Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Estudo de mapeamento sistemático em dependabilidade e métodos ágeis: uma reprodução de estudo de 2012 a 2015

Gabrielly Ferreira Leonardo de Lima

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientadora
Prof.ª Dr.ª Genaína Nunes Rodrigues

Brasília
2016

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Coordenador: Prof. Dr. Homero Luiz Piccolo

Banca examinadora composta por:

Prof.ª Dr.ª Genaína Nunes Rodrigues (Orientadora) — CIC/UnB
Prof.ª Dr.ª Fernanda Lima — CIC/UnB
Prof. Dr. Rodrigo Bonifacio de Almeida — CIC/UnB

CIP — Catalogação Internacional na Publicação

de Lima, Gabrielly Ferreira Leonardo.

Estudo de mapeamento sistemático em dependabilidade e métodos ágeis: uma reprodução de estudo de 2012 a 2015 / Gabrielly Ferreira Leonardo de Lima. Brasília : UnB, 2016.
69 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2016.

1. Métodos Ágeis, 2. dependabilidade, 3. revisão sistemática

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Estudo de mapeamento sistemático em dependabilidade e métodos ágeis: uma reprodução de estudo de 2012 a 2015

Gabrielly Ferreira Leonardo de Lima

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof.ª Dr.ª Genaína Nunes Rodrigues (Orientadora)
CIC/UnB

Prof.ª Dr.ª Fernanda Lima
CIC/UnB

Prof. Dr. Rodrigo Bonifacio de Almeida
CIC/UnB

Prof. Dr. Homero Luiz Piccolo
Coordenador do Bacharelado em Ciência da Computação

Brasília, 17 de fevereiro de 2016

Dedicatória

Eu dedico este trabalho à minha família, ao meu noivo e a todos que estiveram comigo durante esses 6 anos de curso. Cada um foi importante de alguma maneira para que eu chegasse até aqui.

Agradecimentos

Agradeço à minha orientadora, que durante todo o processo me apoiou, ao meu noivo, que sempre esteve me incentivando e me mantendo em casa para que eu pudesse terminar e ao *Google*, sempre presente nas horas de dúvidas indecifráveis.

Resumo

Os métodos ágeis têm sido cada vez mais adotados no desenvolvimento de software, desde sistemas simples até sistemas críticos, como *Spaced-Based Systems* ou *Mission Critical*, porém ainda é pequena a quantidade de estudos publicados que analisam a influência do uso dessas metodologias na dependabilidade do software. Neste trabalho é feita a reprodução do estudo sistemático da literatura desenvolvido por Braga e Leal [7], o qual faz uma análise sobre a utilização dos métodos ágeis e falhas e defeitos, ao mesmo tempo em que é realizada sua atualização estendendo os resultados para abranger os anos de 2013 a 2015. Além disso, são apontadas novas percepções acerca do assunto, como o aumento do uso dos métodos ágeis em sistemas considerados críticos. Este tipo de estudo visa mensurar se a quantidade de trabalhos publicados sobre o assunto é suficiente para responder os questionamentos propostos, além de servir como indicador sobre quais trabalhos ainda não foram desenvolvidos.

Palavras-chave: Métodos Ágeis, dependabilidade, revisão sistemática

Abstract

The use of agile methods has grown, however the number of studies that examine the influence of the use of these methodologies in the dependability of software is still small. This work presents the reproduction of the mapping study on the existing literature of Braga and Leal [7], which examine the effect agile practices have on the occurrence of faults and failures. Moreover it was made an update study extending the results for the years 2013 to 2015. At the end, new perceptions about the theme are presented, like the increasing of agile methods adoption in safety-critical systems.

Keywords: Agile methods, Dependable, systematic review

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A Engenharia de Software experimental é composta basicamente de dois tipos de investigação: estudos primários e estudos secundários. De acordo com Jorge Calmon et al. [11] estudos primários são estudos que visam a avaliação de uma hipótese formulada pelo pesquisador usando testes aplicados em ambientes com condições bem definidas e controladas de acordo com a metodologia do estudo. Estudos secundários são destinados a produzir comparações entre estudos individuais, cientificamente selecionados a partir de um conjunto preliminar, tendo como um dos objetivos produzir generalizações.

