiam tal atividade. Pode-se afirmar que uma solução prática seria a adoção de uma metodologia apropriada, estruturada através de passos sequenciais, e com um formato de guias instrucionais. Você concorda com a formação deste cenário que:”

Gráfico 6.8: Opinião sobre o uso de guias para a implantação de processos adaptados

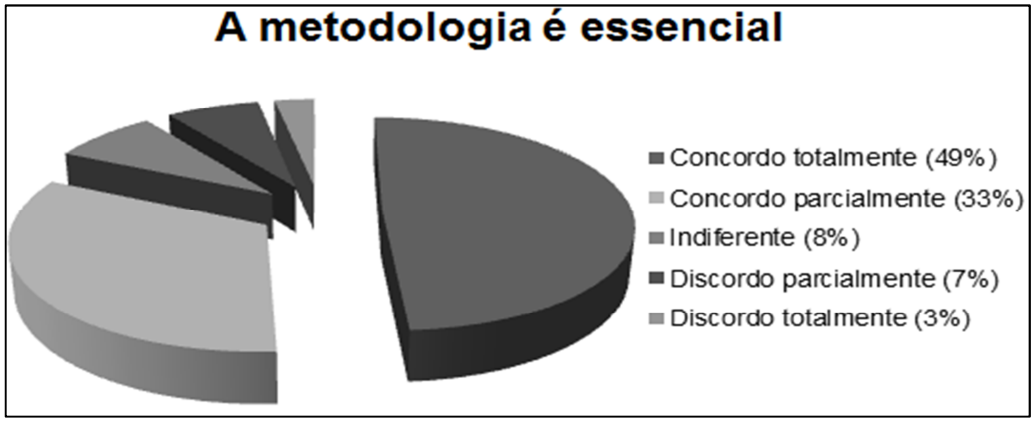

Gráfico 6.9: Opinião sobre a importância da metodologia nos guias de implantação

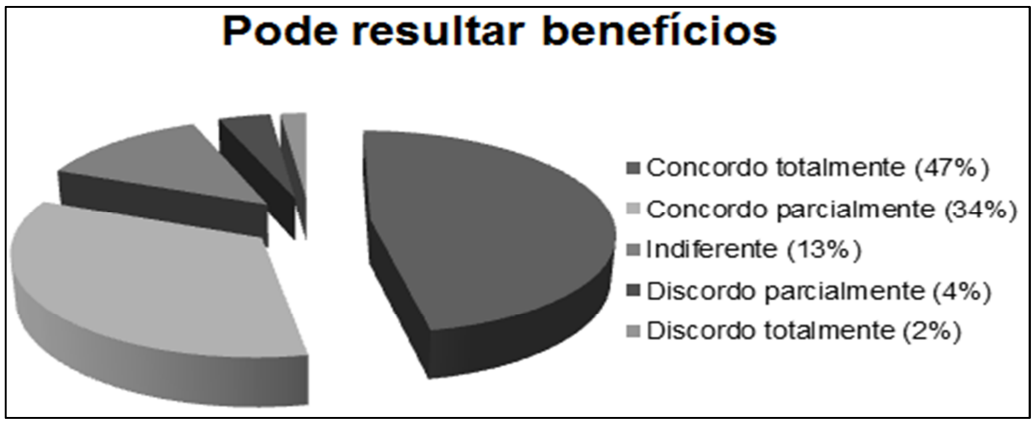

Gráfico 6.10: Opinião sobre os benefícios de utilizar os guias instrucionais para adaptar os processos

Analisando os resultados da “Parte II”, pôde-se comprovar a aderência dos entrevistados em relação à temática de hibridização nos processos de software. Também ficou comprovado que a ideia proposta possui um percentual aceitável de viabilidade para ser adotada como estratégia pelas empresas, instituições ou órgãos que promovam o desenvolvimento de software, visando a formar um ciclo de vida tangível através de processos combinados.

Por fim, na terceira parte do questionário, foram disponibilizadas dez questões. A **primeira questão da Parte III** teve como propósito avaliar a aderência dos profissionais acerca das boas práticas reunidas no guia do processo híbrido de implementação de software, conforme pode ser verificado na seguinte descrição:

“O processo de implementação de software (planejamento) é o primeiro no ciclo de vida do software. Seu propósito é fornecer os subsídios para a organização das atividades do projeto. Pode-se afirmar como boas práticas para um processo adaptado as seguintes:

(Verificar as práticas resumidas e enumeradas no Apêndice A, Parte III, Questão 1)

Você conclui acerca da validade das práticas que:”

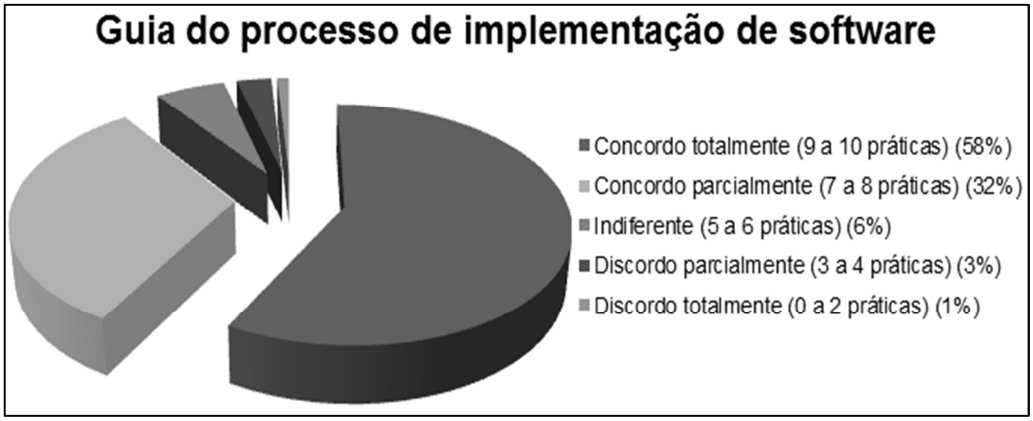

Gráfico 6.11: Aderência ao guia do processo híbrido de implementação de software

Também foi questionado aos entrevistados quais práticas poderiam ser descartadas. Dos 152 que responderam, 80 efetuaram pelo menos um descarte, o que equivale a 53% da amostra total. As três práticas mais descartadas foram: a de número 4, com 21,4%, a de número 5, com 16,7%, e a de número 10, com 14,9% de um total de 168 descartes.

A **segunda questão da Parte III** buscou analisar a aderência dos entrevistados acerca das boas práticas sugeridas no guia do processo híbrido de análise de requisitos de software, conforme pode ser verificado na sua descrição:

"O processo de análise de requisitos é o segundo processo essencial no ciclo de vida do software. Seu propósito é fornecer os subsídios para identificar os requisitos da solução proposta. Pode-se afirmar como boas práticas para um processo adaptado as seguintes:

(Verificar as práticas resumidas e enumeradas no Apêndice A, Parte III, Questão 2)

Você conclui acerca da validade das práticas que:"

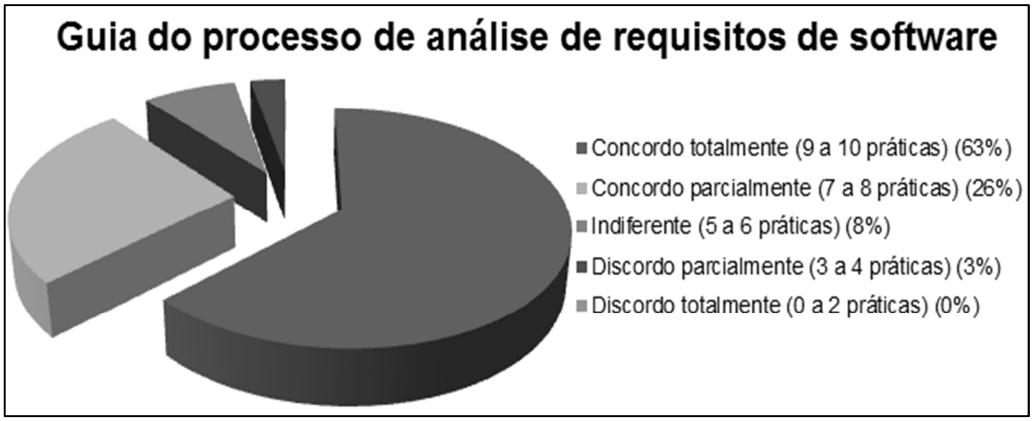

Gráfico 6.12: Aderência ao guia do processo híbrido de análise de requisitos de software

Seguindo-se o modelo da Questão 1, questionou-se quais práticas deste processo poderiam ser descartadas. Dos 152 profissionais que responderam, 64 efetuaram pelo menos um descarte, o que equivale a 42% da amostra total. As três práticas mais citadas foram: a de número 10, com 21,3%, a de número 4, com 16,9%, e a de número 9, com 14%, de um total de 136 descartes.

