

Nelio Muniz Mendes Alves

**INTEGRAÇÃO DE PRINCÍPIOS DE DESENVOLVIMENTO
ÁGIL DE SOFTWARE AO RUP – UM ESTUDO EMPÍRICO**



UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Faculdade de Engenharia Elétrica

Programa de Pós-Graduação em Engenharia Elétrica

Uberlândia
2011

Nelio Muniz Mendes Alves

INTEGRAÇÃO DE PRINCÍPIOS DE DESENVOLVIMENTO ÁGIL DE SOFTWARE AO RUP – UM ESTUDO EMPÍRICO

Tese de Doutorado apresentada
como requisito parcial à obtenção
do grau de Doutor em Ciências.

Banca Examinadora:

Prof. PhD. Edgard Afonso Lamounier Júnior - (Orientador)

Prof. Dr. Alexandre Cardoso – UFU-FEELT

Profa. Dr. Selma Shin Shimizu Melnikoff - USP

Prof. Dr. Marcelo de Almeida Maia – UFU-FACOM

Profa. Dra. Maria Alice Grigas Varella Ferreira - USP

Uberlândia

2011

SUMÁRIO

Agradecimentos	I
Lista de Figuras	III
Lista de Tabelas	IV
Resumo	VI
Abstract.....	VII
Publicações	VIII
1. INTRODUÇÃO.....	9
1.1. Objetivos da pesquisa	14
1.2. Proposta da tese	15
1.3. Organização da tese	15
2. FUNDAMENTOS	17
2.1. Processo de desenvolvimento de software	17
2.2. Modelos de ciclo de vida.....	18
2.3. Paradigma tradicional de desenvolvimento de software	21
2.4. Paradigma ágil	23
2.5. Scrum.....	28
2.6. RUP – Rational Unified Process	29
2.7. CMMI – Capability Maturity Model Integration	32
2.8. Sumário e conclusões	34
3. TRABALHOS RELACIONADOS	35
3.1. Combinação ágil e tradicional	36
3.1.1. XP e processos centrados em arquitetura	36
3.1.2. Técnica para balanceamento ágil e tradicional.....	37
3.1.3. Integração de agilidade no modelo <i>stage-gate</i>	39
3.2. Vantagens e problemas dos métodos ágeis.....	40
3.3. Pesquisas sobre produtividade.....	43
3.4. Sumário e Conclusões	46
4. PROCESSO PROPOSTO	48
4.1. Modelo de ciclo de vida.....	48
4.2. Características.....	50
4.3. Principais artefatos	53

4.4. Papéis sugeridos	55
4.5. Sumário e Conclusões	55
5. ESTUDO EMPÍRICO	57
5.1. Contexto	57
5.1.1. Empresa	58
5.1.2. Mercado	58
5.1.3. Processo	59
5.2. Estratégia e visão geral da pesquisa	61
5.2.1. Questões da pesquisa e proposições	61
5.2.2. Abordagem adotada: estudo de caso	63
5.2.3. Modelo conceitual	64
5.2.4. Seleção do caso e unidades de análise	66
5.3. Coleta de Dados	67
5.3.1. Modelo de medição para produto e processo	67
5.3.2. Indicadores	70
5.3.3. Amostragem	71
5.3.4. Elaboração da entrevista	72
5.3.5. Procedimentos de coleta de dados	74
5.3.6. Cronologia da coleta de dados	75
5.4. Análise de Dados	75
5.5. Ameaças à validade	78
5.5.1. Validade do constructo	78
5.5.2. Validade interna	79
5.5.3. Validade externa	80
5.5.4. Confiabilidade	80
5.6. Resumo e conclusões	81
6. RESULTADOS	82
6.1. Comparação dos projetos	83
6.1.1. Tecnologia	83

6.1.2. Produto	84
6.1.3. Pessoas.....	87
6.2. Comparação das produtividades	88
6.3. Análise qualitativa	90
6.3.1. Recordação dos projetos	90
6.3.2. Entendimento do processo.....	90
6.3.3. Causas do aumento de produtividade	91
6.4. Discussão	94
6.4.1. Proposição P1	94
6.4.2. Proposição P2	99
6.5. Sumário e conclusões	102
7. CONCLUSÕES	104
7.1. Trabalhos futuros	106
REFERÊNCIAS	108
Anexo A – Manifesto Ágil	119
Anexo B – Entrevista.....	121
Anexo C – Avaliação de Qualidade	127
Anexo D – Ferramentas dos Projetos	130
Anexo E – Comparação de Distribuições.....	131
Anexo F – Dados brutos – Etapa 1	132
Anexo G – Dados brutos – Etapa 3	133

Agradecimentos

Meu amigo William Carvalho é a primeira pessoa à qual expresso minha profunda gratidão. William demonstrou sua grandeza como ser humano desde o momento em que, sem pedir nada em troca, se colocou à inteira disposição para me conceder acesso aos seus dados empíricos industriais e para colaborar no estudo de caso. Ele participou de vários encontros comigo para discutirmos sobre a pesquisa, auxiliou na análise qualitativa dos dados de entrevistas e ainda esteve presente nas bancas de qualificação e de defesa de tese, prestigiando o evento e também ficando à disposição da banca avaliadora para esclarecer dúvidas específicas sobre o cotidiano da empresa durante a execução dos projetos do estudo de caso. Sem os dados e a participação do William, a realização deste trabalho não teria sido possível. Ele é a pessoa para a qual expresso minha maior gratidão e reconhecimento na realização deste trabalho.

Agradeço também:

Ao meu amigo, professor e orientador de doutorado, Edgard Lamounier, por sua conduta reta, pelos conselhos e por sua sabedoria: sabe quando deve ser flexível e quando deve ser rigoroso; sabe ter confiança para delegar, ao mesmo tempo em que dá valiosas contribuições nas suas revisões; sabe quando deve apoiar e quando deve advertir.

À minha esposa, namorada, prima e amiga Ellen, que esteve comigo nos últimos anos do doutorado. Agradeço por toda a compreensão e apoio que pôde me dar, mesmo tendo passado por uma fase difícil que foi o afastamento físico de sua família depois que nos casamos há pouco mais de um ano e meio. Agradeço também aos meus demais familiares, especialmente à minha mãe, meu pai e minha irmã, por todo apoio e compreensão. Também agradeço aos amigos pelos mesmos motivos. A família e os amigos oferecem um amparo emocional muito especial. Muitas vezes eles nem precisam dizer nada: somente saber que eles estão ali já é motivo de felicidade e paz.

Aos grandes professores tais como Gonzalez Pecotche, Bob Hoffman e Carl Sagan, que, com seus ensinamentos, cada um ao seu estilo, me permitiram dar passos iniciais no

autoconhecimento, na compreensão do mundo e na reeducação psicológica e emocional. Teria sido muito mais difícil conviver com as dificuldades dos últimos anos sem sua ajuda.

Ao IFTM e à antiga Escola Agrotécnica Federal de Uberlândia, local onde trabalhei durante todo o doutorado, por ter me concedido, mesmo que parcialmente, uma flexibilização de horário que, sem ela, a realização do doutorado seria quase inviável. Agradeço especialmente aos amigos e colegas professores de Informática que formam a equipe atual da área, tanto por serem pessoas maravilhosas, quanto pelo espírito de equipe e por todo apoio que puderam oferecer.

A todas as instituições onde já estudei, incluindo as de ensino fundamental e médio (Bom Jesus, Enéas Guimarães e Escola Estadual de Uberlândia), pois reconheço sua importância fundamental, uma vez que a cada dia, observando a realidade brasileira atual, fica mais claro para mim o quanto é lamentável o resultado quando estas etapas de ensino não são bem feitas. Agradeço especialmente à UFU, onde estudo desde o ano de 1996, onde fiz minha graduação, mestrado e doutorado. Devo especial gratidão a esta instituição que me acolheu por todos esses anos e me permitiu obter toda minha formação acadêmica gratuitamente.

Por último, com toda reverência e respeito, gostaria de fazer uma alusão à ficção “Guerra nas Estrelas”, e agradecer à “Força”. Optei por fazer a referida alusão para me referir ao espiritual sem levantar nenhuma bandeira de religiões ou deuses.

Lista de Figuras

Figura 1 - Dimensões que afetam a escolha do paradigma [5].....	12
Figura 2 - Ciclo de vida em cascata e cascata com realimentação [3].	19
Figura 3 - Ciclo de vida em espiral [3].....	19
Figura 4 - Ciclo de vida entrega evolutiva [3].....	20
Figura 5 - Ciclo de vida quase-espiral [3].	21
Figura 6 - O ciclo de vida do desenvolvimento ágil de sistemas [15].....	25
Figura 7 - Recurso e prazo estimados <i>versus</i> escopo estimado [11]	26
Figura 8 - Scrum - fluxo de construção [47]	29
Figura 9 - Visão geral das disciplinas e fases do RUP ao longo do ciclo de vida de desenvolvimento.....	32
Figura 10 – Visão geral do método de balanceamento ágil e tradicional [5]	38
Figura 11 - Modelo Stage Gate [66].....	39
Figura 12 - Modelo de ciclo de vida proposto (A&D significa Análise & Design).....	49
Figura 13 - Indicação dos subprocessos governados pelos paradigmas ágil e tradicional..	52
Figura 14 - Artefatos	53
Figura 15 – Facetas de contexto industrial em pesquisas de Engenharia de Software [83]	58
Figura 16 – Definição da estratégia de pesquisa [67].....	61
Figura 17 – Modelo conceitual investigado	65
Figura 18 – Tipos básicos de projetos de estudos de caso [67].....	66
Figura 19 – Subconjunto do modelo de medição utilizado na empresa	68
Figura 20 – Cobertura da amostragem dos entrevistados.....	72
Figura 21 – Cronologia da coleta de dados	75
Figura 22 – Tática geral para análise dos dados.....	76
Figura 23 – Períodos em que os projetos do estudo foram desenvolvidos.....	82
Figura 24 – Resultado da comparação da produtividade dos projetos Scrum-RUP e RUP para cada um dos agrupamentos.....	89
Figura 25 – Grau de recordação dos projetos (escala de 0-10, onde 10 = recordo plenamente)	90
Figura 26 – Acareação da influência do domínio da aplicação na produtividade.....	96
Figura 27 – Decisão do teste estatístico para se comparar distribuições.....	131

Lista de Tabelas

Tabela 1 - Características do paradigma tradicional vs. paradigma ágil segundo [11]	10
Tabela 2 - Contextos apropriados para desenvolvimento ágil e tradicional [5]	11
Tabela 3 - Níveis de entendimento e uso de métodos de software [6] [5]	12
Tabela 4 - Definições referentes a processos [3]	18
Tabela 5 - Conceitos do paradigma dirigido por plano [5]	22
Tabela 6 - Exemplos de abordagens dirigidas por plano [5]	23
Tabela 7 - Métodos ágeis mais referenciados na literatura [16]	26
Tabela 8 - Fases e Marcos do RUP	30
Tabela 9 - Níveis de maturidade do CMMI [3]	33
Tabela 10 - Áreas de processo do CMMI por nível de maturidade	33
Tabela 11 – Métodos centrados em arquitetura do SEI	36
Tabela 12 - Vantagens dos métodos ágeis [55]	40
Tabela 13 - Problemas dos métodos ágeis [55]	42
Tabela 14 - Estudos empíricos sobre impacto em produtividade de métodos ágeis	45
Tabela 15 - Processos empregados nos estudos empíricos analisados	45
Tabela 16 - Resumo das informações de cada projeto dos estudos analisados	46
Tabela 17 - Subprocessos	50
Tabela 18 - Artefatos sugeridos	54
Tabela 19 - Papéis	55
Tabela 20 - Princípios ágeis e iterativos contemplados pelos processos	60
Tabela 21 - Síntese das vantagens dos métodos ágeis	62
Tabela 22 - Grupos de fatores que influenciam na produtividade	65
Tabela 23 - Exemplo de medições de requisitos	69
Tabela 24 – Definição das medidas usadas no modelo de medição	69
Tabela 25 - Definição das métricas usadas no modelo de medição	70
Tabela 26 - Tecnologias empregadas nos projetos	83
Tabela 27 - Resultado da comparação dos projetos com relação ao fator tecnologia	84
Tabela 28 - Informações contextuais dos projetos	85
Tabela 29 - Resultado da comparação dos projetos com relação aos	86
Tabela 30 - Participantes dos projetos	87
Tabela 31 - Quantidade de meses gastos para compreensão do processo Scrum-RUP	90

Tabela 32 - Mudanças no processo percebidas pelos entrevistados.....	91
Tabela 33 - Fatores que influenciaram no ganho de produtividade do processo Scrum-RUP em relação ao processo RUP (etapa 1: questão aberta).....	92
Tabela 34 - Fatores relacionados ao Scrum (questão estruturada).....	93
Tabela 35 - Fatores não relacionados ao Scrum (questão estruturada).....	93
Tabela 36 - Fatores responsáveis pelo ganho de produtividade do processo Scrum-RUP, consolidado pelos entrevistados	94
Tabela 37 - Vantagens produtivas do processo Scrum-RUP não previstas.....	100
Tabela 38 - Fatores produtivos não relacionados ao processo	100
Tabela 39 - Fatores que prejudicaram a produtividade	101
Tabela 40 - Resultados consolidados - quantitativos	102
Tabela 41 - Resultados consolidados - causas do aumento de produtividade.....	102

Resumo

Processos de desenvolvimento de software são atualmente imprescindíveis para uma organização obter níveis aceitáveis de produtividade e qualidade. A integração de processos de desenvolvimento de software ágeis e tradicionais é uma área de pesquisa aberta e pouco explorada que tem atraído o interesse das comunidades acadêmica e industrial com o intuito de se aproveitar os pontos fortes das duas abordagens. Entretanto, pouco ainda se sabe sobre os reais benefícios das propostas existentes, pois os estudos ainda são preliminares e as evidências muito esparsas. Esta pesquisa tem o objetivo de investigar as melhores opções de integração ágil e tradicional, definindo um processo híbrido que aproveite os pontos fortes de ambas as abordagens. Foi elaborada uma proposta de integração de práticas do método ágil Scrum dentro de um processo de desenvolvimento baseado no processo RUP – Rational Unified Process com base em algumas indicações e resultados encontrados na literatura. Também foi realizado um estudo de caso comparativo multi-projeto com o intuito de avaliar o impacto em produtividade com a adoção desta proposta híbrida Scrum-RUP. Foram comparadas as produtividades de cinco grupos de projetos similares desenvolvidos em uma empresa CMMI-ML2 de porte médio, dentre os quais alguns usaram o novo processo Scrum-RUP e outros usaram um processo baseado em uma customização RUP que a empresa já utilizava. Também foram realizadas entrevistas com desenvolvedores que participaram dos projetos no intuito de investigar as possíveis causas dos resultados de produtividade. Os resultados quantitativos mostraram que, dos cinco grupos comparados, quatro apresentaram aumento significativo na produtividade dos projetos Scrum-RUP. Os resultados das entrevistas mostraram que as principais causas de aumento de produtividade estavam relacionadas ao processo Scrum-RUP, sendo comunicação, colaboração e diminuição da documentação as mais frequentes. O estudo mostra que é possível inserir práticas Scrum no processo de desenvolvimento de software sem eliminar o rigor nos subprocessos necessários e, mesmo assim, obter ganhos reais de produtividade no desenvolvimento.

Palavras-chave: Processo de desenvolvimento de software, Scrum, Rational Unified Process, produtividade.

Abstract

Software development processes are now essential for an organization to obtain acceptable levels of productivity and quality. The integration of agile and traditional development processes is an open and few explored research area, which has attracted the interest of industrial and academic communities in order to take advantage of the strengths of both approaches. However, little is known about the real benefits of existing proposals, as studies are still preliminary and evidence is very sparse. This research aims to investigate the best options for agile and traditional integration by defining a hybrid process that takes advantage of both approaches. A proposal to integrate the practices of Scrum agile method within a development process based on RUP – Rational Unified Process – was made based on some indications and results in the literature. An empirical study aiming to evaluate the productivity impact of that hybrid Scrum-RUP proposal was also carried out. Five groups of similar projects from a CMMI-ML2 medium-sized company were compared with respect to productivity, some of which were developed using the new Scrum-RUP process and others were developed using the other RUP-based process the company was used to employ. Also interviews were held with developers who participated in the projects to identify the causes of productivity results. Quantitative results have shown that four out of five project groups showed significant productivity increase in Scrum-RUP projects. The results of the interviews have shown that the main causes of productivity increase were related to process, of which the most frequent were communication, collaboration and reduction of documentation. The study shows that it is possible to integrate Scrum practices in the software development process without losing the rigor needed in the desired subprocesses and still get real development productivity gain.

Keywords: Software development process, Scrum, Rational Unified Process, productivity.

Publicações

Esta pesquisa gerou, até o momento, as seguintes publicações:

1) Em [1] é apresentada uma análise das métricas de produtividade quantitativas obtidas na empresa onde o estudo empírico está sendo realizado. O artigo analisa o impacto em estimativa de esforço causado pela integração de Scrum em desenvolvimento de software tradicional:

ALVES, N., CARVALHO, W., LAMOUNIER, E. *Scrum and Plan-driven Process Integration and its Impact on Effort Estimation*. International Conference on Software Engineering and Knowledge Engineering, 2010.

2) Em [2] é apresentada a parte do estudo de caso referente à análise dos dados quantitativos e contextuais:

ALVES, N., CARVALHO, W., CARDOSO, A., LAMOUNIER, E. Um estudo de caso industrial sobre integração de práticas ágeis no RUP. Revista Ciência e Tecnologia. 2011.

3) ALVES, N., CARVALHO, W., CARDOSO, A., LAMOUNIER, E. Investigando o Aumento de Produtividade na Adoção de um Processo Híbrido de Desenvolvimento de Software. *Submetido para publicação*.

1. INTRODUÇÃO

Projetos de desenvolvimento de software geralmente fazem uso de algum processo para obter melhores resultados em produtividade e qualidade. Um processo é um conjunto de ações e atividades inter-relacionadas realizadas para obter um conjunto especificado de produtos, resultados ou serviços [3].

Diversos processos (ou métodos) de desenvolvimento de software foram criados nas últimas décadas [4] e, desde o meio da década de 1990, os novos processos, em geral, podem se encaixar em duas categorias ou paradigmas distintos de acordo com os princípios e a filosofia que eles incorporam: (1) processos tradicionais (ou dirigidos por plano¹) e processos ágeis [5].

A adoção de desenvolvimento ágil de software [6] tem sido uma opção interessante para aumentar a produtividade [7] [8], bem como para melhor responder a mudanças durante a execução de projetos de desenvolvimento de software [9]. Boa parte da indústria de software tem encaminhado no sentido de aderir a este paradigma de desenvolvimento [10], que preconiza diversas práticas ágeis como:

- envolvimento próximo e constante do cliente
- pequenos ciclos de entrega de software com capacidade operacional
- participação colaborativa do grupo de desenvolvimento
- cronogramas fixos de entrega com escopo estimado
- resposta rápida a requisições de mudança

A adoção de agilidade implica em mudanças de paradigma que diferem das abordagens tradicionais. As mudanças do paradigma ágil afetam o modo como diversas questões técnicas, contratuais e gerenciais são abordadas.

A natureza complexa do desenvolvimento de software e a grande variedade de métodos existentes tornam a comparação entre abordagens ágeis e tradicionais difícil e imprecisa. Leffingwell [11] discute essas diferenças, as quais são sumarizadas na Tabela 1.

¹ Nota do autor: a expressão “dirigido por plano” não é usual no idioma Português, mas é relativamente freqüente na literatura no idioma Inglês (“plan-driven”) para referir-se ao paradigma de processo de desenvolvimento de software tradicional, que possui caráter mais previsível, controlado e documentado.

Tabela 1 - Características do paradigma tradicional vs. paradigma ágil segundo [11]

Ponto de vista	Tradicional	Ágil
Medida de sucesso	Conformidade com o plano	Resposta a mudança, código operacional
Cultura gerencial	Chefia e controle	Liderança / colaboração
Requisitos e arquitetura	Grande e no início	Contínuo / emergente
Garantia de teste e qualidade	Grande, planejado / teste tardio	Contínuo / concorrente / teste cedo
Planejamento e cronograma	Detalhado, escopo fixo, tempo e recursos estimados	Planejamento em dois níveis, data fixa, escopo estimado

Ambos os paradigmas (tradicional e ágil) tem seus pontos fortes e fracos. Enquanto o paradigma tradicional tem sido recomendado para projetos de larga escala e alto risco, o paradigma ágil tem se mostrado mais apropriado para projetos de baixo risco e de equipe e tamanho pequenos [5] [4] [12] [13]. De forma geral, projetos grandes e críticos podem ser prejudicados pela falta de rigor e previsibilidade do paradigma ágil, enquanto que projetos pequenos e de baixo risco podem ter um custo e prazo desnecessariamente elevados pela falta de simplicidade e flexibilidade do paradigma tradicional, que geralmente impõe procedimentos complexos e documentação abrangente.

Boehm e Turner também fazem um levantamento das diferenças dos paradigmas ágil e tradicional [5], tentando estabelecer o contexto apropriado de cada um (onde cada um se aplica de forma melhor) sobre quatro perspectivas: de aplicação (projeto), gerencial, técnica e de pessoas. Um resumo dessa classificação é mostrado na Tabela 2.

A Tabela 3 mostra os níveis de entendimento e uso de métodos estabelecidos por [6] para classificar desenvolvedores de software, que depois foram adaptados por [5] para contemplar desenvolvimento ágil e dirigido por plano (tradicional).

Boehm e Turner estabeleceram, ainda, um resumo dos fatores descritos na Tabela 2, consolidando cinco fatores críticos a serem usados para determinar como um projeto ou organização deve levar a cabo o processo de desenvolvimento, ou seja, se deve ser um processo ou mais ágil, ou mais tradicional. Estes cinco fatores são: (1) tamanho do projeto, (2) criticalidade do projeto, (3) dinamismo do ambiente de desenvolvimento com relação a alterações, (4) pessoal, e (5) fatores culturais.

Tabela 2 - Contextos apropriados para desenvolvimento ágil e tradicional [5]

Perspectiva	Ágil	Tradicional
De aplicação		
Objetivos primários	Rápido valor; responder a mudança	Previsibilidade, estabilidade, alta garantia
Tamanho	Equipes e projetos menores	Equipes e projetos maiores
Ambiente	Turbulento; muita alteração; foco em projeto	Estável, pouca alteração, foco em projeto/organização
Gerenciais		
Relação com o cliente	Clientes dedicados <i>on site</i> ; focado em incrementos priorizados	Interação sob demanda com o cliente, focado em provisões contratuais
Planejamento e controle	Planos internalizados; controle qualitativo	Planos documentados, controle quantitativo
Comunicação	Conhecimento implícito interpessoal	Conhecimento explícito documentado
Técnicas		
Requisitos	Estórias e casos de teste informais priorizados; passagem por mudanças não previstas	Projeto formalizado, capacidade, interface, qualidade, alterações de requisitos previsíveis
Desenvolvimento	Design simples, pequenos incrementos, refatoração presumivelmente sem custo	Design extensivo, incrementos longos, refatoração presumivelmente de alto custo
Teste	Casos de teste executáveis definem requisitos	Planos de testes e procedimentos documentados
De pessoas		
Clientes	Dedicados, colaboradores CRACK* <i>in loco</i>	Colaboradores CRACK* nem sempre <i>in loco</i>
Desenvolvedores**	Pelo menos 30% <i>experts</i> nível 2 e 3 <i>full-time</i> ; nenhum nível 1B ou -1	50% nível 3 no início do projeto e 10% todo o tempo; 30% nível 1B trabalháveis, nenhum nível -1.
Cultura	Conforto e autonomia em muitos graus de liberdade	Conforto e autonomia por meio de estruturas de políticas e procedimentos
* CRACK = colaborativos, representativos, autorizados, comprometidos, bem informados.		
** Veja Tabela 3 para a descrição dos níveis mencionados.		

Tabela 3 - Níveis de entendimento e uso de métodos de software [6] [5]

Nível	Descrição
3	Capaz de revisar o método (quebrar suas regras) para se adaptar a uma nova situação imprevista
2	Capaz de adaptar o método para uma nova situação precedente
1A	Com treinamento, capaz de executar passos indiretos/opcionais do método (e.g. dimensionar histórias para se adequarem em incrementos, compor padrões, refatoração composta, integração de COTS complexos). Com experiência, podem se tornar nível 2.
1B	Com treinamento, capaz de executar procedimentos do método (e.g. codificar um método simples, refatoração simples, seguir padrões de codificação e procedimentos de configuração, executar testes). Com experiência pode dominar algumas habilidades do nível 1A.
-1	Pode ter habilidades técnicas, porém incapaz ou relutante a colaborar ou seguir métodos em equipe.

A Figura 1 mostra os cinco fatores identificados por Boehm e Turner organizados em uma escala em forma de estrela, de modo que os itens mais centrais da escala apontam para o uso de um processo ágil, enquanto que os itens mais externos apontam para o uso de um processo tradicional.

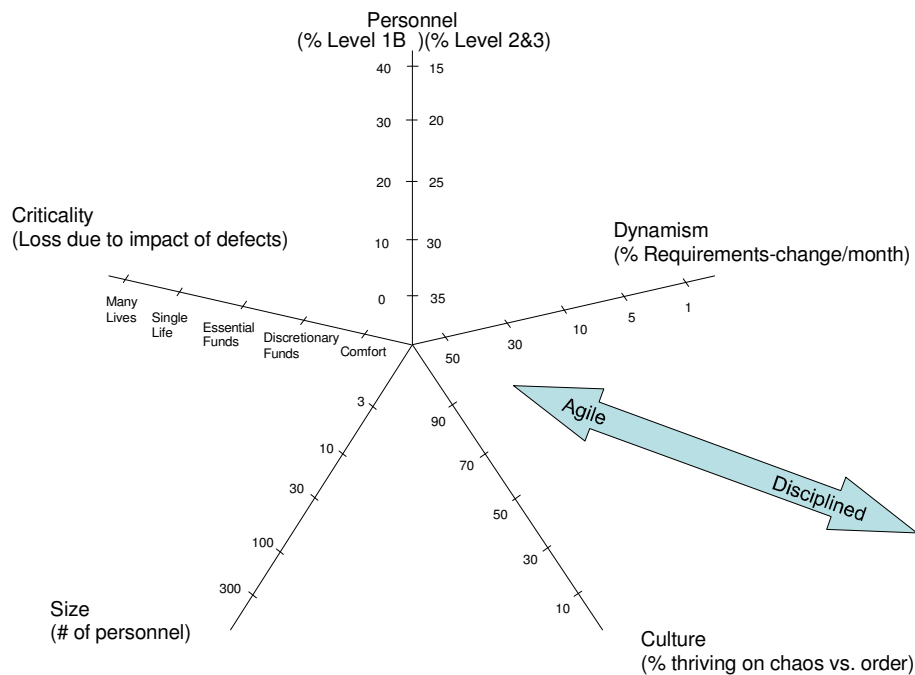


Figura 1 - Dimensões que afetam a escolha do paradigma [5]

Diante do exposto, pode-se notar que há contextos em que o desenvolvimento tradicional é mais apropriado que o desenvolvimento ágil, particularmente naqueles que apresentam uma ou mais das características a seguir:

- **Alta criticalidade:** projetos com alta criticalidade, onde falhas podem acarretar em perdas financeiras significativas ou mortes, necessitam de maior rigor e subprocessos de verificação e validação para prover a alta garantia necessária [5].
- **Projeto e equipe grandes (escala):** a dificuldade dos processos ágeis para lidar com escala é mencionada com frequência na literatura [4]. Como exemplos mais recentes, [14] cita um projeto de larga escala que apresentou, após alguns meses, dentre outros problemas, que as atividades de refatoração começaram a demorar mais que o tempo de uma iteração do processo de desenvolvimento e a taxa de retrabalho aumentou drasticamente, o que vai de encontro ao princípio do acolhimento a mudanças, que é justamente a uma das propostas centrais da agilidade. Em [12] argumenta-se que métodos de desenvolvimento ágil são apropriados para equipes pequenas e, para projetos maiores, outros processos são mais apropriados. A questão da escala tem sido discutida nos meios acadêmicos e industriais com propostas de práticas ágeis de nível técnico-metodológico e corporativo [15] [11], porém o tema ainda carece de evidências satisfatórias [16] [17].
- **Indisponibilidade de recursos humanos suficientes:** para que processos ágeis tenham maior chance de sucesso, é necessária uma maior quantidade de desenvolvedores altamente habilitados durante todo o ciclo de vida de desenvolvimento [5] [18]. Já em processos de desenvolvimento de software tradicionais, essa necessidade é bem menor depois das fases iniciais do projeto.
- **Baixo dinamismo:** projetos com baixa taxa de alterações são mais apropriados para o contexto estável do desenvolvimento tradicional. O desenvolvimento ágil, em tese, é apropriado para altas taxas de alterações [5].
- **Cultura de procedimentos:** desenvolvimento tradicional é mais bem sucedido em uma cultura organizacional onde as pessoas se sentem confortáveis e autônomas tendo seus papéis definidos por políticas e procedimentos claros [5].

Além desses fatores críticos, podem-se considerar os demais fatores já citados na segunda coluna da Tabela 1 e na terceira coluna da Tabela 2 como características que remetem ao desenvolvimento tradicional. Um exemplo significativo é uma situação em que o cliente e/ou fornecedor pode desejar planejamento e contrato antecipados, de modo que mudanças de paradigma como “data fixa – escopo estimado” podem causar muito desconforto em escritórios executivos [11]. Outro exemplo é quando a comunicação *in loco* constante entre cliente e equipe de desenvolvimento é impossibilitada, o que pode ocorrer em situações de terceirização [19] [20].

Ademais, há uma corrente atualmente defendendo que o desenvolvimento ágil de software deve dar mais valor ao processo de elaboração da arquitetura do software [21] [22] [23] [24], uma vez que a arquitetura traz diversos benefícios tais como ser um veículo para gerenciamento de risco, possibilitar comunicação coerente entre envolvidos no projeto, apoiar decisões técnicas e financeiras com maior responsabilidade, além de dar melhor apoio a sistemas críticos [23]. A questão de se conciliar arquitetura com desenvolvimento ágil, entretanto, não é trivial e é um problema emergente, pois ainda se está discutindo o desafio de conciliar as características essenciais de cada abordagem: antecipação *versus* adaptabilidade [24].

Assim, tem-se uma situação em que:

- Por um lado, há um grande interesse na adoção de métodos ágeis por parte da indústria de desenvolvimento de software [17] [10], provavelmente devido a suas promessas de vantagens, notadamente a de ganho de produtividade.
- Por outro lado, há diversos contextos em que o desenvolvimento tradicional é mais apropriado.

Esta situação, bem como o estágio imaturo de pesquisa na área, sugere que estudos devem ser feitos no sentido de apontar e validar soluções para se conciliar as duas necessidades.

1.1. Objetivos da pesquisa

Com certa frequência os autores mais imparciais observam na literatura certo favoritismo em favor de um paradigma ou de outro por parte de acadêmicos e profissionais da indústria [4] [5]. Até mesmo termos como *agilistas* e *tradicionalistas* já foram usados para

identificar os defensores dos paradigmas ágil e tradicional respectivamente [11]. Existe também na literatura propostas de abordagens híbridas que incorporam princípios dos paradigmas ágil e tradicional, as quais serão discutidas em maior detalhe no Capítulo 3.

Entretanto, apesar dos esforços realizados nos últimos anos para levantar as reais implicações do uso de métodos ágeis e de sua combinação com abordagens tradicionais, os estudos ainda são preliminares e esta área de investigação ainda se encontra em um estágio imaturo e pouco explorado [16] [17] [25].

Dentre as três opções de tipos de processo mencionadas (ágil, tradicional e híbrida), esta pesquisa tem o intuito de explorar uma alternativa híbrida a fim de encontrar uma abordagem de processo que aproveite os pontos fortes de cada um dos dois paradigmas ágil e tradicional.

Os objetivos desta pesquisa podem ser especificados como se segue:

- Desenvolver uma proposta de um processo de desenvolvimento de software híbrido que incorpore benefícios de ambos os paradigmas ágil e tradicional.
- Investigar o impacto em produtividade de tal proposta, de modo a verificar se ela resultou em ganho neste sentido.

1.2. Proposta da tese

Para contemplar estes objetivos, foi elaborada uma proposta de combinação de práticas de Scrum [26] com um processo de desenvolvimento de software baseado em RUP [27], ou seja, uma proposta de combinação Scrum-RUP.

A avaliação do impacto em produtividade dessa proposta se deu por meio de um estudo empírico realizado em uma empresa de Tecnologia da Informação CMMI-ML2 de médio porte com sedes em Uberlândia-MG e São Paulo-SP, onde dados de vários projetos foram coletados e analisados. Este estudo analisou o efeito em produtividade do processo Scrum-RUP em relação à situação anterior da empresa em que apenas um processo baseado em RUP era utilizado.

1.3. Organização da tese

A tese é organizada como se segue:

- Capítulo 2 - apresenta uma visão geral dos conceitos relacionados às áreas de conhecimento envolvidas nesta tese, incluindo os processos Scrum e RUP, bem como os conceitos necessários para o entendimento da proposta de combinação Scrum-RUP apresentada no Capítulo 4.
- Capítulo 3 – apresenta uma revisão da literatura com respeito aos trabalhos relacionados a esta pesquisa, bem como aos fundamentos nos quais será baseada a proposta de combinação Scrum-RUP.
- Capítulo 4 – apresenta a proposta de combinação Scrum-RUP, definindo os principais aspectos da mesma. As principais decisões de projeto são discutidas, bem como os principais detalhes.
- Capítulo 5 – contextualiza e descreve em detalhes o estudo empírico realizado nesta pesquisa com o intuito de avaliar o impacto em produtividade do processo Scrum-RUP.
- Capítulo 6 – apresenta e discute os resultados do estudo empírico realizado.
- Capítulo 7 – apresenta as conclusões e discute direções futuras para pesquisa.