Pesquisadores tem aplicado estudos primários para construir a base de conhecimento em Engenharia de Software [5] e dar base para pesquisas relacionadas a engenharia de tecnologias de software, principalmente os estudos que visam avaliar tais tecnologias [1]. Já os estudos secundários, conhecidos também como Revisões Sistemáticas [10], são usados como suporte para a construção de evidências em engenharia de software [22], além de representarem um ponto de partida para o desenvolvimento de novas tecnologias na área. Há casos de aplicação da revisão sistemática como uma das metodologias usadas para inspecionar modelos de arquitetura de software na indústria, por exemplo.

Fazer uma Revisão sistemática requer que cada passo da metodologia seja cuidadosamente planejado para garantir a necessária robustez e consistência dos resultados e conclusões [9]. Tudo deve ser feito antes de começar a execução e todo o processo deve ser documentado, incluindo os resultados intermediários. Outro ponto importante é a definição explícita e precisa das questões que a pesquisa quer responder, tendo em vista a quantidade de informação que é possível se retirar da quantidade de estudos primários selecionados. A questão central da pesquisa da revisão sistemática é a estrutura multidimensional da informação que deve estar relacionada a todos os passos da pesquisa. Os pontos consistentes percebidos entre diferentes pesquisadores na estruturação conceitual feita durante a fase de planejamento levando a garantia de obtenção de resultados similares poderia ser melhorado baseando-se em uma terminologia comum formalizada dos

conceitos envolvidos, representado por uma ontologia [11]. Ainda segundo [11] Entretanto, ontologias tem uma série de limitações, das quais podemos citar como principais o fato de serem difíceis de evoluir e a falta de ferramentas para tradução delas. Mesmo assim, acredita-se que a pesquisa de revisão sistemática pode se beneficiar e muito do uso de ontologias, por causa da sua aplicabilidade na formalização conceitual e representação do conhecimento.

Com o pouco escrito, já é possível notar que uma revisão sistemática é um tipo de pesquisa extremamente técnico, um processo sistemático com uma sequência de passos metodológicos estritos e que devem estar extremamente bem definidos. [11] diz que os passos metodológicos, as estratégias para recuperar evidencias e o foco das questões explicitamente definido e a documentação passo a passo de cada um desses pontos, faz com que o mesmo método possa ser reproduzido por outros pesquisadores, tornando-os habilitados a julgar os resultados anteriores e propor melhorias, além de ser possível de tirar diversas outras conclusões.Esse fato mostra que estudos secundários são ferramentas cientificas extremamente poderosas.

Há na literatura outros exemplos de estudos secundários como os estudos de mapeamento, do inglês *"mapping study"*, muito utilizado na medicina, e usado recentemente em um estudo do Departamento de Ciência da Computação da universidade de Brasília para investigar e reunir o conhecimento existe na literatura sobre métodos ágeis e dependabilidade, visando concluir se há evidencias que técnicas de métodos ágeis são eficientes no desenvolvimento de software confiável. [7]

1.1 Motivação

Nos últimos anos os métodos ágeis ganharam grande visibilidade por supostamente melhorar a gerência de projetos, além de aumentar a qualidade do produto e a satisfação do cliente. Diversos estudos trazem esta temática, porém a maioria focam na qualidade em geral, no sucesso e falha dos projetos ou mesmo na produtividade das equipes, chegando até a abordar como os programadores se sentem ao utilizar essa abordagem. Existem em abudância estudos que comparam a utilização de métodos ágeis e métodos tradicionais, como o cascata, além de estudos excelentes como o apresentado por Dyba *et al.* [12]

Diante desse contexto é estranho a ausência de estudos que abordem mais profundamente a relação entre termos de dependabilidade e métodos ágeis. A explicação pode estar no fato de que pesquisadores tem dificuldade de associar conceitos de dependabilidade com os métodos iterativos, ou na falta de métricas que avaliem a dependabilidade na utilização de tais metodologias. [41] [17]

1.2 Problema

Baseado no estudo de mapeamento sistemático de Artur Braga e Renato Leal [7], O problema foco desta monografia, de forma geral, é tentar reproduzir o trabalho realizado por eles até então. No entanto, uma vez que só são apresentados dados até meados de 2012, foi necessária a atualização para o ano de 2015.