A **terceira questão da Parte III** teve como objetivo avaliar o percentual de aderência dos profissionais acerca das práticas combinadas no guia do processo híbrido de construção de software, conforme pode ser verificado na seguinte descrição:

“O processo de construção de software é o terceiro processo essencial do ciclo de vida do software. Seu propósito é fornecer subsídios para a codificação das soluções projetadas. Pode-se afirmar como boas práticas para um processo adaptado as seguintes:

(Verificar as práticas resumidas e enumeradas no Apêndice A, Parte III, Questão 3)

Você conclui acerca da validade das práticas que:”

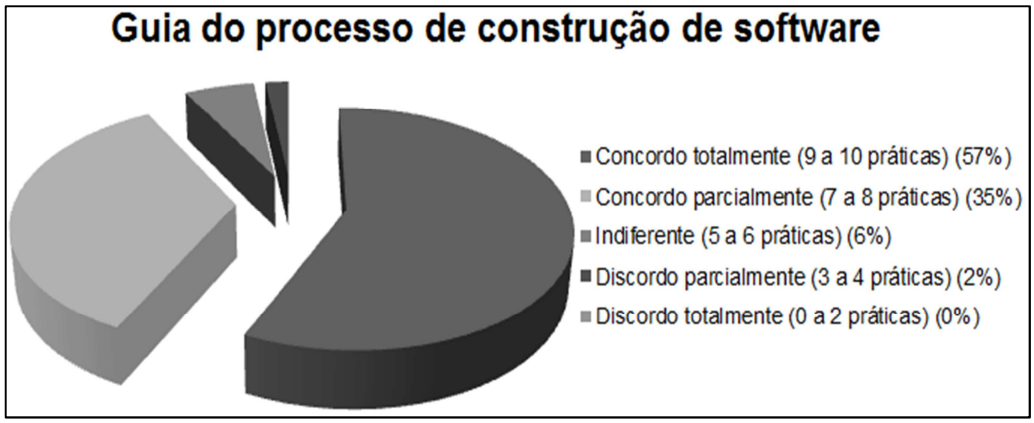

Gráfico 6.13: Aderência ao guia do processo híbrido de construção de software

Ainda foi questionado aos entrevistados quais práticas do guia eles poderiam descartar. Dos 152 que responderam, 69 efetuaram pelo menos um descarte, o que equivale a 45% da amostra total obtida. As três práticas mais mencionadas foram respectivamente: a de número 8, com 27,1%, a de número 6, com 23,3%, e a de número 10, com 12%, de um total de 133 descartes.

A **quarta questão da Parte III** analisou a aderência dos entrevistados acerca das boas práticas de adaptação propostas no guia do processo híbrido de integração de software, conforme pode ser verificado na sua descrição:

“O processo de integração de software é o quarto processo essencial no ciclo de vida do software. Seu propósito é fornecer os subsídios para integrar as unidades funcionais codificadas. Pode-se afirmar como boas práticas para um processo adaptado as seguintes:

(Verificar as práticas resumidas e enumeradas no Apêndice A, Parte III, Questão 4)

Você conclui acerca da validade das práticas que:”

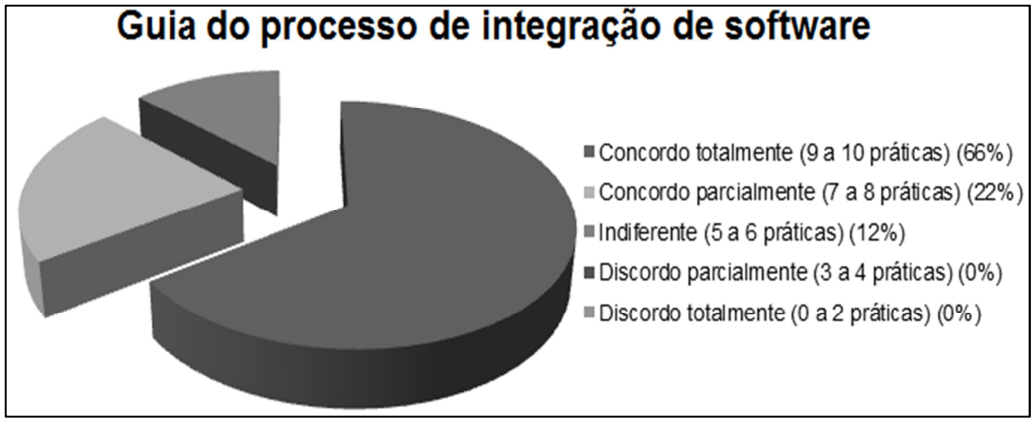

Gráfico 6.14: Aderência ao guia do processo híbrido de integração de software

Os entrevistados ainda foram consultados sobre quais práticas poderiam ser descartadas no guia deste processo. Dos 152 que responderam, 50 deles efetuaram pelo menos um descarte, o que equivale a 33% da amostra total. As três práticas mais citadas foram: a de número 9, com 19,8%, a de número 4, com 17,8%, e a de número 5, com 10,9% de um total de 101 descartes.

A **quinta questão da Parte III** avaliou a aderência dos profissionais acerca das boas práticas de adaptação recomendadas no guia do processo híbrido de qualificação e testes de software, conforme pode ser verificado através do seguinte enunciado:

“O processo de qualificação e testes de software é o quinto processo essencial no ciclo de vida do software. Seu propósito é fornecer subsídios de testes as unidades funcionais integradas. Pode-se afirmar como boas práticas para um processo adaptado as seguintes:

(Verificar as práticas resumidas e enumeradas no Apêndice A, Parte III, Questão 5)

Você conclui acerca da validade das práticas que:”

Gráfico 6.15: Aderência ao guia do processo híbrido de qualificação e testes de software

Além disso, questionou-se aos entrevistados quais práticas não são recomendadas para a formação de processos adaptados de qualificação e testes de software. Dos 152 que responderam, 45 efetuaram pelo menos um descarte, o que equivale a 30% da amostra da pesquisa. As três práticas mais mencionadas foram: a de número 9, com 20,2%, a de número 10, com 19,1%, e a de número 4, com 9,6% de um total de 94 descartes.

A **sexta questão da Parte III** possibilitou avaliar a aderência dos profissionais acerca das boas práticas sugeridas no guia do processo híbrido de gerência da documentação de software, conforme foi questionado através da seguinte descrição:

“O processo de gerência da documentação de software é o primeiro processo de suporte no ciclo de vida do software. Seu propósito é fornecer subsídios de gerência dos documentos produzidos nos demais processos. Pode-se afirmar como boas práticas para um processo adaptado as seguintes:

(Verificar as práticas resumidas e enumeradas no Apêndice A, Parte III, Questão 6)

Você conclui acerca da validade das práticas que:”

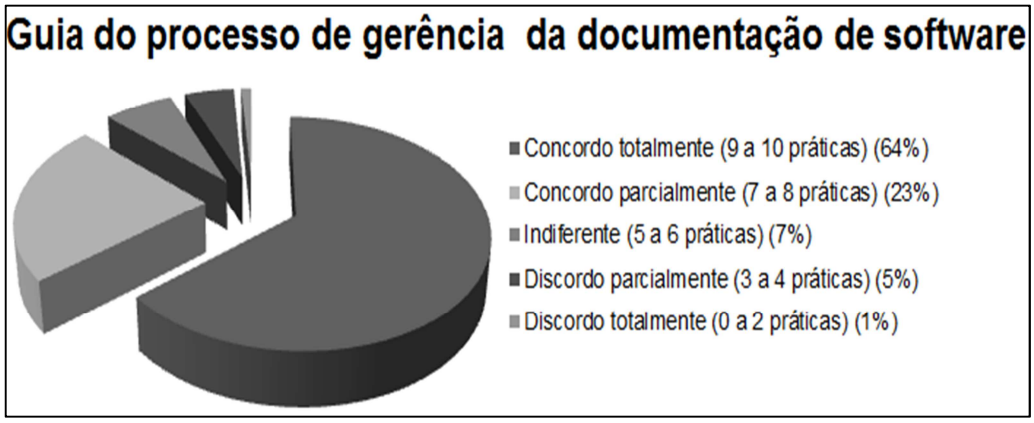

Gráfico 6.16: Aderência ao guia do processo híbrido de gerência da documentação de software

Indagou-se ainda aos entrevistados quais práticas deveriam ser descartadas do guia deste processo. Dos 152 que responderam a pergunta, apenas 56 efetuaram pelo menos um descarte, o que equivale a 37% da amostra obtida. As três práticas mais citadas foram respectivamente: a de número 5, com 18%, a de número 10, com 18%, e a de número 2, com 14,4% de um total de 139 descartes.

A **sétima questão da Parte III** analisou a aderência dos entrevistados acerca das boas práticas de adaptação reunidas no guia do processo híbrido de gerência da configuração de software, conforme pode ser verificado na seguinte descrição:

“O processo de gerência da configuração de software é o segundo processo de suporte do ciclo de vida do software. Seu propósito é fornecer os subsídios para gerir os arquivos do projeto. Pode-se afirmar como boas práticas para um processo adaptado as seguintes:

(Verificar as práticas resumidas e enumeradas no Apêndice A, Parte III, Questão 7)

Você conclui acerca da validade das práticas que:”

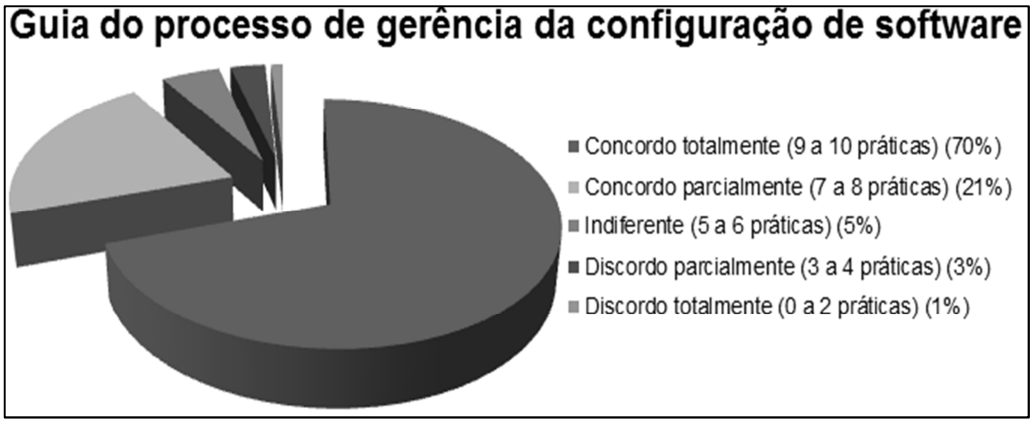

Gráfico 6.17: Aderência ao guia do processo híbrido de gerência da configuração de software

Assim como nas demais questões, perguntou-se aos entrevistados quais práticas poderiam ser descartadas do guia deste processo. Dos 152 que responderam, 39 mencionaram pelo menos uma prática, o que equivale a 26% da amostra total da pesquisa. As três mais descartadas foram: a de número 10, com 20%, a de número 4, com 15%, e a de número 5, com 13,8% de um total de 80 descartes.