2. FUNDAMENTOS

Este capítulo apresenta uma visão geral dos conceitos relacionados às áreas de conhecimento envolvidas nesta tese, incluindo os processos Scrum e RUP, bem como os conceitos necessários para o entendimento da proposta de combinação Scrum-RUP apresentada no Capítulo 4.

2.1. Processo de desenvolvimento de software

Segundo o CMMI - Capability Maturity Model Integration [28], um processo é definido como “*um conjunto de ações e atividades inter-relacionadas realizadas para obter um conjunto específico de produtos, resultados ou serviços*”. Um processo de desenvolvimento de software pode ser conceituado tomando tal definição no contexto de software.

De maneira geral, as atividades de um processo de desenvolvimento de software são executadas por pessoas que desempenham **papéis**. As entidades utilizadas e/ou produzidas pelas atividades geralmente recebem o nome de **artefatos** ou, **itens de trabalho** ou **produtos de trabalho**. A referência [3] apresenta um sumário das principais definições referentes a processo, das quais algumas foram selecionadas na Tabela 4.

Embora esta tese trate especificamente de processo de desenvolvimento de software, processos podem ser definidos, no campo da Engenharia de Software, também para atividades de manutenção, aquisição e contratação de software.

As abordagens de processos de desenvolvimento de software são geralmente especificadas na forma dos chamados métodos² de desenvolvimento de software, ou então na forma de *frameworks* de processo (quando a especificação possui natureza semi-estruturada que permite flexibilidade de uso, customização, etc.). Por exemplo, Scrum e RUP (que participam do contexto desta pesquisa) são comumente referidos na literatura como *frameworks* de processo. Há também *modelos de referência* para melhoria de

² É comum na literatura o uso do termo “metodologia” quando na verdade se está referindo ao método. Aquele termo é coerentemente criticado no sentido de que deveria se referir ao estudo dos métodos. Entretanto, culturalmente “metodologia” adquiriu, até certo ponto, *status* de método no mundo da Engenharia de Software.

processos, como é o caso do CMMI (também participante do contexto industrial desta pesquisa), que pode ser entendido ora como um conjunto de práticas, ora como um conjunto de requisitos.

Tabela 4 - Definições referentes a processos [3]

Termo	Definição
Papel	Conjunto correlato de proficiências, competências e responsabilidades, desempenhado por uma ou mais pessoas.
Processo de desenvolvimento	Conjunto de passos e diretrizes para o desenvolvimento de um produto.
Produto	1. Um objeto produzido, quantificável, e que pode ser um item final ou um item componente. 2. Resultado que se pretende entregar a um cliente ou usuário.
Produto de trabalho ou artefato	1. Qualquer coisa utilizada, produzida ou modificada por tarefas. 2. Coisa útil que resulta de um processo.
Projeto	1. Conjunto gerido de recursos inter-relacionados, que entrega um ou mais produtos a um cliente ou usuário, com início definido e que, tipicamente, opera conforme um plano. 2. Um empreendimento temporário realizado para criar um produto, serviço ou resultado distinto.

2.2. Modelos de ciclo de vida

O ponto de partida para a definição da macro-estrutura ou arquitetura de um processo de desenvolvimento de software é a escolha de um modelo de ciclo de vida. Apresenta-se a seguir uma breve discussão dos principais modelos de ciclo de vida, com base na discussão apresentada em [3], na qual ele utiliza Diagramas de Atividades da UML [29] para representar o *workflow* de subprocessos preconizado por cada modelo de ciclo de vida.

Um dos modelos de ciclo de vida mais clássicos é o ciclo de vida em *cascata*, em que os principais subprocessos são executados em sequência (Figura 2 (a) [3]). Este modelo de ciclo de vida possui a vantagem da simplicidade de gestão com pontos de controle bem definidos, porém é um ciclo de vida muito rígido que exige que as atividades de requisitos, análise e projeto sejam muito bem feitas, senão o risco de se encontrar dificuldades de correção de erros nas fases finais se torna muito elevado. Uma variante do modelo cascata é o modelo *cascata com realimentação*, o qual permite que, de um subprocesso, pode-se retornar ao subprocesso anterior (Figura 2 (b) [3]).

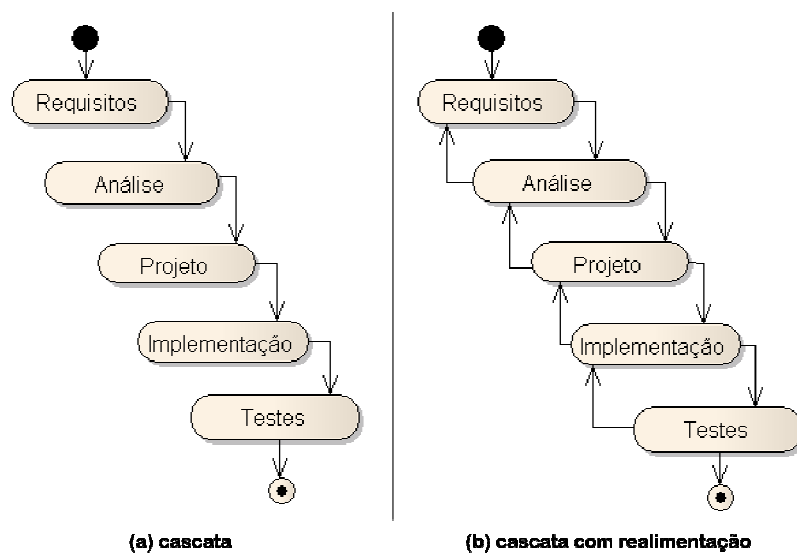


Figura 2 - Ciclo de vida em cascata e cascata com realimentação [3].

O ciclo de vida em *espiral* preconiza que o produto é desenvolvido em uma série de iterações (Figura 3). Cada iteração passa por todos subprocessos e, ao fim de cada iteração, uma *release* parcial do produto é liberada para avaliação.

É possível haver variantes ou modelos de ciclo de vida mistos, como o modelo de entrega evolutiva (Figura 4) [3].

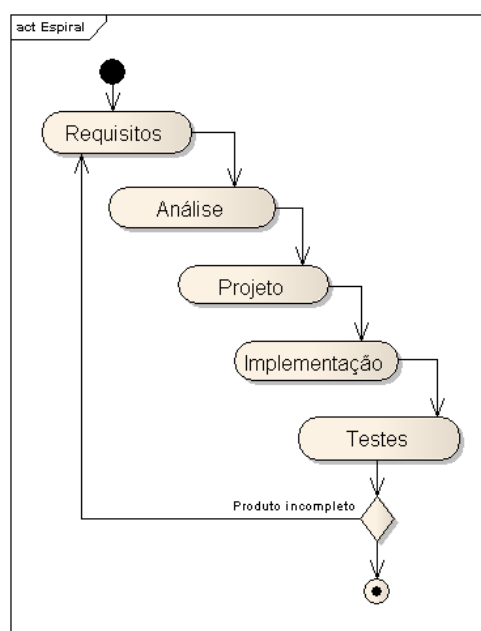


Figura 3 - Ciclo de vida em espiral [3].

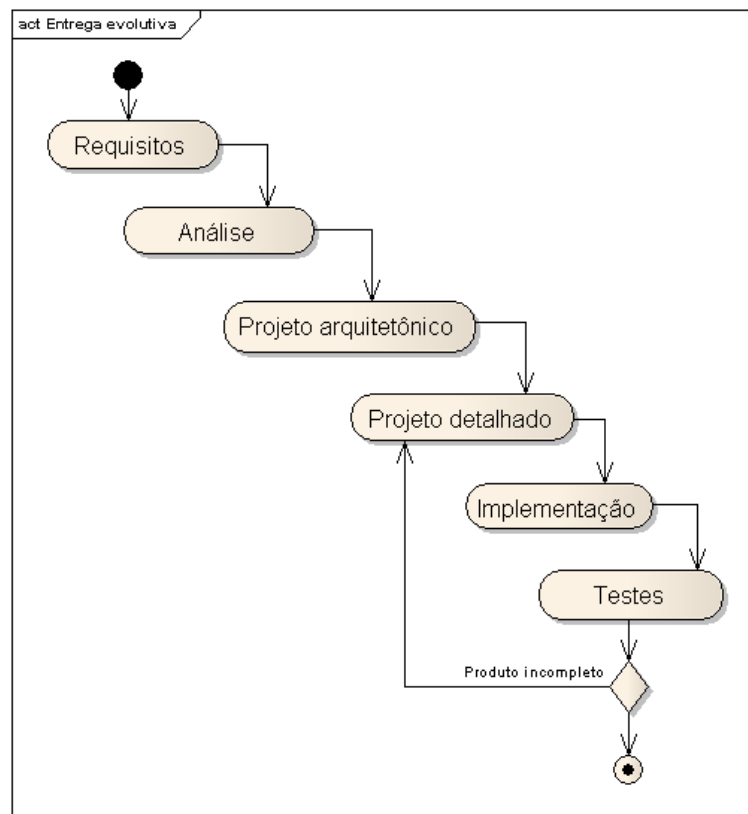


Figura 4 - Ciclo de vida entrega evolutiva [3].

Nesta variante, os subprocessos de requisitos, análise e projeto arquitetônico são executados em cascata, com o objetivo de produzir uma arquitetura de sistema robusta, a fim de estabilizar as principais decisões de projeto nas etapas iniciais e permitir que o desenvolvimento deste ponto em diante não sofra esforços drásticos de retrabalho.

Na prática, entretanto, estes modelos de ciclo de vida não são utilizados pelos processos de desenvolvimento de software da forma como foram apresentados. Na realidade, sempre existe uma fase de iniciação, na qual é feita pelo menos uma definição mínima dos requisitos do produto, para delimitar seu escopo, e uma fase de transição na qual o produto completo é implantado em seu ambiente definitivo. A este modelo dá-se o nome de quase-espiral (Figura 5) [3].

Há também outros modelos como o *time-boxed* (tempo delimitado), no qual um release é aquilo que se consegue fazer dentro de determinado prazo. Este modelo será discutido na Seção 2.4.

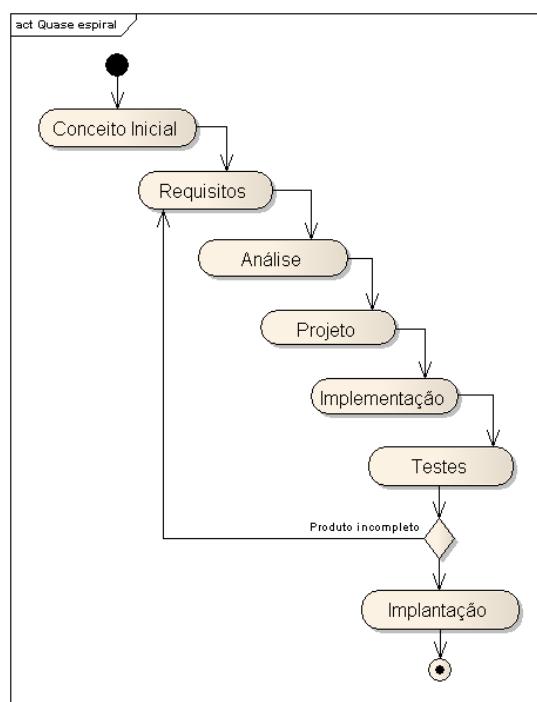


Figura 5 - Ciclo de vida quase-espiral [3].

2.3. Paradigma tradicional de desenvolvimento de software

Diferentemente do paradigma “ágil” (discutido na próxima seção), não há um nome unificado para designar o paradigma de desenvolvimento de software que não incorpora as mudanças para paradigma ágil discutidas no Capítulo 1. Há diversos termos na literatura que remetem às abordagens “não ágeis” de processo de desenvolvimento de software. Termos como “tradicional”, “dirigido por plano”, “disciplinado” e “pesado”³ são os mais encontrados na literatura. Com certa frequência os autores dão preferência à utilização do termo “dirigido por plano”, talvez por entenderem que o termo “tradicional” pode ser erroneamente associado a “antiquado”; e que o termo “pesado” pode sugerir que o processo, se não for ágil, necessariamente deva ser extremamente complexo; e que o termo “disciplinado” sugere que não há disciplina em métodos ágeis, o que é criticado por defensores do paradigma ágil [11].

O paradigma de processo de desenvolvimento de software tradicional é baseado na aplicação sistemática de princípios de Engenharia, incluindo as áreas de Qualidade e Engenharia de Sistemas. Os métodos dentro deste paradigma geralmente focam em

³ Nota do autor: tradução adaptada do equivalente em inglês “heavy-weight”.

planejamento e na elaboração de arquiteturas bem estabilizadas [5]. Seus principais objetivos geralmente são previsibilidade, estabilidade e alta garantia. Este paradigma valoriza artefatos bem definidos, verificação, validação, e conhecimento e interface registrados em contratos e especificações. Há conceitos inerentemente relacionados ao mundo do paradigma dirigido por plano, os quais são sumarizados na Tabela 5 [5].

Tabela 5 - Conceitos do paradigma dirigido por plano [5]

Conceito	Descrição
Melhoria de processo	Um programa de atividades delineado para melhorar o desempenho e maturidade dos processos da organização. Melhoria de processos cresceu a partir do trabalho de gestão de qualidade de Deming, Crosby e Juran, e tem o objetivo de aumentar a capacidade dos processos de trabalho.
Capacidade de processo	A habilidade inerente de um processo produzir resultados planejados. À medida que a capacidade de cada processo aumenta, ele se torna previsível e mensurável, e as maiores causas de baixa qualidade e produtividade são controladas ou eliminadas.
Maturidade organizacional	Pela regular melhoria de sua capacidade de processo, uma organização é dita madura. Maturidade engloba não só a capacidade individual de processo, mas também a aplicação geral de processos padrão na organização. Processos comuns são mantidos e pessoas são treinadas na sua aplicação. Projetos adaptam os artefatos comuns para atender suas necessidades e, uma vez que produzem artefatos comuns, a organização pode começar a medir sua eficiência e eficácia, e melhorá-las baseando-se em métricas.
Grupo de processo	Uma coleção de especialistas que facilitam a definição, manutenção e melhoria dos processos utilizados pela organização.
Gestão de risco	Um processo organizado e analítico para identificar incertezas que podem causar danos ou perdas (identificar riscos), avaliar e quantificar os riscos identificados, e desenvolver e aplicar planos de gestão de riscos para prevenir ou lidar com eles.
Verificação	Verificação confirma que os produtos de trabalho (e.g. especificações, projetos, modelos) refletem apropriadamente os requisitos apropriados para eles (construir o produto corretamente).
Validação	Validação confirma a adequação ou valor de um produto de trabalho para o propósito ao qual se destina (construir o produto correto).
Arquitetura de sistema de software	Uma arquitetura de sistema de software define uma coleção de componentes de software e de sistema, conectores e restrições; uma coleção de declarações de necessidades dos envolvidos no sistema; e uma lógica que mostra que os componentes, conectores e restrições definem o sistema que, se implementado, satisfaria a coleção de declarações de necessidades dos envolvidos.

A Tabela 6 apresenta alguns exemplos de abordagens consideradas por Boehm como representantes do paradigma tradicional ou dirigido por plano [5].

Tabela 6 - Exemplos de abordagens dirigidas por plano [5]

Abordagem	Proponentes	Descrição	Referências
Cleanroom	Harlan Mills, IBM	Usa controle estatístico de processo a verificação matemática para se desenvolver software com confiabilidade certificada. Toda a abordagem é focada em código livre de erros.	[30] [31]
CMMI	SEI, DoD, NDIA, Roger Bate, Jack Ferguson, Mike Phillips	É um modelo de referência de capacidade e maturidade de processos, e também um conjunto de métodos de avaliação que tratam uma variedade de disciplinas usando um vocabulário e arquitetura comuns, e um conjunto de áreas de processo. Veja seção 2.7 para mais detalhes.	[28] [32]
PSP / TSP	Watts Humphrey, SEI	PSP é um <i>framework</i> estruturado de formulários, orientações e procedimentos para se desenvolver software. Ele é direcionado para o uso de auto-avaliação para melhorar as habilidades individuais de programação. TSP baseia-se no PSP e apóia o desenvolvimento de software de porte industrial por meio do controle e planejamento de equipes.	[33] [34]

2.4. Paradigma ágil

Desenvolvimento ágil de software é um paradigma que remete ao ressurgimento da programação como uma arte e não como um processo industrial [5]. Williams e Cockburn afirmam que o desenvolvimento ágil tem tudo a ver com abraçar *feedback* e alterações e, também, que os métodos ágeis são construídos para aceitar, ao invés de rejeitar, altas taxas de alterações [9]. Além disso, o desenvolvimento ágil enfatiza a importância do planejamento adaptativo, simplicidade e liberação contínua de software com valor operacional em pequenas iterações com tempo fixado. Em oposição ao paradigma dirigido por plano, os métodos ágeis tratam o desafio de um mundo imprevisível confiando na criatividade das pessoas e não nos processos [35] [36].

Em 2001, um grupo de profissionais, incluindo vários autores de métodos ágeis, definiu um conhecido documento denominado “Manifesto para Desenvolvimento Ágil de Software” [37], que estabelece quatro principais valores do paradigma ágil:

Passamos a valorizar:

- *Indivíduos e interações mais que processos e ferramentas*
- *Software em funcionamento mais que documentação abrangente*
- *Colaboração com o cliente mais que negociação de contratos*
- *Responder a mudanças mais que seguir um plano*

Além destes quatro valores, tal documento apresenta doze princípios (veja Anexo A para detalhes), muitos dos quais relacionados com as características do paradigma ágil mencionadas no capítulo 1.

Boehm e Turner [5] estabelecem que um método realmente ágil deve conter todos os seguintes atributos:

- Iterativo (diversos ciclos)
- Incremental (não entregar o produto completo de uma vez)
- Auto-organizável (equipes determinam a melhor forma de trabalhar)
- Emergente (processos, princípios e estruturas de trabalho são descobertas durante o projeto ao invés de serem predeterminadas)

A partir dos princípios e valores gerais propostos para o paradigma ágil, profissionais da indústria e acadêmicos propuseram diversas “práticas ágeis” presentes na literatura. Estas práticas geralmente possuem caráter:

- **Técnico-metodológico.** Exemplos: design simplificado, refatoração, desenvolvimento dirigido por teste (TDD), integração contínua.
- **Gerencial.** Exemplos: iterações curtas, reuniões diárias, retrospectivas, programação em pares, planejamento em dois níveis, *time-boxing* (data fixa com escopo estimado).

Ramsin e Paige [4] apontam que uma das críticas mais frequentes na literatura sobre os processos ágeis é que para eles não há a especificação de processo clara e desambigua. Ambler [15] apresenta uma figura geral do ciclo de vida do desenvolvimento ágil

dividindo o tempo do ciclo de vida do desenvolvimento em 3 fases (Iniciação (*Inception*), Elaboração & Construção (*Elaboration & Construction*) e Transição (*Transition*)) mais 1 fase de Produção (*Production*) (Figura 6). É importante notar que, segundo essa proposta de Ambler, o ciclo de vida do desenvolvimento ágil é bem aderente ao ciclo quase-espiral (Figura 5), uma vez que incorpora as etapas de *Conceito Inicial* e *Implantação* (fases Iniciação e Transição respectivamente – veja Figura 6), mediadas por uma etapa iterativa de desenvolvimento – Elaboração & Construção. Um ponto essencial a ressaltar aqui é o princípio da arquitetura emergente, significando que, no desenvolvimento ágil, o esforço em se construir a arquitetura do sistema é distribuído ao longo de toda a fase *Elaboration & Construction*.

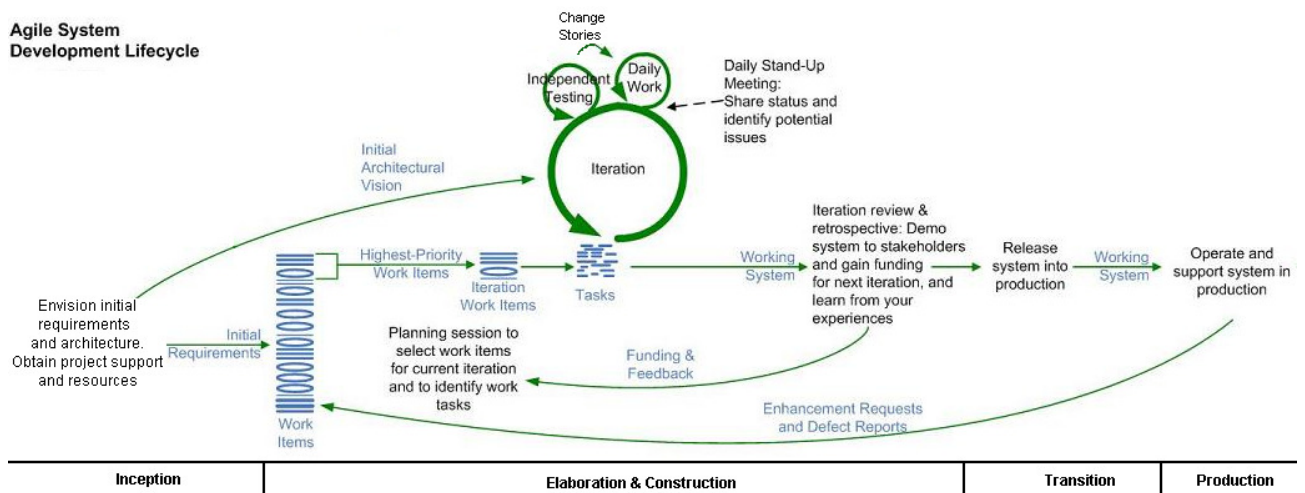


Figura 6 - O ciclo de vida do desenvolvimento ágil de sistemas [15]

Além do modelo quase-espiral, os métodos ágeis também incorporam um modelo de ciclo de vida *time-boxed*, que estabelece intervalos de tempo fixos e é entregue aquilo que se consegue fazer [3]. Esta é talvez a ruptura de paradigma mais drástica proposta pelas abordagens ágeis, como enfatizado por Leffingwell [11]: é uma inversão da pirâmide sobre “qual aspecto determina qual”. Tradicionalmente, fixa-se o escopo e estimam-se recursos e prazo. Porém no modelo *time-boxed* fixa-se recursos e prazo, e então o escopo é estimado (Figura 7).

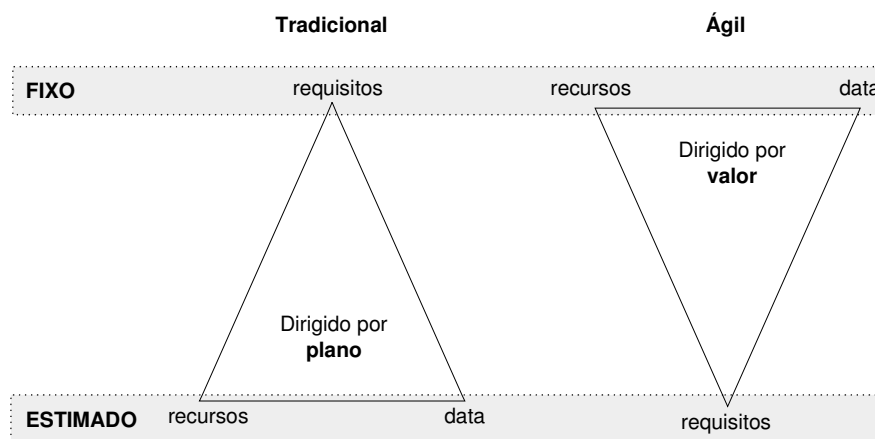


Figura 7 - Recurso e prazo estimados *versus* escopo estimado [11]

Os requisitos no paradigma ágil geralmente são descritos na forma de *estórias* ou *estórias de usuário* (*user stories*), que são textos narrativos que expressam o entendimento do software em termos de unidades de funcionalidade visíveis ao usuário. Uma forma canônica de uma *estória* pode ser vista como mostrado a seguir [38]:

“Como um <tipo de usuário> eu quero <objetivo> para que <razão>”.

Por exemplo:

“Como um Agente de Viagens, eu quero remarcar uma viagem para que eu ganhe tempo quando fizer isso frequentemente”.

A Tabela 7 a seguir mostra uma breve descrição dos processos ágeis mais citados na literatura [16]:

Tabela 7 - Métodos ágeis mais referenciados na literatura [16]

Processo	Propo- nentes	Descrição	Refe- rências
DSDM – Dynamic Software Development Method	Dane Faulkner e outros	Divide projetos em três fases: pré-projeto, ciclo de vida do projeto e pós-projeto. Nove princípios norteiam DSDM: envolvimento com usuário, autonomia da equipe do projeto, entrega frequente, tratar necessidades presentes de negócio, desenvolvimento iterativo e incremental, permite alterações, escopo geral definido antes de o projeto iniciar,	[39]

		teste durante o ciclo de vida, e comunicação eficiente e efetiva.	
Scrum	Schwaber, Sutherland e Beedle	Um <i>framework</i> para processo de desenvolvimento ágil com ênfase em práticas de gerenciamento de projeto. Discutido em detalhes na seção 2.5.	[40] [26]
XP, XP2	Kent Beck, Eric Gamma e outros	Foca em melhores práticas de desenvolvimento. Consiste em doze práticas: <i>planning game</i> , pequenas <i>releases</i> , metáforas, <i>design</i> simplificado, teste, refatoração, programação em pares, propriedade coletiva, integração contínua, semana de 40 horas, clientes <i>on site</i> e padrões de codificação. A versão revisada “XP2” consiste das seguintes “práticas primárias”: sentar-se juntos, toda equipe, <i>workspace</i> informativo, trabalho energizado, programação em pares, estórias, ciclo semanal, ciclo trimestral, descanso, <i>build</i> de 10 minutos, integração contínua, programação <i>test-first</i> e <i>design</i> incremental. Há também 11 “práticas corolárias”.	[41] [42]
Crystal methodologies	Alistair Cockburn	Uma família de métodos para equipes co-localizadas de tamanhos e criticalidades diferentes: <i>Clear, Yellow, Orange, Red, Blue</i> . O método mais ágil, <i>Crystal Clear</i> , foca na comunicação de pequenas equipes desenvolvendo software que não seja crítico com relação a vidas. Características: entrega frequente, melhoria reflexiva, comunicação osmótica, segurança pessoal, foco, fácil acesso a usuários <i>experts</i> , e requisitos para o ambiente técnico.	[43]
FDD – Feature-Driven Development	Peter Coad e Jeff DeLuca	Combina desenvolvimento ágil e dirigido por modelos com ênfase em modelo de objetos inicial, divisão do trabalho em <i>features</i> , e <i>design</i> iterativo para cada <i>feature</i> . Afirma ser adequado para desenvolvimento de sistemas críticos. Uma iteração de uma <i>feature</i> consiste em duas fases: <i>design</i> e desenvolvimento.	[44]
Lean Software Development	Mary e Tom Poppendieck	Uma adaptação dos princípios de <i>lean production</i> (produção magra), o sistema de produção da Toyota para desenvolvimento de software. Consiste de sete princípios: eliminação de desperdício, ampliação do aprendizado, decidir o mais tarde possível, entregar o mais rápido possível, dar autonomia à equipe, criar integridade, e ver o todo.	[45]

2.5. Scrum

Scrum [40] [26] é um *framework* focado em práticas de gestão para desenvolvimento ágil de software. O *framework* Scrum consiste de um conjunto de *Scrum teams* (equipes Scrum) e seus papéis associados, *time-boxes* (eventos com tempo fixado), artefatos e regras. Ele organiza o desenvolvimento de software em iterações de 2 a 4 semanas chamadas *Sprints*.

Schwaber e Sutherland [46] definem que *Scrum teams* são concebidos para otimizar flexibilidade e produtividade. Para esta finalidade, eles são auto-organizáveis, multifuncionais e trabalham em iterações. Cada *Scrum team* possui três papéis: (1) o *ScrumMaster*, que é responsável por assegurar que o processo seja compreendido e seguido; (2) o *Product Owner*, que é responsável por maximizar o valor do trabalho que o *Scrum team* realiza; e (3) a Equipe (*Team*), que realiza o trabalho. A Equipe consiste de desenvolvedores com todas habilidades para transformar os requisitos do *Product Owner* em um pedaço do produto no final da *Sprint*.

Scrum estabelece *time-boxes* – eventos com tempo fixado – para criar regularidade. Estes eventos incluem:

- *Release Planning Meeting*
- *Sprint Planning Meeting*
- *Sprint*
- *Daily Scrum*
- *Sprint Review*
- *Sprint Retrospective*

O coração do Scrum é a *Sprint*, uma iteração de 2 a 4 semanas. Toda *Sprint* utiliza o mesmo *framework* Scrum e produz um incremento do produto final que é potencialmente liberável. Uma *Sprint* inicia-se imediatamente após a outra.

Uma visão pictórica do fluxo do ciclo de vida da construção de software baseado em Scrum é mostrada na Figura 8 [47].

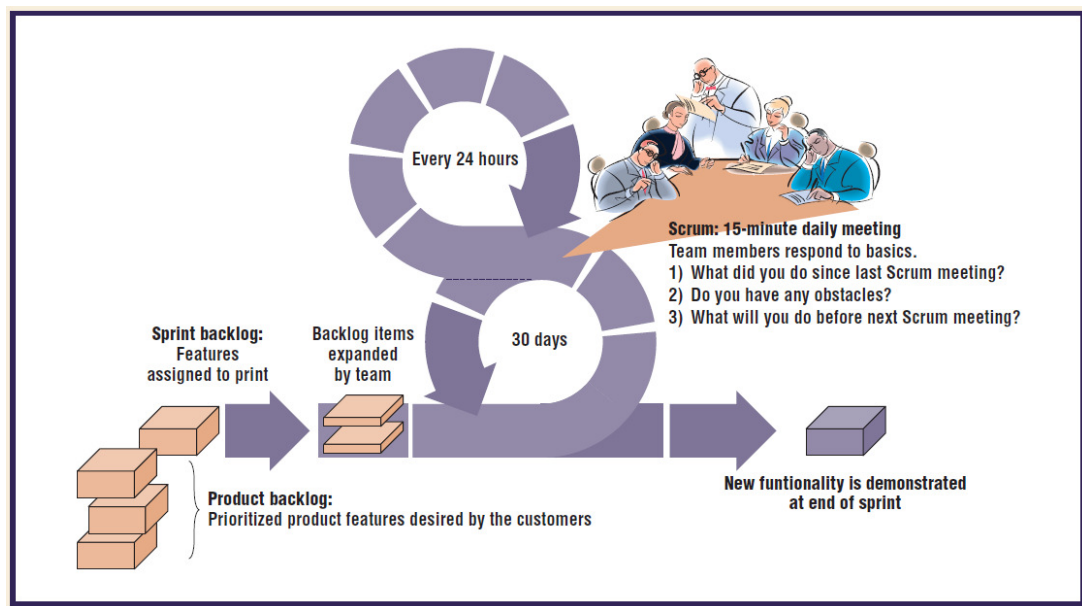


Figura 8 - Scrum - fluxo de construção [47]

Quatro principais **artefatos** são empregados pelo Scrum. O *Product Backlog* é uma lista priorizada de tudo o que se necessita para o produto. O *Sprint Backlog* é uma lista de tarefas para transformar o *Product Backlog* de uma *Sprint* em um incremento de produto potencialmente liberável (*release* parcial). Um *Release Burndown* mede o *Product Backlog* realizado e não realizado no tempo, para um planejamento de *release*. Um *Sprint Burndown* mede o *Sprint Backlog* realizado e não realizado no tempo, para uma *Sprint*.

Regras vinculam *time-boxes*, papéis e artefatos. Um exemplo de regra é que apenas membros da Equipe (*Team*) – as pessoas comprometidas em converter o *Product Backlog* em um incremento – podem falar durante uma reunião *Daily Scrum*.

2.6. RUP – Rational Unified Process

O RUP - Rational Unified Process é um processo amplamente adotado na indústria de software [27]. Ele foi derivado pela Rational Corporation (atualmente uma divisão da IBM) a partir de vários métodos e técnicas de análise e projeto orientado a objetos. Especificações anteriores na mesma linhagem do RUP, a partir dos quais este deriva, são o processo MBase [48] [49] [50] e o USDP - Unified Software Development Process [51].

O RUP possui um conjunto de características centrais:

- **Dirigido por casos de uso** – *casos de uso* [52] são utilizados para capturar os requisitos e correspondem a um conjunto de cenários correlatos que descrevem a interação entre um tipo de usuário (ator) e o sistema. Por terem esta natureza, os casos de uso são adotados ao longo do processo pois ajudarão a assegurar que o sistema se comporte como desejado do ponto de vista do usuário. Os casos de uso servem de base para diversas disciplinas como análise&projeto, implementação e testes.
- **Centrado na arquitetura** – a arquitetura é o coração dos esforços da equipe para se modelar o sistema. Como um único modelo não é capaz de cobrir todos os aspectos de um sistema, o RUP prevê diversos modelos e visões arquiteturais. A arquitetura do sistema deve ser estabilizada ao final da fase de elaboração e servirá de alicerce para o desenvolvimento restante.
- **Iterativo** – o processo é executado em iterações.
- **Incremental** – cada iteração acrescenta uma porção de funcionalidade ao produto.
- **Focado em risco** – o processo visa mitigar os principais riscos nas fases iniciais. Os produtos produzidos em cada iteração, especialmente na fase de elaboração, devem ser selecionados de forma a assegurar que os maiores riscos sejam tratados primeiramente.

O RUP define um ciclo de desenvolvimento dividido em quatro fases (iniciação, elaboração, construção e transição), cada uma com um marco a ser atingido ao seu final. As fases do RUP podem ser entendidas como uma divisão do ciclo de vida de desenvolvimento **no tempo**. A Tabela 8 apresenta um resumo dos marcos de cada fase.