1.3 Objetivos

1.3.1 Objetivo Geral

Este trabalho tem como objetivo geral, além de reproduzir atualizando o estudo de Artur Braga e Renato Leal [7], investigar e reunir de forma sistemática o conhecimento disponível na literatura especializada sobre os temas, através de um agrupamento dos estudos relevantes que abordam a ocorrência de falhas e defeitos com a utilização de métodos ágeis, agora aplicando o estudo para os anos 2013 a 2015.Tem como objetivo também encontrar durante a reprodução do trabalho lacunas na literatura e explorá-la. Outros objetivos incluem identificar se há falta de literatura especializada sobre os temas apontados como lacunas e deste modo trazer mais atenção ao tema além de servir como base de consulta para que outros estudos possam ser realizados. Para atingir tais objetivos foi realizado um estudo de mapeamento sistemático com questões que, ao serem respondidas, colocam em evidência onde estão sendo colocados, ou não, os esforços da comunidade acadêmica, além das práticas ágeis que propiciam uma maior qualidade de software quando abordados em termos de dependabilidade.

1.3.2 Objetivos Específicos

As questões de pesquisa, que serão apresentadas no Capítulo 4, buscam elucidar os seguintes objetivos:

- Identificar quais são as lacunas no trabalho usado como base e fazer o mapeamento focando neles.
- Identificar quais são os grupos de pesquisa sobre o tema, classificando-os entre a academia e o mercado.
- Identificar as principais práticas de métodos ágeis que auxiliam no desenvolvimento de um software fidedigno.
- Identificar quais são os atributos de dependabilidade citados como os mais importante para o desenvolvimento de software fidedigno

1.4 Metodologia

Neste trabalho primeiramente foi realizada a reprodução do estudo de Artur Braga e Renato Leal [7], o qual foi realizado em duas etapas: inicialmente foi feito um levantamento do referencial teórico sobre os assuntos de métodos ágeis e dependabilidade para a melhor compreensão do objeto de estudo. Em seguida houve a realização do estudo sistemático *(systematic mapping study)* para a identificação, classificação e análise dos estudos de relevância para o trabalho base. Após essa reprodução, foram listados quais outros pontos relacionados a métodos ágeis e dependabilidade a literatura não aborda e a partir dessa lista houve a realização de um novo mapeamento sistemático. A ultima etapa foi a escrita desta monografia.

1.5 Organização do trabalho

Este trabalho se divide em 7 capítulos, os quais:

Capítulo 2 São apresentadas a teoria e os principais conceitos de dependabilidade abordados nesse trabalho, como os classificaremos e sua importância em um software. O objetivo desse capítulo é dar ao leitor a fundamentação teórica necessária para entender todo o tema deste estudo.

Capítulo 3 Traz um referencial teórico sobre métodos ágeis que fazem parte do contexto desse estudo. Serão apresentadas as principais práticas utilizadas por estes métodos, introduzidas pelo manisfesto ágil [14] .Serão abordadas tambem algumas práticas específicas como o SCRUM de Schwaber [39], o Lean Development de Poppendieck e Poppendieck [36] e o XP *(extreme programming)* de Beck [6].

Capítulo 4 Apresenta a teoria sobre a realização do estudo de mapeamento sistemático *(mapping studies)*, explicando cada passo realizado e narrando como foi realizado os passos neste trabalho, apresentando os resultados de cada etapa.

Capítulo 5 Traz a reprodução e atualização do estudo base, mostrando a comparação dos resultados encontrados na estapa de levatamento dos dados.

Capítulo 6 São apresentados os resultados obtidos e é feita a análise utilizando os mesmos critérios que o estudo base. São respondidas tambem as questões de pesquisa definidas neste trabalho e é feita uma análise dos *gaps* encontrados.

Capítulo 7 São apresentadas as limitações e ameaças à validade do trabalho, assim como são trazidas propostas de trabalhos futuros a partir dos resultados encontrados e finaliza-se com as conclusões do trabalho.

Capítulo 2

Dependabilidade

Dependabilidade de sistemas computacionais *(dependability)* é a habilidade de entregar um serviço que pode, justificadamente, ser confiado (Avizienis et al. [4]). Além do conceito, para o entendimento do trabalho são necessários também três conceitos básicos, são eles: as ameaças à dependabilidade do sistema, os atributos de dependabilidade e os meios que permitem alcançá-la.

Será apresentada a seguir descrição dos atributos de dependabilidade, focando no atributo confiabilidade *(reliability)*, e então poderemos falar mais detalhadamente sobre as ameaças e meios de se obter um sistema no qual se possa confiar.