A **oitava questão da Parte III** teve como objetivo avaliar o percentual de aderência dos profissionais entrevistados acerca das boas práticas inseridas no guia do processo híbrido de garantia da qualidade de software, conforme pode ser verificado na descrição do seu enunciado:

“O processo de garantia da qualidade de software é o terceiro processo de suporte no ciclo de vida do software. Seu propósito é fornecer os subsídios de garantias de funcionamento dos demais processos. Pode-se afirmar como boas práticas para um processo adaptado as seguintes:

(Verificar as práticas resumidas e enumeradas no Apêndice A, Parte III, Questão 8)

Você conclui acerca da validade das práticas que:”

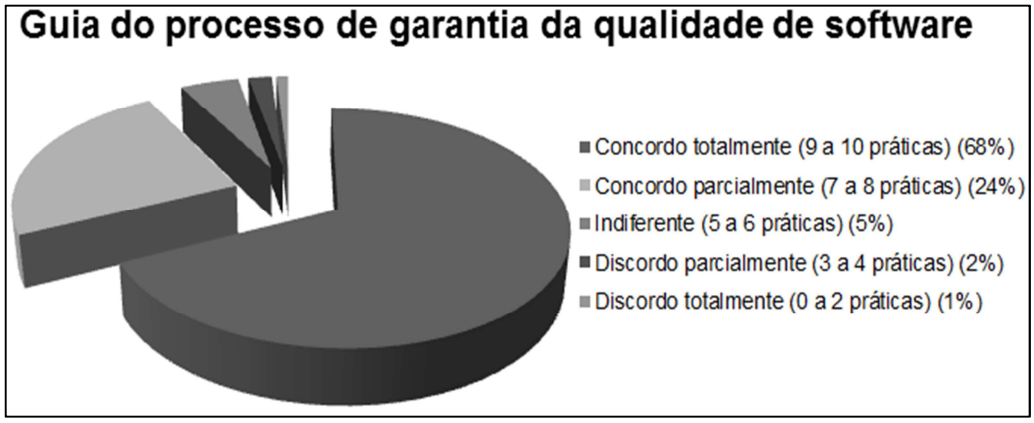

Gráfico 6.18: Aderência ao guia do processo híbrido de garantia da qualidade de software

Perguntou-se a todos os entrevistados quais seriam as práticas que os mesmos descartariam neste processo. Dos 152 que responderam, 38 efetuaram pelo menos um descarte, o que equivale a 25% da amostra. As três práticas mais citadas foram: a de número 7, com 18,8%, a de número 10, com 15%, e a de número 3, com 12,5% de um total de 80 descartes.

A penúltima e **nona questão da Parte III** buscou analisar a aderência dos entrevistados acerca das práticas combinadas no guia do processo híbrido de verificação de software, conforme foi questionado com a seguinte descrição:

“O processo de verificação de software é o quarto processo de suporte no ciclo de vida do software. Seu propósito é fornecer os subsídios de verificação das funcionalidades do software. Pode-se afirmar como boas práticas para um processo adaptado as seguintes:

(Verificar as práticas resumidas e enumeradas no Apêndice A, Parte III, Questão 9)

Você conclui acerca da validade das práticas que:”

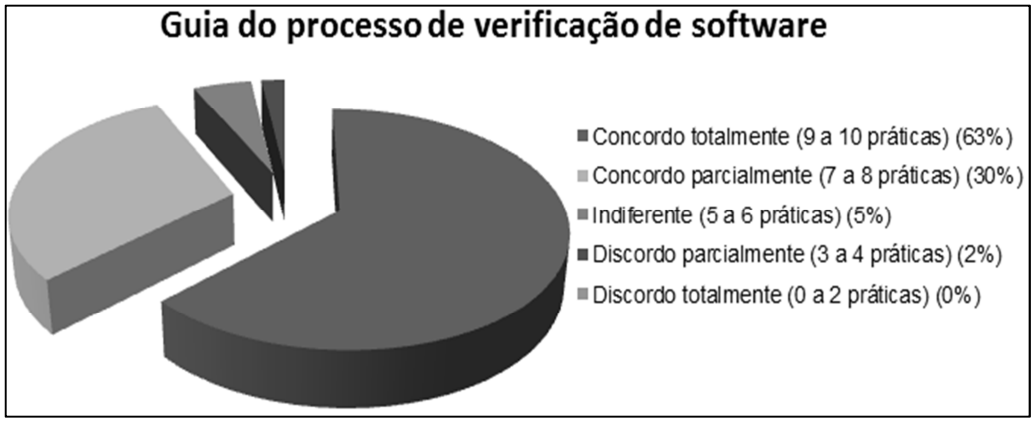

Gráfico 6.19: Aderência ao guia do processo híbrido de verificação de software

Questionou-se aos entrevistados quais práticas poderiam ser descartadas em um processo de verificação adaptado de software. Dos 152 que responderam, 55 mencionaram pelo menos uma prática, o equivalente a 36% da amostra obtida na pesquisa. As três práticas que obtiveram uma maior porcentagem foram respectivamente: a de número 5, com 15,4%, a de número 6, com 15,4%, e a de número 2, com 13,5% de um total de 104 descartes.

Por fim, a **décima questão da Parte III** propôs avaliar o percentual de aderência dos profissionais entrevistados acerca das boas práticas ágeis atribuídas no guia do processo híbrido de validação de software, conforme pode ser observado na seguinte descrição:

“O processo de validação de software é o quinto processo de suporte no ciclo de vida do software. Seu propósito é fornecer os subsídios de validação das funcionalidades do software. Pode-se afirmar como boas práticas para um processo adaptado as seguintes:

(Verificar as práticas resumidas e enumeradas no Apêndice A, Parte III, Questão 10)

Você conclui acerca da validade das práticas que:”

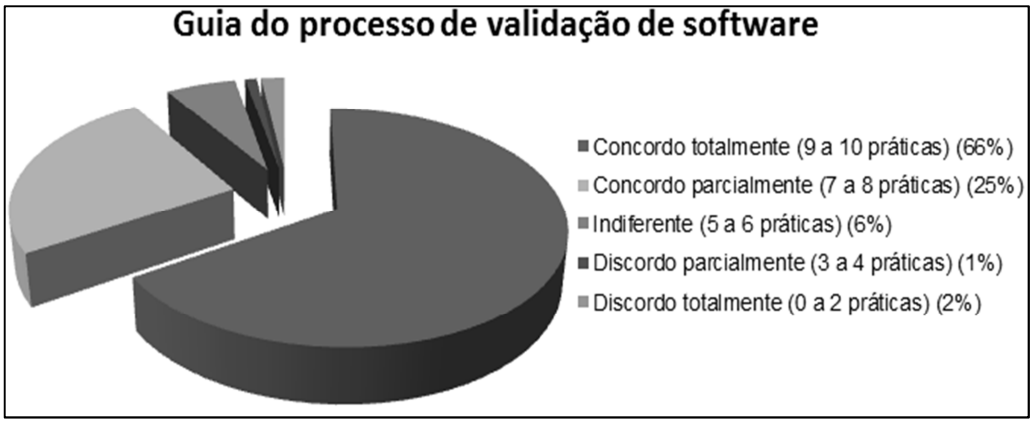

Gráfico 6.20: Aderência ao guia do processo híbrido de validação de software

Também foi questionado aos entrevistados quais seriam as práticas que poderiam ser descartadas do guia do processo de validação. Dos 152 que responderam, apenas 47 mencionaram pelo menos uma prática, o que equivale a 31% da amostra da pesquisa. As três práticas mais descartadas foram: a de número 8, com 14,7%, a de número 5, com 12,7%, e a de número 7, com 12,7%, de um total de 102 descartes.

É importante ressaltar que não foi possível comprovar o intervalo de confiança da amostra, assim como seu nível de confiabilidade. A estatística foi baseada em uma combinação matemática simples, onde foram contabilizados os valores da amostra total, e de descarte dos entrevistados, e as frequências total e individual de descarte de cada prática, para que se tornasse possível definir os percentuais de aceitação e rejeição dos guias.

Ao longo das análises são citadas apenas as três práticas mais descartadas. As frequências e demais porcentagens de descarte podem ser observadas no Apêndice B (Guias do processo de implementação de software) e no Apêndice C (Guias dos processos de suporte de software), de acordo com a classificação estipulada para as categorias dos processos.

6.4 Considerações finais

A partir da avaliação realizada neste capítulo, foi possível inferir algumas conclusões importantes, referentes à expansão da temática discutida, da proposta de trabalho e da aderência dos guias e práticas sugeridas.

Em relação aos resultados da Parte I, algo que se esperava apenas se confirmou. Os profissionais que despertam mais interesse pela adaptação dos processos ágeis de software são provenientes de cenários menores, formados por no máximo 25 integrantes. A resistência é mais aparente onde está concentrado um maior número de envolvidos, sendo empresas, instituições ou órgãos que possuam um tempo maior de consolidação no mercado, por conta do esforço que precisa ser arcado para prover a adaptação das hierarquias mais rígidas.