Tabela 8 - Fases e Marcos do RUP

Fase	Marco
Iniciação	Objetivos do ciclo de vida: decidir se continuar ou cancelar o projeto.
Elaboração	Estabilização da arquitetura: examinar os objetivos e escopo detalhados do sistema, a escolha da arquitetura e a resolução dos maiores riscos.
Construção	Capacidade operacional inicial: toda funcionalidade foi desenvolvida e todos testes-alfa foram completados. Além do software, o manual do usuário foi desenvolvido, bem como uma descrição da <i>release</i> atual.
Transição	Liberação do produto: o resultado da validação e aceitação dos itens do produto por parte do cliente.

O RUP (assim como o USDP) utiliza a UML - Unified Modeling Language [29] como notação padrão dos modelos que compõe os principais resultados das atividades do processo.

As atividades do processo são agrupadas no RUP em nove disciplinas:

- Modelagem de Negócios
- Requisitos
- Análise & Design
- Implementação
- Testes
- Implantação
- Gestão de Projeto
- Gestão de Configuração e Mudança
- Ambiente

O RUP determina que, em cada fase, uma determinada quantidade de esforço seja despendida em cada disciplina. A Figura 9 mostra uma representação gráfica de como deve ser distribuído o esforço de cada disciplina ao longo das fases.

Assim como os modelos ágeis em geral, o RUP também adere ao modelo de ciclo de vida quase-espiral, porém possui um grande volume textual e gráfico de orientações e, devido a esse fato, ele é geralmente visto como um método pesado e dirigido por plano. Entretanto, muitos dos atributos ágeis são incorporados na estrutura do RUP, porém o grau de detalhamento do processo muitas vezes obscurece isso. RUP representa uma forma de processo híbrido que incorpora idéias provenientes de ambos paradigmas ágil e dirigido por plano [5].

O RUP pode ser melhor definido não como um processo pré-definido, mas sim como um *framework* customizável que pode atender uma variedade de realidades, pois sua documentação inclui descrições detalhadas de uma grande quantidade de atividades organizadas de forma semi-ordenada, de tal modo que permite uma flexibilidade na customização. Assim, diversas atividades e artefatos podem ser desconsiderados ou reduzidos, a fim de deixar o grau de rigidez do processo de desenvolvimento adequado à realidade que o contexto atual exige.

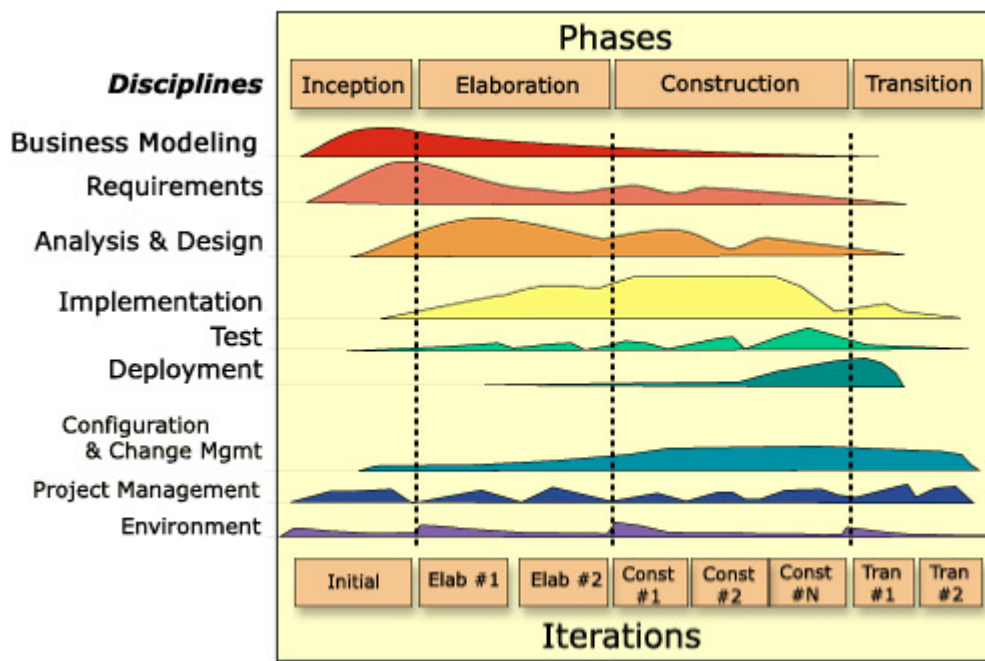


Figura 9 - Visão geral das disciplinas e fases do RUP ao longo do ciclo de vida de desenvolvimento

2.7. CMMI – Capability Maturity Model Integration

O CMMI - Capability Maturity Model Integration [28] não é um método, mas sim um modelo de referência de maturidade de capacidade que descreve um caminho de melhoria evolutiva que parte de processos imaturos e *ad hoc*, até chegar a processos maduros e disciplinados, com o intuito de melhorar de forma significativa a qualidade dos produtos e a eficácia do trabalho [32]. Desenvolvido pelo SEI (Software Engineering Institute) da Universidade Carnegie Mellon, o CMMI procura estabelecer um modelo único para o processo de melhoria corporativo, integrando diferentes modelos e disciplinas.

As práticas recomendadas pelo CMMI são agrupadas em **áreas de processos** (*process areas*), que são conjuntos de práticas relacionadas a uma área que, quando executadas coletivamente, satisfazem a um conjunto de metas considerado importante para a realização de melhorias significativas nessa área. Os modelos nos quais o CMMI se inspirou usavam formas diferentes para medir o domínio das organizações com relação às áreas de processo: alguns usavam **nível de maturidade** (cada nível é um grau de melhoria dos processos, em um conjunto predefinido de áreas de processo, no qual todas as metas dentro do conjunto foram atingidas) e outros o **nível de capacitação** (um indicador do grau

de melhoria de processos dentro de uma área individual). O CMMI admite ambos, mas dar-se-á atenção ao nível de maturidade por este estar relacionado ao contexto industrial desta pesquisa. A Tabela 9 apresenta a lista de níveis de maturidade estabelecidos pelo CMMI.

Tabela 9 - Níveis de maturidade do CMMI [3]

Nível	Nome	Descrição
1	Executado (<i>performed</i>)	Processos informais e <i>ad hoc</i> , às vezes caóticos.
2	Gerido (<i>managed</i>)	Processos planejados e executados conforme políticas.
3	Definido (<i>defined</i>)	Processos bem caracterizados, entendidos e padronizados.
4	Gerido quantitativamente (<i>quantitatively managed</i>)	Processos geridos em função de objetivos quantitativos de qualidade e desempenho.
5	Otimizante (<i>optimizing</i>)	Processos em melhoria contínua.

Como já mencionado, para se obter um nível de maturidade, deve-se atingir as metas de um conjunto de **áreas de processo**. A Tabela 10 apresenta as áreas de processo do CMMI agrupadas por nível de maturidade.

Tabela 10 - Áreas de processo do CMMI por nível de maturidade

Nível de Maturidade	Sigla	Nome	Categoria
2	REQM	Gestão de Requisitos	Engenharia
	PP	Planejamento de Projetos	Gestão de projetos
	PMC	Monitoração e Controle de Projetos	Gestão de projetos
	SAM	Gestão de Acordos com Fornecedores	Gestão de projetos
	MA	Medição e Análise	Suporte
	PPQA	Garantia da Qualidade de Processos e Produtos	Suporte
	CM	Gestão de Configurações	Suporte
3	RD	Desenvolvimento de Requisitos	Engenharia
	TS	Solução Técnica	Engenharia
	PI	Integração de Produtos	Engenharia
	VER	Verificação	Engenharia
	VAL	Validação	Engenharia
	OPF	Focalização dos Processos da Organização	Gestão de processos

	OPD	Definição dos Processos da Organização	Gestão de processos
	OT	Treinamento da Organização	Gestão de processos
	IPM	Gestão Integrada de Projetos	Gestão de projetos
	RSKM	Gestão de Riscos	Gestão de projetos
	DAR	Análise e Resolução de Decisões	Suporte
4	OPP	Desempenho dos Processos da Organização	Gestão de processos
	QPM	Gestão Quantitativa de Projetos	Gestão de projetos
5	OID	Inovação e Implantação na Organização	Gestão de processos
	CAR	Análise e Resolução de Causas	Suporte

2.8. Sumário e conclusões

Este capítulo apresentou de forma sintetizada os principais conceitos relacionados com esta tese, quais sejam:

- Processo de desenvolvimento de software – conceitos relacionados a processo, de modo geral.
- Modelos de ciclo de vida – os principais modelos de ciclo de vida de desenvolvimento de software, ou seja, as formas como os subprocessos podem ser organizados no desenvolvimento (*workflow*).
- Paradigma tradicional de desenvolvimento de software – conceitos e exemplos de processos tradicionais ou dirigidos por plano.
- Paradigma ágil – conceitos e exemplos de processos ágeis.
- Scrum – visão geral do *framework* de gestão de processos Scrum.
- RUP – Rational Unified Process – visão geral dos principais aspectos relacionados ao RUP.
- CMMI - Capability Maturity Model Integration – breve visão geral do modelo de referência CMMI.

As informações apresentadas neste Capítulo 2 têm o intuito de viabilizar a leitura de toda a tese com entendimento. Maiores informações sobre cada tema podem ser adquiridas nas referências apresentadas.

3. TRABALHOS RELACIONADOS

Apesar do grande interesse e dos avanços da indústria na utilização de processos ágeis, muito trabalho ainda precisa ser feito para levantar as reais implicações do uso desses processos e de sua combinação com abordagens tradicionais. Os estudos ainda são preliminares e esta área de investigação ainda se encontra em um estágio imaturo e pouco explorado [16] [17] [25].

Houve uma explosão de propostas de processos de desenvolvimento de software com as mais diversas características nas duas últimas décadas [4]. Entretanto, até o momento existe pouquíssima evidência empírica concernente a processos de desenvolvimento de software [53] [4] [25]. Medir o real valor que um processo tem sobre o outro não é trivial, e a literatura na área é geralmente contaminada com favoritismo e subjetividade, o que é de longe a abordagem apropriada para uma empreitada científica [4].

Muito pouco ainda se sabe sobre os reais benefícios dos métodos ágeis pois, até o presente momento, foram realizados poucos estudos empíricos com um padrão aceitável de confiabilidade e validade [16] [17]. Dentre os poucos estudos empíricos existentes sobre abordagens ágeis, a grande maioria refere-se ao método XP – eXtreme Programming [41]. Com relação a Scrum, como já mencionado no Capítulo 1, este *framework* tem sido amplamente adotado na indústria, embora tenha sido a abordagem ágil “menos pesquisada se comparada com sua popularidade na indústria” [16]. A integração dos métodos ágeis com métodos tradicionais é uma área altamente inexplorada [54].

Este capítulo apresenta trabalhos relacionados ao tema desta tese, subdividindo-os em três seções com propósitos específicos:

- **Combinação ágil e tradicional:** são apresentadas e discutidas as propostas de combinação de processo ágil e tradicional com o propósito de analisar seus resultados quantitativos e qualitativos para auxiliar no levantamento de características que a proposta de integração Scrum-RUP desta pesquisa deve levar em conta. Foram selecionados os trabalhos de dois grupos mundialmente relevantes na área, bem como um trabalho de pesquisa empírica sobre o tema.
- **Vantagens e desvantagens dos métodos ágeis:** com base em um recente trabalho de Petersen e Wohlin [55], as vantagens e desvantagens dos métodos ágeis são

sumarizadas e discutidas, de acordo com os resultados empíricos existentes até então.

- **Produtividade em desenvolvimento ágil:** são apresentados e discutidos os estudos empíricos existentes sobre o impacto de produtividade pelo uso de processos ágeis. Tomou-se por base os trabalhos selecionados na revisão de literatura feita por Dybå e Dingsøyr [16].

3.1. Combinação ágil e tradicional

3.1.1. XP e processos centrados em arquitetura

Nord e sua equipe [56] [13] discutem como o processo de elaboração de arquitetura no desenvolvimento com o método XP pode ser enriquecido com a aplicação das orientações de diversos métodos específicos para arquitetura desenvolvidos pelo SEI da Universidade Carnegie Mellon, os quais são mostrados na Tabela 11.

Tabela 11 – Métodos centrados em arquitetura do SEI

Método	Propósito	Referência
Quality Attribute Workshop (QAW)	Auxilia a equipe de desenvolvimento a compreender o problema levantando requisitos de atributos de qualidade na forma de cenários. Basear-se em cenários e objetivos de negócio assegura que os desenvolvedores tratem os problemas certos.	[57]
Attribute-Driven Design (ADD)	Define a arquitetura do software baseando o processo de projeto nos cenários de atributos de qualidade priorizados que o software precisa contemplar. Este método ajuda a identificar as coisas mais importantes a se fazer para assegurar que o projeto esteja no caminho certo de atender os principais atributos de qualidade e entregar valor ao cliente.	[58]
Trade-off Analysis Method (ATAM) e Cost-Benefit Analysis Method (CBAM)	Oferecem orientações detalhadas para análise do projeto e obtenção de <i>feedback</i> rápido sobre riscos. O ATAM oferece aos arquitetos de software um <i>framework</i> para a compreensão de <i>trade-offs</i> e riscos que se apresentam à medida que tomam decisões arquiteturais. O CBAM auxilia os arquitetos a considerar o retorno do investimento (ROI) de cada decisão arquitetural e oferece orientações a respeito de <i>trade-offs</i> econômicos envolvidos.	[58] [59]

A proposta de [56] consiste, em essência, de um conjunto de indicações de como os métodos desenvolvidos da Carnegie Mellon podem ser executados de forma aderente a alguns princípios ou práticas da filosofia ágil que o método XP incorpora. Por exemplo, quando eles descrevem como o método QAW deve ser usado em combinação com XP, eles expressam:

Scenarios can (...) help the customer prepare the acceptance test suite that will grow with the product. Typically, customers develop user stories for requirements and then work on acceptance test cases for the end of development. Many customers don't know how to build these test cases. The quality attribute scenarios can give them information, if not encourage them to build the test cases. This way of using the scenarios fits in with XP's "test first" or "build for the test" philosophy; test cases are available to test whether or not the code implements the requirements, early in the development process [13].

O trabalho de [56], entretanto, não é um trabalho de pesquisa. Ele consiste em uma proposta e não apresenta corroboração com resultados advindos de algum estudo empírico com rigor metodológico. Considerando a proposta pertinente, ela tem a vantagem de oferecer práticas e orientações detalhadas para elaboração da arquitetura do software sob vários aspectos (veja Tabela 11), o que pode enriquecer muito o gerenciamento de risco, comunicação coerente entre envolvidos no projeto, apoio a decisões técnicas e financeiras com maior responsabilidade, e apoio a sistemas críticos. O principal ponto fraco da proposta, como mencionado, é a falta de evidências empíricas quantitativas e qualitativas para identificar e fundamentar os seus reais benefícios. Mesmo assim este trabalho foi incluído na discussão devido ao seu foco em arquitetura de software (que será um dos temas discutidos na nossa proposta), e também por causa da relevância do SEI e de seus trabalhos na área de melhoria de processos.

3.1.2. Técnica para balanceamento ágil e tradicional

Boehm e Turner publicaram em 2004 um livro sobre balanceamento de processo ágil e dirigido por plano [5], juntamente com outras publicações relacionadas [60] [61] [62] [63].

Eles identificaram um conjunto de fatores críticos em um projeto para determinar o quanto o desenvolvimento do mesmo deve ser ágil ou dirigido por plano. Tais fatores são aqueles apresentados no Capítulo 1, Figura 1: (1) tamanho do projeto, (2) criticalidade do projeto, (3) dinamismo do ambiente de desenvolvimento com relação a alterações, (4)

peçoal, e (5) fatores culturais. De acordo com a proposta apresentada, a chave para o balanceamento adequado entre agilidade e “disciplina” é o risco, analisando tal risco à luz dos cinco fatores mencionados, como mostrado na Figura 10.

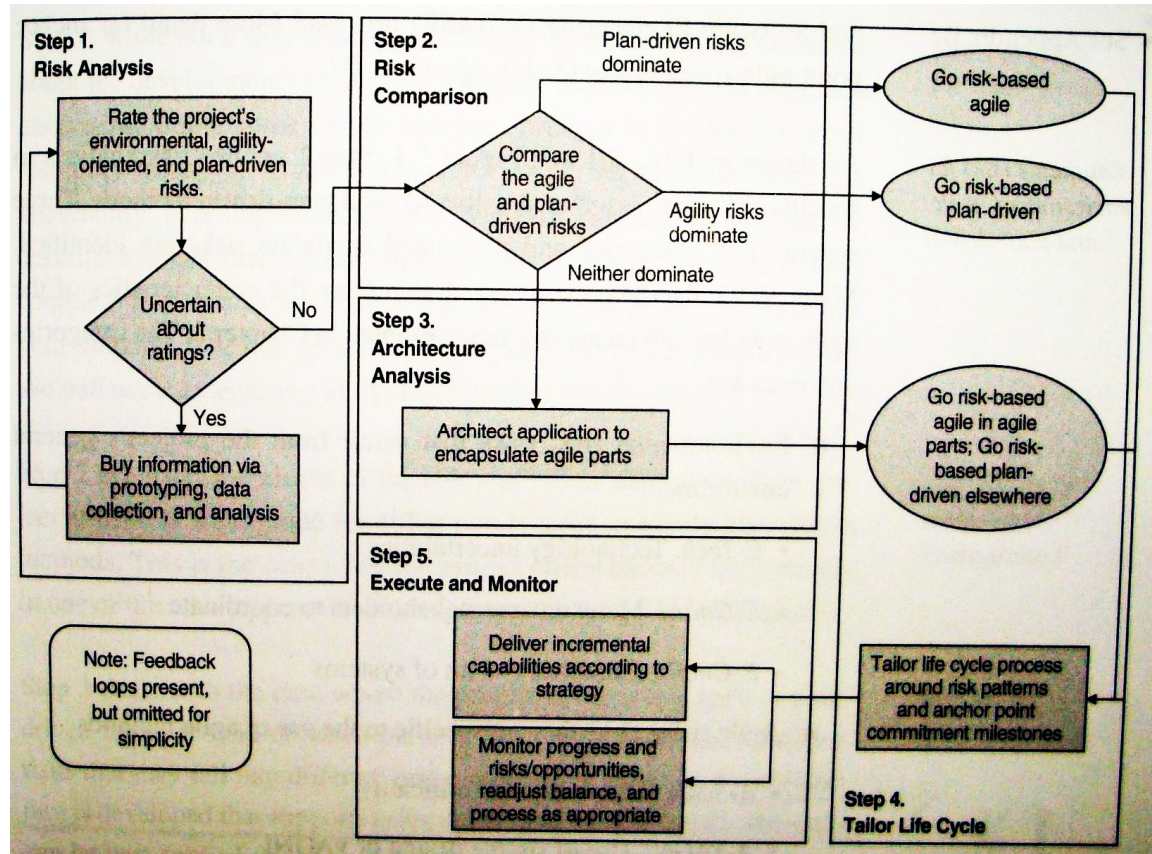


Figura 10 – Visão geral do método de balanceamento ágil e tradicional [5]

Eles apresentam dois casos onde a abordagem proposta por eles funcionou adequadamente, mas eles apresentam, dentre suas conclusões, que sua abordagem não está totalmente desenvolvida, mas funcionou bem nas situações onde foi aplicada.

A principal vantagem deste trabalho é que a escolha dos cinco fatores, bem como a identificação do risco como ponto chave para balanceamento, foi baseada na experiência de vários projetos de muitas organizações. O trabalho de Boehm e Turner também apresenta uma abrangente discussão sobre processos ágeis *versus* processos tradicionais. Eles não oferecem, entretanto, um *framework* ou explicação de “como” deve ser a integração ágil e tradicional por plano, mas somente uma ferramenta para auxiliar a decidir o quanto de cada um dos dois paradigmas deve ser aplicado em um dado projeto.

3.1.3. Integração de agilidade no modelo *stage-gate*

Karlström e Runeson [64] [65] realizam um estudo sobre a integração do método XP no modelo *stage gate* (Figura 11) [66] – um tipo de modelo de gestão de projetos tipicamente adotado na indústria, considerando que o desenvolvimento de software não é uma atividade isolada e, geralmente, corresponde a subprojetos em um ambiente composto de desenvolvimento de sistemas, marketing, planejamento de produção, etc. O modelo *stage gate* prescreve estágios sequenciais bem definidos por quais um projeto deve passar (similar à filosofia do modelo cascata, ou ao conceito de *fases* no RUP), de modo a dar suporte não somente à comunicação dentro do projeto, mas também para tomada de decisão pelos envolvidos. A passagem de um estágio para outro é denominada *gate*.

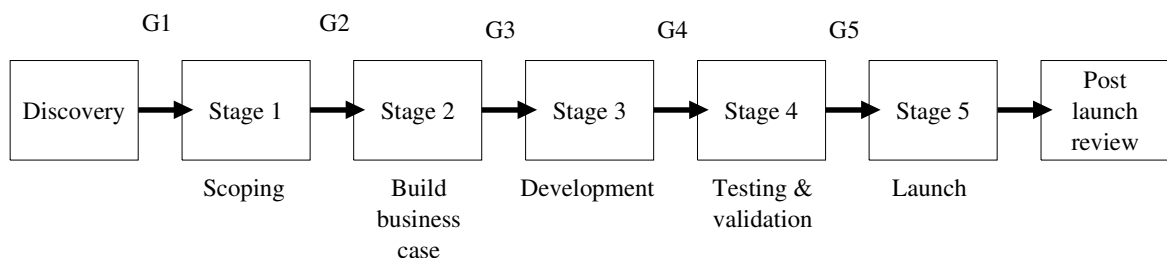


Figura 11 - Modelo Stage Gate [66]

Em seu estudo de caso qualitativo baseado em entrevistas com profissionais de três grandes empresas, eles perceberam que métodos ágeis oferecem ferramentas poderosas para micro planejamento, controle diário do trabalho e relato de progresso. Eles também perceberam que as equipes podem se comunicar de forma bem mais efetiva quando se valem de software funcionando e reuniões face a face, do que em documentos escritos. Em contrapartida, o modelo *stage-gate* oferece aos métodos ágeis formas de coordenar trabalho com outras equipes e de comunicar com as áreas de marketing e alta gestão. A identificação destes aspectos em que o processo ágil ajuda no modelo *stage-gate* e vice-versa constituem a contribuição central e o ponto forte da pesquisa.

Um ponto fraco deste trabalho é que Karlström e Runeson limitam o escopo de seu estudo focando no *gate 3* (veja Figura 11), ou seja, nas implicações e questões relacionadas na passagem do estágio 2 para o estágio 3, quando se utiliza o método XP para desenvolvimento de software neste contexto. Além disso, o estudo se baseia principalmente em entrevistas com profissionais, o que torna o estudo muito suscetível a vieses devido à natureza deste método de coleta de dados [67].

3.2. Vantagens e problemas dos métodos ágeis

Petersen e Wohlin [55] desenvolveram uma pesquisa focada em levantar as vantagens e problemas dos métodos ágeis com base na revisão de literatura de Dybå e Dingsøyr [16], e também incluindo os resultados de outro estudo [68] e de seu próprio estudo de caso industrial realizado na empresa Ericsson AB. A diferença do estudo de Petersen e Wohlin para os demais é que se tratou de um projeto de larga escala, enquanto que os demais estudos se referiam a projetos de menor porte. As vantagens dos métodos ágeis identificadas por Petersen e Wohlin, já incluindo aquelas identificadas no seu estudo, são sumarizadas na Tabela 12.

Tabela 12 - Vantagens dos métodos ágeis [55]

Identificador	Descrição da vantagem	Referências
V01	Melhor transferência de conhecimento para melhor comunicação e feedback frequente de cada iteração.	[69], [64], [70], [68], [55]
V02	Clientes são percebidos pelos programadores como muito valiosos, permitindo aos desenvolvedores ter discussões e ter rápido feedback.	[64], [71], [70], [72]
V03	Programação em pares facilita o aprendizado se os parceiros são trocados regularmente.	[72]
V04	Controle do processo, transparência e qualidade são aumentados por meio de integração contínua e tarefas pequenas e gerenciáveis.	[64], [55]
V05	XP é muito guiado pelo aspecto técnico, fortalecendo os engenheiros e assim aumentando sua motivação.	[64],
V06	Equipes pequenas e reuniões face a face frequentes aumentam a cooperação e ajudam a obter melhores insights no processo de desenvolvimento.	[70], [55]
V07	O ambiente de trabalho é percebido como pacífico, confiável, responsável, e preservador da qualidade de vida profissional.	[73]
V08	Clientes gostam da participação ativa em projetos pois isso os permite controlar o projeto e o processo de desenvolvimento, além de se manterem atualizados.	[7]
V09	Os desenvolvedores percebem o ambiente de trabalho confortável e se sentem mais produtivos utilizando programação em pares.	[74]
V10	Programadores estudantes percebem a qualidade do código utilizando programação em pares.	[74]
V11	Equipes pequenas com pessoas tendo diferentes papéis requerem apenas pequenas porções de documentação pois esta é substituída pela comunicação direta facilitando o aprendizado e entendimento de cada um.	[55]
V12	Pequenos pacotes de requisitos permitem que se implemente e libere pacotes de requisitos rapidamente, o que leva à redução da volatilidade de requisitos nos projetos.	[55]
V13	O desperdício de trabalho não utilizado (documento de requisitos, componentes implementados, etc.) é reduzido pois os pequenos pacotes iniciados são sempre implementados.	[55]

Apenas as vantagens V12 e V13 foram consideradas inéditas por Petersen e Wohlin. As demais vantagens identificadas em seu estudo de caso foram consideradas por eles como equivalentes às outras já existentes (V01, V04 e V06 – veja Tabela 12). A vantagem V11, embora também considerada equivalente à V06 por Petersen e Wohlin, foi destacada de forma exclusiva na Tabela 12, pois a questão de se substituir documentação por comunicação é um aspecto central que se deseja discutir nesta tese mais adiante.

As principais vantagens dos métodos ágeis são relacionadas com benefícios de comunicação levando a um melhor aprendizado e transferência de conhecimento (V01, V02, V06, V11), sendo que V06 também pode ser relacionada com melhor colaboração entre a equipe e V11 com a redução de documentação. Petersen e Wohlin também encontraram vantagens relacionadas a micro gerenciamento (V04, V12, V13), confirmando os resultados de [64] discutidos na seção anterior. Além disso, é enfatizado que as pessoas se sentem confortáveis utilizando métodos ágeis (V07, V09) e desenvolvedores se sentem mais motivados (V05), apesar de que um estudo recente detectou maior stress nos desenvolvedores após a adoção de Scrum [75]. Os demais estudos se referem a *feedback* entre cliente e desenvolvedores (V02, V08) e benefícios da programação em pares (V03, V09, V10).

Similarmente, os problemas identificados por Petersen e Wohlin são sumarizadas na Tabela 13. É importante ressaltar que a lista de problemas está consideravelmente maior do que a lista de vantagens porque a maioria dos problemas identificados no estudo de caso de Petersen e Wohlin foram inéditos, ou seja, não estavam presentes nos estudos anteriores. Este fenômeno pode ser devido ao fato de que o projeto analisado por eles em seu estudo e caso era de larga escala, o que explorou esta particular fraqueza dos métodos ágeis [12] [4] e, portanto, gerou novos problemas não identificados pelos estudos anteriores que trataram projetos menores. Para facilitar a visão geral dos problemas listados na Tabela 13, os mesmos foram agrupados em três categorias:

- **Problemas gerais:** considerados aqueles relacionados aos aspectos centrais da filosofia ágil, tais como os prejuízos no *design* da solução devido ao baixo enfoque em arquitetura (P02), difícil adaptação para testes contínuos (P01), necessidade de membros altamente qualificados (P05), desvantagens da baixa documentação (P06) e deficiência em contextos de larga escala (P03, P11, P12, P13, P14, P15, P17, P18).

- **Problemas específicos:** considerados aqueles relacionados a alguma prática mais específica, tais como as desvantagens de se ter o cliente *onsite* (P09) e as desvantagens da programação em pares (P04, P10).
- **Problemas gerenciais:** considerados aqueles que afetam a atividade dos gerentes de projeto, tais como a perda de controle (P07), aumento da dinâmica na gestão de questões técnicas (P08) e de *releases* internas (P16).

Tabela 13 - Problemas dos métodos ágeis [55]

Identificador	Descrição do problema	Referências
P01	A realização de testes contínuos requer muito esforço (exemplo: a criação de um ambiente de teste integrado é difícil para diferentes plataformas e dependências entre sistemas).	[70], [55]
P02	Não há foco suficiente em arquitetura no desenvolvimento ágil, acarretando decisões de projeto ruins.	[76], [77], [55]
P03	Desenvolvimento ágil é fraco em projetos de grande escala.	[12], [55]
P04	Programação em pares é percebida como exaustiva e ineficiente.	[7], [78], [72]
P05	Membros da equipe precisam ser altamente qualificados para obter sucesso usando agilidade.	[79]
P06	As equipes são altamente coesas, o que significa que a comunicação entre as equipes funciona bem. Porém, a comunicação entre equipes é prejudicada.	[64], [68]
P07	O fortalecimento dos engenheiros torna os gestores temerosos inicialmente, e assim requer treinamento suficiente para os gestores.	[64]
P08	Implementação começa muito cedo, assim questões técnicas aparecem muito cedo do ponto de vista gerencial.	[64]
P09	Clientes <i>onsite</i> tem que se comprometer com todo desenvolvimento, o que os coloca sob stress.	[19]
P10	Da perspectiva dos estudantes, programação em pares não é aplicável se um parceiro é muito mais experiente que o outro.	[80]
P11	A entrega dos requisitos para serem projetados demora devido ao processo de decisão ser complexo.	[55]
P12	A lista de prioridades é essencial no modelo da empresa para se trabalhar e é difícil de criar e manter.	[55]
P13	Atividades de <i>design</i> possuem livre capacidade devido a longos períodos de espera pois em engenharia de requisitos a tomada de decisões complexas tomam muito tempo.	[55]
P14	Redução de cobertura de testes nos projetos devido à falta de testes independentes e escassez de projetos, demandando que a última versão do sistema compense a cobertura.	[55]
P15	O processo da empresa requer que se produza muita documentação de testes.	[55]
P16	A gestão de configuração requer alto esforço para coordenar o alto número de releases internas.	[55]
P17	O desenvolvimento do ambiente de configuração para selecionar partes para a customização de soluções demanda um longo tempo devido ao início tardio do trabalho de empacotamento do produto e uso de bibliotecas de programação sequencial.	[55]
P18	O trabalho de empacotamento do produto é aumentado pois ele ainda é visto de um ponto de vista técnico, mas não de um ponto de vista comercial.	[55]

O trabalho de Dybå e Dingsøyr [16] e o posterior aprimoramento e consolidação feitos por Petersen e Wohlin [55] desempenharam um papel muito importante na elaboração da pesquisa desta tese. Por um lado, as vantagens identificadas são importantes para melhorar a qualidade da construção da pesquisa, pois ajudam no levantamento de hipóteses e a formulação da explicação dos resultados esperados. Por outro lado, os problemas identificados são importantes para auxiliar na elaboração do modelo de processo proposto, de modo a tentar mitigar alguns dos problemas existentes.

3.3. Pesquisas sobre produtividade

Nesta seção são apresentados os resultados de estudos sobre impacto em produtividade da aplicação de processos ágeis em comparação com processos tradicionais.

Pode-se encontrar na literatura diversos artigos que relatam as vantagens obtidas após a adoção de métodos ágeis e, especificamente, sobre os ganhos obtidos em produtividade. Entretanto, a grande maioria destes trabalhos consiste de relatos de experiência, ou de lições aprendidas, ou de informações baseadas em evidência anedótica [16]. Além disso, nestes trabalhos a qualidade da descrição dos processos empregados e das informações contextuais é variável, o que dificulta a comparação dos trabalhos e a consequente inclusão dos mesmos no corpo de conhecimento científico na área. Assim, como mencionado no início deste capítulo, não será dada especial atenção a tais trabalhos nesta tese, até mesmo para evitar a explosão desnecessária de referências que na verdade iriam acrescentar pouca informação relevante relacionada ao objetivo desta pesquisa. Assim, tomou-se por base os trabalhos selecionados na revisão de literatura feita por Dybå e Dingsøyr [16], com o intuito de incluir apenas aqueles trabalhos que, segundo a avaliação de qualidade aplicada por eles, possuem um grau satisfatório de rigor e credibilidade. Os trabalhos foram, então, analisados e a discussão sobre os mesmos é apresentada a seguir.

Ilieva et al. [7] compararam a produtividade de dois projetos similares, onde um deles usou métodos tradicionais e outro usou um método híbrido baseado em PSP e XP. Ambos os projetos possuíam tamanho por volta de 900 pessoas-horas, equipe de 4 pessoas e usaram a tecnologia J2EE. Por outro lado, os projetos eram aplicações diferentes: o projeto desenvolvido com método tradicional correspondia a um *framework* para se desenvolver sistemas de gestão financeira, enquanto que o projeto desenvolvido com o método XP & PSP híbrido era uma customização de um produto para outro cliente. A

produtividade foi medida durante três iterações de cada projeto. No geral, os resultados mostraram um aumento de 42% em produtividade, medida em LOC/hora, para a equipe ágil. O aumento de produtividade foi maior para a primeira iteração, enquanto que praticamente não houve diferença em produtividade na última iteração, pois apenas correções de *bugs* e modificações foram requeridas nesta iteração, o que requereu considerável esforço, porém nenhum novo código foi desenvolvido.