2.1 Atributos

Atributos são qualidades do sistema, que podem definir a dependabilidade utilizando medidas qualitativas e quantitativas no funcionamento de um sistema ou em sua documentação [7]. De acordo com Avizienis et al. [4] o conceito de dependabilidade pode ser dividido em:

1. Disponibilidade (*Availability*): probabilidade de um sistema estar funcionando corretamente em um dado instante do tempo.
2. Confiabilidade (*Reliability*): probabilidade de um sistema estar funcionando corretamente de acordo com as especificações dadas, dentro de certas condições em um período de tempo.
3. Segurança (*Safety*): ausência de consequências catastróficas para os usuários (humanos ou outros sistemas) e/ou para o próprio ambiente em que o sistema está inserido.

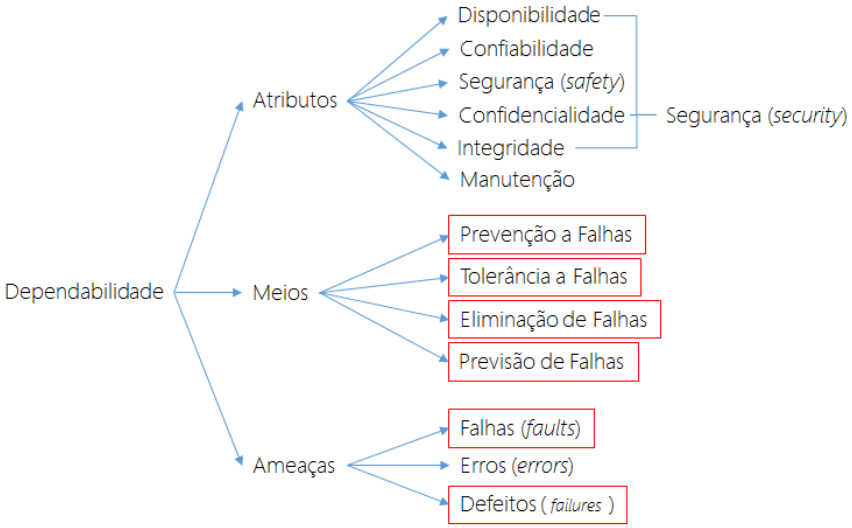

Figura 2.1: Desenvolvimento incremental [4] Tradução de Avizienis et al. Os destaques em vermelho foram o foco da análise do trabalho Base desse estudo [7].

4. Integridade (*Integrity*): ausência de alterações que modifiquem o comportamento do sistema. O sistema mantém-se conforme entregue e os dados continuam válidos.

5. Manutenibilidade (*Maintainability*): habilidade de receber reparos e modificações, com baixa indisponibilidade e com o mínimo de impacto para os usuários, sejam estas modificações correções de falhas ou implementação de novas funcionalidades.

6. Confidencialidade (*Confidentiality*): a ausência de divulgação não-autorizada de informações. Alguns autores não o classificam como um atributo primário e sim um atributo derivado quando se avalia a questão de segurança em nível de controle de acesso.

Na Figura 2.1 é apresentada tambem o requisto não-funcional de segurança (*security*), o qual corresponde a junção dos atributos confidencialidade, disponibilidade e integridade.

No trabalho base deste estudo o foco foi confiabilidade (*reliability*) de sistemas computacionais. Como o principal intuito do trabalho é realizar a reprodução do que foi feito no já citado a seguir é apresentada uma definição mais detalhada para este atributo.

2.1.1 Confiabilidade

É a probabilidade de que um sistema desempenhará satisfatoriamente (ou adequadamente) o seu propósito específico por um dado período até a ocorrência da primeira falha [27], [25] e [31]. Segundo Lyu [26], quando falamos de dependabilidade o atributo com maior destaque é a confiabilidade , além de ser a medida mais usada em sistemas em

que mesmo curtos períodos de operação incorreta são inaceitáveis. Elá é considerada um fator chave para expressar sucesso ou fracasso de uma aplicação.

Por ser uma função de probabilidade podemos realizar previsões e medições a partir de uma equação matemática na qual a função de confiabilidade R(t) é a probabilidade de que o sistema irá funcionar corretamente sem falha no intervalo de tempo de 0 a t e T é uma variável aleatória representando o tempo de falha ou tempo para falha, conforme apresentado na equação (2.1) a seguir dada por (Maciel et al. [31]):

$$\mathrm{R(t)} = \mathrm{P(T > t)}, \mathrm{t} >= 0 \tag{2.1}$$

E a probabilidade de falha, ou inverso da confiabilidade, é o representada por:

$$\mathrm{F(t)} = 1 - \mathrm{R(t)} = \mathrm{P(T > t)} \tag{2.2}$$

conhecida como a função de distribuição de T, a qual nos diz que um componente não apresentará defeitos até o tempo t.