Tornou-se claro, com a avaliação da Parte II, que os profissionais apontam que a causa mais comum responsável por gerar problemas no ciclo de vida do software está relacionada principalmente com gestão ineficiente dos processos. Isto significa que a falta do alinhamento das atividades pode impactar diretamente na possibilidade de alcançar bons resultados, quando não estão sendo atribuídas soluções plausíveis mediante as necessidades encontradas.

A aceitação da temática de processos adaptados obteve resultados relevantes. De acordo com a avaliação, a prática de combinar processos diferentes reporta-se como uma demanda existente para o mercado de software, sendo uma estratégia que vem sendo bem aceita para solucionar os problemas de maneira rápida e eficiente. A proposta de guias foi considerada pelos profissionais como uma contribuição fundamental, que fornece os subsídios necessários para tornar os processos mais flexíveis.

Por fim, as últimas conclusões firmadas sobre os resultados obtidos com a avaliação realizada através da Parte III dão conta de que a maior aderência dos guias híbridos foi proveniente dos processos de suporte de software. Percebeu-se que as práticas mais descartadas foram as dos processos de implementação, tais como análise, codificação, testes etc., pelo fato de a adaptação dos ambientes sempre variar de acordo com o modelo de negócios, tipo ou abrangência no mercado.

Capítulo

7

Conclusões

Este capítulo apresenta as conclusões inferidas a partir da realização da pesquisa, com uma visão geral sobre as contribuições alcançadas. São listadas as limitações sobre o desenvolvimento da pesquisa e as restrições características com sua aplicação. Por fim, são discutidas as perspectivas de expansão da proposta, e as mudanças que podem ser viabilizadas no futuro para melhorar os guias de implantação dos processos híbridos de software.

7.1 Contribuição da pesquisa

O avanço das pesquisas sobre os processos híbridos de software tem contribuído de maneira significante para a melhoria do ciclo de vida do software ao longo dos últimos anos. A certeza de que a fusão não só pode prover “processos bons”, mas sim, “processos grandiosos”, subsidia atrair mais profissionais a aceitarem os desafios de utilizar esta estratégia, que tem sido considerada como tendência para o mercado de software (JAKOBSON e SUTHERLAND, 2009).

A contribuição almejada por este trabalho foi a de habilitar os gerentes de projetos e os demais componentes das equipes a adaptarem os seus processos ágeis de software. Em um âmbito de que a grande dificuldade encontrada é substituir as rotinas originais do processo, a melhoria pode está assegurada através de um aprendizado contínuo, possível de ser adquirido a cada nova implementação, desde que o planejamento prévio seja efetivado como é proposto com o uso dos guias.

Espera-se que esta dissertação sirva também como estímulo para que os profissionais da mesma área concebam trabalhos semelhantes e deem continuidade de expansão na pesquisa. A base estrutural de qualquer ciclo de vida é formalizada com a especificação de

“como se deve iniciar”, referindo-se a presente proposta de implantação, para que depois possam ser dados os próximos passos com a síntese de guias com foco para a melhoria de processos híbridos de software, maturidade de processos híbridos de software, além de várias áreas relacionadas que podem ser desmembradas.

Considerando a estrutura do trabalho, contextualização sobre o tema, revisão literária, o mapeamento realizado, e por fim, os guias apresentados ao longo das seções, ainda podem ser listados como fontes de contribuição desta pesquisa:

- Disponibilizar os resultados deste trabalho para que sirvam como base de dados e referência para o surgimento de novas pesquisas quantitativas, cito a adaptação da metodologia de pesquisa, o tratamento dos dados, as limitações, conclusões etc.
- Apresentar aos leitores os trabalhos relevantes relacionados com a área de adaptação dos processos de software, cito livros, artigos, e periódicos, com seus principais autores e/ou defensores, e as ideias de autoria do presente autor sugeridas na pesquisa.
- Transparecer aos profissionais resistentes ao uso de padrões, que a síntese de soluções simples para os problemas nos processos de software podem ser realizadas com o uso de normas da ISO®, cito a ISO/IEC e IEEE 12207 (ISO, 2008a) e adjacentes.
- Comprovar que as metodologias ágeis possuem estruturas, e objetivos diferentes quanto ao atendimento das necessidades com base na gerência e engenharia para a produção, com o *Scrum* (SCHWABER, 2007), *eXtreme Programming* (BECK, 1999), e *Feature Driven Development* (PALMER e FELSING, 2002), e que suas atividades podem ser combinadas para formar ciclos de vida alternativos de software.

Além de tudo que foi citado, pode-se apresentar como contribuição principal um conjunto de boas práticas distribuídas ao longo de dez processos, através de guias instrucionais de livre acesso, que podem ser utilizadas em projetos de diferentes portes, sem relevar padrões, tecnologias, ou pessoas para seu uso.

7.2 Limitações da pesquisa

Ao longo do desenvolvimento da pesquisa, as dificuldades encontradas para garantir sua validade foram sendo contornadas pelo autor. Entretanto, como quaisquer trabalhos acadêmicos, as limitações foram contabilizadas para que se tonasse possível delimitar seu campo de aplicação, destacando-se principalmente como foco a proposta dos guias.

Para facilitar esta visualização, foram listadas como principais limitações da pesquisa:

- A impossibilidade de associar quais são os processos que possuem maior demanda pelos profissionais para serem formatados através de guias para a implantação híbrida.
- A impossibilidade de especificar quais hierarquias (pequena, média ou grande porte) de projeto são recomendadas para que os resultados alcançados sejam mais efetivos.
- Comparar e recomendar quais metodologias ágeis utilizadas no mercado estariam melhor susceptíveis a realizarem a prática da adaptação (VERSIONE, 2010).
- Mensurar estatisticamente a eficiência de uso dos guias propostos no trabalho, com a aplicação das boas práticas em um projeto real para o desenvolvimento de software.
- Limitar a pesquisa a apenas uma amostra, com a participação de engenheiros de software dispersos em um universo que possui uma forte heterogeneidade, como é o caso das regiões do Brasil (AGILCOOP, 2011).
- Delimitar quais práticas estariam recomendadas para serem atribuídas aos respectivos cargos que atuam no ciclo de vida do software, de acordo com os perfis identificados.
- Por fim, disponibilizar uma maneira de predição, ou medição, de quais seriam os resultados possíveis de serem alcançados com a aplicação dos guias, de maneira que os profissionais pudessem obter uma forma de prever a eficiência das práticas sugeridas.

7.3 Trabalhos futuros

A evolução dos guias de processos híbridos de software desta dissertação está enquadrada como meta principal a ser alcançada no futuro. O desenvolvimento da metodologia, sua estruturação em quatro passos e a quantidade de práticas contidas ao longo das seções do capítulo 5, remetem às perspectivas de que este trabalho possa ser transformado e reconhecido como um *framework*.

Para que se torne possível atingir essa meta, é necessário inicialmente que as limitações sejam avaliadas e possivelmente traçadas como objetivos para a realização deste projeto. De fato, a melhoria nos guias demanda um estudo mais profundo que não só depende dos fatores técnicos agregados ao ciclo, como os processos e atividades, mas sim, de uma

análise categórica dos fatores não técnicos que influenciam a infraestrutura organizacional, como os perfis de atuação dos profissionais, as tendências de mercado por regiões etc.

O que se conclui como perspectiva a ser desempenhada no futuro é veicular uma forma de alinhar ambos os fatores (técnicos e não técnicos) para identificar quais processos se tornam universais nos cenários de desenvolvimento de software para que se possa construir o *framework*. Neste caso, poderão ser providas análises sequenciais que estejam direcionadas para os variados campos de atuação no mercado pela qual os guias podem abranger.

Portanto, para dar continuidade à expansão dessa proposta, estão listados como trabalhos futuros:

- Aplicar os guias em vários projetos, de diferentes empresas, instituições ou órgãos que promovem o desenvolvimento de software, realizando uma análise comparativa da efetividade dos resultados das práticas em cada um deles.
- Realizar um levantamento das práticas mais utilizadas pelos profissionais para solucionar os problemas de gerência em processos de pequeno, médio e grande porte, e dentre elas as que remetem os melhores resultados; o mesmo para as práticas de engenharia.
- Realizar um levantamento acerca dos fatores externos que motivam os profissionais a combinarem os processos, metodologias, ou *frameworks*, por exemplo, com a realização de pesquisas ou revisões sistemáticas.
- Realizar um levantamento acerca das principais dificuldades encontradas pelos profissionais para manter um ciclo de vida híbrido eficiente, e quais as decisões mais comuns as serem tomadas para corrigir os problemas.
- Expandir a pesquisa de campo em um âmbito de sua aplicação, cito em empresas, instituições ou órgãos que promovem o desenvolvimento de software nacional ou internacionalmente.
- Por fim, efetivar a adoção de uma ou mais técnicas estatísticas que reportem comprovar com maior precisão todas as hipóteses e análises realizadas em futuras implantações dos guias.

Referências

ABBAS, N..; GRAVELL, A.; WILLS, G..; **Historical Roots of Agile Methods: Where did "Agile Thinking" Come from?.** School of ECS of University of Southampton, 2008.

ABNT. ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISO/IEC 12207, Tecnologia de informação – Processos de ciclo de vida de software**. Brasil, 1998.

ABNT. ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, **NBR ISO/IEC 12207, Tecnologia de informação – Processos de ciclo de vida de software**. Brasil, 2009

ABRAHAMSSON, Pekka; WARSTAB, Juhani; SIPONEN, Mikko T.; RONKAINEN, Jussi; **New Directions on Agile Methods: A Comparative Analysis**, ICSE, 2003.

AGILCOOP. **Métodos Ágeis no Brasil: Estado da prática em times e organizações**. Relatório Técnico do Departamento de Ciência da Computação, UME/USP, São Paulo, 2011.