Layman et al. [8] compararam um *release* desenvolvido em linguagens Java e C++ com métodos tradicionais durante 18 meses, com um novo *release* desenvolvido em linguagem Java com métodos ágeis durante 3,5 meses. Os resultados mostraram um ganho de 46% em produtividade, medida em LOC/hora, para o novo *release* ágil comparada com o *release* anterior tradicional. Entretanto, a equipe ágil tinha considerável maior domínio e perícia na linguagem de programação, além de maior experiência em gestão de projetos, pois três membros que trabalharam no novo *release* haviam também trabalhado no *release* anterior. Vale ressaltar também que o primeiro projeto foi parcialmente desenvolvido na linguagem C++ que pode ter contribuído para a perda de produtividade.

Os resultados do estudo de Wellington et al. [81] mostraram uma perda de 44% em produtividade, medida em LOC/hora, de uma equipe utilizando uma variação do método XP comparada a uma equipe tradicional utilizando uma variação do método TSP. Este estudo foi realizado com estudantes durante um semestre letivo de atividades. A equipe tradicional utilizou um tempo para projetar o software a ser desenvolvido, porém no estudo percebeu-se que eles não seguiram o projeto durante a implementação e, além disso, utilizou-se de muitos códigos usando “copiar e colar”. Esta observação leva ao questionamento sobre o ganho de produtividade da equipe tradicional ter sido apenas aparente.

Benediktsson et al. [82] realizaram um experimento com estudantes no qual quinze equipes desenvolveram produtos de software comparáveis utilizando quatro diferentes abordagens (V-Model, Incremental, Evolucionário e XP). A maior diferença em produtividade, medida em LOC/hora, foi entre as equipes que utilizaram o V-Model e as equipes que utilizaram XP, com XP sendo, em média, 337% mais produtivo que o V-Model. Entretanto, este ganho de produtividade foi devido ao fato de que a equipe XP ter entregado 3,5 vezes mais linhas de código sem entregar mais funcionalidade.

A Tabela 14 a seguir resume as informações gerais dos quatro estudos que se acabou de discutir.

Tabela 14 - Estudos empíricos sobre impacto em produtividade de métodos ágeis

Estudo	Abordagem de pesquisa	Profissionais ou Estudantes	Descreve empresa / mercado?	Descreve a experiência ou habilidade das pessoas?	Descreve as ferramentas / técnicas utilizadas?	Dados qualitativos	Dados quantitativos	Ganho em produtividade do processo ágil
[7]	Estudo de caso único	Profissionais	sim	Não	não	não	sim	42%
[8]	Estudo de caso único	Profissionais	não	Sim	sim	sim	sim	46%
[81]	Estudo de caso único	Estudantes	n/a	Não	não	sim	sim	-44%
[82]	Experimento	Estudantes	n/a	Não	sim	não	sim	338%

A seguir são apresentadas outras considerações sobre estes estudos:

Ferramentas: apenas dois dos quatro estudos identificados citam as ferramentas de desenvolvimento utilizadas nos projetos (veja Tabela 14).

Processos: ao analisar a descrição dos processos tradicionais e ágeis empregados nos quatro estudos identificados, percebeu-se que a qualidade e o grau de detalhe das descrições são variáveis. A Tabela 15 resume a maneira de como os processos foram descritos nos artigos. Na descrição dos processos geralmente faltam elementos importantes como a lista de artefatos produzidos. Os processos tradicionais são os menos especificados, gerando falta de clareza a respeito de “com o quê o método ágil foi comparado”.

Tabela 15 - Processos empregados nos estudos empíricos analisados

Estudo	Projetos do grupo controle					Projetos sujeitos				
	Processo empregado	Lista práticas obedecidas?	Lista papéis?	Lista artefatos?	Lista subprocessos / workflow?	Processo empregado	Lista práticas obedecidas?	Lista papéis?	Lista artefatos?	Lista subprocessos / workflow?
[7]	n/d	não	não	não	não	XP & PSP hybrid	sim	sim	sim	sim
[8]	"Waterfall with some XP practices"	não	não	não	não	XP	sim	não	não	n/a
[81]	TSP variant	não	não	não	não	XP variant	sim	não	não	n/a
[82]	V-Model	não	não	sim	sim	XP	não	não	sim	n/a

Pessoas: apenas um dos quatro estudos identificados cita a experiência ou habilidade das pessoas participantes nos projetos (veja Tabela 14).

Projetos: a Tabela 16 apresenta um resumo das informações dos projetos envolvidos nos estudos analisados.

Tabela 16 - Resumo das informações de cada projeto dos estudos analisados

Projetos do grupo de controle						Projetos sujeitos				
Estudo	Descrição do software	Tamanho da equipe	Tamanho do projeto	Duração do projeto	Linguagem	Descrição do software	Tamanho da equipe	Tamanho do projeto	Duração do projeto	Linguagem
[7]	"a framework for developing financial management systems".	4	908 pessoas horas	n/d	J2EE	"a personalization of the product for another client".	4	900 pessoas horas	n/d	J2EE
[8]	"a scriptable GUI environment for external customers to develop customized end user and business software".	10	108 pessoas meses	18 meses	Java, C++	(outra <i>release</i> do mesmo software)	10	14.7 pessoas meses	3,5 meses	Java
[81]	n/d	17	n/d	1 semestre	n/d	n/d	16	n/d	1 semestre	n/d
[82]	"an interactive package centered on a database system for home or family usage with an appropriate web-based user interface" (com pequenas diferenças entre cada um, de acordo com os autores)	3-4	515 pessoas horas	n/d	J2EE	(idem)	3-4	736 pessoas horas	n/d	J2EE
		3-4	907 pessoas horas	n/d	J2EE		3-4	669 pessoas horas	n/d	J2EE
		3-4	919 pessoas horas	n/d	J2EE		3-4	373 pessoas horas	n/d	J2EE
		3-4	651 pessoas horas	n/d	J2EE		-	-	-	-

3.4. Sumário e Conclusões

Os trabalhos de Nord et. al [13] e Boehm e Turner [5] podem ser considerados como marcos relevantes na discussão sobre a importância de se utilizar práticas ágeis sem abrir mão de princípios de desenvolvimento de software tradicional, tais como a atenção à elaboração de arquitetura fundamentada em boas práticas de Engenharia, de modo a enriquecer o gerenciamento de risco, comunicação entre envolvidos, garantia a sistemas críticos e apoio a decisões técnicas e financeiras. Complementando esta visão, os resultados de Karlström e Runeson [64] sugerem que métodos ágeis podem colaborar com abordagens tradicionais oferecendo melhor micro planejamento, controle diário do

trabalho, relato de progresso, além de comunicação mais efetiva se valendo software funcionando e reuniões face a face, do que por meio de documentos escritos.

Com relação às vantagens e desvantagens dos métodos ágeis, Petersen e Wohlin [55] apresentaram uma lista consolidada, a qual foi apresentada e discutida neste capítulo. As vantagens e desvantagens serão levadas em conta no Capítulo 4 nas discussões sobre as decisões da proposta de processo híbrido. Além disso, as vantagens também servirão como base para a elaboração de proposições de pesquisa.

Os estudos existentes sobre impacto em produtividade pela adoção de processos ágeis confirmam que esta área de investigação ainda se encontra em um estágio prematuro e pouco explorado, pois se pode ver que ainda contamos com resultados intrigantes e contraditórios, além de que a quase totalidade dos estudos foram limitados ao escopo do método ágil XP e as informações contextuais são relatadas com qualidade variável.

Assim, pode-se afirmar que o campo dos métodos ágeis para desenvolvimento de software e sua integração com abordagens tradicionais ainda requerem muita pesquisa no sentido de investigar as melhores possibilidades de integração, além de se verificar as reais implicações em produtividade que a agilidade proporciona.

4. PROCESSO PROPOSTO

Este capítulo tem o intuito de cumprir o primeiro objetivo desta pesquisa, apresentando uma proposta de um processo de desenvolvimento de software híbrido que incorpore benefícios de ambos os paradigmas ágil e tradicional.

Nas seções que se seguem, é apresentada a proposta de processo Scrum-RUP híbrido desenvolvida nesta tese, que consiste em um *framework* para integração de práticas do método ágil Scrum dentro de um processo de desenvolvimento baseado no RUP. O termo *framework* é empregado no sentido de que o processo aqui descrito não possui caráter rígido, nem tem o objetivo de explicar detalhes sobre como cada atividade deve ser executada. Ao invés disso, a descrição apresentada neste capítulo define e discute as principais decisões tomadas na concepção e elaboração do processo, tais como aspectos do modelo de ciclo de vida, características gerais, as porções ágeis e tradicionais do processo, artefatos e papéis.

Assim, desde que seja respeitada sua estrutura geral, esta proposta pode sofrer pequenas adaptações de acordo com as exigências e especificidades do contexto no qual for aplicado. Esta flexibilidade vale especialmente para os detalhes sobre como as atividades do processo devem ser realizadas, pois cada organização implementa detalhes específicos diferentes em seu processo, dependendo de sua cultura organizacional, necessidades, dentre outros fatores.

4.1. Modelo de ciclo de vida

O modelo de ciclo de vida deste *framework* é um modelo híbrido que incorpora idéias do modelo de ciclo de vida *quase-espiral*, *entrega evolutiva* e *time-boxed* (Figura 12), pelos seguintes motivos:

- **Quase-espiral:** o ciclo de vida proposto é parcialmente aderente ao modelo *quase-espiral* pois há um subprocesso *Conceito Inicial* responsável pelo levantamento inicial dos requisitos e questões arquitetônicas mais importantes, obtenção de recursos e infraestrutura para o projeto. Em sequência há um conjunto de subprocessos executados parcialmente em estilo iterativo.

- **Entrega evolutiva:** diferentemente do ciclo de vida de desenvolvimento puramente ágil, a etapa de elaboração da arquitetura é antecipada para os momentos iniciais do projeto, como no RUP [27]. Assim, pode-se entender que o ciclo de vida proposto incorpora parcialmente a idéia do modelo entrega evolutiva.
- **Time-boxed:** a modelo *time-boxed* também se enquadra parcialmente no ciclo de vida deste *framework* pois seu desenvolvimento, como em Scrum, é subdividido em iterações de tempo fixo (geralmente duas semanas) com possibilidade de entrega de incremento do produto ao final de cada uma. Entretanto, a aplicação de *time-boxing* aqui não é totalmente aderente à proposta ágil: há uma diferença que caracteriza a natureza híbrida deste *framework*, a qual será discutida a seguir na Seção 4.2.

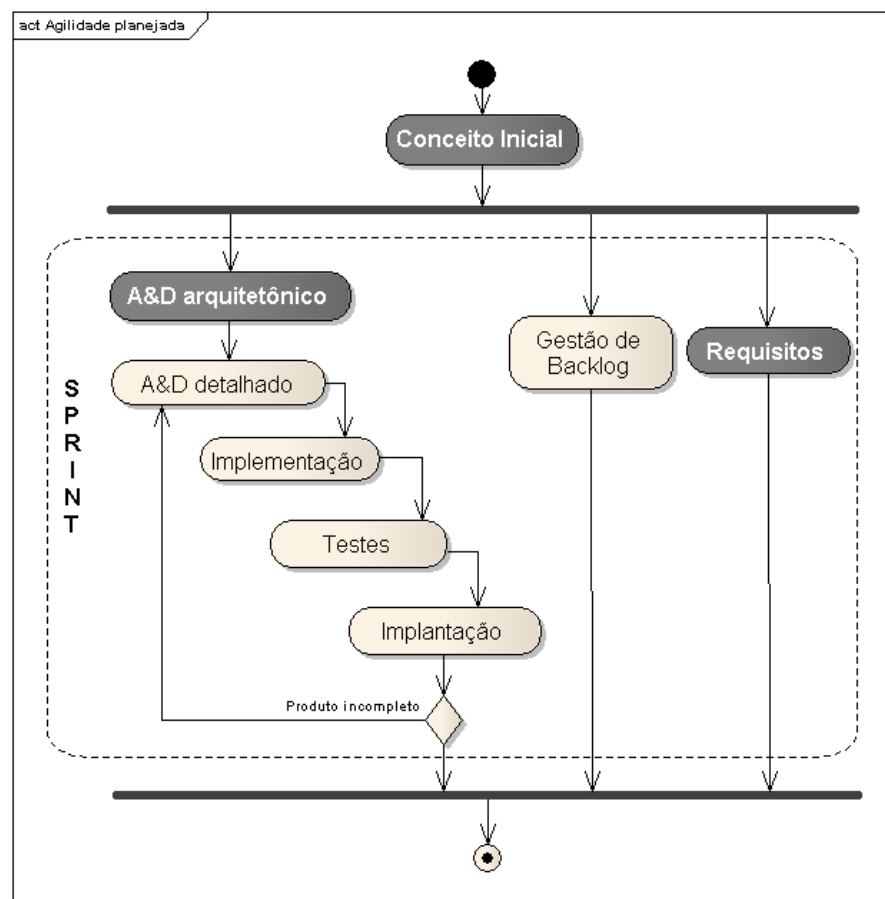


Figura 12 - Modelo de ciclo de vida proposto (A&D significa Análise & Design)

A Tabela 17 apresenta uma breve descrição do propósito de cada subprocesso indicado no modelo de ciclo de vida apresentado na Figura 12.

Tabela 17 - Subprocessos

Subprocesso	Propósito
Conceito Inicial	Este subprocesso se destina a identificar os requisitos do sistema e os aspectos arquiteturais comprometedores, se houver. A partir deste levantamento, aplica-se alguma técnica de medição de funcionalidade e estimativa de esforço para se apresentar uma proposta comercial ao cliente, contendo a definição de escopo e prazo, como é tipicamente feito em projetos dentro do paradigma tradicional.
Análise&Design Arquitetônico	Ocupa-se em elaborar a arquitetura do sistema e mitigar os principais riscos do projeto.
Requisitos	Compreende fundamentalmente atividades de especificação de casos de uso [52] e validação dos mesmos frente ao cliente.
Análise&Design Detalhado	Corresponde às atividades de produção de todos os tipos de artefatos de design não arquitetônico, como realizações de casos de uso e protótipos de interface com usuário.
Implementação	Destinado a realizar atividades de codificação, testes de unidade, integração e empacotamento.
Testes	Corresponde a atividades de planejamento e execução de testes.
Implantação	Atividades de implantação em ambientes de homologação e produção, bem como produção de documentação.
Gestão de Backlog	Corresponde ao trabalho de manter o Product Backlog atualizado gerando e priorizando histórias (<i>user stories</i>) [38].

4.2. Características

A seguir são apresentadas outras considerações sobre o modelo de ciclo de vida do *framework* (Figura 12) com relação à integração dos paradigmas ágil (representado aqui por Scrum) e tradicional:

- **Time-boxing híbrido:** Neste *framework*, pode-se considerar que o *time-boxing* possui caráter híbrido pois, embora se estabeleça ciclos de tempo fixo para potencial entrega de incrementos no produto, a definição de escopo é feita previamente na etapa de *Conceito Inicial*, o que significa que não se tem a mesma situação de inversão de pirâmide mencionada na seção 2.4, ou seja, não é exatamente o caso de se ter recursos e tempo fixos com escopo estimado. Esta

decisão foi tomada com o intuito de atender aos contextos em que se deseja a definição de escopo já no início do projeto.

- **Sprints:** Toda a “macro etapa” pontilhada do ciclo de vida (veja Figura 12) deve ser executada na forma de *sprints* – as iterações de curta duração definidas pelo Scrum. Esta decisão tem o intuito de favorecer a dinâmica e o micro gerenciamento da equipe, uma vez que os incrementos do produto serão menores, bem como os planejamentos e retrospectivas das iterações.
- **Arquitetura:** Pode-se notar que o ciclo de vida (Figura 12) indica que o subprocesso de *Análise&Design arquitetônico* deve ser finalizado em um momento do projeto (quanto antes melhor) e, daí então, o ciclo de vida indica que as atividades subsequentes de análise e design ocorrem em nível detalhado, uma vez que a arquitetura do sistema já se encontra estabilizada. Neste *framework* a arquitetura do software não possui caráter emergente como nas abordagens ágeis. A decisão por dar enfoque rigoroso na elaboração de arquitetura possui em parte o intuito de mitigar os riscos de se incorrer em *design* pobre devido à falta de foco em arquitetura [76], [77], [55], como mencionado no Capítulo 3.
- **Requisitos:** As atividades de especificação e validação podem ocorrer por toda a “macro etapa” pontilhada do ciclo de vida (veja Figura 12), porém tal esforço deve se concentrar nas iterações iniciais do projeto, especialmente no sentido de priorizar os requisitos que tenham impacto na elaboração da arquitetura. Os requisitos funcionais são especificados como casos de uso, pois se deseja manter o princípio da orientação por casos de uso do RUP.
- **Testes:** De modo a contemplar o princípio de ser guiado por casos de uso, a especificação de casos de testes se faz necessária para que se possa assegurar que os casos de uso estão sendo devidamente atendidos, assim como para assegurar níveis aceitáveis de qualidade e formalidade na validação do produto. Esta decisão não exclui a possibilidade da realização de princípios ágeis na atividades de teste tais como testes contínuos e rápido *feedback*, uma vez que o modelo de ciclo de vida do processo de desenvolvimento é organizado em curtas iterações de tempo fixo (*time-boxes*) e, além disso, várias atividades de teste podem ser desenvolvidas de forma mais dinâmica em conjunto com atividades de realização e implementação, tais como testes exploratórios ou testes de unidade.

- **Gestão de Backlog:** O subprocesso denominado *Gestão de Backlog* deve ocorrer por toda a “macro etapa” pontilhada indicada na Figura 12. Para manter o *Backlog* atualizado, este subprocesso deve utilizar informações geradas pelos subprocessos de *Requisitos*, *Análise & Design Arquitetônico* e *Testes*. Neste *framework* o subprocesso *Gestão de Backlog* pode ser visto como a interface de comunicação entre a porção ágil e a porção tradicional do processo.
- **Particionamento ágil e tradicional:** Os subprocessos destacados na Figura 12 – *Conceito Inicial*, *Requisitos*, *Análise&Design arquitetônico* devem possuir caráter tradicional. O subprocesso *Teste* se situa em uma zona indefinida de predominância ágil ou tradicional. Os demais subprocessos possuem caráter ágil, como mostrado na Figura 13. Esta escolha foi tomada pois partiu-se do pensamento de que os objetivos primários do paradigma tradicional (previsibilidade, estabilidade e alta garantia – veja Tabela 2) são mais bem atendidos se a arquitetura for elaborada nas etapas iniciais do processo e se os requisitos forem especificados e validados formalmente.



Figura 13 - Indicação dos subprocessos governados pelos paradigmas ágil e tradicional

O enfoque em arquitetura, bem como a gestão formal de requisitos e a especificação de casos de teste, também podem reduzir a necessidade de profissionais altamente qualificados, geralmente requeridos para se empregar um método ágil com sucesso [79] [5], pois a documentação produzida por estes subprocessos pode servir como guia para membros da equipe menos experientes. Outro problema que também pode ser tratado por essas decisões é com relação ao prejuízo na comunicação entre equipes identificada em estudos anteriores [65] [68]. Este prejuízo de comunicação aparentemente pode ser

Tabela 18 - Artefatos sugeridos

Artefato	Descrição
SRS1	<i>Especificação de Requisitos de Software</i> ou <i>Especificação Funcional</i> – Documento que contém um conjunto de requisitos que define completamente o comportamento externo do sistema a ser desenvolvido. Em particular, este documento SRS1 presume que os requisitos ainda estão apenas levantados, com uma descrição sucinta em detalhe suficiente apenas para se medir o escopo do projeto, que será base para definição de cronograma e custo ao fim da etapa de Conceito Inicial.
SRS2	Evolução do SRS1, contendo detalhamento de requisitos e especificação dos casos de uso para validação.
Arquitetura	<i>Documento de Arquitetura do Sistema</i> – abrange as decisões significantes sobre a organização do sistema e a seleção dos elementos estruturais e suas interfaces pelas quais o sistema é composto juntamente com seu comportamento especificado na colaboração entre tais elementos. Além de estrutura e funcionalidade, também pode especificar questões de usabilidade, funcionalidade, desempenho, resiliência, reuso, inteligibilidade, estética e <i>tradeoffs</i> entre economia e tecnologia.
Modelo de Dados	Modelo de dados do sistema.
<i>Product Backlog</i>	<i>Product Backlog</i> do Scrum [46] - lista de requisitos tipicamente na forma de histórias (<i>user stories</i>). Em uma <i>Sprint</i> tem-se o <i>Sprint Backlog</i> , que é a lista de histórias, priorizadas a partir do <i>Product Backlog</i> , para a tal <i>Sprint</i> .
<i>Release Burndown</i>	<i>Release Burndown</i> do Scrum [46] – representação gráfica dos esforço restante, ao longo do tempo, para execução do projeto. Em nível de <i>Sprint</i> existe o artefato equivalente <i>Sprint Burndown</i> .
Realização	Conjunto de modelos que descreve como um caso de uso é realizado em nível de projeto, em termos de elementos colaborativos.
Interface com usuário	Conjunto de artefatos relacionados à interface com o usuário do sistema, como desenhos de telas, páginas <i>web</i> , mapas de navegação, bem como a realização dos mesmos.
Elemento de implementação	Partes físicas que compõe a implementação do sistema, tais como arquivos de código (fonte, executável), arquivos de dados, diretórios e ajuda <i>online</i> , dentre outros.
<i>Build</i>	Versão operacional do sistema ou parte do sistema que mostra um subconjunto das capacidades a serem providas para produto final.
Caso de teste	Conjunto correlato de cenários de teste e seus elementos (<i>inputs</i> , condições de execução, resultados esperados, etc.) com propósito de verificar algum aspecto do item a ser testado.
Evidência de teste	Relato quantitativo e/ou qualitativo da execução de um teste, com o propósito de evidenciar o resultado obtido no mesmo.
Manual do usuário	Documento que auxilia o usuário a aprender, utilizar e operar o produto.
Guia de implantação	Documento que provê instruções requeridas para instalar o produto em ambiente de produção.

4.4. Papéis sugeridos

A Tabela 19 apresenta os principais papéis sugeridos por este *framework*, que são os responsáveis pelos artefatos sugeridos na Seção 4.3.

Tabela 19 - Papéis

Papel	Responsável pelos artefatos
Analista de Requisitos	SRS1
	SRS2
	Manual do usuário
Arquiteto de Software	Arquitetura
Analista de Banco de Dados	Modelo de dados
Product Owner	<i>Product Backlog</i>
Scrum Master	<i>Release Burndown</i>
Projetista	Realização
Web Designer	Interface com usuário
Implementador	Elemento de implementação
	Guia de implantação
Integrador	<i>Build</i>
Analista de Teste	Caso de teste
Testador	Evidência de teste

4.5. Sumário e Conclusões

Este capítulo apresentou a proposta do *framework* de processo Scrum-RUP desta tese. Este processo foi apresentados a partir das seguintes perspectivas:

- Modelo de ciclo de vida, incluindo os subprocessos envolvidos e *workflow*
- Características
- Artefatos
- Papéis

A essência desta proposta de integração de Scrum em desenvolvimento de software baseado em RUP pode ser entendida como um ponto de corte no espectro ágil-tradicional, acompanhado de uma interface entre as duas partes cortadas. Na “parte tradicional” ficaram os subprocessos *Requisitos* e *Análise & Design Arquitetônico*, enquanto que na

“parte ágil” ficaram os subprocessos *Análise & Design Detalhado*, *Implementação* e *Implantação*, e o subprocesso *Teste* se situa em uma região intermediária dos dois paradigmas, como mostrado na Figura 13. O subprocesso responsável por fazer a interface entre as duas partes é o *Gestão de Backlog*.

5. ESTUDO EMPÍRICO

Este capítulo apresenta o delineamento do estudo empírico realizado com o intuito cumprir o segundo objetivo da pesquisa, que é o de investigar o impacto em produtividade da proposta de combinação Scrum-RUP apresentada no Capítulo 4. Cada seção aborda um aspecto específico do delineamento.

5.1. Contexto

O passo inicial para estabelecer os direcionadores para a estratégia e condução da pesquisa foi a definição do ambiente ou local onde o estudo seria realizado. Foi feito, então, um contato com uma empresa que utilizava um processo baseado em uma customização do RUP e que concordou em disponibilizar dados de projetos desenvolvidos em um intervalo de tempo de cerca de um ano e meio. Dentre estes projetos, alguns seriam desenvolvidos utilizando o mencionado processo RUP customizado e outros seriam desenvolvidos utilizando o novo processo Scrum-RUP.

Para que seja possível tirar conclusões com maior validade ao se agregar evidências sobre estudos empíricos industriais, é importante descrever o contexto no qual eles foram realizados [83].

Boas descrições de contexto são essenciais para permitir que diferentes estudos possam ser comparados. Atualmente, não existe um *framework* que padronize quais atributos devem ser descritos no contexto de uma pesquisa sobre métodos ágeis, embora sua necessidade já tenha sido reconhecida [55].

Para descrever o contexto desta pesquisa, foi utilizada a *ckecklist* proposta recentemente em [83], que define as seis facetas nas quais o contexto deve ser descrito (Figura 15): empresa, mercado, processo, produto, pessoas e práticas/ferramentas/técnicas.

As informações sobre empresa, mercado e processo (incluindo práticas/técnicas) serão discutidas nesta seção, enquanto que as informações de produto, pessoas e ferramentas serão mostradas no Capítulo 6, na Seção 6.1.

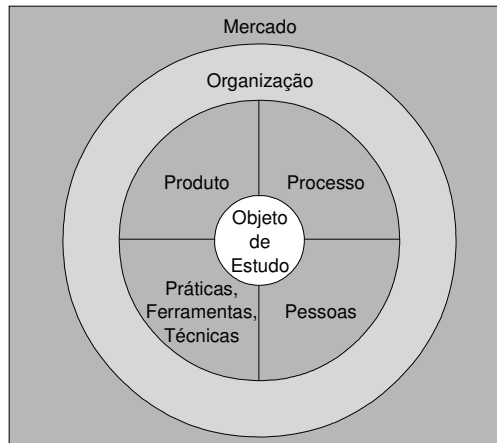


Figura 15 – Facetas de contexto industrial em pesquisas de Engenharia de Software [83]

5.1.1. Empresa

O estudo foi realizado em uma empresa brasileira de Tecnologia da Informação (TI) de tamanho médio (cerca de 200 colaboradores) com unidades localizadas nas cidades de Uberlândia-MG e São Paulo-SP, cujas atividades são focadas em desenvolvimento de projetos de software customizados, consultoria e *outsourcing* de serviços de TI.

Ela foi certificada em 2006 como uma empresa CMMI-ML2. Mesmo antes de iniciar os esforços para a obtenção do CMMI-ML2, a empresa executava seus projetos de desenvolvimento de software usando uma customização do RUP aderente às necessidades da empresa, e posteriormente, aderente às práticas do CMMI.

Os colaboradores são em sua maioria assalariados ou prestadores de serviço. A empresa não trabalha com sistema de recompensa.

5.1.2. Mercado

Os clientes da empresa são grandes corporações que fazem uso intensivo de software para viabilizar seus negócios. Em sua maioria, seus clientes estão sediados no eixo Rio-São Paulo. Como exemplo, citamos: globo.com, R7, UOL, VIVO, BoldCron, Claro, TecBan, iG, Oi e Nokia Siemens.

Seus clientes geralmente exigem negociação e planejamento prescritivos para acordo contratual formal e antecipado. Muitos de seus clientes não trabalham com uma estreita interação com o fornecedor durante o projeto. Além disso, muitos de seus clientes não pretendem trabalhar com o modelo de data fixa e escopo estimado como recomendado pela filosofia ágil (veja Figura 7 na Seção 2.4). Ao invés disso, eles preferem fixar escopo

contratualmente, estimando custo e prazo. Todos os projetos incluídos neste estudo seguiram este modelo contratual.

A empresa é altamente experiente no gerenciamento de requisitos por meio de casos de uso (até porque utilizam RUP há vários anos) e estimação de esforço para desenvolvimento de software utilizando Pontos de Função como base para medição de funcionalidade de software.

5.1.3. Processo

Esta pesquisa se propõe a comparar a produtividade do processo Scrum-RUP proposto no Capítulo 4 com a produtividade do processo anteriormente utilizado pela empresa, que é um processo baseado em uma customização/simplificação do RUP, o qual será chamado, a partir deste ponto, simplesmente de RUP. Sendo assim, é importante descrever as diferenças entre os processos RUP e Scrum-RUP utilizados na empresa, para que se possa ter uma noção precisa do que foi modificado e, assim, poder tirar conclusões mais válidas sobre o impacto em produtividade.

As diferenças dos processos RUP e Scrum-RUP, a partir dos principais pontos de vistas destacados no Capítulo 4, são apresentadas a seguir:

- **Modelo de ciclo de vida:** o processo baseado em RUP usado pela empresa segue o modelo de ciclo de vida comumente proposto pela filosofia RUP (veja Seção 2.6). Já o modelo de ciclo de vida do processo Scrum-RUP sofreu algumas alterações, conforme mostrado na Seção 4.1. Os subprocessos são os mesmos para ambos os processos RUP e Scrum-RUP.
- **Artefatos:** ambos os processos produzem os mesmos artefatos com graus similares de detalhe. A exceção, entretanto, é que a Realização dos requisitos é consideravelmente menos detalhada no processo Scrum-RUP (menos documentação). A Seção 6.3.2 discute essa diminuição de documentação no contexto específico no qual o estudo desta tese foi realizado.
- **Papéis:** ambos os processos tem os mesmos papéis, exceto que o processo RUP não possui *Product Owner* e *Scrum Master*. Vale a observação de que, na empresa em estudo, alguns dos papéis propostos na Seção 4.4 são mesclados em um só.
- **Características:** outros aspectos que diferem o processo RUP do processo Scrum-RUP é que neste a comunicação e colaboração entre a equipe é mais frequente, e as

iterações são menores (*sprints*). Embora o subprocesso *Gestão de Backlog* tenha sido declarado explicitamente no Capítulo 4, o gerente de qualidade de processos explicou que isso não é uma diferença significativa se comparada ao processo RUP, pois neste já havia uma atividade equivalente de gestão de tarefas a serem atribuídas aos desenvolvedores.

A Tabela 20 apresenta um resumo dos princípios/práticas ágeis e iterativas contempladas pelos processos deste estudo de caso, e também auxilia na identificação de suas diferenças.

Tabela 20 - Princípios ágeis e iterativos contemplados pelos processos

	Desenvolvimento iterativo	XP	Scrum	Processo RUP usado na empresa	Processo Scrum-RUP usado na empresa
Iterações e incrementos (*)(**)	x	x	x	x	x
<i>Releases</i> internas e externas (*)	x			x	x
<i>Time boxing</i> (*)	x	x	x		x
Não se modifica projetos iniciados (<i>planning up front</i>) (*)(**)	x		x	x	x
Cliente <i>on site</i> (*)(**)		x	x		
Interação face a face frequente (*)		x	x		x
Equipes auto organizáveis (*)(**)		x	x		
Processo empírico (*)(**)		x	x		
Disciplina sustentável (*)		x			
<i>Product backlog</i> flexível (*)		x	x	x	x
Tomada de decisão rápida (*)			x	x	x
Integração frequente (*)(**)		x	x		
<i>Design</i> simplificado (*)		x			x
Refatoração (*)(**)		x			
Equipe detém o domínio do código (*)		x			
Estimativa de esforço prévia (**)	x			x	x
Retrospectiva (**)			x		x
Reuniões diárias (**)			x		x
Definição prévia de arquitetura	x			x	x
Gestão formal de requisitos / especificação detalhada de requisitos	x			x	x

Os itens da Tabela 20 foram determinados com base em uma lista proposta por Larman [84] e recentemente usada por Petersen e Wohlin [55]. Também foram incluídas práticas presentes em [75]. Os itens marcados com (*) estão presentes em [84] e os itens marcados com (**) estão presentes em [75].

5.2. Estratégia e visão geral da pesquisa

Diversos fatores influenciam na definição da abordagem (ou estratégia) de pesquisa. Um dos principais direcionadores é a questão (ou questões) que a pesquisa pretende responder. Além disso, a abordagem também é direcionada em virtude das limitações impostas pela natureza da pesquisa, que podem incluir falta de controle sobre variáveis, aspectos éticos, custo, tempo, acesso a informações, dentre outras. Yin [67] apresenta uma proposta parcial para auxiliar na definição da abordagem de pesquisa, a qual é mostrada na Figura 16.

Estratégia	Forma da questão de pesquisa	Exige controle sobre eventos comportamentais?	Focaliza acontecimentos contemporâneos?
Experimento	Como, por quê	Sim	Sim
Levantamento	Quem, o que, onde, quantos, quanto	Não	Sim
Análise de arquivos	Quem, o que, onde, quantos, quanto	Não	Sim/não
Pesquisa histórica	Como, por quê	Não	Não
Estudo de Caso	Como, por quê	Não	Sim

Figura 16 – Definição da estratégia de pesquisa [67]

5.2.1. Questões da pesquisa e proposições

Com o segundo objetivo da pesquisa em mente, que é o de investigar a influência da proposta de combinação Scrum-RUP na produtividade, e tendo em conta que o parâmetro de comparação no contexto da pesquisa será uma customização do processo RUP, este estudo de caso pretende responder a seguinte questão de pesquisa:

Como o processo Scrum-RUP impacta a produtividade de desenvolvimento, em comparação ao processo RUP?

Naturalmente, é desejável responder esta questão sob dois pontos de vista: quantitativo e causal. Assim, a questão apresentada foi subdividida em duas questões de pesquisa:

Q1: Há diferença de produtividade entre equipes RUP e equipes Scrum-RUP? Se sim, quanto é esta diferença?

Q2: Qual a explicação dos resultados das produtividades de equipes RUP e Scrum-RUP?