AGILE MANIFESTO. **Manifesto for Agile Software Development**. 2012. Disponível em:< http://www.agilemanifesto.org> Acesso em: 4 Jan. 2012.

AMBLER, Scott. **Agile Modeling: Effective Practices for Extreme Programming and the Unified Process**. New York: Wiley, 2002.

ANDERSON, D. J.; **Comparing eXtreme Programming and Feature Driven Development in Academic and Regulated Environments**. Harvard University Library, 2004.

BASILI, V., ZAZWORKA, N., SHULL, F.; **Tool supported detection and judgment of nonconformance in process execution**. ESEM, p. 312-323, 2009.

BECK K. **Programação Extrema Explicada**. Ed. Bookman, São Paulo, 1999.

BOEHM, Barry; TURNER Richard; **Management Challenges to Implementing Agile Processes in Traditional Development Organizations**. IEEE Library, p. 30-37, Set. 2005.

BORGES Junior, A.; SOUZA, H.; ALVES, P.; ROCHA, R.; MEIRA, S.; **Uma Experiência no Gerenciamento Ágil de Projetos de Software com Equipes Grandes e Distribuídas**. Anais do Workshop de Gerenciamento de Processos Ágeis (SBQS), São Paulo, 2011.

BUENO, Silveira. **Minidicionário da língua portuguesa**. São Paulo: FTD, 2000.

BUGLIONE, L.; ABRAN, A.; **Improving Estimations in Agile Projects: Issues and Avenues**. 4th Software Measurement European Forum, 2007.

CAMPELO, Renata Endriss Carneiro. **XP-CMM2: Um Guia para Utilização de Extreme Programming em um Ambiente Nível 2 do CMM**. Dissertação de Mestrado do Centro de Informática da Universidade Federal de Pernambuco, 2003.

CHO, J.; **A Hybrid Software Development and Method for Large Scale Projects: Rational Unified Process with SCRUM**. Colorado State University, Pueblo, 2009.

COAD, Peter; LEFEBVRE, Eric; DELUCA, Jeff. **JAVA Modeling Color with UML: Enterprise Components and process**. Prentice Hall; 1a ed., 1999.

COHEN, Stephen J.; MONEY, William H.; Bridge Methods: **Complementary Steps Integrating Agile Development Tools & Methods with Formal Process Methodologies**. Proceedings of the 41st Hawaii International Conference on System Sciences, 2008.

COX, Brad. **Planning the Software Industrial Revolution**. Stepstone Corp®, Nov. 1990.

DeLUCA, Jeff. **The Original FDD Processes**, 1998. Disponível em:<http://www.nebulon.com/articles/fdd/originalprocesses.html> Acesso em 14 Abr. 2012.

DeLUCA, Jeff. **Feature Driven Development Overview**. Nebulon Pty Corporation®, 2005.

DONAIRES, Omar Sacilotto. **Uma abordagem sistêmica ao mapeamento e melhoria do processo de desenvolvimento de software**. Centro Universitário de Franca. Revista FACEF Pesquisa, v.12, n.2, 2008.

FERREIRA, R.; LIMA, F.; **Metodologias Ágeis: Um Novo Paradigma de Desenvolvimento de Software**. II WOSES, p.107-114, Vila Velha, 2006.

FIGUEIREDO, Alexandre M.; **FDD em uma casca de banana. Um guia de rápido aprendizado para a Feature Driven Development**, 2007.

FOWLER, Martin. **The New Agile Methodology**. 2001. Disponível em: <http://www.martinfowler.com/articles/newMethodology.html> Acesso em: 12 Mar. 2012

HIGHSMITH, Jim. COCKBURN, Alistair; **Agile Software Development: The Business of Innovation**. Published in IEEE Library, 2001.

FUGGETTA, Alfonso. **Software Process: A Roadmap**. Future of Software Engineering Limerick Ireland Copyright, ACM, 2000.

GHEORGE, M.; HOLCOMBE, M.; SYED-ABDULLAH, S.; **The Impact of an Agile Methodology on the Well Being of Development Teams**. Citeseerx Libray, 2006.

GIUFFRA, C.E., VILAIN, P.. **Modelagem da Interação do Usuário no Desenvolvimento Ágil**. Departamento de Informática da Universidade Federal de Santa Catarina. 2010. Disponível em: <http://periodicos.unesc.net/index.php/sulcomp/article/viewFile/321/328> Acesso em: 27 Jan. 2012.

GRAHAN, Ian; SELLERS, Brian Henderson; YOUNESSI, Houman. **The OPEN Process Specification**. Addison Wesley, United Kingdom, 1997.

GONZÁLEZ, Carlos Sánchez. **ONess: un proyecto open source para el negocio textil mayorista desarrollado con tecnologías open source innovadoras**. Facultade de Informática - Departamento de Tecnoloxías da Información e as Comunicacións. Universidad de la Coruña, 2004.

HIGHSMITH, Jim; **Agile Project Management: Agile Project Management: Creating Innovative Products**. 2 ª ed.: Addison, Wesley, Boston, 2010.

HOLL, Rodrigo; **Metodologia ágil de desenvolvimento de software FDD**, 2010.

HOYER, C., CHROUST. G., **Evolving standard process reference models for product line development**. Proceedings of the 32nd Euromicro Conference on Software Engineering and Advanced Applications, pages 320–327, 2006.

HUMPHREY, Watts S.; **Characterizing the Software Process: A Maturity Framework.** USA 1988.

IBM. **Rational Unified Process Best Practices for Software Development Teams**, USA, 1998.

IBM. **Improving Software Economics: Top 10 Principles of Achieving Agility at Scale**. Rational Software Improving Software Economics Whitepaper, May., 2009.

IEEE. INSTITUTE OF ELETRICAL AND ELECTRONICS ENGINEERS. **What are standards?**. 2012. Disponível em <http://standards.ieee.org/develop/overview.html> Acesso em: 05 fev. 2012.

IRRAZABAL, E.; VÁSQUEZ, F.; DIAZ, R.; GARZÁS, J.; **Applying ISO/IEC 12207:2008 with SCRUM and Agile Methods.** Communications in Computer and Information Science, Volume 155, Parte 5, p. 169-180, 2011.

ISO. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, **Joining In. Participating in Internacional Standardization**. Suíça 2007. Disponível em: < http://www.iso.org/iso/joining_in_2007.pdf > Acesso em: 01 fev. 2012.

ISO. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **Systems and software engineering cycle ISO/IEC 12207 Processes**, Suíça, 2008a.

ISO. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **Systems and software engineering cycle ISO/IEC 15288 Processes.** Suíça, 2008b.

ISO. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **About us.** 2012. Disponível em: <http://www.iso.org/iso/home/about.html> Acesso em: 14 fev. 2012.

JAKOBSEN, C. R.; SUTHERLAND, Jeff.; **Scrum and CMMI – Going from Good to Great Are you ready-ready to be done-done?**. Scrum Training Institute, 2009.

KANWAL, F.; JUNAID, K.; FAHIEM, M.; **A Hybrid Software Architecture Evaluation Method for FDD: An Agile Process Model**. CiSE. p.1-5, 2010.

KERLINGER, Fred; **Metodologia da pesquisa em ciências sociais: um tratamento conceitual**, E.P.U/EDUSP, São Paulo, 1980.

KNIBERG, Henrik; **Scrum e XP direto das trincheiras: Como nós fazemos Scrum**. C4Media Inc®, 2007.

KOSCIANSKI, André; SOARES, Michel dos Santos. **Qualidade de Software: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software**. 2ª ed.: Novatec, São Paulo, 2007.

LAKEY, Peter; A **Hybrid Software Process Simulation Model for Project Management. Cognitive Concepts**, 2003.

LARMAN, Creg; **Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process**. Prentice Hall, 2a edition, 2001.

LAYMAN, L., WILLIAMS, L., DAMIAN, D., BURES, H. **Essential communication practices for Extreme Programming in a global software development team**. Information and Software Technology, 48, p. 781-794, 2006.

LIMA, R. S.; **Estudos Empíricos em Engenharia de Software: Avaliando o Benchmarking Framework Através de um Survey e Replicando uma Avaliação de Desempenho**. Dissertação de Mestrado do Centro de Informática da Universidade Federal de Pernambuco, 2010.

LIKERT, R.; **A Technique for the Measurement of Attitudes**. Archives of Psychology, p. 1-55, 1932.

LYCETT, Mark; MACREDIE, Robert D.; PATEL, Chaitali.; PAUL, Ray J.; **Migrating Agile Methods to Standardized Development Practice**. IEEE Library, p. 79-84, Set. 2003.

MAGDALENO, A., WERNER, C., ARAUJO, R.; **Revisão Quasi-Sistemática da Literatura: Conciliação de processos de desenvolvimento de software**. Programa de Engenharia de Sistemas de Computação da COPPE-UFRJ. Rio de Janeiro, 2009.

MARÇAL, Ana Sofia; **SCRUMMI: Um processo de gestão ágil baseado no SCRUM e aderente ao CMMI**. Dissertação de Mestrado da Universidade de Fortaleza, 2007.

MARCONI, M. e LAKATOS, E. **Técnicas de Pesquisa**. 7ª ed. São Paulo, Atlas, 2009.

MATTIELO-FRANCISCO, M. F.; **Verificação e validação na terceirização de software embarcado em aplicações espaciais**. Simpósio Brasileiro de Qualidade de Software, 2006.

NASIR, M. H.; AHMAD, R.; HASSAN, N.; **Resistance Factors in the Implementation of Software Process Improvement Project in Malaysia**. Journal of Computer Science, v.4, n.3 p 221-219, 2008.

NERUR, S.; BALIJEPALLY, V.; **Theoretical Reflections on Agile Development Methodologies**. Communications of the ACM Library, p.79-83, 2007.