Com base nas questões da pesquisa, proposições de pesquisa são formuladas. Proposições de pesquisa guiam o pesquisador para a direção na qual procurar por evidências de modo a responder as questões de pesquisa de um estudo de caso [67]. Uma proposição é similar a uma hipótese, especificando o que será esperado como resultado do estudo. As proposições a seguir foram elaboradas para esta pesquisa:

P1 (relacionada à questão Q1): espera-se que haverá ganho de produtividade das equipes Scrum-RUP em relação às equipes RUP. Embora alguns estudos registraram ganho de produtividade da ordem de 40% ou mais, optaremos pela mesma proposição feita por [7], de se esperar 20% de ganho em produtividade, uma vez que o processo Scrum-RUP a ser testado possui caráter híbrido e mantém uma quantidade razoável de documentação a ser produzida pelos desenvolvedores.

P2 (relacionada à questão Q2): para se estabelecer as causas de aumento de produtividade esperadas, tomou-se por base as vantagens dos métodos ágeis apresentadas na Seção 3.2, as quais foram agrupadas em categorias, conforme mostrado na Tabela 21.

Tabela 21 - Síntese das vantagens dos métodos ágeis

Categoria	V01	V02	V03	V04	V05	V06	V07	V08	V09	V10	V11	V12	V13
Melhor comunicação	x	x											
Programação em pares			x						x	x			
Micro gerenciamento				x								x	x
Conforto					x		x		x				
Maior colaboração						x							
Interação com o cliente		x						x					
Diminuição de documentação											x		

Com base nas categorias identificadas, quatro delas foram selecionadas para esta proposição como fatores que se espera que sejam motivos de aumento de produtividade no processo Scrum-RUP. São elas:

- **Melhor comunicação**, pois se entende que pequenos problemas durante o projeto podem ser mais rapidamente resolvidos por meio de uma comunicação mais simples e eficiente.
- **Micro gerenciamento**, pois se entende que pequenas unidades de gerenciamento no projeto são mais fáceis de lidar e, portanto, proporcionam a liberação de incrementos de produtos mais rapidamente, bem como a redução de desperdício.
- **Maior colaboração**, pois ajuda a obter melhores *insights* sobre o projeto e sobre a resolução de problemas.
- **Diminuição da documentação**, pois naturalmente que, se menos artefatos são produzidos, a produtividade do projeto como um todo tende a aumentar.

As categorias “programação em pares” e “interação com o cliente” não foram incluídas aqui como fatores de aumento de produtividade esperados pois, no contexto do estudo de caso, estas práticas não foram utilizadas. Também não foi incluído “conforto” pois não foram identificados motivos convincentes para justificar este aspecto como fator que favorece a produtividade. Porém, independente disso, o estudo de caso naturalmente está aberto à possibilidade de identificar este fator dentre seus resultados.

5.2.2. Abordagem adotada: estudo de caso

As questões de pesquisa fazem com que a mesma tenha um caráter tanto exploratório quanto explanatório. Elas sugerem que se deseja investigar o quanto a produtividade será impactada (**Q1**) e também as causas deste impacto (**Q2**).

Estes objetivos sugerem, idealmente em princípio, um estudo que tenha condições de fornecer resultados conclusivos para generalização estatística⁴. Abordagens que geralmente tem melhor condição de fornecer tais resultados são o experimento e o quase-

⁴ Uma condição necessária para se dizer que um resultado pode ser generalizado estatisticamente, é que o mesmo procedeu da análise estatística de um teste realizado sobre uma amostra significativa de indivíduos. Neste sentido, poder-se-ia, por exemplo, estimar intervalos de confiança para dizer o quanto a produtividade de um processo é maior que a do outro, ou até mesmo dizer o quanto um fator influencia na produtividade.

experimento [85]. Entretanto, como esta pesquisa vai lidar com projetos reais, os tratamentos não serão determinados pelo pesquisador, o que implica que não há controle sobre as variáveis independentes dos projetos analisados e, portanto, os dados são observacionais e não experimentais. Assim, a abordagem do experimento fica descartada. Além disso, a quantidade de projetos disponibilizados para este estudo certamente não seria suficiente para configurar uma amostra significativa e aleatória para análise quantitativa em nível de projetos, o que leva a se descartar também a abordagem de quase-experimento.

Assim, pelo fato desta pesquisa contar com essas restrições e, por se tratar do estudo de um fenômeno contemporâneo no contexto da empresa participante, a abordagem de pesquisa escolhida foi a do estudo de caso, que é uma abordagem de pesquisa empírica que objetiva a investigação de um fenômeno contemporâneo em seu contexto [67]. Estudos de caso não geram os mesmos resultados na forma de relações causais como os experimentos, mas eles fornecem uma compreensão aprofundada do fenômeno em estudo [86], o que fortalece sua contribuição exploratória, podendo oferecer resultados mais ricos para levantamento de hipóteses e trabalhos futuros.

Estudos de caso são particularmente importantes para avaliação industrial de métodos e ferramentas de Engenharia de Software [87]. De modo geral, a abordagem de estudo de caso é bem apropriada para muitos tipos de pesquisa em Engenharia de Software, pois os objetos de estudo são fenômenos contemporâneos, os quais são difíceis de controlar experimentalmente.

5.2.3. Modelo conceitual

Para responder as questões de pesquisa, foi necessário, inicialmente, definir os fatores que exercem influência na produtividade e a forma como eles serão medidos.

Medir produtividade não é uma tarefa trivial e há dezenas de fatores propostos na literatura [88], o que sugere ser impraticável medir todos os fatores possíveis. Uma alternativa, então, é buscar identificar os principais grupos de fatores e daí encontrar indicadores adequados para os mesmos. Para isto, tomou-se como base uma meta análise feita por [89], onde os principais grupos de fatores que influenciam a produtividade são identificados, os quais são sumarizados na Tabela 22.

Tabela 22 - Grupos de fatores que influenciam na produtividade de projetos de software [89]

	[90]	[91]	[92]	[93]	[94]
Tecnologias / ferramentas		x	x	x	x
Equipe / pessoas	X	x	x	x	
Processo		x	x	x	x
Ambiente organizacional					x
Sistema de recompensa					x
Produto	X	x	x		

De acordo com esta meta análise, os fatores de produtividade mais citados são: Tecnologias, Equipe, Processo e Produto, que serão os quatro fatores considerados para este estudo. Pode-se ter uma noção dos outros dois fatores mencionados (ambiente organizacional e sistema de recompensa) na descrição do contexto organizacional da Seção 5.1.

Assim, com base nos fatores escolhidos, foi elaborado o modelo conceitual⁵ que este estudo pretende investigar, o qual é mostrado na Figura 17.

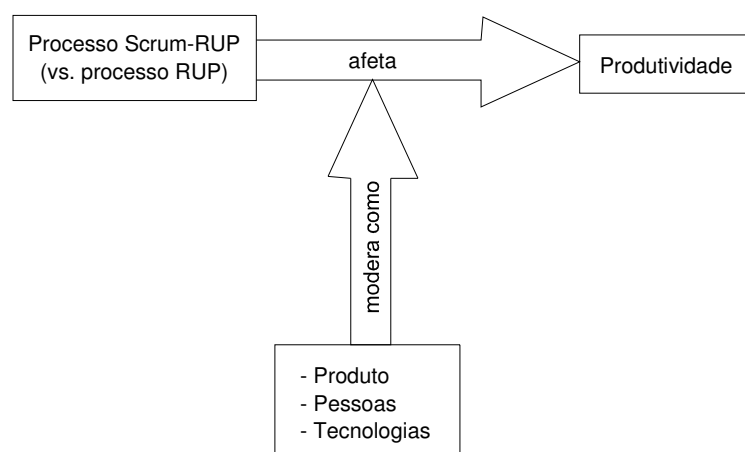


Figura 17 – Modelo conceitual investigado

⁵ Este modelo conceitual segue o mesmo padrão do usado em [104].

5.2.4. Seleção do caso e unidades de análise

O caso selecionado para estudo foi a unidade da empresa localizada na cidade de Uberlândia-MG, por ser localizada na mesma cidade onde residem os pesquisadores, por ser a unidade onde está alocado o gerente de qualidade de processos (que foi o principal contato da empresa) e por atender satisfatoriamente às demandas básicas do estudo, tais como maturidade de processo, número e qualidade de colaboradores e demanda de projetos.

Cada projeto disponibilizado pela empresa corresponderá a uma unidade de análise do estudo de caso.

Assim, considerando os tipos básicos de projetos de estudos de caso apontados por Yin [67], este estudo se enquadra na configuração de projeto de caso único com múltiplas unidades de análise (incorporado), conforme mostrado no canto inferior esquerdo da Figura 18.

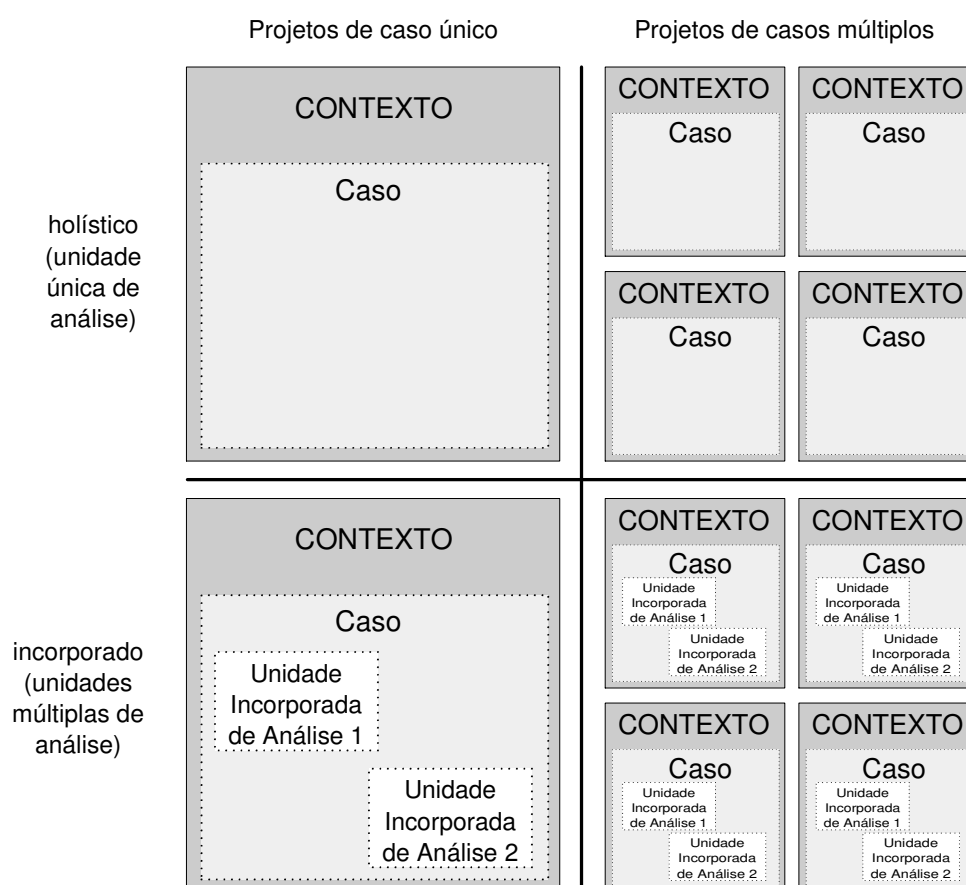


Figura 18 – Tipos básicos de projetos de estudos de caso [67]

Ademais, como as questões de pesquisa pretendem comparar o processo RUP (antes utilizado pela empresa) com o novo processo Scrum-RUP, trata-se de um estudo de caso comparativo, de modo que as unidades de análise serão agrupadas em dois grupos: projetos RUP (que serão identificados pelos nomes R1, R2, etc.) e projetos Scrum-RUP (que serão identificados pelos nomes S1, S2, etc.).

5.3. Coleta de Dados

Este estudo fez uso de diversas fontes de evidências, consistindo de dados quantitativos e qualitativos. O uso de diversas fontes de evidências permite a utilização da técnica de triangulação [67] [95], que compreende uma interação entre as mesmas para sustentar os constructos, proposições ou hipóteses, visando analisar a convergência das fontes de evidência. Além disso, foi feito o uso de dados quantitativos e qualitativos. Dados qualitativos podem ser utilizados para enriquecer resultados quantitativos com informações explanatórias que podem auxiliar no entendimento das causas e no tratamento de questões complexas envolvendo comportamento humano [8].

As fontes de evidências utilizadas neste estudo foram:

- **Medidas de produto** (dados quantitativos)
- **Medidas de processo** (dados quantitativos)
- **Informações contextuais** (dados qualitativos e quantitativos)
- **Entrevistas** (dados qualitativos)

Nas subseções seguintes serão discutidos os aspectos relevantes que tomaram parte na condução do estudo no que diz respeito à coleta de dados.

5.3.1. Modelo de medição para produto e processo

As medidas de produto e processo foram obtidas com base em um subconjunto do modelo de medições utilizado na empresa, que consiste de um modelo de medição de produtividade baseado em dados quantitativos referentes a cada requisito funcional. Em outras palavras, para cada requisito funcional são registrados diversos campos referentes às suas características e ao esforço gasto para sua produção. O subconjunto de medidas e

métricas utilizado neste estudo é mostrado por meio de um Diagrama de Classes da UML [29] na Figura 19.

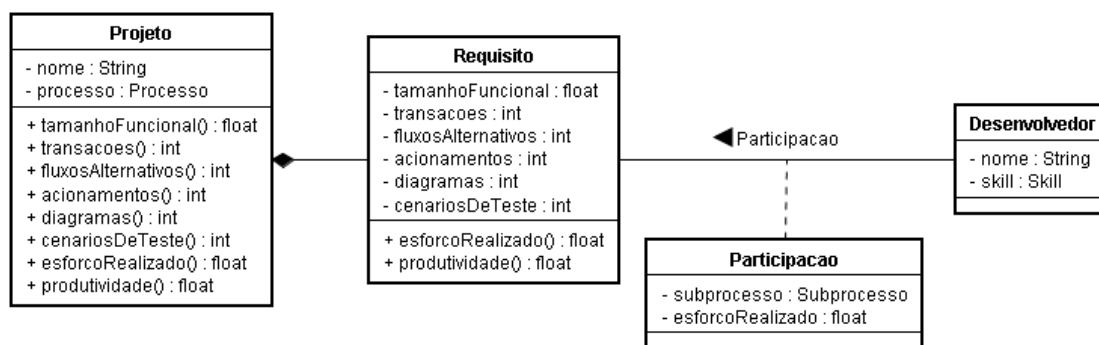


Figura 19 – Subconjunto do modelo de medição utilizado na empresa

A seguir algumas considerações sobre este modelo de medição:

- O registro de uma Participação é que faz o elo entre o desenvolvedor e o requisito: ele diz qual foi a participação do desenvolvedor em um dado requisito, indicando o esforço realizado pelo desenvolvedor em um determinado subprocesso.
- Para efeitos de medição, um projeto nada mais é que um conjunto de requisitos. Isso pode ser observado no modelo pela relação de composição entre Projeto e Requisito.
- No modelo de medição da empresa, a participação de um desenvolvedor na produção de um requisito é na verdade associada com o papel desempenhado por ele, e não com o subprocesso. Entretanto, como a empresa mescla alguns papéis da Tabela 19 (e.g. eles mesclam “Testador” e “Analista de Teste” como um único papel “Analista de Teste”), entendeu-se que seria melhor associar a participação dos desenvolvedores por subprocesso, para fins deste estudo.

A critério de esclarecimento, para melhor entendimento do modelo de medição, a Tabela 23 mostra as medições dos primeiros 5 requisitos do projeto R2. Foram acrescentados os valores do esforço realizado e da produtividade também para fins de entendimento. É importante notar que a entidade **Projeto** possui diversas métricas que, como será mostrado a seguir, são todas calculadas a partir das medidas dos requisitos.

Tabela 23 - Exemplo de medições de requisitos

Requisito #	tamanhoFuncional	transacoes	fluxosAlternativos	Acionamentos	diagramas	cenariosDeTeste	esforcoRealizado()	produtividade()
1	6	5	2	2	3	14	71,00	0,084507
2	7	4	1	3	3	12	83,50	0,083832
3	5	4	1	3	3	12	58,50	0,085470
4	7	7	1	2	4	20	89,50	0,078212
5	5	6	4	3	3	17	56,50	0,088496

A definição das medidas do modelo de medição é mostrada na Tabela 24. As siglas das unidades utilizadas são:

- pf – Ponto de função. O tamanho dos projetos é medido em Pontos de Função (pf), que é uma medida reconhecida pela ISO que serve para mensurar o tamanho do requisito em termos de funcionalidade. Seu cálculo é feito de acordo com o método IFPUG [96].
- ph – Pessoa hora.

Tabela 24 – Definição das medidas usadas no modelo de medição

Entidade	Medida	Tipo	Unidade	Descrição
Projeto	<i>nome</i>	Texto	-	Nome do projeto. Neste estudo, poderá assumir os valores R1, R2, R4, R8, S1, S2, S4, S5, S6 e S8.
Projeto	<i>processo</i>	Processo	-	Processo empregado na execução do projeto. Neste estudo, poderá assumir os valores RUP e Scrum-RUP.
Requisito	<i>tamanhoFuncional</i>	Numérico	pf	Quantidade de pontos de função do requisito (tamanho funcional do requisito).
Requisito	<i>transacoes</i>	Numérico	-	Quantidade de transações do caso de uso correspondente ao requisito.
Requisito	<i>fluxosAlternativos</i>	Numérico	-	Quantidade de fluxos alternativos do caso de uso correspondente ao requisito.
Requisito	<i>acionamentos</i>	Numérico	-	Quantidade de acionamentos do caso de uso correspondente ao requisito.
Requisito	<i>diagramas</i>	Numérico	-	Quantidade de diagramas produzidos na realização (projeto detalhado) do requisito.
Requisito	<i>cenariosDeTeste</i>	Numérico	-	Quantidade de cenários de teste do caso de teste correspondente ao requisito.

Desenvolvedor	<i>nome</i>	Texto	-	Nome do desenvolvedor participante do projeto. Para este estudo, o nome registrado é fictício e um mesmo desenvolvedor pode ser registrado - mais de uma vez se seu nível de skill mudar de um projeto para outro.
Desenvolvedor	<i>skill</i>	Skill	-	Nível de habilidade do desenvolvedor. Pode assumir os valores Júnior, Pleno e Sênior.
Participação	<i>subprocesso</i>	Subprocesso	-	Subprocesso no qual o desenvolvedor participou na produção de um requisito. Pode assumir os valores: R = Requisitos; D = Análise & Design; I = Implementação; T = Testes.
Participação	<i>esforcoRealizado</i>	Númerico	ph	Quantidade de horas gastas pelo desenvolvedor na produção do requisito.

A definição das métricas do modelo de medição é mostrada na Tabela 25:

Tabela 25 - Definição das métricas usadas no modelo de medição

Entidade	Métrica	Tipo	Unidade	Cálculo
Requisito	<i>esforcoRealizado()</i>	Númerico	ph	Soma dos esforços realizados em todas participações de desenvolvedores no requisito.
Requisito	<i>produtividade()</i>	Númerico	pf/ph	<i>tamanhoFuncional</i> / <i>esforcoRealizado()</i>
Projeto	<i>tamanhoFuncional()</i>	Númerico	pf	Somatório de <i>tamanhoFuncional</i> dos requisitos do projeto.
Projeto	<i>transacoes()</i>	Númerico	-	Somatório de <i>transacoes</i> dos requisitos do projeto.
Projeto	<i>fluxosAlternativos()</i>	Númerico	-	Somatório de <i>fluxosAlternativos</i> dos requisitos do projeto.
Projeto	<i>acionamentos()</i>	Númerico	-	Somatório de <i>acionamentos</i> dos requisitos do projeto.
Projeto	<i>diagramas()</i>	Númerico	-	Somatório de <i>diagramas</i> dos requisitos do projeto.
Projeto	<i>cenariosDeTeste()</i>	Númerico	-	Somatório de <i>cenariosDeTeste</i> dos requisitos do projeto.
Projeto	<i>esforcoRealizado()</i>	Númerico	ph	Somatório de <i>esforcoRealizado()</i> dos requisitos do projeto.
Projeto	<i>produtividade()</i>	Númerico	pf/ph	<i>tamanhoFuncional()</i> / <i>esforcoRealizado()</i>

5.3.2. Indicadores

A seguir são apresentados os indicadores utilizados para medir os fatores do modelo conceitual apresentado na Figura 17:

- **Produto:** para medir este fator, serão usadas as medidas dos projetos do modelo de medição mostrado na Subseção 5.3.1., quais sejam: tamanho funcional, número de transações, fluxos alternativos, acionamentos, diagramas e cenários de teste. Serão usadas também as características contextuais de produto sugeridas em [83], tais como domínio da aplicação, tipo de sistema e customização. Dessa forma entende-se que as características dos produtos serão capturadas de forma satisfatória, uma

vez que esses indicadores cobrem aspectos funcionais, não funcionais, e contextuais.

- **Pessoas:** para medir este fator, será utilizado o tamanho da equipe do projeto, bem como os graus de senioridade (júnior, pleno, sênior) de cada colaborador. Além disso, será feito um mapeamento das participações de cada colaborador por senioridade e por subprocesso. A senioridade dos desenvolvedores foi medida de maneira parcialmente subjetiva (por julgamento de *experts* e conforme seu desempenho por métricas de processo) e corresponde ao seu enquadramento funcional pelo departamento de recursos humanos da empresa.
- **Tecnologias:** para medir este fator, foi considerada a linguagem de programação, bem como as ferramentas de desenvolvimento (ide's, frameworks, cases, testes, e de infraestrutura) utilizadas no projeto.
- **Produtividade:** a produtividade é medida conforme especificado no modelo de medição, ou seja, é o tamanho funcional dividido pelo número de pessoas hora. A empresa não trabalha com medição baseada em LOC (linhas de código) por considerar, segundo sua experiência relatada pelo seu gerente de qualidade de processos, que a quantidade de LOC varia muito em função da expertise do desenvolvedor. Assim, haveria um risco muito alto de que o uso de LOC não refletisse o tamanho real do software. Embora o uso conjugado de medidas de produtividade baseadas em LOC e pontos de função acrescentaria mais informação à análise de dados, considerou-se que a limitação de utilizar apenas pontos de função não prejudica os objetivos do estudo, nem sua utilidade industrial, uma vez que o cliente paga por funcionalidade e não por LOC.

Vale lembrar que o fator *Processo* assumirá apenas os dois tratamentos que se deseja comparar: Scrum-RUP (que consiste no processo de desenvolvimento de software descrito no Capítulo 4) e RUP (que consiste no processo de desenvolvimento de software antes utilizado na empresa, cujas diferenças em relação ao processo Scrum-RUP foram descritas na Seção 5.1.3).

5.3.3. Amostragem

Neste estudo, o conceito de amostragem se aplica apenas para as entrevistas, na seleção dos entrevistados. Vale lembrar que o conceito de amostragem não se aplica aos projetos

selecionados para o estudo, uma vez que múltiplos casos (ou, neste caso, unidades de análise) não são unidades de amostragem e não devem ser escolhidos por essa razão, nem devem ser usados para conceber generalização estatística, como mencionado no segundo capítulo de [67].

A Figura 20 resume a cobertura da amostragem, ou seja, o número de entrevistados por projeto, por subprocesso e por nível de senioridade. Note que há imbricações de entrevistados por projeto e por subprocesso, pois o mesmo entrevistado pode participar de mais de um projeto ou mais de um subprocesso. No total, 11 pessoas foram entrevistadas.

Projetos													
R1	R2	R3	R4	R5	R7	S1	S2	S3	S4	S5	S6	S7	S8
2	2	0	2	0	2	1	1	1	2	4	5	4	4

Subprocessos			
Requisitos	Análise & Design	Implementação	Teste
1	2	6	5

Senioridade		
Junior	Plenum	Senior
2	4	5

Figura 20 – Cobertura da amostragem dos entrevistados

Os entrevistados foram selecionados na tentativa de que todos projetos, atividades e níveis de senioridade fossem cobertos. Cada projeto do estudo deveria ser representado por pelo menos duas pessoas, se possível. Pessoas de todos os níveis de senioridade foram selecionadas, porém, como pode ser visto na Figura 20, deu-se preferência aos colaboradores com maior senioridade no intuito de reduzir a possibilidade de vieses de aprendizado de um projeto para outro. As atividades relacionadas com os principais subprocessos (Requisitos, Análise & Design, Implementação e Testes) também foram cobertas, porém deu-se preferência em Implementação e Testes porque foi assumido que estes subprocessos são os mais afetados pelo novo processo Scrum-RUP.

5.3.4. Elaboração da entrevista

Entrevistas podem ser estruturadas, não estruturadas ou semi-estruturadas [95]. Foi escolhido o formato semi-estruturado devido ao fato de que tanto questões abertas como de

múltipla escolha ou dicotômicas eram desejadas, dependendo do propósito. As questões abertas foram empregadas geralmente na primeira questão de um determinado tema com o objetivo de exercer menor influência na resposta do entrevistado e abrir a possibilidade do surgimento de novas respostas não pensadas pelo pesquisador [97]. Quando não há necessidade de questão aberta, foi dada preferência para questões de múltipla escolha ou dicotômica, para tratar especificamente os pontos desejados, bem como para facilitar e reduzir vieses no processo de análise [97].

Com o objetivo de atender aos objetivos da pesquisa e, em especial, de auxiliar na resposta da questão de pesquisa **Q2** (veja Seção 5.2.1), a entrevista foi arquitetada nas seguintes partes:

1. **Informações iniciais:** são fornecidas instruções ao respondente, bem como informações para auxiliá-lo a recordar dos projetos desenvolvidos com o processo Scrum-RUP.
2. **Classificação do respondente:** são perguntadas informações básicas sobre a experiência do respondente.
3. **Entendimento do processo:** são feitas algumas perguntas para verificar se o respondente entendeu o processo Scrum-RUP e sua diferença com o processo RUP antes utilizado.
4. **Impressões do processo Scrum-RUP:** é desenvolvida uma sequência de perguntas para saber as razões do aumento de produtividade do processo Scrum-RUP, de acordo com o julgamento do respondente. O levantamento das causas do aumento de produtividade foi feito em três etapas:
 - A primeira consistiu em uma pergunta aberta para explorar as impressões do entrevistado sem influenciá-lo.
 - A segunda etapa consistiu de duas questões estruturadas mostrando uma lista de possíveis fatores para serem classificados, com o objetivo de ajudar o entrevistado a lembrar de fatores que por ventura não tenha identificado na primeira etapa. A lista de fatores foi parcialmente derivada de [88].
 - A terceira etapa solicitou do entrevistado uma lista consolidada de fatores que afetaram a produtividade, ordenados por importância.

O item 4 da entrevista pressupõe que houve aumento de produtividade nas equipes Scrum-RUP. Isso significa que a entrevista só terá sentido de ser aplicada se os dados quantitativos indicarem referido aumento de produtividade.

5.3.5. Procedimentos de coleta de dados

5.3.5.1. Medidas de produto e processo

A coleta dos dados quantitativos, ou seja, das medidas numéricas da Tabela 24, é feita de maneira semi-automatizada. A empresa conta com um sistema informatizado o qual possui uma base de dados com o modelo de medição completo implementado. Este sistema fica acessível aos desenvolvedores do projeto, de modo que eles mesmos fazem o lançamento dos dados. Assim, por exemplo, um implementador, ao concluir uma determinada tarefa, faz o registro manualmente do instante em que a terminou. Da forma semelhante, os dados de produto (exemplos: número de transações de um requisito, número de cenários, etc.) são preenchidos por aquele colaborador responsável pelo dado em questão.

5.3.5.2. Dados contextuais

Os dados contextuais foram obtidos por meio de registros históricos que a empresa mantém a partir da realização de seus projetos. Assim, não houve participação ativa do pesquisador e o procedimento consistiu de simples acesso aos registros.

5.3.5.3. Entrevistas

Para realizar a entrevista, foi construído um questionário, o qual está disponível no Anexo B. O questionário foi aplicado pelo próprio gerente de qualidade de processos da empresa, que apresentava os objetivos do estudo ao respondente, depois dava instruções de preenchimento e, em seguida, solicitava que retornasse o questionário.

Inicialmente o questionário foi apresentado na forma impressa, mas os próprios entrevistados solicitaram que o mesmo fosse fornecido na forma eletrônica, de modo que pudessem preencher o questionário no próprio computador. Foram dadas instruções específicas sobre a importância de seguir a ordem das questões [97] e não “espiar” as questões subsequentes, sob o risco de invalidar o instrumento.

Após o retorno dos respondentes, o gerente de qualidade de processos reuniu os questionários e os enviou em sua forma bruta para o pesquisador.

5.3.6. Cronologia da coleta de dados

O desenvolvimento da coleta de dados, do ponto de vista cronológico, é mostrado na Figura 21. Pode-se notar que as entrevistas foram aplicadas após a finalização dos projetos deste estudo. Embora a decisão de aplicar as entrevistas só no final possa ter a desvantagem de que os respondentes possam ter graus variados de esquecimento dos projetos, por outro lado tem a vantagem de poder reduzir vieses de ansiedade ou diferenças de atenção dos desenvolvedores pelo fato de estarem sendo observados.

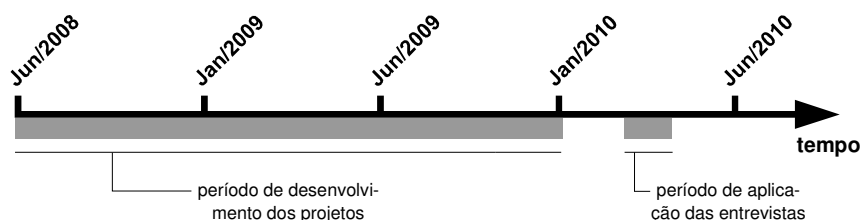


Figura 21 – Cronologia da coleta de dados

5.4. Análise de Dados

A análise das evidências de um estudo de caso é um dos aspectos menos desenvolvidos e mais complicados ao realizar estudos de caso [67]. A análise de dados deve ser previamente planejada e explicitada no trabalho [98], ou seja, deve ser especificada como parte do protocolo de estudo de caso [67].

Pode-se encontrar em [67] recomendações de algumas estratégias gerais, bem como de algumas técnicas analíticas específicas para análise de dados em estudos de caso. A estratégia geral escolhida foi a de “basear-se em proposições teóricas”, que determina que a análise de dados seja orientada pelas proposições do estudo de caso, relacionadas às questões de pesquisa (veja Seção 5.2.1). A técnica analítica específica de “adequação ao padrão” foi utilizada neste estudo, que consiste em comparar um padrão fundamentalmente empírico com outro de base prognóstica (baseado nas proposições), de modo que, se os padrões coincidirem, os resultados podem ajudar o estudo de caso a reforçar sua validade interna [67].

Com base na estratégia geral e técnica analítica específica escolhida, e considerando todas as quatro fontes de evidências utilizadas no estudo (veja Seção 5.3), foi elaborada uma tática geral para análise de dados, a qual é mostrada na Figura 22, com vistas a verificar as proposições de pesquisa **P1** e **P2** (veja Seção 5.2.1). Essencialmente, as medidas quantitativas de processo tem o intuito de medir a produtividade para auxiliar a investigar a proposição **P1**, enquanto que as informações contextuais e as medidas de produto tem o intuito de auxiliar na comparação dos projetos e na acareação dos resultados obtidos. Já os dados qualitativos das entrevistas tem o intuito de enriquecer a compreensão do fenômeno em estudo, a fim de auxiliar na investigação da proposição **P2**, além de serem usados para triangulação com os resultados obtidos na fase anterior da análise.

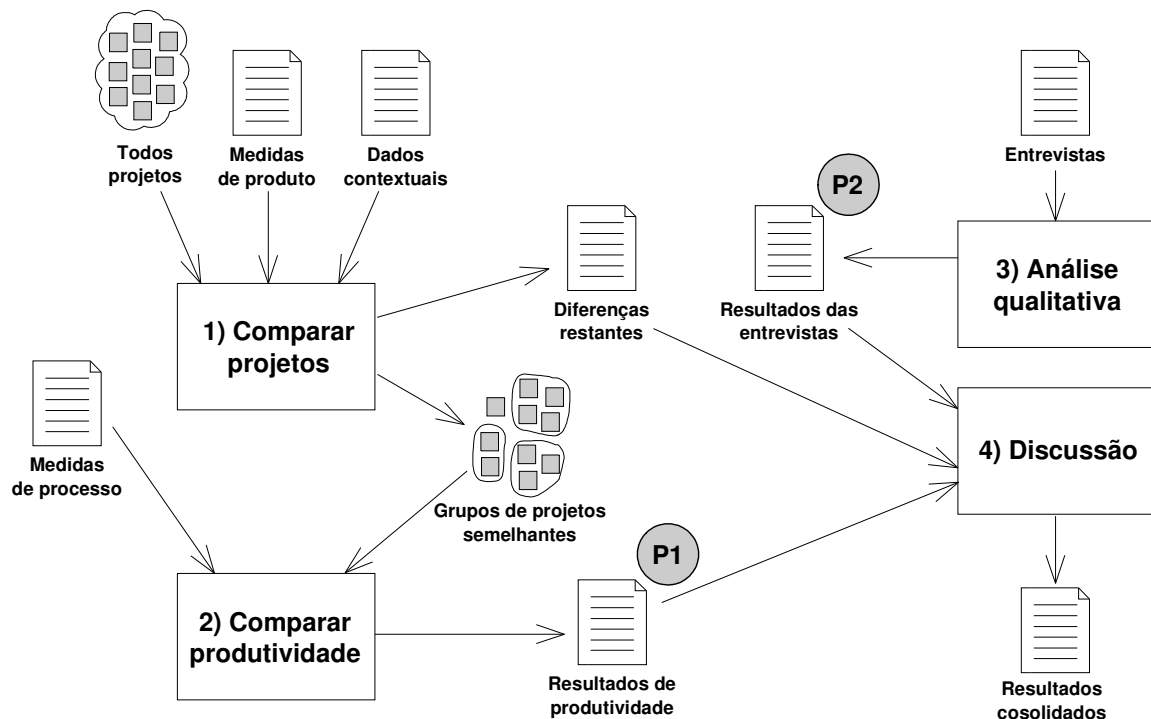


Figura 22 – Tática geral para análise dos dados

A seguir uma breve descrição dos procedimentos de análise de dados mostrados na Figura 22:

1) Comparar projetos

O objetivo deste procedimento é a separação dos projetos em agrupamentos que possuam tratamentos semelhantes com relação aos três fatores moderadores do efeito na produtividade (Produto, Pessoas e Tecnologia - veja Figura 17), de modo

a fazer o máximo possível para isolar estes fatores e selecionar projetos semelhantes para comparação.

Como os dados são observacionais (sem controle dos fatores por parte do pesquisador), os agrupamentos não foram previamente definidos antes da análise de dados, mas sim irão emergir durante a análise de dados de acordo com as medidas relevantes encontradas.

Poderá ainda haver, entre os projetos, diferenças que não sejam tão significativas a ponto de haver necessidade de separá-los em agrupamentos por causa dessas diferenças. Nestes casos, tais diferenças deverão ser anotadas como “Diferenças restantes”, para discussão posterior.

Neste procedimento de comparação de projetos, a subjetividade deverá ser evitada ao máximo, até mesmo com o uso de ferramentas estatísticas para auxiliar a encontrar diferenças significativas.

2) Comparar a produtividade

A produtividade dos projetos RUP e Scrum-RUP será comparada quantitativamente para cada um dos agrupamentos identificados na comparação dos projetos.

3) Análise qualitativa

Será realizado um procedimento de análise qualitativa com os dados das entrevistas. Para as perguntas de múltipla escolha e dicotômicas da entrevista, o processo de análise é relativamente direto, consistindo da tabulação dos resultados em planilhas/matrizes, seguida da priorização dos itens mais frequentes. Para as questões abertas, entretanto, será usado o processo de codificação [95], que consiste no seguinte: a partir dos dados brutos (a resposta de uma questão aberta em sua forma original), partes da narrativa (palavras, frases ou mesmo parágrafos) são marcadas com um código que represente uma categoria de interesse para o estudo. Como será utilizada a técnica analítica de *adequação ao padrão*, quatro códigos serão previamente definidos conforme a proposição **P2**:

- Melhor comunicação
- Diminuição da documentação
- Maior colaboração

- Micro gerenciamento

Os demais códigos serão exploratórios, ou seja, deverão emergir de acordo com as respostas dos respondentes.

Para reduzir vieses de pesquisador, além da definição deste protocolo de análise, será utilizada a técnica de triangulação de pesquisadores, por meio da qual dois pesquisadores fazem a análise isoladamente e, em seguida, os resultados são cruzados e os itens divergentes são discutidos até se chegar a um consenso, similarmente como feito em [16].

4) Discussão

De posse dos resultados quantitativos de produtividade, das diferenças restantes detectadas entre os projetos, e dos resultados das entrevistas, estes dados serão discutidos com o intuito de fazer uma acareação dos resultados obtidos por meio de triangulação e levantar as possibilidades de influência dos fatores na produtividade, de acordo com os dados.

5.5. Ameaças à validade

Estudos com componentes empíricos possuem ameaças à validade. Entretanto, o sucesso de um estudo empírico pode ser mais bem assegurado se essas ameaças forem identificadas antecipadamente de modo que ações para mitigá-las possam ser realizadas. Há quatro tipos de ameaças à validade em estudos de caso [67]: validade do constructo, validade interna, validade externa e confiabilidade (ou validade da conclusão).

5.5.1. Validade do constructo

Validade do constructo está relacionada com a obtenção das medidas corretas dos conceitos estudados. Uma das ameaças à validade do constructo é em relação às medidas dos fatores moderadores do efeito na produtividade. Com relação ao produto, embora tenham sido usados indicadores baseados em informações contextuais e medidas quantitativas, é possível que outras características técnicas não mencionadas possam influenciar na complexidade do produto, além do que a complexidade pode estar relacionada à expertise dos desenvolvedores [99]. Especificamente sobre a medida de

tamanho do software (que também é o fundamento para se medir a produtividade), a medida utilizada foi baseada somente em pontos de função não ajustados, que mede o tamanho do software em termos de funcionalidade, porém negligencia suas características não funcionais. Uma alternativa para isto seria o uso do Fator de Ajuste de Valor⁶ (VAF), porém estudos empíricos indicaram que este fator de ajuste não apresenta correspondência à realidade [100] [101]. Além disso, pode-se considerar que a medida do tamanho de um software pode ser melhor obtida por meio da combinação de diversas medidas [102]. Duas iniciativas foram tomadas para se mitigar as ameaças à validade na medição do produto: (1) realizou-se o agrupamento de projetos para que apenas projetos semelhantes fossem comparados, e (2) foram utilizados os dados das entrevistas com o intuito de realizar uma acareação dos resultados por meio de triangulação.

Outra ameaça é com relação aos indicadores usados para medir as pessoas, que podem ser questionados com relação ao seu poder de capturar características específicas relevantes à capacidade e experiência da equipe [88]. Para mitigar estas ameaças, foram utilizados os dados das entrevistas com o intuito de realizar uma acareação dos resultados por meio de triangulação.

5.5.2. Validade interna

Validade interna se aplica mais a pesquisas explanatórias e se refere à relação causal entre os tratamentos e os resultados. Este estudo de caso, embora predominantemente exploratório, possui um leve caráter explanatório, mesmo que no sentido de levantar possíveis causas para a geração de hipóteses para outros estudos. Por este motivo, é importante considerar a validade interna. Uma das ameaças à validade interna deste estudo é com relação à escolha dos fatores que influenciam na produtividade e, conseqüentemente, à possível existência de fatores de confusão. Há um risco de que os resultados de produtividade não sejam apenas devido aos fatores determinados no modelo conceitual (veja Seção 5.2.3). As entrevistas tem o intuito de mitigar esta ameaça, de modo

⁶ No processo de estimativa do tamanho de software por Análise de Pontos de Função (APF), uma das etapas finais é multiplicar a medida de pontos de função por um Fator de Ajuste de Valor (VAF), obtido com base em 14 fatores denominados Características Gerais de Sistemas (CGS). O VAF varia entre 0,65 a 1,35 e é uma proposta para ajustar a medida do tamanho do software em conseqüência de suas características não funcionais. Entretanto, estudos indicaram que o VAF não apresenta correspondência à realidade (LOKAN, 2000) (CALAZANS, LISBOA e OLIVEIRA, 2005).

que os respondentes possam expressar suas percepções sobre os efeitos de produtividade e, assim, identificar outras causas inicialmente não pensadas no estudo. Além disso, a proposição **P2** tem o intuito de prognosticar as causas do aumento de produtividade com base nos resultados da literatura, de modo que, se forem confirmadas pelos resultados, proverá maior solidez às conclusões sobre as causas.

5.5.3. Validade externa

Validade externa diz respeito a saber se os resultados são generalizáveis além do estudo realizado, o que constitui um grande obstáculo ao se realizar estudos de caso [67], devido à sua limitação a um dado contexto. Deve-se então estabelecer o domínio ao qual as descobertas de um estudo de caso podem ser generalizadas. Para isto, buscou-se descrever uma boa descrição do contexto, abarcando todas as facetas identificadas em [83]. Além disso, fez-se o uso de proposições teóricas de modo a fazer replicações dos resultados existentes de outros estudos.

Pesquisadores se tornam mais confiantes em uma teoria quando resultados similares emergem de diferentes contextos [87]. O registro de variáveis contextuais de vários estudos empíricos permite aos pesquisadores a construção de evidências por meio de uma família de estudos. A replicação de estudos de caso pode tratar algumas ameaças à validade [103].

5.5.4. Confiabilidade

A confiabilidade diz respeito a até que ponto um pesquisador pode seguir os mesmos procedimentos do estudo e chegar às mesmas constatações e conclusões. A ameaça identificada aqui é o viés de pesquisador nos procedimentos do estudo. No caso da amostragem da entrevista, buscou-se cobrir todas as características dos entrevistados conforme mencionado na Seção 5.3.3. Os procedimentos de coleta de dados sofreram pouca participação do pesquisador, uma vez que os dados quantitativos eram registrados pelos próprios desenvolvedores, as informações contextuais eram obtidas em registros históricos e as entrevistas foram feitas por meio de questionário enviado aos entrevistados (veja Seção 5.3.5). O procedimento mais ameaçado pelo viés de pesquisador é o de análise de dados, particularmente os procedimentos de comparação dos projetos e análise qualitativa de dados. Para reduzir vieses na comparação dos projetos, foram usadas

ferramentas estatísticas e ferramentas de filtragem e ordenação de dados sempre que aplicáveis. Já na análise qualitativa, na qual a codificação pode sofrer vieses de acordo com a experiência e conhecimento do pesquisador, as seguintes ações foram tomadas para reduzir esta ameaça:

- Foi elaborada a descrição detalhada do protocolo de estudos de caso, que inclui a descrição detalhada dos procedimentos de análise de dados.
- Foi incluído no protocolo o procedimento de triangulação de pesquisador (veja Seção 5.4), incluindo um membro da academia (autor desta tese) e um membro da indústria (gerente de qualidade de processos da empresa participante), de modo a cobrir as duas áreas e enriquecer a discussão de consenso na triangulação.
- Foi feito um procedimento de verificação dos resultados da análise junto ao gerente de qualidade de processos da empresa para que este verificasse se a realidade estava aparentemente ali refletida.

5.6. Resumo e conclusões

Este capítulo apresentou o delineamento do estudo de caso industrial realizado com o intuito de avaliar o impacto em produtividade da proposta Scrum-RUP desta tese. A seguir os principais pontos:

- O contexto do estudo foi descrito fornecendo informações sobre a empresa, seu mercado e processo, como recomendado por [83]. As demais facetas contextuais (produto, pessoas e ferramentas) serão descritas no próximo capítulo pois estão intimamente ligadas à análise de dados.
- Os dados quantitativos e qualitativos do estudo foram descritos, assim como os procedimentos de coleta e análise de dados a serem aplicados.
- As ameaças à validade foram identificadas e discutidas, inclusive as atitudes tomadas no intuito de mitigá-las.

A descrição apropriada do delineamento do estudo é fundamental para assegurar o entendimento do mesmo, bem como para evidenciar a qualidade do estudo.

6. RESULTADOS

Este capítulo apresenta os resultados da análise de dados deste estudo de caso, bem como a discussão dos mesmos. As seções do capítulo seguirão a estrutura dos procedimentos estabelecidos na tática geral de análise mostrada na Seção 5.4 (veja Figura 22).

A empresa disponibilizou os dados de 16 projetos para análise, os quais são divididos em dois grupos:

- **Projetos RUP:** 8 projetos foram desenvolvidos utilizando o processo RUP ao longo do período de Junho/2008 a Junho/2009. Estes projetos foram identificados por R1, R2, R3, R4, R5, R6, R7, R8.
- **Projetos Scrum-RUP:** os outros 8 projetos foram desenvolvidos utilizando o processo Scrum-RUP (proposta desta pesquisa - veja Capítulo 4) ao longo do período de Setembro/2008 a Dezembro/2009. Estes projetos foram identificados por S1, S2, S3, S4, S5, S6, S7, S8.

A Figura 23 a seguir mostra a cronologia do desenvolvimento dos projetos do estudo. A visão cronológica dos projetos auxilia no entendimento do estudo, bem como na discussão de possíveis vieses de aprendizado.

	2008							2009											
PROJETO	jun	jul	ago	set	out	nov	dez	jan	fev	mar	abr	mai	jun	jul	ago	set	out	nov	dez
R1																			
R2																			
R3																			
R4																			
R5																			
R6																			
R7																			
R8																			
S1																			
S2																			
S3																			
S4																			
S5																			
S6																			
S7																			
S8																			

Figura 23 – Períodos em que os projetos do estudo foram desenvolvidos

6.1. Comparação dos projetos

Esta seção apresenta o resultado da separação dos projetos em agrupamentos que possuam tratamentos semelhantes com relação aos três fatores moderadores do efeito na produtividade (Tecnologia, Produto e Pessoas – veja Figura 17), bem como um levantamento das “Diferenças restantes” (veja Figura 22) entre os projetos considerados semelhantes.

6.1.1. Tecnologia

O primeiro critério adotado para agrupamento de projetos semelhantes foram os indicadores da tecnologia utilizada em cada projeto (a linguagem e as ferramentas utilizadas), os quais são mostrados na Tabela 26 (consulte o Anexo D para breve explanação das siglas das ferramentas dos projetos).

Tabela 26 - Tecnologias empregadas nos projetos

Projeto	Linguagem	Eclipse	MSVS	EA	Subversion	Maven	EJB Container	Erwin	Spring	Velocity	JSF	Jboss	Hibernate	Jetty	Jmeter	JUnit	Selenium	PowerDesigner	Struts
R1	Java EE	X		x	x	x	x	x	x	x		x	x					x	
R2	Java EE	X		x	x	x	x	x	x	x		x	x	x	x			x	x
R3	.NET		x		x													x	
R4	Java EE	x		x	x	x	x	x	x	x		x	x	x	x	x		x	x
R5	.NET		x		x													x	
R6	C++	x		x	x			x											
R7	49% Java EE; 51% C++	x		x	x							x	x					x	x
R8	Java EE	x		x	x	x	x	x	x	x		x	x	x		x	x		x
S1	Java EE	x		x	x	x	x				x	x	x			x	x		x
S2	Java EE	x		x	x	x	x				x	x	x			x	x		x
S3	.NET		x		x													x	
S4	Java EE	x		x	x	x	x				x					x			
S5	Java EE	x		x	x	x	x				x				x			x	
S6	Java EE	x		x	x	x	x			x	x	x	x				x	x	x
S7	65% Java EE; 35% C++	x		x	x	x						x	x				x		x
S8	Java EE	x		x	x	x	x					x	x	x		x		x	

A linguagem, por ser o indicador mais relevante, foi utilizada para montar os agrupamentos, enquanto que as diferenças no uso de ferramentas foram anotadas como diferenças restantes. A seguir algumas considerações sobre a análise:

- **Linguagem:**
 - Para cada linguagem, se existe pelo menos um projeto de cada processo (RUP e Scrum-RUP), um agrupamento desta linguagem é formado.
 - O projeto R6 foi excluído da análise pois ele era o único de linguagem C++.
- **Ferramentas:**
 - Foi feita uma análise manual buscando identificar, para cada agrupamento, as ferramentas que possam ter um efeito espúrio na diferença de produtividade dos projetos RUP e Scrum-RUP.

O resultado da comparação é mostrado na Tabela 27.

Tabela 27 - Resultado da comparação dos projetos com relação ao fator tecnologia

Agrupamento	Projetos	Diferenças restantes
1 - Projetos Java EE	R1, R2, R4, R8, S1, S2, S4, S5, S6, S8	1) Apenas projetos RUP usam Erwin, Spring e Velocity (exceção: S6 usa Velocity). 2) Apenas projetos Scrum-RUP usam JSF (exceção: S8 não usa JSF). 3) Há outras pequenas diferenças no uso de ferramentas, porém sem correlação aparente com o uso dos processos RUP e Scrum-RUP.
2 - Projetos .NET	R3, R5, S3	(nenhuma)
3 - Projetos Java EE / C++	R7, S7	1) Percentualmente, a porção Java EE de S7 é um pouco maior que a de R7. 2) Somente S7 usa Maven e Selenium. 3) Somente R7 usa PowerDesigner.

6.1.2. Produto

O próximo critério utilizado para refinar e/ou subdividir os agrupamentos da Tabela 27 foram os indicadores do fator produto (as informações contextuais dos projetos, bem como suas medidas quantitativas). As informações contextuais dos projetos, de acordo com o

checklist de [83], são mostradas na Tabela 28 (informações sobre maturidade dos produtos não foi aplicável, e informações sobre a qualidade não foram disponíveis).

Tabela 28 - Informações contextuais dos projetos

Projeto	Domínio da aplicação	Qte de requisitos	Tamanho (pf)	Tipo de sistema	Customização
R1	Portal Conteúdo e e-commerce	39	169	Aplicação web	Não
R2	Financeiro	27	100	Aplicação web	Não
R3	Financeiro	60	298	Aplicação web	Não
R4	Telecom	28	169	Aplicação web	Não
R5	Financeiro	29	155	Aplicação web	Não
R7	Financeiro	24	152	Aplicação web	Não
R8	Infra estrutura de rede	34	176	Aplicação web	Sim
S1	Portal Conteúdo e e-commerce	18	80	Aplicação web	Não
S2	Telecom	25	123	Aplicação web	Não
S3	Portal Conteúdo e e-commerce	24	128	Aplicação web	Não
S4	Portal Conteúdo e e-commerce	40	185	Aplicação web	Não
S5	Telecom	32	159	Aplicação web	Não
S6	Portal Conteúdo e e-commerce	45	234	Aplicação web	Não
S7	Financeiro	48	249	Aplicação web	Não
S8	Portal Conteúdo e e-commerce	28	168	Aplicação web	Não

As informações contextuais foram utilizadas para refinar e/ou subdividir os agrupamentos, enquanto que as diferenças nas medidas quantitativas foram anotadas como diferenças restantes. A seguir algumas considerações sobre a análise:

- **Informações contextuais:**
 - O projeto R8 foi excluído da análise por ser de um domínio muito específico (infraestrutura de rede) e por ser o único com customização.
 - O agrupamento “1” (veja Tabela 27) foi subdividido em domínios *Telecom* e *Portal Conteúdo e e-commerce*.
 - Embora o agrupamento “2” (veja Tabela 27) esteja comparando projetos de domínios de aplicação diferentes, não foi possível subdividi-lo devido à inexistência de uma combinação de mesmo domínio da aplicação.

- **Medidas quantitativas:**

- A comparação dos projetos com relação às medidas quantitativas foi feita com base na comparação estatística das distribuições de cada um dos 6 atributos dos requisitos dos projetos (veja Figura 19).
- O teste não paramétrico de *Kruskal-Wallis* foi utilizado, porque as distribuições de todos os atributos não atenderam as condições para o uso da alternativa paramétrica *ANOVA One-way* – veja Anexo E. O software utilizado foi o Minitab 16.

O resultado da comparação com relação ao fator produto, já combinada com a comparação com relação ao fator tecnologia, é mostrado na Tabela 29.

Tabela 29 - Resultado da comparação dos projetos com relação aos fatores tecnologia e produto

Agrupamento	Projetos	Diferenças restantes
1 - Projetos Java EE	R1, R2, R4, S1, S2, S4, S5, S6, S8	1) Apenas projetos RUP usam Erwin, Spring e Velocity (exceção: S6 usa Velocity). 2) Apenas projetos Scrum-RUP usam JSF (exceção: S8 não usa JSF). 3) Há outras pequenas diferenças no uso de ferramentas, porém sem correlação aparente com o uso dos processos RUP e Scrum-RUP. 4) R4, S2, S5 = <i>Telecom</i> ; R1, S1, S4, S6, S8 = <i>Portal Conteúdo e e-commerce</i> . 5) S2: difere dos outros no número de cenários de teste dos requisitos. 6) R2 e S8: difere dos outros no tamanho funcional dos requisitos.
1.1 – Projetos Java EE; Telecom	R4, S2, S5	(vide agrupamento 1)
1.2 – Projetos Java EE; Portal Conteúdo e e-commerce	R1, S1, S4, S6, S8	(vide agrupamento 1)
2 - Projetos .NET	R3, R5, S3	1) R3, R5 = <i>Financeiro</i> ; S3 = <i>Portal Conteúdo e e-commerce</i> .
3 - Projetos Java EE / C++; Financeiro	R7, S7	1) Percentualmente, a porção Java EE de S7 é um pouco maior que a de R7. 2) Somente S7 usa Maven e Selenium. 3) Somente R7 usa PowerDesigner.

6.1.3. Pessoas

As informações sobre as equipes dos projetos são mostradas na Tabela 30, que detalha as quantidades de desenvolvedores participantes por subprocesso (veja Figura 12) de cada projeto. Para indicar quantos desenvolvedores de cada nível de senioridade participou em cada subprocesso, foi usada uma notação numérica X/Y/Z que indica, na ordem Senior/Pleno/Júnior, a quantidade de colaboradores de cada um destes níveis de senioridade.

Tabela 30 - Participantes dos projetos

Projeto	Tamanho da equipe	SUBPROCESSO			
		Requisito	A & D	Implementação	Testes
R1	4	1/0/0	1/0/0	0/3/0	0/3/0
R2	4	1/0/0	1/0/0	0/3/0	0/3/0
R3	5	1/0/0	1/0/0	1/1/1	1/1/1
R4	5	1/0/0	1/0/0	1/1/1	1/1/1
R5	5	1/0/0	1/0/0	1/2/0	1/2/0
R7	6	1/0/0	1/0/0	2/1/0	2/2/0
S1	7	1/0/0	1/0/0	1/1/1	0/1/1
S2	8	1/0/0	1/0/0	0/3/1	0/1/1
S3	7	1/0/0	1/0/0	1/2/0	0/1/1
S4	8	1/0/0	1/0/0	1/1/3	1/1/0
S5	9	1/0/0	1/0/0	1/1/3	1/1/0
S6	12	1/0/0	1/1/0	1/1/4	1/1/1
S7	14	1/0/0	2/0/0	1/4/4	1/0/2
S8	13	1/0/0	2/0/0	3/0/4	1/1/2

A seguir algumas considerações sobre a análise:

- Pode-se perceber que o tamanho da equipe varia significativamente de um projeto para outro. Segundo o gerente de processos da empresa, isso foi devido à mudança na tática de alocação de recursos para *Implementação* e *Testes*, que se tornaram mais rotativos depois da adoção do processo Scrum-RUP.

- Os padrões das equipes possuem semelhanças perceptíveis especialmente no que diz respeito aos subprocessos de *Requisitos* e *A&D* (as únicas exceções são os projetos S6, S7 e S8 que alocam duas pessoas para *A&D* ao invés de apenas uma).
- Por outro lado, os padrões das equipes variam muito para os subprocessos de *Implementação* e *Testes*.

Devido à notável granularidade das diferenças entre os padrões das equipes nos subprocessos de *Implementação* e *Testes*, tornou-se inviável a subdivisão ou refinamento dos agrupamentos de projetos para cada padrão. Assim, decidiu-se considerar todas as informações das equipes como “Diferenças restantes” entre os projetos, as quais serão utilizadas para acareação na comparação dos mesmos.

As produtividades dos projetos RUP e Scrum-RUP serão então comparadas em cada agrupamento definido na seção anterior e, depois na acareação, tentar-se-á encontrar algum padrão nas equipes que possa ter relação com as diferenças de produtividade encontradas.

6.2. Comparação das produtividades

Para comparar quantitativamente as produtividades de cada um dos agrupamentos, foi feito o seguinte procedimento para cada agrupamento:

- Calcular o ganho médio em produtividade dos projetos Scrum-RUP em relação aos projetos RUP.
- Comparar as distribuições dos requisitos dos projetos Scrum-RUP e RUP para auxiliar a verificar se a diferença de produtividade entre os projetos Scrum-RUP e RUP é significativa. Para isso foi utilizado novamente o teste de Kruskal-Wallis pois a produtividade dos requisitos de alguns projetos não atendeu as condições para o uso da alternativa paramétrica *ANOVA One-way* (veja Anexo E). O software utilizado foi o Minitab 16.

O resultado das comparações dos projetos Scrum-RUP e RUP de cada agrupamento é mostrado na Figura 24.

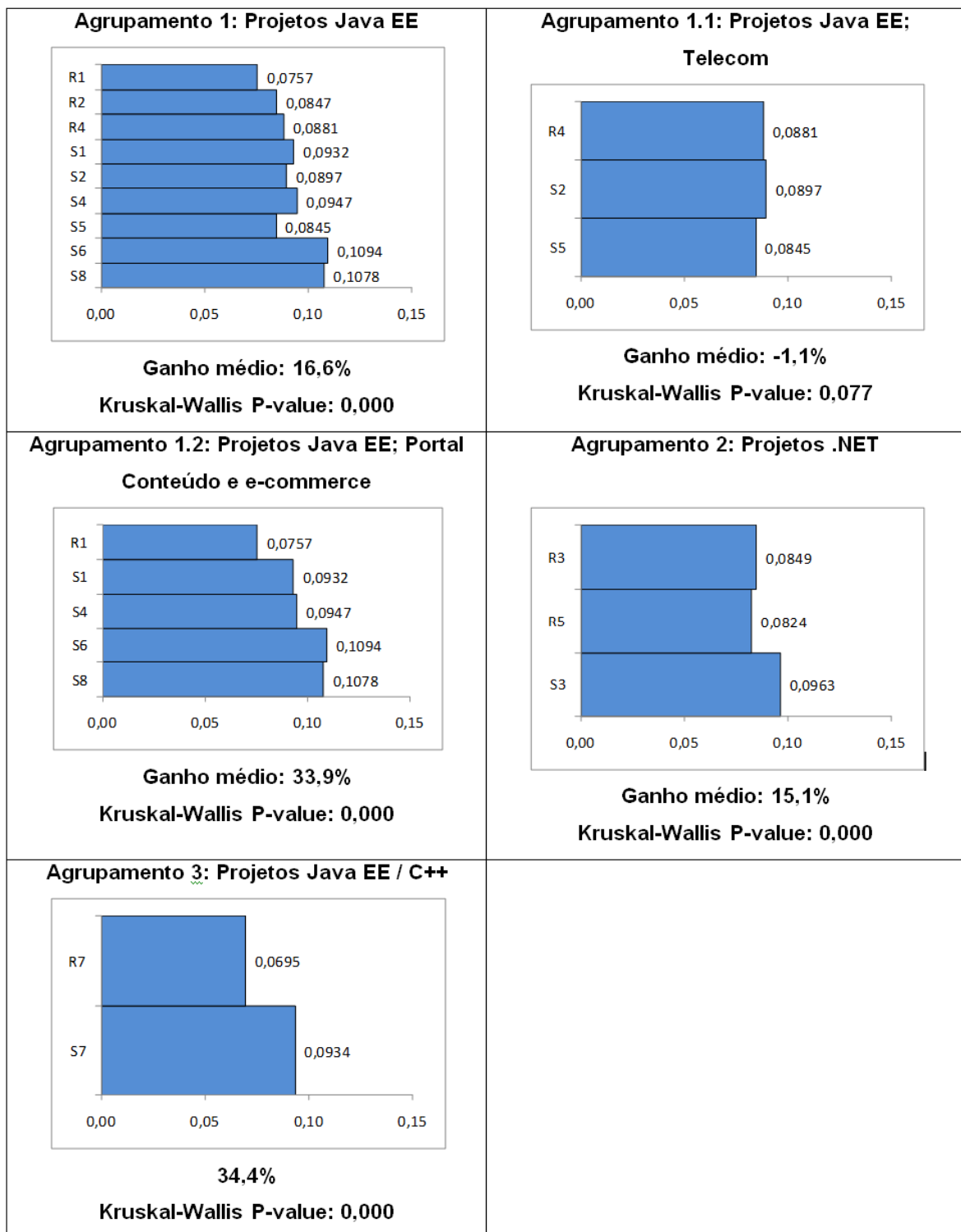


Figura 24 – Resultado da comparação da produtividade dos projetos Scrum-RUP e RUP para cada um dos agrupamentos

6.3. Análise qualitativa

A seguir é apresentada a análise dos dados das entrevistas efetuadas com os 11 desenvolvedores. O leitor pode consultar o Anexo B para detalhes do questionário aplicado.

6.3.1. Recordação dos projetos

Os entrevistados, em média, consideram que lembram dos projetos com um grau de confiança de 77% (veja Figura 25). Este grau de recordação foi considerado satisfatório para contribuir com a validade das entrevistas.

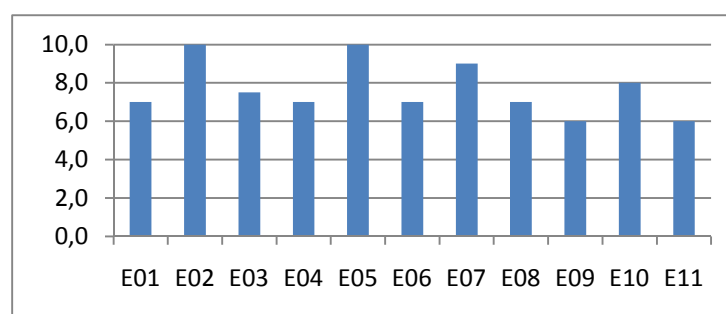


Figura 25 – Grau de recordação dos projetos (escala de 0-10, onde 10 = recordeo plenamente)

6.3.2. Entendimento do processo

Em média, os entrevistados afirmaram ter demorado mais de dois meses para compreender o novo processo Scrum-RUP (veja Tabela 31). Curiosamente, os dois entrevistados que afirmaram ter demorado mais tempo (6 meses) para compreender o processo eram profissionais sênior, dos quais se esperava um rápido entendimento. Os colaboradores participantes dos projetos do estudo receberam orientações e treinamentos informais sobre o novo processo Scrum-RUP, na forma de testes piloto, por dois meses previamente à sua adoção.

Tabela 31 - Quantidade de meses gastos para compreensão do processo Scrum-RUP

E01	E02	E03	E04	E05	E06	E07	E08	E09	E10	E11	Média
6	0	3	5	3	0	1	6	2	0	0	2,36

Foi perguntado aos entrevistados quais as características do novo processo Scrum-RUP (pergunta aberta do Item 3 da entrevista – veja Anexo B). Após o procedimento de

codificação-triangulação (veja Seção 5.4) das respostas, as características mais percebidas pelos entrevistados foram identificadas e são apresentada na Tabela 32. Pode-se perceber que os três itens mais citados na Tabela 32 pertencem ao conjunto das quatro características previstas na proposição **P2** (veja Seção 5.2.1).

Tabela 32 - Mudanças no processo percebidas pelos entrevistados

Código	Frequência
Diminuição da documentação	11
Melhor comunicação	8
Maior colaboração	6
Centralização de informações	4
Indisciplina / desorganização	3
Micro gerenciamento	3
Maior agilidade	2
Gerência participativa	1
Gestão facilitada	1
Melhoria de entendimento	1
Menos contato com o cliente	1

Foi feita uma acareação informal junto ao gerente de qualidade de processos da empresa com o intuito de obter uma compreensão melhor sobre como foi a diminuição de documentação no contexto específico no qual o estudo foi realizado. Por restrições de sigilo, não foi possível incluir nesta tese um exemplo real para comparar a documentação de um requisito do processo RUP e do processo Scrum-RUP, mas as principais diferenças gerais relatadas foi que no processo RUP eram feitos diagramas detalhados juntamente com explicações adicionais para especificar a Realização de um requisito, enquanto que no processo Scrum-RUP os requisitos são traduzidos em Estórias de Usuário e os diagramas eram evitados, sendo especificados somente para requisitos complexos, ou seja, aqueles com maiores tamanhos funcionais, ou que possuam algum risco elevado.

6.3.3. Causas do aumento de produtividade

A seguir são apresentados os resultados da análise das três etapas da entrevista que objetivavam levantar as causas do aumento de produtividade (veja perguntas no item “4” no questionário do Anexo B).

ETAPA 1. Para a questão aberta sobre as causas do aumento de produtividade, foi feito o procedimento de codificação-triangulação apresentado na Seção 5.4. Os resultados da análise são mostrados na Tabela 33.

Houve quatro porções de texto que receberam dois códigos ao mesmo tempo (“Melhor comunicação” e “Micro gerenciamento”), mas as duplicidades de códigos de um mesmo entrevistado foram eliminadas. Os dados brutos e seus códigos podem ser conferidos no Anexo F.

Tabela 33 - Fatores que influenciaram no ganho de produtividade do processo Scrum-RUP em relação ao processo RUP (etapa 1: questão aberta)

Código	Frequência
Melhor comunicação	10
Maior colaboração	6
Diminuição da documentação	5
Micro gerenciamento	4
Ferramentas (teste)	3
Subprocesso de testes	2
Aumento gradual da equipe	1
Equipes homogêneas	1
Ferramentas (frameworks)	1
Gerência participativa	1
Pressão do gerente	1

ETAPA 2. Os resultados das questões estruturadas são mostrados na Tabela 34 e Tabela 35. Cada fator foi avaliado de acordo com a seguinte pontuação:

- Cada ocorrência de “caiu muito” vale -2
- Cada ocorrência de “caiu pouco” vale -1
- Cada ocorrência de “aumentou pouco” vale 1
- Cada ocorrência de “aumentou muito” vale 2

Vale notar que, como o número de entrevistados foi 11, o *score* final de um fator pode assumir a faixa de valores de -22 a 22.