NUNES, V.; MAGDALENO, A.; WERNER, C.; **Modelagem de contexto sobre o domínio de processos de desenvolvimento de software**. COPPE/UFRJ, 2010.

OLIVEIRA FILHO, A.; ALVES, A.; **Proposta de Processo SCRUM utilizando conceitos da Gestão do Tempo definida pelo PMBOK**. Pontifícia Universidade Católica de Goiás, 2010.

OSTERWEIL, Leon; **Software Processes are software too**. University of Colorado Boulder, Colorado, 1987.

PALMER, Stephen.; FELSING, John Mac.; **Practical Guide to Feature-Driven Development**. Ed.: Prentice-Hall, 2002.

PATEL, Jinalben; LEE, Roger; KIM, Haeng-Kon; **Architectural View in Software Development Life-Cycle Practices**. 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS), 2007.

PENNOTTI, M.; PYSTER, A.; **Developing and validating a framework for integrating systems and software engineering**. 3rd Annual Systems Conference of IEEE, 2009.

PINSONNEAULT, A., KRAEMER, K., **Survey research methodology in management information systems: an assessment**. Journal of Management Information System, v. 10, n. 2, p. 75-105, 1993.

PRESSMAN, Roger S.; **Engenharia de Software**. 6ª ed.: McGraw-Hill, São Paulo, 2006.

RAMACHANDRAN, Muthu; **Guidelines Based Software Engineering for Developing Software Components**. Journal of Software Engineering and Applications, 5, 1-6. 2012.

RAMAKRISHNAN, Sita.; **On Integrating Architecture Design into Engineering Agile Software Systems.** Informing Science and Information Technology, Vol. 7, p. 10-15, 2010.

RETAMAL, A. Muniz; **Feature Driven Development: Agilidade com Responsabilidade**. Borland® Corporation, 2005.

RITCHIE J.; **Design and Adoption of a Web Application Development Methodology**. Applied Project for Master of Business Administration Athabasca University, 2009.

ROCHA, Rodrigo; ARCORVERDE, Daniel; BRITO, Rebeka; ARÔXA, Bruno; COSTA, Catarina; SILVA, Fabio; ALBUQUERQUE, Jones; MEIRA, S.L.R.; **Uma Experiência na**

Adaptação do RUP em Pequenas Equipes de Desenvolvimento Distribuído. II Workshop de Desenvolvimento Distribuído de Software – WDDS, Campinas, 2008.

RODRÍGUEZ, Laura C.; MORA, Manuel.; ALVAREZ, Francisco J.; **A Descriptive/Comparative Study of the Evolution of Process Models of Software Development Life Cycles (*PM-SDLCs*)**, p. 298-301, 2005.

RORIZ FILHO, Heitor; **Scrum é um framework de gerenciamento de projetos**. Abril/2011. Disponível em:< bit.ly/RC7eW3> Acesso em: 16 Mar. 2012.

RYCHLÝ, M.; TICHÁ, P.; **A Tool for Supporting Feature-Driven Development**. Department of Information and Technology of Brno University. 2007.

SALGADO, A.; MELCOP, T.; ACCHAR, J.; REGO, P.; IRIGOYEN, A.; FERREIRA, F.; KATSURAYAMA, E.; MONTONI, M.; ZANETTI, D.; **Aplicação de um Processo Ágil para Implantação de Processos de Software baseado em Scrum na Chemtech**. Anais do IX Simpósio Brasileiro de Qualidade de Software, 2010.

SEI. SOFTWARE ENGINEERING INSTITUTE. **CMMI-SE/SW, v1.1 Staged Representation**. Carnegie Mellon University, 2001. Disponível em: < http://www.sei.cmu.edu/pub/documents/02.reports/pdf/02tr002.pdf Acesso em: 02 Mar. 2012.

SCHWABER, Ken; **Agile Project Management with Scrum**. 1ª ed. Washington: Microsoft Press, Washington, 2004.

SCHWABER, Ken; **The Enterprise and Scrum** 1ª ed.: Microsoft Press, Washington 2007.

SIAKAS, Kerstin V.; GEORGIADOU Elli.; **Agile Methodologies and software Process Improvement**. Conference on Computer Science and Information Systems, 2005.

SFETSOS, P.; STAMELOS, I.; **Empirical Studies on Quality in Agile Practices: A Systematic Literature Review**. Seventh International Conference on the Quality of Information and Communications Technology, 2010.

SOARES, M.; **Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software**. Departamento de Ciência da Computação UFBA, Salvador, 2004.

SOFTEX. **Melhoria de Processo do Software Brasileiro Guia Geral MPS.BR**, 2011.

SMOJVER, K.; BELANI, H.; CAR, Z.; **Building a Hybrid Process Model for a Complex Software System Integration**. Faculty of Electrical Engineering and Computing of University of Zagreb, 2009.

SOMMERVILLE, Ian; SAWYER, Peter; **Requirements Engineering: A Good Practice Guide.** John Wiley and sons Ltd, USA, 1997.

SOMMERVILLE, Ian. **Engenharia de Software**. 9 ed.: Pearson, São Paulo, 2011.

SOUZA, Adriana Silveira; OLIVEIRA, Juliano Lopes; **Experiências de Implantação de Processo de Software em Goiás**. Pro-Qualiti e SOFTEX do Brasil, 2006.

TELES, Vinícius M.; **Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade**. Novatec Editora, 2006.

TIAN, D.; **A Test-Driven Web application model based on layered approach**. International Conference on Information Theory and Security (ICITIS), p.160-163, 2010.

VASCO, Carlos; VITHOFT, Marcelo; ESTANTE, Paulo. **Comparação entre Metodologias RUP e XP**. Pontifícia Universidade Católica do Paraná (PUCPR), 2005.

VILAÇA, Luísa Lemos; MACHADO, Marcelly Manhães; CARVALHO, Marcos Philipe Ribeiro; AMOY NETTO, Thiago; PEREIRA FILHO, Z. Ribeiro; MACIEL, Gustavo da Silva; **Anomalias no processo de operações portuárias: Atividade pré-embarque**. Revista Científica Perspectivas Online, Volume 1, número 1, 2011.

VISÃO ÁGIL. **Agile e MPS.BR Case de implantação de processos ágeis com MPS.BR**. Ano II, 1ª ed., p. 10-17, 2007.

VERSIONONE, **Agile made easier**. State of Agile Development Survey. 5th Internal Conference for Software Process Improvement, 2010.

XIAOGUANG, W.; LINPIN, L., ZHuoning, C.; **Research on Organizational-Level Software Process Improvement Model and Its Implementation**. Proceedings of International Symposium on Computer Science and Computational Technology, Volume 02, p 285-289, 2008.

XU, P.; **Knowledge Support in Software Process Tailoring.** Proceedings of the 38th Annual Hawaii International Conference on System Sciences. p.87-99, Hawaii, 2005.

ZAKI, K.; MOAWAD, R.; **A Hybrid Disciplined Agile Software Process Model**. 7th International Conference Informatics and Systems, 2010.

Apêndice

A

Questionário de validação

PARTE I – IDENTIFICAÇÃO DOS PERFIS DAS EMPRESAS E PROFISSIONAIS

Nome do entrevistado:_____

Nome da empresa, instituição, ou órgão (opcional):_____ Cargo:_____

Quantidade de profissionais:_____ Tempo de mercado:_____

Cidade/UF:_____ e-mail (opcional):_____

PARTE II – VALIDAÇÃO DA PROPOSTA DE PESQUISA

Questão 1: Vários profissionais encontram-se despreparados atualmente para corrigir os problemas de adequação e execução nos processos de desenvolvimento. Pode-se afirmar que algumas das causas estão associadas à ineficiência de organização das atividades de gerência, assim como das atividades engenharia de produção para o ciclo de vida do software. Você concorda que a ineficiências para o desenvolvimento software provêm da:

Gerência dos processos	Engenharia dos processos
☐ Concordo totalmente	☐ Concordo totalmente
☐ Concordo parcialmente	☐ Concordo parcialmente
☐ Indiferente	☐ Indiferente
☐ Discordo parcialmente	☐ Discordo parcialmente
☐ Discordo totalmente	☐ Discordo totalmente

Questão 2: Diante dos problemas citados na **Questão 1**, os profissionais recorrem à soluções alternativas que consistem em adaptar a arquitetura de seu processo principal com a inclusão de boas práticas provenientes de um segundo processo. Pode-se afirmar que é uma boa estratégia, desde que o processo suporte plenas condições para efetivar as mudanças. Você concorda com formação deste cenário que:

É uma alternativa a analisar	O planejamento é essencial	Pode resultar benefícios
☐ Concordo totalmente	☐ Concordo totalmente	☐ Concordo totalmente
☐ Concordo parcialmente	☐ Concordo parcialmente	☐ Concordo parcialmente
☐ Indiferente	☐ Indiferente	☐ Indiferente
☐ Discordo parcialmente	☐ Discordo parcialmente	☐ Discordo parcialmente
☐ Discordo totalmente	☐ Discordo totalmente	☐ Discordo totalmente

Questão 3: A resistência dos profissionais relativa a prática da fusão de processos de software é motivada pelo fato de não existirem meios tangíveis que norteiam tal atividade. Pode-se afirmar que uma solução prática seria a adoção de uma metodologia apropriada, estruturada através de passos sequenciais e formatada através de guias instrucionais.

Você concorda com formação deste cenário que:

É uma alternativa a analisar	A metodologia é essencial	Pode resultar benefícios
☐ Concordo totalmente	☐ Concordo totalmente	☐ Concordo totalmente
☐ Concordo parcialmente	☐ Concordo parcialmente	☐ Concordo parcialmente
☐ Indiferente	☐ Indiferente	☐ Indiferente
☐ Discordo parcialmente	☐ Discordo parcialmente	☐ Discordo parcialmente
☐ Discordo totalmente	☐ Discordo totalmente	☐ Discordo totalmente

PARTE III – VALIDAÇÃO DA METODOLOGIA E BOAS PRÁTICAS DOS GUIAS

Questão 1: O processo de implementação de software (planejamento) é o primeiro no ciclo de vida do software. Seu propósito é fornecer subsídios para a organização das atividades do projeto. Pode-se afirmar como boas práticas para um processo adaptado as seguintes:

1- Coletar os dados do cliente/Traçar seu perfil.
2- Identificar o problema/Qualificar a viabilidade de solução.
3- Analisar o cenário/Verificar condições de mudanças.
4- Sintetizar um escopo didático para representação do problema.
5- Sintetizar um escopo didático para representar as propostas de soluções.
6- Atribuir prioridades de resolução ao problema.
7- Definir quantidade de fases e tecnologias a serem implantadas.
8- Definir os critérios os perfis dos profissionais para o projeto.
9- Conceber artefatos que representem a solução encontrada.
10- Conceber artefatos contábeis de custos, recursos materiais e humanos.

Você conclui acerca da validade das práticas que:

☐ Concordo totalmente (9 a 10 práticas)
☐ Concordo parcialmente (7 a 8 práticas)
☐ Concordo indiferente (5 a 6 práticas)
☐ Discordo parcialmente (3 a 4 práticas)
☐ Discordo totalmente (0 a 2 práticas)

Quais práticas você descartaria (citar números)? _____

Questão 2: O processo de análise de requisitos é o segundo processo essencial no ciclo de vida do software. Seu propósito é fornecer subsídios para identificar os requisitos da solução proposta. Pode-se afirmar como boas práticas para um processo adaptado as seguintes:

1- Selecionar os documentos de projeto que facilitarão a análise.
2- Definir o meio de participação do cliente/*stakeholder* (presencial ou virtual).
3- Definir os métodos/técnicas de captação de requisitos por público-alvo.
4- Utilizar palavras da área de negócios para facilitar a coleta.
5- Representar requisitos e atores, estados e comportamentos.
6- Padronizar a descrição dos requisitos. Ex.: RF-XY
7- Classificar os requisitos por prioridades nas iterações. Ex.: (1-5)
8- Definir os requisitos não funcionais por categorias. Ex.: (cliente e software)
9- Conceber as listas de requisitos classificadas por volumes/Versões.
10- Conceber rascunhos do fluxo dos requisitos/Diagramas de UML.

Você conclui acerca da validade das práticas que:

☐ Concordo totalmente (9 a 10 práticas)
☐ Concordo parcialmente (7 a 8 práticas)
☐ Concordo indiferente (5 a 6 práticas)
☐ Discordo parcialmente (3 a 4 práticas)
☐ Discordo totalmente (0 a 2 práticas)

Quais práticas você descartaria (citar números)?_____

Questão 3: O processo de construção de software é o terceiro processo essencial no ciclo de vida do software. Seu propósito é fornecer subsídios para a codificação das soluções projetadas. Pode-se afirmar como boas práticas para um processo adaptado as seguintes:

1- Selecionar os documentos de requisitos que facilitarão a codificação.
2- Planejar um tempo-base para as atividades de codificação.
3- Elaborar o cronograma de iterações, com datas e atividades.
4- Definir regras de atribuição/seleção das atividades.
5- Definir regras de organização/padronização do código-fonte.

6- Utilizar ferramentas web para reuniões. Ex.: VoIP, MSN, etc..
7- Compartilhar o código-fonte em repositórios/Trabalhar em pares ou trios.
8- Documentar o código-fonte com hipertexto. Ex.: Páginas HTML.
9- Conceber pacotes de dados que possam ser mapeados para a integração.
10- Conceber relatórios acerca dos problemas de codificação.

Você conclui acerca da validade das práticas que:

☐ Concordo totalmente (9 a 10 práticas)

☐ Concordo parcialmente (7 a 8 práticas)

☐ Concordo indiferente (5 a 6 práticas)

☐ Discordo parcialmente (3 a 4 práticas)

☐ Discordo totalmente (0 a 2 práticas)

Quais práticas você descartaria (citar números)?_____

Questão 4: O processo de integração de software é o quarto processo essencial no ciclo de vida do software. Seu propósito é fornecer subsídios para integrar as unidades funcionais codificadas. Pode-se afirmar como boas práticas para um processo adaptado, as seguintes:

1- Verificar a existência dos arquivos necessários para integração.
2- Planejar um tempo-base para as atividades de integração.
3- Definir um cronograma/fases/metodologia para gerenciar a integração.
4- Realizar a checagem do código-fonte por grupos. Ex.: Dois ou três.
5- Mapear os caminhos e paths de configuração, etc., através de marcação. Ex.: XML
6- Mapear a ordem da integração das unidades funcionais.
7- Realizar a integração através de grupos. Ex.: Três ou mais
8- Atribuir controle de mudanças para integração. Ex.: [nome da aplicação] + versão
9- Conceber códigos-fontes semelhantes aos originais para realizar simulações.
10- Padronizar a estrutura raiz (web) ou instaladores de arquivos executáveis (local).

Você conclui acerca da validade das práticas que:

☐ Concordo totalmente (9 a 10 práticas)

☐ Concordo parcialmente (7 a 8 práticas)

☐ Concordo indiferente (5 a 6 práticas)

☐ Discordo parcialmente (3 a 4 práticas)

☐ Discordo totalmente (0 a 2 práticas)

Quais práticas você descartaria (citar números)?_____

Questão 5: O processo de qualificação e testes de software é o quinto processo essencial no ciclo de vida do software. Seu propósito é fornecer subsídios de testes às unidades funcionais integradas. Pode-se afirmar como boas práticas para um processo adaptado as seguintes:

1- Verificar a existência dos arquivos e executáveis para realizar os testes.
2- Listar todos os testes que serão realizados no processo.
3- Definir um roteiro com as características para cada teste.
4- Especificar um protocolo (passo-a-passo) de montagem dos cenários de testes.
5- Definir os métodos de testes internos e externos. Ex.: Ambiente e cliente.
6- Controlar os testes internos por versão. Ex.: [nome da aplicação] + versão + *Alpha*.
7- Controlar os testes externos por versão: Ex.: [nome da aplicação] + versão + *Beta.*
8- Mapear os erros encontrados. Ex.: [código de referência] + nome do erro
9- Conceber artefatos similares a fichas técnica do software.
10- Conceber um cronograma futuro de otimização para o software.

Você conclui acerca da validade das práticas que:

☐ Concordo totalmente (9 a 10 práticas)
☐ Concordo parcialmente (7 a 8 práticas)
☐ Concordo indiferente (5 a 6 práticas)
☐ Discordo parcialmente (3 a 4 práticas)
☐ Discordo totalmente (0 a 2 práticas)

Quais práticas você descartaria (citar números)?_____

Questão 6: O processo de gerência da documentação de software é o primeiro processo de suporte no ciclo de vida do software. Seu propósito é fornecer subsídios de gerência dos documentos produzidos nos demais processos. Pode-se afirmar como boas práticas para um processo adaptado as seguintes:

1- Definir quais processos precisam gerar documentação.
2- Definir o porte das documentações. Ex.: Extensiva, intermediária, ou ágil.
3- Padronizar a estrutura dos documentos através de *templates*.
4- Definir as ferramentas específicas para a produção de cada documento.
5- Conceber redações técnicas dos profissionais para o conteúdo dos documentos.
6- Conceber os manuais para o software. Ex.: Usabilidade e técnico.
7- Fixar critérios de gerenciamento, manipulação e acesso para os documentos.
8- Utilizar técnicas arquivistas para armazenagem e mudanças nos documentos.
9- Mapear a busca dos documentos produzidos através de índices.
10- Utilizar um diário de relatos sobre problemas de documentação.

Você conclui acerca da validade das práticas que:

☐ Concordo totalmente (9 a 10 práticas)
☐ Concordo parcialmente (7 a 8 práticas)
☐ Concordo indiferente (5 a 6 práticas)
☐ Discordo parcialmente (3 a 4 práticas)
☐ Discordo totalmente (0 a 2 práticas)

Quais práticas você descartaria (citar números)?_____

Questão 7: O processo de gerência da configuração de software é o segundo processo de suporte no ciclo de vida do software. Seu propósito é fornecer subsídios para gerir os arquivos do projeto. Pode-se afirmar como boas práticas para um processo adaptado as seguintes:

1- Verificar se os artefatos estão identificados/Coletar suas informações técnicas.
2- Verificar sempre se a infraestrutura/ambiente está compatível à demanda de projeto.
3- Definir um roteiro para controle dos tipos de mudanças. Ex.: *Upload* ou *download*.
4- Atribuir escalas de acompanhamento nas mudanças. Ex.: iniciada, em andamento, etc..
5- Categorizar os tipos de arquivos para armazená-los organizadamente.
6- Gerenciar as mudanças no repositório através de solicitações. Ex.: CR.
7- Criar cópias dos arquivos para garantir a integridade. Ex.: Cópia de edição e *backup*.
8- Controlar as mudanças por versões. Ex.: número da *release* + número da mudança.
9- Conceber artefatos das pendências de *baselines* por release/Mudanças.
10- Conceber relatos com configurações pré-estabelecidas para uso futuro.