Tabela 34 - Fatores relacionados ao Scrum (questão estruturada)

	caiu muito	caiu pouco	neutro	aumentou pouco	aumentou muito	Score final
Comunicação da equipe	0	0	0	4	7	18
Cooperação da equipe	0	0	0	5	6	17
Distribuição física da equipe	0	1	2	3	5	12
Resposta a requisições de mudança	0	1	1	8	1	9
Artefatos	1	1	1	5	3	8
Moral / Motivação	0	0	5	6	0	6
Adaptabilidade	0	0	6	5	0	5
Entrega evolutiva	0	1	6	4	0	3
Flexibilidade	0	2	5	4	0	2
Foco	1	2	3	5	0	1
Dinâmica / flexibilidade	0	4	5	2	0	-2
Tamanho da iteração	0	3	7	1	0	-2
Papéis	0	2	9	0	0	-2
Respeito	0	4	6	1	0	-3
Confiança	1	3	7	0	0	-5
Gestão de backlog	2	3	4	2	0	-5
Time-boxing	3	6	2	0	0	-12

Tabela 35 - Fatores não relacionados ao Scrum (questão estruturada)

	caiu muito	caiu pouco	neutro	aumentou pouco	aumentou muito	Score final
Mudança na equipe durante o projeto		3	1	3	4	8
Experiência do time		1	4	6		5
Cronograma apertado			8	2	1	4
Capacidade do time			7	4		4
Linguagem		1	7	3		2
Ferramentas de desenvolvimento		1	7	3		2
Política de recompensa			11			0
Complexidade da interface com usuário		4	5	2		-2
Cultura da empresa	1	3	5	2		-3
Complexidade do produto	2	6	3			-10
Capacidade do líder	3	4	4			-10
Riscos dos requisitos	3	7		1		-12
Projetos similares / reaproveitamento	9	2				-20

É importante fazer a ressalva aqui que foi feita uma acareação informal para investigar porque o item “Projetos similares / reaproveitamento” obteve um resultado tão negativo. A resposta encontrada nesta acareação foi que, como os métodos ágeis privilegiam a entrega imediata de valor para o cliente, a criação de componentes fica prejudicada, pois isso

implicaria a inserção de *overhead* nas tarefas para construí-los de forma genérica e reutilizável.

ETAPA 3. Para a última questão sobre as causas do aumento de produtividade (que pede para o entrevistado consolidar os fatores que aumentaram a produtividade por ordem de prioridade), também foi feito o procedimento de codificação-triangulação da Seção 5.4, só que desta vez incluindo também a ordem de prioridade dada pelos entrevistados a cada fator. Os resultados da análise são mostrados na Tabela 36.

Houve seis porções de texto que receberam dois códigos ao mesmo tempo (“Melhor comunicação” e “Micro gerenciamento”), mas as duplicidades de códigos de um mesmo entrevistado foram eliminadas. Os dados brutos e seus códigos podem ser conferidos no Anexo G.

Tabela 36 - Fatores responsáveis pelo ganho de produtividade do processo Scrum-RUP, consolidado pelos entrevistados

Colocação	Código	Ocorrências em 1º Lugar	Ocorrências em 2º Lugar	Ocorrências em 3º Lugar	Ocorrências em 4º Lugar	Total
1	Melhor comunicação	5	3	2*	0	10
2	Maior colaboração	4	4	0	0	8
3	Diminuição da documentação	2	2	2	1	7
4	Micro gerenciamento	0	2	4	1	7
5	Pressão do gerente	0	0	1	0	1
6	Subprocesso de teste	0	0	1	0	1

* as duas ocorrências de “Melhor comunicação” em 3º. lugar vieram de itens que receberam também o código “Micro gerenciamento”.

6.4. Discussão

6.4.1. Proposição P1

A proposição P1 prevê um aumento de 20% na produtividade do processo Scrum-RUP em relação ao processo RUP. De acordo com os resultados quantitativos, com exceção do agrupamento 1.1 (veja Figura 24), todos os demais agrupamentos apresentaram aumento significativo de produtividade dos projetos Scrum-RUP sobre os projetos RUP, porém os ganhos de produtividade foram variáveis entre os agrupamentos.

Resta agora fazer, com base nas “Diferenças restantes” (Tabela 29 e Tabela 30) e nos resultados da análise qualitativa, uma acareação destes resultados para explorar a possibilidade de que o aumento de produtividade não tenha sido devido ao processo, mas sim a outros fatores.

Para auxiliar na identificação de fatores espúrios, os dados dos projetos foram dispostos em uma planilha do Microsoft Excel e, daí, foram utilizadas as ferramentas de filtro (para selecionar os projetos de cada agrupamento) e ordenação (por produtividade), de modo a verificar possíveis correspondências entre a produtividade e o fator investigado.

A discussão dos resultados consistirá de duas etapas: primeiramente será feito um levantamento, para cada um dos fatores, das suas possíveis influências na produtividade. Em seguida, os resultados das entrevistas serão confrontados com as possibilidades levantadas (triangulação).

6.4.1.1. Acareação dos fatores

PRODUTO:

Medidas de produto: em geral os projetos não apresentaram diferenças significativas em relação às medidas de produto, com pequenas exceções nos projetos R2, S2 e S8.

Domínio da aplicação: o domínio da aplicação pode ser uma ameaça à validade dos agrupamentos 1 e 2 que misturam domínios de aplicação diferentes. Nestes dois agrupamentos o domínio da aplicação pareceu influenciar a produtividade, porém o processo Scrum-RUP também acompanhou a correlação com os projetos mais produtivos, conforme mostrado na Figura 26. Para os demais agrupamentos, apenas o agrupamento 1.1 não apresentou ganho significativo de produtividade em projetos de domínio de aplicação iguais. Assim, estes indícios indicam que não se pode descartar a possibilidade de que o processo teve parte nos fatores que aumentam a produtividade. Resta saber até que ponto o domínio da aplicação também exerceu influência.

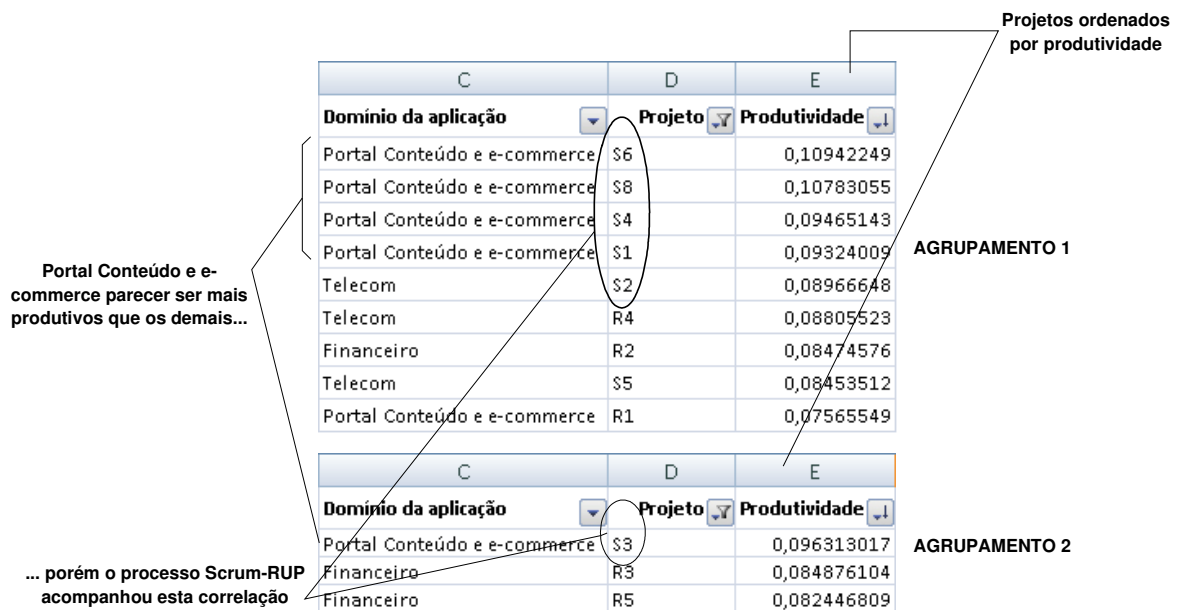


Figura 26 – Acareação da influência do domínio da aplicação na produtividade

TECNOLOGIA

Linguagem de programação: como os projetos foram agrupados por linguagem de programação, este fator não representa maiores ameaças. A única exceção é no agrupamento 3 (projetos Java EE / C++), onde a porção Java EE do projeto S7 é proporcionalmente maior que a do projeto R7; mais precisamente, 16% maior. Isso levanta a possibilidade de que o projeto S7 foi mais produtivo por causa da maior porção Java EE, já que Java EE é (devido ao seu nível de abstração e simplicidade de sintaxe) naturalmente mais produtiva que C++. Entretanto, para que a linguagem fosse uma causa de aumento de produtividade nos projetos (de 34,4%) tão significativa a ponto de se desconsiderar o processo, a diferença de produtividade das linguagens teria que ser muito maior que os 34,4%, o que não corresponde à realidade. Assim, além da linguagem, outros fatores devem ter influenciado no aumento de 34,4% em produtividade dos projetos.

Ferramentas: o uso de quatro ferramentas parece estar correlacionado com a produtividade, sendo JSF positivamente, e Erwin, Spring e Velocity negativamente. Entretanto, a correlação negativa do Erwin pode ser contestada pois o não uso desta ferramenta em projetos Scrum-RUP está em grande parte associado à diminuição da documentação neste processo, o que torna o não uso da ferramenta uma

consequência óbvia. Sendo assim, as ferramentas que aparentemente apresentam algum tipo de correlação com a produtividade são apenas JSF, Spring e Velocity.

PESSOAS

Tamanho da equipe: em todos os agrupamentos, o tamanho da equipe aparentemente teve correlação com a produtividade, o que levanta a possibilidade de que a produtividade em equipes maiores é maior.

Senioridade da equipe: não foi encontrada correlação aparente entre a produtividade e a senioridade dos desenvolvedores que atuaram nos subprocessos de *Implementação* e *Testes* (únicos subprocessos em que as equipes se diferenciaram significativamente). Sendo assim, não se pode dizer que a senioridade destes colaboradores influenciou na produtividade.

Assim, resumindo esta acareação dos possíveis fatores espúrios de aumento de produtividade, os três principais fatores identificados foram:

- **Domínio da aplicação:** projetos de Portal Conteúdo e e-commerce foram mais produtivos que os demais.
- **Ferramentas:** projetos que utilizaram JSF foram mais produtivos que os demais, enquanto que projetos que utilizara, Spring e Velocity foram menos produtivos.
- **Tamanho da equipe:** projetos de equipes maiores foram mais produtivos.

6.4.1.2. Triangulação com os resultados das entrevistas

Os três fatores espúrios identificados na subseção anterior foram confrontados com os resultados das entrevistas no intuito de identificar se aqueles foram confirmados por estas. Para isto, foram consultados os fatores que os respondentes consideraram que aumentaram a produtividade (Tabela 33, Tabela 34, Tabela 35 e Tabela 36). A seguir as discussões:

Domínio da aplicação: nenhum dos respondentes mencionou algo similar ao domínio da aplicação como causa do aumento de produtividade.

Ferramentas: pode-se verificar que algumas respostas apontaram o uso de ferramentas como fator de aumento de produtividade, porém com frequência ou pontuação baixa em relação aos demais fatores:

- **Etapa 1:** Ferramentas (teste): 3 ocorrências; e Ferramentas (frameworks): 1 ocorrência.
- **Etapa 2:** Ferramentas de desenvolvimento: pontuação 2 (em uma faixa de -22 a 22 pontos).

Tamanho da equipe: algumas respostas podem estar relacionadas com o tamanho da equipe, as quais são destacadas a seguir:

- **Etapa 1:** Aumento gradual da equipe: 1 ocorrências; e Subprocesso de testes: 2 ocorrências.
- **Etapa 2:** Mudança da equipe durante o projeto: pontuação 8 (em uma faixa de -22 a 22 pontos).

É importante notar que o resultado da entrevista indica que, na verdade, podem haver outros detalhes (além do tamanho) da equipe que aumentaram a produtividade, como a mudança da equipe durante o projeto (que naturalmente implica em um número maior de participantes) e/ou especificamente à equipe de testes.

Assim, dos três fatores espúrios inicialmente identificados na seção anterior, apenas dois deles (“Ferramentas” e “Tamanho da equipe”) foram parcialmente sustentados pela triangulação, sendo que o “Tamanho da equipe” parece ficar melhor representado como o “Aumento gradual da equipe”. A lista de possíveis fatores espúrios assim atualizada é mostrada a seguir:

- **Ferramentas:** automatização de testes e frameworks (sendo JSF o mais sustentado pelas evidências).
- **Aumento gradual da equipe:** equipes maiores foram mais produtivas. Aparentemente esta correlação pode ter relação com o subprocesso de testes e com a rotatividade da equipe.

6.4.2. Proposição P2

A proposição P2 prevê que o processo Scrum-RUP vai aumentar a produtividade pelos seguintes motivos: melhor comunicação, melhor colaboração, diminuição da documentação e micro gerenciamento. Com base nos resultados das entrevistas sobre os fatores que aumentaram a produtividade (Tabela 33, pontuações positivas da Tabela 34 e Tabela 35, e Tabela 36) e os fatores que prejudicaram a produtividade (pontuações negativas da Tabela 34 e Tabela 35), os seguintes pontos de discussão foram destacados com relação à proposição P2:

CONFIRMAÇÃO DA PROPOSIÇÃO: os quatro motivos listados na proposição P2 foram confirmados pelos resultados e estão entre aqueles com as frequências/pontuações mais altas, sendo “Melhor comunicação” o mais frequente. A seguir considerações mais específicas sobre esta confirmação:

- Quando foi perguntado aos entrevistados de forma aberta (sem sugerir nenhum fator) as causas da produtividade, pode-se perceber, conforme Tabela 33, que os entrevistados já consideraram “Melhor comunicação” como o fator mais importante para o aumento de produtividade (10 ocorrências), seguido dos outros três fatores previstos na proposição **P2**. Mesmo após a etapa 2, na qual diversos outros fatores foram sugeridos para que o entrevistado se lembrasse de mais detalhes, a confirmação da proposição P2 foi reforçada, como mostrado na consolidação dos fatores na etapa 3, conforme Tabela 36.
- É interessante notar que, quando foi perguntado aos entrevistados sobre o que mudou no processo Scrum-RUP em relação ao processo RUP (Tabela 32), todos eles destacaram a diminuição da documentação (11 ocorrências). Porém, quando responderam especificamente sobre as causas da produtividade, eles consideraram a “Melhor comunicação” e “Maior colaboração” mais importantes que a “Diminuição da documentação”.

OUTRAS VANTAGENS DO PROCESSO Scrum-RUP: os resultados das três etapas das perguntas sobre produtividade foram avaliados no intuito de se encontrar outros fatores relacionados ao processo Scrum-RUP que possam ter aumentado a produtividade. A partir

desta avaliação, percebeu-se que os resultados sugerem que o subprocesso de *Testes* pode ter influenciado na produtividade. Além disso, outros cinco fatores de menor frequência/pontuação também apareceram nos resultados, que podem ser considerados como pontos isolados (*outliers*). A Tabela 37 a seguir mostra todos os seis fatores indicados juntamente com os resultados relacionados de cada etapa.

Tabela 37 - Vantagens produtivas do processo Scrum-RUP não previstas

Fator	Etapa 1	Etapa 2	Etapa 3
Subprocesso de testes	Ferramentas (testes) (3) Subprocesso de testes (2)	Resposta a requisições de mudança (9)	Subprocesso de testes (0/0/1/0)
Adaptabilidade	-	Adaptabilidade (5) Flexibilidade (2)	-
Moral da equipe	-	Moral / motivação (6)	-
Gerência participativa	Gerência participativa (1)	-	-
Entrega evolutiva	-	Entrega evolutiva (3)	-
Foco	-	Foco (1)	-

FATORES PRODUTIVOS NÃO RELACIONADOS AO PROCESSO: além dos fatores de produtividade relacionados ao processo, os respondentes entendem que (1) a pressão do gerente sobre a equipe, (2) o uso de ferramentas e (3) o aumento gradual da equipe contribuíram para o aumento de produtividade nos projetos Scrum-RUP. Além disso, outros fatores de menor frequência ou pontuação também apareceram nos resultados. Um resumo é mostrado na Tabela 38.

Tabela 38 - Fatores produtivos não relacionados ao processo

Fator	Etapa 1	Etapa 2	Etapa 3
Pressão do gerente	Pressão do gerente (1)	Cronograma apertado (4)	Pressão do gerente (0/0/1/0)
Aumento gradual da equipe	Aumento gradual da equipe (1)	Mudança da equipe durante o projeto (8)	-
Ferramentas	Ferramentas (testes) (3) Ferramentas (frameworks) (1)	Linguagem (2) Ferramentas de testes (2)	-
Características da equipe	Equipes homogêneas (1)	Experiência da equipe (4) Capacidade da equipe (4)	-

Pode-se notar que, além dos dois fatores espúrios já identificados na subseção anterior (“Ferramentas” e “Aumento gradual da equipe”), os resultados das entrevistas apontaram para mais dois:

- **Pressão do gerente:** os resultados indicam que, paralelamente à adoção do processo Scrum-RUP, houve maior cobrança da equipe, em especial por parte do gerente de qualidade.
- **Características da equipe:** os resultados também indicam alguns pontos isolados (*outliers*) relacionados a algumas características das equipes (homogeneidade, experiência e capacidade) como fatores que exerceram alguma influência no aumento de produtividade.

FATORES QUE PREJUDICARAM A PRODUTIVIDADE: os fatores que prejudicaram a produtividade, de acordo com as respostas dos entrevistados (veja Tabela 34 e Tabela 35) são mostrados a seguir na Tabela 39 em ordem de importância conforme sua pontuação. Fatores relacionados ao processo estão destacados.

Tabela 39 - Fatores que prejudicaram a produtividade

Fator	Itens
Características do produto	Projetos similares / reaproveitamento (-20) Riscos dos requisitos (-12) Complexidade do produto (-10) Complexidade da interface com o usuário (-2)
Time-boxing (PROCESSO)	Time-boxing (-12) Tamanho da iteração (-2)
Capacidade do líder	Capacidade do líder (-10)
Relacionamento	Confiança (-5) Respeito (-3)
Gestão de backlog (PROCESSO)	Gestão de backlog (-5)
Cultura da empresa	Cultura da empresa (-3)
Dinâmica / flexibilidade (PROCESSO)	Dinâmica / flexibilidade (-2)
Papéis (PROCESSO)	Papéis (-2)

6.5. Sumário e conclusões

Este capítulo apresentou os resultados quantitativos e qualitativos encontrados no estudo de caso industrial multi-projeto realizado nesta pesquisa. Os resultados consolidados da análise de dados são mostrados a seguir na Tabela 40 e Tabela 41.

Tabela 40 - Resultados consolidados - quantitativos

Agrupamento #	Descrição	Ganho médio em produtividade do processo Scrum-RUP
1	Projetos Java EE	16,6%
2	Projetos Java EE / Telecom	-1,1%
3	Projetos Java EE / Portal conteúdo e e-commerce	33,9%
4	Projetos .NET	15,1%
5	Projetos Java EE e C++ / Financeiro	34,4%

Tabela 41 - Resultados consolidados - causas do aumento de produtividade

Fator	Relacionado ao processo Scrum-RUP?	Suporte evidencial dos dados contextuais	Suporte evidencial das entrevistas
Comunicação	sim	n/a	muito alto
Colaboração	sim	n/a	Alto
Diminuição da documentação	sim	n/a	Alto
Micro gerenciamento	sim	n/a	Médio
Subprocesso de testes	sim	n/a	Médio
Pressão do gerente	não	n/a	Médio
Aumento gradual da equipe	não	sim	Médio
Ferramentas	não	JSF	Baixo
Características da equipe	não	não	Baixo
Adaptabilidade	sim	n/a	muito baixo
Moral da equipe	sim	n/a	muito baixo
Entrega evolutiva	sim	n/a	muito baixo
Gerência participativa	sim	n/a	muito baixo
Foco	sim	n/a	muito baixo

Além destes resultados da Tabela 41 sobre os fatores que contribuíram para o aumento da produtividade, foram apresentados na Tabela 39 os fatores que, segundo as entrevistas, prejudicaram a produtividade.

É importante ressaltar que na análise qualitativa, os pontos isolados (*outliers*) são tratados de forma diferente da análise quantitativa. Enquanto os pontos isolados podem ser ignorados (até com o uso de ferramentas estatísticas) na análise quantitativa, eles desempenham um papel importante na análise qualitativa, podendo explicar, calibrar e até apoiar uma proposição [95], além de poderem ser utilizados como base para investigações futuras.

Os resultados sustentaram em boa parte os quatro motivos previstos na proposição **P2**, os quais são discutidos a seguir:

- **Melhor comunicação:** a melhoria da comunicação entre a equipe foi o fator mais frequentemente identificado dentre os entrevistados, corroborando os resultados de [69], [64], [70], [68], e [55].
- **Maior colaboração:** o segundo fator mais citado pelos entrevistados foram os relacionados com a colaboração entre os membros da equipe, o que corrobora os resultados de [70] e [55]. As respostas dos entrevistados relacionadas a este fator se referiam à colaboração da equipe, integração da equipe, e o uso de reuniões frequentes, onde pequenos problemas eram resolvidos mais facilmente e a equipe era alinhada com relação ao projeto.
- **Diminuição de documentação:** os resultados também mostraram que a diminuição da documentação implicou no aumento de produtividade, assim como apontado no estudo de [55]. A diminuição da documentação, entretanto, foi criticada informalmente pelos entrevistados deste estudo e há algum tempo já é identificada como um problema potencial em projetos de software [5].
- **Micro gerenciamento:** não com tanto suporte evidencial quando os três primeiros, mas o micro gerenciamento foi o quarto fator mais destacado entre os entrevistados, o que corroborou razoavelmente os resultados de [64], [55].

7. CONCLUSÕES

Esta tese apresentou uma proposta de um processo híbrido Scrum-RUP, integrando algumas práticas de Scrum em um processo de desenvolvimento baseado em RUP. As decisões tomadas na concepção e elaboração do processo Scrum-RUP foram consideravelmente tomadas com base nas discussões encontradas na literatura, bem como nos principais resultados da pesquisa na área de métodos ágeis.

Um estudo de caso industrial multi-projeto foi conduzido em uma empresa localizada em Uberlândia-MG com o intuito validar a proposta no sentido de avaliar, conforme estipulado na questão de pesquisa, como o processo Scrum-RUP impacta a produtividade de desenvolvimento, em comparação ao processo RUP que a empresa estava acostumada a utilizar. Esta questão de pesquisa foi desmembrada em duas outras questões **Q1** e **Q2**, para as quais as conclusões são apresentadas a seguir.

Q1: Há diferença de produtividade entre equipes RUP e equipes Scrum-RUP? Se sim, quanto é esta diferença?

Os resultados quantitativos mostraram ganho significativo de produtividade em quatro dos cinco agrupamentos de projetos semelhantes (veja Tabela 40). Os ganhos variaram da ordem de 15% a 34%, enquanto que no agrupamento que não apresentou diferença significativa, o ganho foi de -1,1%. Estes resultados apontam que houve ganho de produtividade nos projetos Scrum-RUP, porém não explicam até que ponto o processo foi o responsável por tal ganho. Entretanto, pode-se concluir com relativa segurança que o processo foi em parte responsável pelo ganho de produtividade, devido aos resultados consolidados após as entrevistas, nos quais as causas relacionadas ao processo Scrum-RUP tiveram significativa predominância (veja Tabela 41).

Não se pode inferir conclusões sobre o ganho teórico do processo Scrum-RUP em comparação ao processo RUP principalmente devido ao fato de que a quantidade de projetos não foi grande o suficiente para representar uma amostra significativa para generalização estatística.

Q2: Qual a explicação dos resultados das produtividades de equipes RUP e Scrum-RUP?

Vale recordar que, conforme mencionado no Capítulo 5, esta pesquisa não teve o intuito de fornecer explicações causais com poder de generalização estatística, mas sim investigar as possíveis causas dos efeitos na produtividade, para fornecer uma base para reflexão, bem como auxiliar no levantamento de hipóteses em outros estudos.

Os resultados do estudo de caso mostram que, dentre os fatores que aumentaram a produtividade, aqueles relacionados ao processo Scrum-RUP possuem suporte evidencial claramente mais expressivo do que os fatores não relacionados ao processo Scrum-RUP, o que fortalece a tese de que o processo teve parte significativa nas causas do ganho de produtividade.

É importante observar que [67] destaca que, quando uma proposição previamente estabelecida é confirmada pelos resultados, pode-se inferir uma sólida conclusão sobre os efeitos esperados. Ademais, conforme apresentado na Seção 6.5, dos quatro fatores previstos na proposição **P2**, a (1) comunicação obteve suporte evidencial muito alto, enquanto que (2) colaboração e (3) diminuição da documentação obtiveram suporte alto, e (4) micro gerenciamento obteve suporte médio (além desses quatro fatores, ainda surgiu um quinto não previsto: o subprocesso de *Testes*, com suporte evidencial também médio).

Assim, pode-se concluir que a “Melhor comunicação” é uma característica ágil consideravelmente benéfica, que já foi corroborada por diversos estudos empíricos anteriores e agora por este. Pode-se afirmar com propriedade que já existe considerável embasamento científico para esta conclusão. Não com tanta solidez, porém ainda com relativo grau de confiança, pode-se concluir que os fatores “Maior colaboração”, “Diminuição da documentação” e “Micro gerenciamento” também exerceram papel significativo no ganho de produtividade do processo Scrum-RUP, pois estes tiveram suporte evidencial alto e também foram replicações de outros estudos anteriores. Para os demais fatores, não se pode tirar conclusões sólidas sobre sua influência na produtividade, mas estes abrem novas possibilidades de pesquisa em trabalhos futuros.

Vale ressaltar também que os resultados indicaram ameaças específicas à validade interna: os quatro fatores de aumento de produtividade não relacionados ao processo (pressão do gerente, aumento gradual da equipe, uso de ferramentas e características da equipe – veja Tabela 41), bem como os fatores de diminuição de produtividade relacionados ao processo (time-boxing, gestão de backlog, dinâmica/flexibilidade e papéis – veja Tabela 39) abrem a possibilidade de questionamento com relação à influência do

processo no ganho de produtividade encontrado. O que mitiga estas ameaças, como já mencionado na Seção 5.5.2, são os resultados das entrevistas (que favoreceram o processo), bem como à confirmação do padrão prognosticado na proposição **P2**.

Esta proposta de processo Scrum-RUP se mostrou útil no contexto onde foi aplicada. O leitor pode analisar as informações contextuais e tentar replicar os resultados em outros projetos. O processo Scrum-RUP apresentado aqui pode apresentar ganhos de produtividade ao mesmo tempo que preserva parcialmente as especificações de requisitos, arquitetura e casos de teste, de modo a permitir modelos contratuais mais rigorosos, com fechamento de escopo antecipado. Este processo pode ser útil, inclusive, em contextos empresariais onde se deseja utilizar agilidade e não se tem à disposição um representante *on site* da área de negócio do lado do cliente, como preconizado pela filosofia ágil. A gestão e especificação formal de requisitos supre a dificuldade que o cliente teria se a documentação fosse esparsa e se ele não pudesse manter um contato próximo com a equipe de desenvolvimento.

O estudo mostra que é possível inserir práticas Scrum no processo de desenvolvimento de software sem eliminar o rigor nos subprocessos necessários e, mesmo assim, obter ganhos reais de produtividade no desenvolvimento.

7.1. Trabalhos futuros

Com base nos resultados da pesquisa, novos estudos podem ser realizados, de modo a investigar questões sobre a influência dos fatores identificados de aumento e diminuição de produtividade. Por exemplo, a identificação do “Aumento gradual da equipe” como fator de aumento de produtividade é um ponto interessante, pois em outros estudos com processos ágeis já foi percebido justamente o oposto: que é mais difícil realocar recursos em equipes ágeis devido ao fato de que no desenvolvimento ágil o conhecimento é tácito, ao invés de ser registrado em documentos [84]. Assim, este estudo sugere que a abordagem híbrida proposta aqui proporcionou a utilização de agilidade não dificultando a realocação de recursos, mas sim a facilitando; o que pode ser um interessante tema de pesquisas futuras.

Entretanto, além deste fator, todos os outros fatores de produtividade sumarizados na Tabela 41 (inclusive os pontos isolados) também podem ser investigados ainda em

processos híbridos de desenvolvimento de software, uma vez que tais fatores foram identificados por este estudo e, portanto, já possuem uma base inicial para a justificativa de estudos mais específicos sobre eles.

A proposta do processo Scrum-RUP também pode ser revista no sentido de investigar os fatores de diminuição de produtividade apontados neste estudo (veja Tabela 39), tais como time-boxing, gestão de backlog, dinâmica/flexibilidade e papéis.

Pesquisas futuras também podem ser realizadas no sentido de aprimorar e validar os modelos de medições utilizados nesta pesquisa, pois a área de Engenharia de Software ainda não consolidou padrões ou *frameworks* de medição de estudos, bem como de comparação entre estudos no mesmo tema [55].

Um fenômeno interessante percebido durante a fase de codificação das respostas, foi que alguns entrevistados “aproveitaram” o espaço da resposta para acrescentar críticas sobre algumas características, dentre as mais criticadas estão a redução de documentação e o acúmulo de informações por parte do líder. Em outras, palavras, os desenvolvedores se sentiram desconfortáveis com a troca de documentação por conhecimento tácito. Diante dessas críticas, estudos podem ser realizados no sentido de avaliar os reais prejuízos destas práticas. Um exemplo de uma área a que possa interessar profundamente uma pesquisa desse tipo é na área de manutenção de projetos de software, pois uma equipe que for prestar manutenção em um projeto com pouca documentação pode encontrar problemas sérios ao realizar seu trabalho.

Também podem ser realizados estudos para avaliar a robustez e os efeitos do processo Scrum-RUP em contextos diferentes do estudo realizado nesta tese, como por exemplo em projetos de larga escala, equipes geograficamente distribuídas, dentre outros.

REFERÊNCIAS

1. ALVES, N.; CARVALHO, W.; LAMOUNIER JÚNIOR, E. **Scrum and Plan-driven Process Integration and its Impact on Effort Estimation**. International Conference on Software Engineering and Knowledge Engineering. Redwood, USA: KSI. 2010.
2. ALVES, N. et al. Um estudo de caso industrial sobre integração de práticas ágeis no RUP. **Revista Ciência e Tecnologia**, n. 24, 2011.
3. PÁDUA FILHO, W. **Engenharia de Software**. 3. ed. Rio de Janeiro: LTC, 2009.
4. RAMSIN, R.; PAIGE, R. F. Process-Centered Review of Object Oriented Software Development Methodologies. **ACM Computing Surveys**, v. 40, n. 1, p. 1-89, 2008.
5. BOEHM, B.; TURNER, R. **Balancing Agility and Discipline: A Guide for the Perplexed**. Reading: Addison-Wesley, 2004.
6. COCKBURN, A. **Agile Software Development**. Reading: Addison-Wesley, 2002.
7. ILIEVA, S.; IVANOV, P.; STEFANOVA, E. **Analyses of an Agile Methodology Implementation**. Euromicro Conference. Rennes, France: IEEE Computer Society Press. 2004.
8. LAYMAN, L.; WILLIAMS, L.; CUNNINGHAM, L. **Exploring Extreme Programming in Context: an Industrial Case Study**. Agile Development Conference. Salt Lake City, USA: IEEE Computer Society. 2004.
9. WILLIAMS, L.; COCKBURN, A. Agile software development: it's about feedback and change. **IEEE Computer**, v. 36, n. 6, p. 39-43, 2003.
10. AMBLER, S. Dr. Dobb's Journal Agile Adoption Survey. **Ambysoft Home Page**, 2008. Disponível em: <<http://www.ambysoft.com/surveys/agileFebruary2008.html>>. Acesso em: 1 Junho 2010.

11. LEFFINGWELL, D. **Scaling Software Agility**. Reading: Addison Wesley, 2006.
12. COHEN, D.; LINDVALL, M.; COSTA, P. **An Introduction to Agile Methods**. Amsterdam: Elsevier, v. 62, 2004.
13. NORD, R.; TOMAYKO, J. Software Architecture-Centric Methods and Agile Development. **IEEE Software**, v. 23, n. 2, p. 47-53, 2006.
14. KRUCHTEN, P. **Software Architecture and Agile Software Development - An Oxymoron?** Workshop Software Architecture Challenges in the 21st Century. Los Angeles: University of Southern California. 2009.
15. AMBLER, S. **Agile Software Development at Scale**. IFIP European Conference on Software Engineering Techniques. Poznan: Springer. 2007.
16. DYBÅ, T.; DINGSØYR, T. Empirical studies of agile Software development: A systematic review. **Information and Software Technology**, v. 50, n. 9, p. 833-859, 2008.
17. DYBÅ, T.; DINGSØYR, T. What Do We Know about Agile Software Development? **IEEE Software**, v. 25, n. 5, p. 6-9, 2009.
18. QUMER, A.; HENDERSON-SELLERS, B. A Framework to Support the Evaluation, Adoption and Improvement of Agile Methods in Practice. **Journal of Systems and Software**, v. 81, n. 11, p. 1899-1919, 2008.
19. MARTIN, A.; BIDDLE, R.; NOBLE, J. **When XP Met Outsourcing**. International Conference on Extreme Programming and Agile Processes in Software Engineering. Garmisch-Partenkirchen: Springer. 2004.
20. ALVES, N. et al. **From Strategy to Solution: A Lightweight Semi-Prescriptive Approach for Software Development Lifecycle with Outsourcing Support**. International Conference on Software Engineering and Knowledge Engineering. Boston: KSI. 2009.