Você conclui acerca da validade das práticas que:

☐ Concordo totalmente (9 a 10 práticas)
☐ Concordo parcialmente (7 a 8 práticas)
☐ Concordo indiferente (5 a 6 práticas)
☐ Discordo parcialmente (3 a 4 práticas)
☐ Discordo totalmente (0 a 2 práticas)

Quais práticas você descartaria (citar números)?_____

Questão 8: O processo de garantia da qualidade de software é o terceiro processo de suporte no ciclo de vida do software. Seu propósito é fornecer subsídios de garantias de funcionamento nos demais processos. Pode-se afirmar como boas práticas para um processo adaptado as seguintes:

1- Definir uma lista de critérios da qualidade a serem analisados nos processos.
2- Definir uma lista de critérios da qualidade a serem analisados para o produto.
3- Criar um protocolo de verificação dos processos dividido em fases.
4- Definir variáveis de análise por processo. Ex.: tempo de realização, atraso, etc.
5- Classificar o processo pela a importância. Ex.: Crítico, indispensável, complementar.
6- Elaborar verificações acerca da padronização do código-fonte. Ex.: Leitura de código.
7- Representar os resultados das verificações através de *rankings*. Ex.: (1-5)
8- Avaliar regularmente o conhecimento e as condições de trabalho dos profissionais.
9- Mapear os problemas e as soluções encontradas. Ex.: Evitar repetição.
10- Conceber um informativo com as regras de trabalho. Ex.: Troca de profissionais.

Você conclui acerca da validade das práticas que:

☐ Concordo totalmente (9 a 10 práticas)
☐ Concordo parcialmente (7 a 8 práticas)
☐ Concordo indiferente (5 a 6 práticas)
☐ Discordo parcialmente (3 a 4 práticas)
☐ Discordo totalmente (0 a 2 práticas)

Quais práticas você descartaria (citar números)?_____

Questão 9: O processo de verificação de software é o quarto processo de suporte no ciclo de vida do software. Seu propósito é fornecer subsídios de verificação das funcionalidades do software. Pode-se afirmar como boas práticas para um processo adaptado as seguintes:

1- Definir uma ambientação adequada para a interação com o cliente/*stakeholder*.
2- Identificar os recursos e dinâmicas necessárias da verificação. Ex.: Datashow, slides.
3- Definir um roteiro através de tópicos-chave para o projeto e o software.
4- Dividir a verificação da documentação em textual e técnica. Ex.: Ortografia e termos.
5- Analisar a ergonomia através da padronização de passos. Ex.: Função X, até 4 passos.
6- Realizar a verificação com profissionais experientes em regime de rodízio.
7- Reuniões em partes: A primeira só dos profissionais e a segunda com o cliente.
8- Conceber atas nas reuniões para registrar os problemas e decisões tomadas.
9- Conceber uma lista com os itens verificados com êxito, e a outra com os pendentes.
10- Comunicar regularmente ao cliente a agenda e detalhes das próximas verificações.

Você conclui acerca da validade das práticas que:

☐ Concordo totalmente (9 a 10 práticas)
☐ Concordo parcialmente (7 a 8 práticas)
☐ Concordo indiferente (5 a 6 práticas)
☐ Discordo parcialmente (3 a 4 práticas)
☐ Discordo totalmente (0 a 2 práticas)

Quais práticas você descartaria (citar números)?_____

Questão 10: O processo de validação de software é o quinto processo de suporte no ciclo de vida do software. Seu propósito é fornecer subsídios de validação das funcionalidades do software. Pode-se afirmar como boas práticas para um processo adaptado as seguintes:

1- Traçar perfis de usuário/Conhecimento de informática para facilitar as instruções.
2- Definir o método e periodicidade de acompanhamento a cada reunião. Ex.: Mudanças.
3- Definir um roteiro com as instruções de teste/uso a serem repassadas ao cliente.
4- Definir casos com a validação individual e em grupos do cliente. Ex.: Operações.
5- Definir simulações de análise para funcionalidades. Ex.: Tempo médio de operações.
6- Distribuir FAQs temporários para os clientes para ganho de tempo no suporte.
7- Pesquisas de satisfação/Distribuir caixas de sugestões de melhoria no software.
8- Distribuir cópias do software através do mapeamento de versões/funcionalidades.
9- Conceber um relatório de mudanças solicitadas pelo cliente no software a cada release.
10- Conceber um relatório de ocorrências/problemas encontrados pelo cliente.

Você conclui acerca da validade das práticas que:

☐ Concordo totalmente (9 a 10 práticas)
☐ Concordo parcialmente (7 a 8 práticas)
☐ Concordo indiferente (5 a 6 práticas)
☐ Discordo parcialmente (3 a 4 práticas)
☐ Discordo totalmente (0 a 2 práticas)

Quais práticas você descartaria? (citar números)_____

Apêndice

B

Porcentagens de descarte dos guias de implementação de software

Guia híbrido/Processo	Número/ Prática	Frequência de descarte	Total de Descartes/Guia	Porcentagem de descarte/Prática
Implementação de software	1	4	168	2,4%
Implementação de software	2	4	168	2,4%
Implementação de software	3	11	168	6,5%
Implementação de software	4	36	168	21,4%
Implementação de software	5	28	168	16,7%
Implementação de software	6	7	168	4,2%
Implementação de software	7	19	168	11,3%
Implementação de software	8	18	168	10,7%
Implementação de software	9	16	168	9,5%
Implementação de software	10	25	168	14,9%
Análise de requisitos	1	7	136	5,1%
Análise de requisitos	2	8	136	5,9%
Análise de requisitos	3	10	136	7,4%
Análise de requisitos	4	23	136	16,9%
Análise de requisitos	5	8	136	5,9%
Análise de requisitos	6	7	136	5,1%
Análise de requisitos	7	7	136	5,1%
Análise de requisitos	8	18	136	13,2%
Análise de requisitos	9	19	136	14,0%
Análise de requisitos	10	29	136	21,3%
Construção de software	1	5	133	3,8%
Construção de software	2	8	133	6,0%
Construção de software	3	3	133	2,3%
Construção de software	4	9	133	6,8%
Construção de software	5	6	133	4,5%
Construção de software	6	31	133	23,3%
Construção de software	7	8	133	6,0%
Construção de software	8	36	133	27,1%
Construção de software	9	11	133	8,3%
Construção de software	10	16	133	12,0%
Integração de software	1	5	101	5,0%
Integração de software	2	10	101	9,9%
Integração de software	3	9	101	8,9%
Integração de software	4	18	101	17,8%
Integração de software	5	11	101	10,9%
Integração de software	6	4	101	4,0%
Integração de software	7	11	101	10,9%
Integração de software	8	4	101	4,0%
Integração de software	9	20	101	19,8%
Integração de software	10	9	101	8,9%
Qualificação e testes de software	1	8	94	8,5%
Qualificação e testes de software	2	3	94	3,2%
Qualificação e testes de software	3	6	94	6,4%
Qualificação e testes de software	4	9	94	9,6%
Qualificação e testes de software	5	7	94	7,4%
Qualificação e testes de software	6	9	94	9,6%
Qualificação e testes de software	7	9	94	9,6%
Qualificação e testes de software	8	6	94	6,4%
Qualificação e testes de software	9	19	94	20,2%
Qualificação e testes de software	10	18	94	19,1%

Apêndice

C

Porcentagens de descarte dos guias de suporte de software

Guia híbrido/Processo	Número/ Prática	Frequência de descarte	Total de Descartes/Guia	Porcentagem de descarte/Prática
Gerência da documentação de software	1	6	139	4,3%
Gerência da documentação de software	2	20	139	14,4%
Gerência da documentação de software	3	6	139	4,3%
Gerência da documentação de software	4	9	139	6,5%
Gerência da documentação de software	5	25	139	18,0%
Gerência da documentação de software	6	5	139	3,6%
Gerência da documentação de software	7	13	139	9,4%
Gerência da documentação de software	8	18	139	12,9%
Gerência da documentação de software	9	12	139	8,6%
Gerência da documentação de software	10	25	139	18,0%
Gerência da configuração de software	1	4	80	5,0%
Gerência da configuração de software	2	5	80	6,3%
Gerência da configuração de software	3	9	80	11,3%
Gerência da configuração de software	4	12	80	15,0%
Gerência da configuração de software	5	11	80	13,8%
Gerência da configuração de software	6	9	80	11,3%
Gerência da configuração de software	7	8	80	10,0%
Gerência da configuração de software	8	1	80	1,3%
Gerência da configuração de software	9	5	80	6,3%
Gerência da configuração de software	10	16	80	20,0%
Garantia da qualidade de software	1	7	80	8,8%
Garantia da qualidade de software	2	3	80	3,8%
Garantia da qualidade de software	3	10	80	12,5%
Garantia da qualidade de software	4	9	80	11,3%
Garantia da qualidade de software	5	6	80	7,5%
Garantia da qualidade de software	6	7	80	8,8%
Garantia da qualidade de software	7	15	80	18,8%
Garantia da qualidade de software	8	9	80	11,3%
Garantia da qualidade de software	9	2	80	2,5%
Garantia da qualidade de software	10	12	80	15,0%
Verificação de software	1	1	104	1,0%
Verificação de software	2	14	104	13,5%
Verificação de software	3	9	104	8,7%
Verificação de software	4	14	104	13,5%
Verificação de software	5	16	104	15,4%
Verificação de software	6	15	104	14,4%
Verificação de software	7	13	104	12,5%
Verificação de software	8	7	104	6,7%
Verificação de software	9	3	104	2,9%
Verificação de software	10	12	104	11,5%
Validação de software	1	11	102	10,8%
Validação de software	2	7	102	6,9%
Validação de software	3	9	102	8,8%
Validação de software	4	9	102	8,8%
Validação de software	5	13	102	12,7%
Validação de software	6	11	102	10,8%
Validação de software	7	13	102	12,7%
Validação de software	8	15	102	14,7%
Validação de software	9	7	102	6,9%
Validação de software	10	7	102	6,9%