21. BABAR, M.; ABRAHAMSSON, P. **Architecture-Centric Methods and Agile Approaches**. International Conference on Agile Processes and Extreme Programming in Software Engineering. Limerick: Springer. 2008.
22. PARSONS, R. **Architecture and Agile Methodologies - How to Get Along**. Working IEEE/IFIP Conference on Software Architecture. Vancouver: IEEE Computer Society. 2008.
23. ERDOGMUS, H. Architecture Meets Agility. **IEEE Software**, v. 26, n. 5, p. 2-4, 2009.
24. ABRAHAMSSON, P.; BABAR, M.; KRUCHTEN, P. Agility and Architecture: Can They Coexist? **IEEE Software**, v. 27, n. 2, p. 16-22, 2010.
25. ROMBACH, D.; SEELISCH, F. **Formalisms in Software Engineering: Myths Versus Empirical Facts**. Central and East European Conference on Software Engineering Techniques. Poznan: Springer. 2007. p. 13-25.
26. SCHWABER, K.; BEEDLE, M. **Agile Software Development with Scrum**. Upper Saddle River: Prentice Hall, 2001.
27. KRUCHTEN, P. **Rational Unified Process: An Introduction**. 3. ed. Reading: Addison-Wesley, 2003.
28. SEI. **Capability Maturity Model Integration**, 2002. Disponível em: <<http://www.sei.cmu.edu/cmmi>>. Acesso em: 1 Julho 2010.
29. JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. **Unified Modeling Language User Guide**. 2. ed. Reading: Addison-Wesley, 2005.
30. BECKER, S.; WHITTAKER, J. **Cleanroom Software Engineering Practices**. 4. ed. Hershey: Idea Group Publishing, 1997.
31. PROWELL, S. et al. **Cleanroom Software Engineering: Technology and Process**. Reading: Addison-Wesley, 1999.

32. CMMI PRODUCT TEAM, T. **CMMI for Development, Version 1.2. CMU/SEI-2006-TR-008**. Software Engineering Institute. Pittsburgh. 2006.
33. HUMPHREY, W. **Introduction to the Personal Software Process**. Reading: Addison-Wesley, 1997.
34. HUMPHREY, W. **Introduction to Team Software Process**. Reading: Addison-Wesley, 2000.
35. NERUR, S.; MAHAPATRA, R.; MANGALARAJ, G. Challenges of Migrating to Agile Methodologies. **Communications of the ACM**, v. 48, n. 5, p. 72-78, 2005.
36. DYBÅ, T. Improvisation in Small Software Organizations. **IEEE Software**, v. 17, n. 5, p. 82-87, 2000.
37. BECK, K. et al. **Manifesto for Agile Software Development**, 2001. Disponível em: <<http://www.agilemanifesto.org>>. Acesso em: 1 Julho 2010.
38. COHN, M. **User Stories Applied: For Agile Software Development**. Reading: Addison-Wesley, 2004.
39. STAPLETON, J. **DSDM: Business Focused Development**. 2. ed. Upper Saddle River: Pearson Education, 2003.
40. SCHWABER, K. **SCRUM development process**. Conference on Object-Oriented Programming Systems, Languages and Applications. Austin: Springer. 1995.
41. BECK, K. **Extreme Programming Explained: Embrace Change**. Reading: Addison-Wesley, 2000.
42. BECK, K. **Extreme Programming Explained: Embrace Change**. 2. ed. Reading: Addison-Wesley, 2004.
43. COCKBURN, A. **Crystal Clear: A Human-Powered Methodology for Small Teams**. Reading: Addison-Wesley, 2004.

44. PALMER, S.; FELSING, J. **A Practical Guide to Feature-driven Development**. Upper Saddle River: Prentice Hall, 2002.
45. POPPENDIECK, M.; POPPENDIECK, T. **Lean Software Development – An Agile Toolkit for Software Development Managers**. Reading: Addison-Wesley, 2003.
46. SCHWABER, K.; SUTHERLAND, J. **Scrum Guide**, 2010. Disponível em: <<http://www.scrum.org/scrumguides/>>. Acesso em: 1 Julho 2010.
47. BOEHM, B.; TURNER, R. Management Challenges to Implementing Agile Processes in Traditional Development Organizations. **IEEE Software**, v. 22, n. 5, p. 30-39, 2005.
48. BOEHM, B. Anchoring the Software Process. **IEEE Software**, v. 13, n. 4, p. 73-82, 1996.
49. BOEHM, B. et al. Using the WinWin Spiral Model: A Case Study. **IEEE Computer**, v. 31, n. 7, p. 33-44, 1998.
50. BOEHM, B.; BROWN, W.; PORT, D. **Model-Based Architecting & Software Engineering**, 1995. Disponível em: <<http://sunset.usc.edu/research/MBASE>>. Acesso em: 1 Julho 2010.
51. JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. **Unified Software Development Process**. Reading: Addison-Wesley, 1999.
52. COCKBURN, A. **Writing Effective Use Cases**. Reading: Addison-Wesley, 2000.
53. ABRAHAMSSON, P. et al. **New Directions on Agile Methods: A Comparative Analysis**. International Conference on Software Engineering. Oregon: IEEE Computer Society. 2003.
54. GALAL-EDEEN, G.; RIAD, A.; SEYAM, M. **Agility versus Discipline: Is Reconciliation Possible?** International Conference on Computational & Experimental Engineering and Sciences. Cairo: IEEE Computer Society. 2007.

55. PETERSEN, K.; WOHLIN, C. A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. **Journal of Systems and Software**, v. 82, n. 9, p. 1479-1490, 2009.
56. NORD, R.; TOMAYKO, J.; WOJCIK, R. **Integrating Software-Architecture-Centric Methods into Extreme Programming (XP)**, tech. report CMU/SEI-2004-TN-036. Software Engineering Institute. Pittsburgh. 2004.
57. BARBACCI, M. R. et al. **Quality Attribute Workshops (QAWs)**, 3rd ed., tech. report CMU/SEI-2003-TR-016. Software Engineering Institute, Carnegie Mellon University. Pittsburgh. 2003.
58. BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software Architecture in Practice**. 2. ed. Reading: Addison-Wesley, 2003.
59. CLEMENTS, P.; KAZMAN, R.; KLEIN, M. **Evaluating Software Architectures: Methods and Case Studies**. Reading: Addison-Wesley, 2002.
60. BOEHM, B.; TURNER, R. **Balancing Agility and Discipline: Evaluating and Integrating Agile and Plan-Driven Methods**. International Conference on Software Engineering. Washington, DC, USA: IEEE Computer Society. 2004a.
61. BOEHM, B.; TURNER, R. **Observations on Balancing Discipline and Agility**. Agile Development Conference. Salt Lake City, Utah, USA: IEEE Computer Society. 2003.
62. BOEHM, B.; TURNER, R. **Rebalancing Your Organization's Agility and Discipline**. Agile Universe Conference. New Orleans, LA, USA: Springer. 2003a.
63. BOEHM, B.; TURNER, R. Using Risk to Balance Agile and Plan-driven Methods. **Computer**, v. 36, n. 6, p. 57-66, 2003b.
64. KARLSTRÖM, D.; RUNESON, P. Integrating Agile Software Development into Stage-gate Managed Product Development. **Empirical Software Engineering**, v. 11, n. 2, p. 203-225, 2006.

65. KARLSTRÖM, D.; RUNESON, P. Combining Agile Methods with Stage-Gate Project Management. **IEEE Software**, v. 22, n. 3, p. 43-49, 2005.
66. COOPER, R. **Winning at New Products**. 3. ed. Cambridge: Perseus Publishing, 2001.
67. YIN, R. **Estudo de Caso - Planejamento e Métodos**. 3. ed. Porto Alegre: Bookman, 2005.
68. PIKKARAINEN, M. et al. The Impact of Agile Practices on Communication in Software Development. **Empirical Software Engineering**, v. 13, n. 3, p. 303-337, 2008.
69. BAHLI, B.; ABOU-ZEID, E. **The Role of Knowledge Creation in Adopting XP Programming Model: an Empirical Study**. International Conference on Information and Communications Technology. Cairo, Egypt: IEEE Computer Society. 2005.
70. SVENSSON, H.; HÖST, M. **Introducing an Agile Process in a Software Maintenance and Evolution Organization**. European Conference on Software Maintenance and Reengineering. Manchester, UK: IEEE Computer Society. 2005.
71. MANN, C.; MAURER, F. **A Case Study on the Impact of Scrum on Overtime and Customer Satisfaction**. Agile Development Conference. Denver, CO, USA: IEEE Computer Society. 2005.
72. TESSEM, B. **Experiences in Learning XP Practices: A Qualitative Study**. Extreme Programming and Agile Processes in Software Engineering. Genova, Italy: Springer. 2003.
73. ROBINSON, H.; SHARP, H. **The Characteristics of XP Teams**. Extreme Programming and Agile Processes in Software Engineering. Garmisch-Partenkirchen, Germany: Springer. 2004.
74. MANNARO, K.; MELIS, M.; MARCHESI, M. **Empirical Analysis on the Satisfaction of IT Employees Comparing XP Practices with Other Software**

- Development Methodologies.** Extreme Programming and Agile Processes in Software Engineering. Garmisch-Partenkirchen, Germany: Springer. 2004.
75. LI, J.; MOE, N.; DYBÅ, T. **Transition from a plan-driven process to Scrum: a longitudinal case study on software quality.** Empirical Software Engineering and Measurement. Bolzano/Bozen, Italy: ACM. 2010.
76. MCBREEN, P. **Questioning Extreme Programming.** Boston, MA, USA: Pearson Education, 2003.
77. STEPHENS, M.; ROSENBERG, D. **Extreme Programming Refactored: The Case Against XP.** Berkeley, CA: Apress, 2003.
78. MACKENZIE, A.; MONK, S. From Cards to Code: How Extreme Programming Re-Embodies Programming as a Collective Practice. **Computer Supported Cooperative Work**, v. 13, n. 1, p. 91-117, 2004.
79. MERISALO-RANTANEN, H.; TUURE, T.; MATTI, R. Is Extreme Programming Just Old Wine in New Bottles: a Comparison of Two Cases. **Journal of Database Management**, v. 16, n. 4, p. 41-61, 2005.
80. MELNIK, G.; MAURER, F. **Perceptions of Agile Practices: A Student Survey.** Extreme Programming and Agile Methods - XP/Agile Universe. Chicago, IL, USA: Springer. 2002.
81. WELLINGTON, C.; BRIGGS, R.; GIRARD, C. **Comparison of Student Experiences with Plan-driven and Agile Methodologies.** ASEE/IEEE Frontiers in Education Conference. Indianapolis, IN : IEEE Computer Society. 2005.
82. BENEDIKTSSON, O.; DALCHER, D.; THORBERGSSON, H. Comparison of software development life cycles: a multiproject experiment. **IEE Proceedings Software**, v. 53, n. 3, p. 87, 2006.
83. PETERSEN, K.; WOHLIN, C. **Context in Industrial Software Engineering.** Empirical Software Engineering and Measurement. Lake Buena Vista, Florida, USA:

IEEE Computer Society. 2009a.

84. LARMAN, C. **Agile and Iterative Development: A Manager's Guide**. Upper Saddle River: Pearson Education, 2003.
85. SHADISH, W.; COOK, T.; CAMPBELL, D. **Experimental and Quasi-Experimental Designs for Generalized Causal Inference**. 2. ed. Florence, KY, USA: Cengage Learning, 2001.
86. RUNESON, P.; HÖST, M. Guidelines for Conducting and Reporting Case Study Research in Software Engineering. **Empirical Software Engineering**, v. 14, n. 2, p. 131-164, 2009.
87. KITCHENHAM, B.; PICKARD, L.; PFLEEGER, S. Case Studies for Method and Tool Evaluation. **IEEE Software**, v. 12, n. 4, p. 52-62, 1995.
88. WAGNER, S.; RUHE, M. **A Systematic Review of Productivity Factors in Software Development**. International Workshop on Software Productivity Analysis and Cost Estimation. Beijing: IEEE Computer Society. 2008.
89. CUNHA, J.; THOMAZ, J.; MOURA, H. **Um Modelo de Avaliação de Produtividade de Projetos de Software Baseado em uma Abordagem Multicritério**. Simpósio Brasileiro de Qualidade de Software. Florianópolis: SBC. 2008.
90. SCACCHI, W. Understanding Software Productivity. **Advances in Software Engineering and Knowledge Engineering**, v. 4, p. 37-70, 1995.
91. WHITE, K. **Software Engineering Management for Productivity and Quality**. International Conference on Accelerator and Large Experimental Physics Control Systems. Trieste, Italy: [s.n.]. 1999.
92. BOEHM, B. et al. **Software Cost Estimation with COCOMO II**. Upper Saddle River: Prentice Hall, 2000.

93. CHIANG, R.; MOOKERJEE, S. Improving Software Team Productivity. **Communications of the ACM**, v. 47, n. 5, p. 89-93, 2004.
94. CLINCY, V. Software Development Productivity and Cycle Time Reduction. **Journal of Computing Sciences in Colleges**, v. 19, n. 2, p. 278-287, 2003.
95. SEAMAN, C. Qualitative Methods in Empirical Studies on Software Engineering. **IEEE Transactions on Software Engineering**, v. 25, n. 4, p. 557-572, 1999.
96. ISO/IEC20926. **Software and systems engineering - Software measurement - IFPUG functional size measurement method**. International Organization for Standardization (ISO). Geneva 20, Switzerland. 2009.
97. MATTAR, F. **Pesquisa de marketing: metodologia, planejamento, execução e análise**. 2. ed. São Paulo: Atlas, 1994.
98. MIGUEL, P. Estudo de caso na engenharia de produção: estruturação e recomendações para sua condução. **Produção**, 2007.
99. ARISHOLM, E.; SJØBERG, D. Evaluating the Effect of a Delegated versus Centralized Control Style on the Maintainability of Object-Oriented Software. **IEEE Transactions on Software Engineering**, v. 30, n. 8, p. 521-534, 2004.
100. LOKAN, C. An Empirical Analysis of Function Point Adjustment Factors. **Information and Software Technology**, v. 42, n. 4, p. 649-659, 2000.
101. CALAZANS, A.; LISBOA, I.; OLIVEIRA, M. **Avaliação das Características Gerais de Sistemas na Análise de Pontos de Função - APF por Meio da Aplicação do GQM - Goal, Questions, Metrics**. Simpósio Internacional de Melhoria de Processos de Software. São Paulo, Brazil: [s.n.]. 2005.
102. KITCHENHAM, B.; MENDES, E. Software Productivity Measurement Using Multiple Size Measures. **IEEE Transactions on Software Engineering**, v. 30, n. 12, p. 1023-1035, 2004.

103. BASILI, V.; SHULL, F.; LANUBILE, F. Building Knowledge Through Families of Experiments. **IEEE Transactions on Software Engineering**, v. 25, n. 4, p. 456-473, 1999.
104. ARISHOLM, E. et al. Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise. **IEEE Transactions on Software Engineering**, v. 33, n. 2, p. 65-86, 2007.

Anexo A – Manifesto Ágil

Manifesto para Desenvolvimento Ágil de Software

Estamos descobrindo maneiras melhores de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo. Através deste trabalho, passamos a valorizar:

Indivíduos e interações mais que processos e ferramentas
Software em funcionamento mais que documentação abrangente
Colaboração com o cliente mais que negociação de contratos
Responder a mudanças mais que seguir um plano

Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

© 2001, os autores acima

Esta declaração pode ser copiada livremente em qualquer formato, mas somente integralmente através desta declaração.

Princípios por trás do Manifesto Ágil

Nós seguimos estes princípios:

Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.

Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento.

Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.

Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.

Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.

Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.

O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.

Software funcionando é a medida primária de progresso.

Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.

Contínua atenção à excelência técnica e bom design aumenta a agilidade.

Simplicidade--a arte de maximizar a quantidade de trabalho não realizado--é essencial.

As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.

Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

Anexo B – Entrevista

INSTRUMENTO DE VALIDAÇÃO

Objetivos do estudo

“Investigar as causas do aumento médio de produtividade (em torno de 15%) indicado pelas métricas nos projetos onde se mesclou Scrum no processo de desenvolvimento.”

IMPORTANTE: A ordem das questões é muito importante para assegurar a validade do instrumento. Assim, por favor responda as questões na ordem em que aparecem sem “espiar” as questões subsequentes.

1) Recordação

A critério de recordação, os primeiros projetos onde se mesclou Scrum no processo são sumarizados a seguir:

Projeto	Tech	PF	Horas	2008					2009											
				ago	set	out	nov	dez	jan	fev	mar	abr	mai	jun	jul	ago	set	out	nov	dez
S1	J2EE	80	858																	
S2	J2EE	123	1372																	
S3	.NET	128	1329																	
S4	J2EE	185	1945																	
S5	J2EE	159	1881																	
S6	J2EE	234	2139																	
S7	J2EE & C++	249	2666																	
S8	J2EE	168	1558																	

2) Classificação do respondente

- Desde que ano trabalha (profissionalmente / em empresa) com desenvolvimento de software? _____
- E nesta empresa? _____

- De quais dos projetos citados você participou (responda “sim”, “não”, ou “não recordo”)?

Projeto	Participou?	Caso a coluna anterior seja “sim”, atribua um valor de 0 a 10 para o quanto você se recorda do desenvolvimento deste projeto (10 = recordo totalmente)
S1		
S2		
S3		
S4		
S5		
S6		
S7		
S8		

3) Entendimento do processo

- Por favor descreva com suas palavras o que mudou no processo de desenvolvimento ao se introduzirem atividades típicas de SCRUM:

- Este entendimento ocorreu já no início da adoção (S/N)? ____
- Se não, quanto tempo levou para você obter um entendimento satisfatório da mudança?

4) Impressões nos projetos onde se mesclou Scrum no processo

- Por favor descreva com suas palavras quais alterações ocorreram na produção de artefatos e documentações (há artefatos que foram suprimidos ou acrescentados)?

- Quais fatores, no seu julgamento, contribuíram para o aumento de produtividade?

Fatores relacionados ao Scrum	Fatores NÃO-relacionados ao Scrum

- Classifique a influência dos seguintes “fatores relacionados ao Scrum” que, na sua opinião, afetaram a produtividade nos projetos citados:

Fator	Produtividade				
	Caiu muito	Caiu pouco	Neutro	Aumentou pouco	Aumentou muito
Cooperação do time					
Comunicação do time					
Moral / Motivação					
Flexibilidade					
Respeito					
Confiança					
Dinâmica / flexibilidade					
Adaptabilidade					
Foco					
Tamanho da iteração					
Time-boxing					
Gestão de backlog					
Entrega evolutiva					
Resposta a requisições de mudança					
Papéis					
Artefatos					
Distribuição física da equipe					
Outro? _____					
Outro? _____					
Outro? _____					
Outro? _____					

- Classifique a influência dos seguintes “fatores NÃO-relacionados ao Scrum” que, na sua opinião, afetaram a produtividade nos projetos citados:

Fator	Produtividade				
	Caiu muito	Caiu pouco	Neutro	Aumentou pouco	Aumentou muito
Mudança na equipe durante o projeto					
Projetos similares / reaproveitamento					
Complexidade do produto					
Complexidade da interface com usuário					
Riscos dos requisitos					
Linguagem					
Ferramentas de desenvolvimento					
Política de recompensa					
Cronograma apertado					
Experiência do time					
Capacidade do líder					
Capacidade do time					
Cultura da empresa					
Outro? _____					
Outro? _____					
Outro? _____					
Outro? _____					

- Você considera que sua produtividade individual foi alterada no período específico dos primeiros projetos citados?

() caiu pouco

- () neutro
- () aumentou pouco
- () aumentou consideravelmente
- () aumentou muito

- Agora faça uma lista consolidada, em ordem de importância (1 = mais importante), dos fatores (relacionados e não-relacionados ao Scrum) que aumentaram a produtividade e dê uma breve explicação do como/porque você entende que eles aumentaram a produtividade.

	Fator	Como / Por que?
1		
2		
3		
4		
5		
6		
7		
8		

Anexo C – Avaliação de Qualidade

A estudo empírico desta tese foi submetido à avaliação de qualidade utilizada em [16], a qual é mostrada a seguir. Esta avaliação é específica sobre processos ágeis e sobre o momento atual de maturidade de pesquisa nesta área, mas serve como uma base para reflexão. O estudo obteve a nota 11 de um total de 11 questões.

PARTE I – QUESTÕES DE TRIAGEM

1. Is this a research paper? Consider: –Is the paper based on research (or is it merely a “lessons learned” report based on expert opinion? (+)	(x) Sim () Não
2. Is there a clear statement of the aims of the research? Consider: –Is there a rationale for why the study was undertaken? (+) –Is the study’s focus or main focus on Agile Software Development? (+/-) –Does the study present empirical data? (+) –Is there a clear statement of the study’s primary outcome (i.e. time-to-market, cost, or product or process quality)? (+)	(x) Sim () Não
3. Is there an adequate description of the context in which the research was carried out? Consider whether the researcher has identified: –The industry in which products are used (e.g. banking, telecommunications, consumer goods, travel, etc) (+) –The nature of the software development organization (e.g. in-house department or independent software supplier) (+) –The skills and experience of software staff (e.g. with a language, a method, a tool, an application domain) (+) –The type of software products used (e.g. a design tool, a compiler) (+) –The software processes being used (e.g. a company standard process, the quality assurance procedures, the configuration management process) (+)	(x) Sim () Não

QUESTÕES DETALHADAS (aplicáveis se as respostas da questão 1 e pelo menos uma dentre as questões 2 e 3 forem “Sim”)

Research design 4. Was the research design appropriate to address the aims of the research? Consider: – Has the researcher justified the research design (e.g. have they discussed how they decided which methods to use)? (+)	(x) Sim () Não
Sampling 5. Was the recruitment strategy appropriate to the aims of the research? Consider: –Has the researcher explained how the participants or cases were identified and selected? (+) –Are the cases defined and described precisely? (+)	(x) Sim () Não

–Were the cases representative of a defined population? (+/-) –Have the researchers explained why the participants or cases they selected were the most appropriate to provide access to the type of knowledge sought by the study? (+) –Was the sample size sufficiently large? (+/-)	
Control group 6. Was there a control group with which to compare treatments? Consider: –How were the controls selected? (+) –Were they representative of a defined population? (+/-) –Was there anything special about the controls? (n/a) –Was the non-response high? Could non-respondents be different in any way? (n/a)	(x) Sim () Não
Data collection 7. Was the data collected in a way that addressed the research issue? Consider: –Were all measures clearly defined (e.g. unit and counting rules)? (+) –Is it clear how data was collected (e.g. semi-structured interviews, focus group etc.)? (+) –Has the researcher justified the methods that were chosen? (+) –Has the researcher made the methods explicit (e.g. is there an indication of how interviews were conducted, did they use an interview guide)? (+/-) –If the methods were modified during the study, has the researcher explained how and why? (n/a) –Whether the form of the data is clear (e.g. tape recording, video material, notes etc.) (+) –Whether quality control methods were used to ensure completeness and accuracy of data collection (+/-)	(x) Sim () Não
Data analysis 8. Was the data analysis sufficiently rigorous? Consider: –Was there an in-depth description of the analysis process? (+) –If thematic analysis was used, is it clear how the categories/ themes were derived from the data? (n/a) –Has sufficient data been presented to support the findings? (+/-) –To what extent has contradictory data been taken into account? (+/-) –Whether quality control methods were used to verify the results (+/-)	(x) Sim () Não
Reflexivity (research partnership relations/recognition of researcher bias) 9. Has the relationship between researcher and participants been considered adequately? Consider: –Did the researcher critically examine their own role, potential bias and influence during the formulation of research questions, sample recruitment, data collection, and analysis and selection of data for presentation? (+) –How the researcher responded to events during the study and whether they considered the implications of any changes in the research design. (n/a)	(x) Sim () Não
Findings 10. Is there a clear statement of findings? Consider: –Are the findings explicit (e.g. magnitude of effect)? (+) –Has an adequate discussion of the evidence, both for and against the researcher's arguments, been demonstrated? (+) –Has the researcher discussed the credibility of their findings (e.g. triangulation, respondent validation, more than one analyst)? (+) –Are limitations of the study discussed explicitly? (+) –Are the findings discussed in relation to the original research questions? (+) –Are the conclusions justified by the results? (+)	(x) Sim () Não
Value of the research 11. Is the study of value for research or practice?	(x) Sim () Não

Consider: –Does the researcher discuss the contribution the study makes to existing knowledge or understanding (e.g. do they consider the findings in relation to current practice or relevant research-based literature)? (+) –Does the research identify new areas in which research is necessary? (+) –Does the researcher discuss whether or how the findings can be transferred to other populations, or consider other ways in which the research can be used? (+)	
---	--

Anexo D – Ferramentas dos Projetos

Neste anexo são apresentadas as referências para as ferramentas utilizadas nos projetos do estudo de caso. Foi dada a preferência ao artigo Wikipedia (ao invés do site oficial) para evitar desatualizações da referência devido a possíveis eventos futuros.

IDE

Eclipse – [http://en.wikipedia.org/wiki/Eclipse_\(software\)](http://en.wikipedia.org/wiki/Eclipse_(software))

MSVS – http://en.wikipedia.org/wiki/Microsoft_Visual_Studio

CASE

EA – <http://www.sparxsystems.com/products/ea/>

Power Designer – <http://en.wikipedia.org/wiki/PowerDesigner>

Erwin – http://en.wikipedia.org/wiki/CA_ERwin_Data_Modeler

FRAMEWORK

Spring – http://en.wikipedia.org/wiki/Spring_Framework

JSF – http://en.wikipedia.org/wiki/JavaServer_Faces

Hibernate – http://en.wikipedia.org/wiki/Hibernate_%28Java%29

Struts – http://en.wikipedia.org/wiki/Apache_Struts

Velocity – http://en.wikipedia.org/wiki/Apache_Velocity

TESTES

Jmeter – <http://en.wikipedia.org/wiki/JMeter>

Junit – <http://en.wikipedia.org/wiki/JUnit>

Selenium – [http://en.wikipedia.org/wiki/Selenium_\(software\)](http://en.wikipedia.org/wiki/Selenium_(software))

OUTRAS

Subversion – http://en.wikipedia.org/wiki/Apache_Subversion

Maven – http://en.wikipedia.org/wiki/Apache_Maven

Jetty – [http://en.wikipedia.org/wiki/Jetty_\(Web_server\)](http://en.wikipedia.org/wiki/Jetty_(Web_server))

Jboss – http://en.wikipedia.org/wiki/JBoss_application_server

Anexo E – Comparação de Distribuições

Para se comparar as distribuições de N amostras, um dos testes mais preferidos é o teste paramétrico *ANOVA One-way*, que testa a hipótese nula de que não há diferença estatística entre as médias das N amostras.

Entretanto, o teste *ANOVA One-way* exige que a variável testada possua distribuição normal e que as variâncias de todas as amostras sejam estatisticamente iguais.

Caso alguma dessas condições não seja satisfeita, ainda pode-se optar pela alternativa não paramétrica, que é teste de *Kruskal-Wallis*, que testa a hipótese nula de que não há diferença estatística entre as medianas das N amostras.

Uma representação gráfica desta estrutura de decisão para a comparação de distribuições é mostrada a seguir na Figura 27.

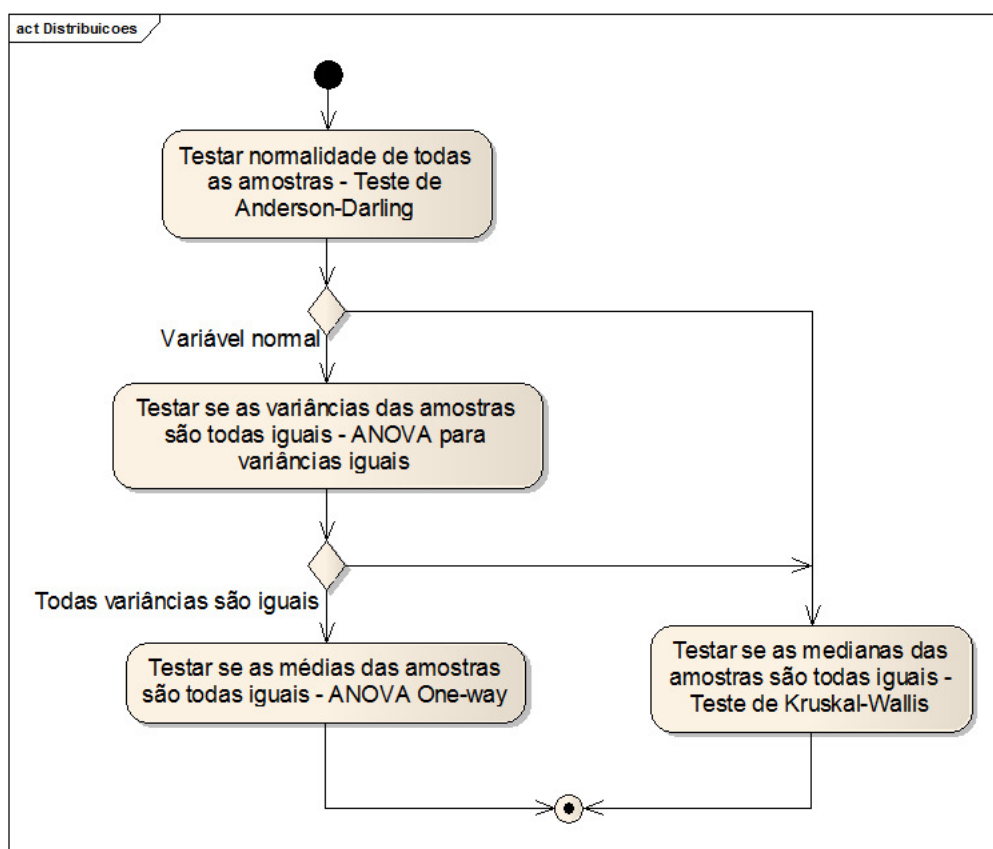


Figura 27 – Decisão do teste estatístico para se comparar distribuições

Anexo F – Dados brutos – Etapa 1

ENTRE-VISTADO	PORÇÃO DE TEXTO (DADOS BRUTOS)	CÓDIGO
E01	Menos documentação	Diminuição da documentação
E01	Maior interação entre desenvolvedores	Maior colaboração
E01	Mais detalhes do projeto são passados para o time	Melhor comunicação
E02	Documentação mais superficial	Diminuição da documentação
E02	Pressão do gerente	Pressão do gerente
E02	Aumento de contato entre colegas do time	Melhor comunicação
E03	Aumento gradual da equipe de testes	Aumento gradual da equipe
E03	e automatização de testes	Ferramentas (teste)
E03	Maior envolvimento dos analistas nas questões do projetos (embora o Scrum Master centralize o conhecimento do projeto)	Gerência participativa
E03	Maior interação entre os membros do time	Maior colaboração
E03	Reuniões de revisão periódicas (no final dos sprints) ajudam a corrigir o que está errado mais cedo	Micro gerenciamento & Melhor Comunicação
E04	Menos documentação	Diminuição da documentação
E04	Ferramentas de testes	Ferramentas (teste)
E04	Mais contato entre os analistas do projeto	Melhor comunicação
E04	Planejamentos e revisões constantes	Micro gerenciamento & Melhor Comunicação
E05	Mais contato entre os desenvolvedores	Melhor comunicação
E06	Proximidade física	Maior colaboração
E06	Melhoria da comunicação	Melhor comunicação
E06	Monitoramento informal frequente	Micro gerenciamento & Melhor Comunicação
E06	Testes	Subprocesso de testes
E07	Menos tempo documentando	Diminuição da documentação
E07	Equipes homogêneas	Equipes homogêneas
E07	Comunicação	Melhor comunicação
E08	Colaboração da equipe	Maior colaboração
E08	Facilidade de interação com Scrum Master	Melhor comunicação
E08	Reuniões periódicas para resolver impedimentos	Micro gerenciamento & Melhor Comunicação
E09	Frameworks de desenvolvimento	Ferramentas (frameworks)
E09	Colaboração	Maior colaboração
E09	Comunicação	Melhor comunicação
E09	Monitoramento frequente; Cobranças localizadas	Micro gerenciamento
E10	Documentação mais resumida	Diminuição da documentação
E10	Automatização de testes	Ferramentas (teste)
E10	Equipe em baias próximas	Maior colaboração
E11	Comunicação entre analistas da equipe	Melhor comunicação
E11	Equipe de testes	Subprocesso de testes

Anexo G – Dados brutos – Etapa 3

ENTRE-VISTADO	PORÇÃO DE TEXTO (DADOS BRUTOS)	ORDEM	CÓDIGO
E01	Maior interação com colegas	1	Maior colaboração
E01	Maior contato com cliente	2	Melhor comunicação
E01	Solicitações de mudança mais transparentes	3	Subprocesso de teste
E01	Plannings mais granulares (a gestão de backlog ficou mais fácil)	4	Micro gerenciamento
E02	Menos documentação	1	Diminuição da documentação
E02	Mais integração com time	2	Maior colaboração
E03	Maior interação entre membros do time	1	Maior colaboração
E03	Menos documentação	2	Diminuição da documentação
E03	Reuniões periódicas	3	Micro gerenciamento & Melhor Comunicação
E04	Mais contato com as pessoas.	1	Melhor comunicação
E04	Reuniões diárias.	2	Micro gerenciamento & Melhor Comunicação
E04	Menos detalhes de projeto	3	Diminuição da documentação
E05	Documentação flexível	4	Diminuição da documentação
E05	Comunicação	1	Melhor comunicação
E05	Interação entre equipe	2	Maior colaboração
E05	QA alocado o tempo todo	3	Pressão do gerente
E06	Melhoria da comunicação	1	Melhor comunicação
E06	Proximidade física	2	Maior colaboração
E06	Monitoramento informal frequente	3	Micro gerenciamento & Melhor Comunicação
E07	Comunicação direta entre equipe	1	Melhor comunicação
E07	Pontos de controle diários	2	Micro gerenciamento & Melhor Comunicação
E07	Documentação ligh	3	Diminuição da documentação
E08	Colaboração	1	Maior colaboração
E08	Documentação menos densa	2	Diminuição da documentação
E08	Alinhamentos constantes	3	Micro gerenciamento & Melhor Comunicação
E09	Colaboração	1	Maior colaboração
E09	Comunicação	2	Melhor comunicação
E09	Monitoramento frequente	3	Micro gerenciamento & Melhor Comunicação
E10	Menos documentação	1	Diminuição da documentação
E10	Participação de todos nas reuniões	2	Melhor comunicação
E11	Comunicação entre equipe.	1	Melhor comunicação
E11	Resolução mais rápida de pendências e ajuda mútua.	2	Maior colaboração