Confiabilidade não é disponibilidade. Um sistema pode ser disponível e possuir períodos de inoperância, entretanto estes períodos devem ser curtos a ponto de não comprometer a qualidade do serviço.

2.2 Ameaças

São fatores que podem afetar um sistema e diminuir sua dependabilidade, podendo ser divididas nos três conceitos seguintes: falhas *(faults)*, erros *(errors)* e defeitos *(failures)* [10]:

Falha (*fault*) ou falha é um estado incorreto do hardware ou do software podendo resultar em problemas caso seja percebida no contexto físico, erros no contexto da informação e defeitos no contexto do usuário. Um sistema pode apresentar uma ou mais falhas e não apresentar erros, a chamada falha dormente (ou latente). Existe tambem a falha ativa, que ocorre quando efetivamente produz um erro. O tempo entre o surgimento da falha e sua ativação, ou produção do erro, é chamado de latência da falha.

Erro (*error*) é uma manifestação da falha no sistema, correspondendo a um estado do sistema que pode levar a outras falhas ou pode não causar nenhuma interrupção de algum serviço, apenas inconsistência entre dados esperados e percebidos.

Defeito (*failure*) é um desvio do que seria o comportamento correto do sistema (*correct service*). Segundo Avizienis et al. [4] pode ocorrer porque alguma funcionalidade

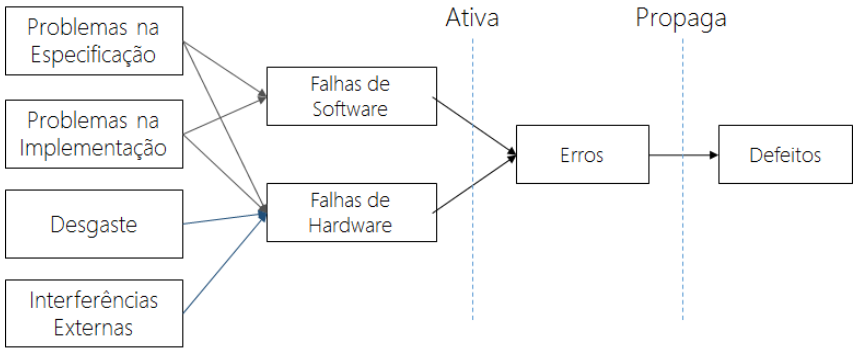

Figura 2.2: Relação de Causa-Efeito entre Falhas, Erros e Defeitos. Retirada de [7]

não segue adequadamente os requisitos ou porque os requisitos não descrevem a funcinalidade adequadamente.

A Figura 2.2 demonstra o fluxo de relação entre falha, erro e defeito.

A seguir são apresentadas uma descrição mais detalhada sobre os conceitos de falhas e defeitos, temas estudados no rabalho base.

Falhas

De acordo com Avizienis et al. [4] falhas de um sistema podem ser classificadas em oito classes elementares. Cada falha pode pertencer à uma dessas classes ou à uma combinação, as quais são chamadas classes de falhas combinadas. Essas se restringem a 31 tipos de falhas, agrupadas em três grandes categorias: falhas de desenvolvimento, físicas (afetam o hardware) ou de intenção. Esta categorização está representada na Figura 2.3.

Fase de criação ou ocorrência Nesta fase são classificadas de acordo com a fase do ciclo de vida na qual elas surgem e são divididas em falhas de desenvolvimento e falhas operacionais. Falhas de desenvolvimento podem ocorrer durante a implementação ou durante a manutenção do sistema e as falhas operacionais ocorrem na entrega do sistema.

Limites do sistema Aqui as falhas são classificadas de acordo com o sua localização, podendo ser internas ou externas.

Causas fenomenológicas Classificadas entre falhas humanas ou falhas naturais. A primeira está relacionada a todo tipo de falha que pode ser causada por erro humano, como má escolha no métodos de desenvolvimento a ser usado, testes incompletos ou mal elaborados, entre outros. Já as falhas naturais estão relacionadas ao ambiente

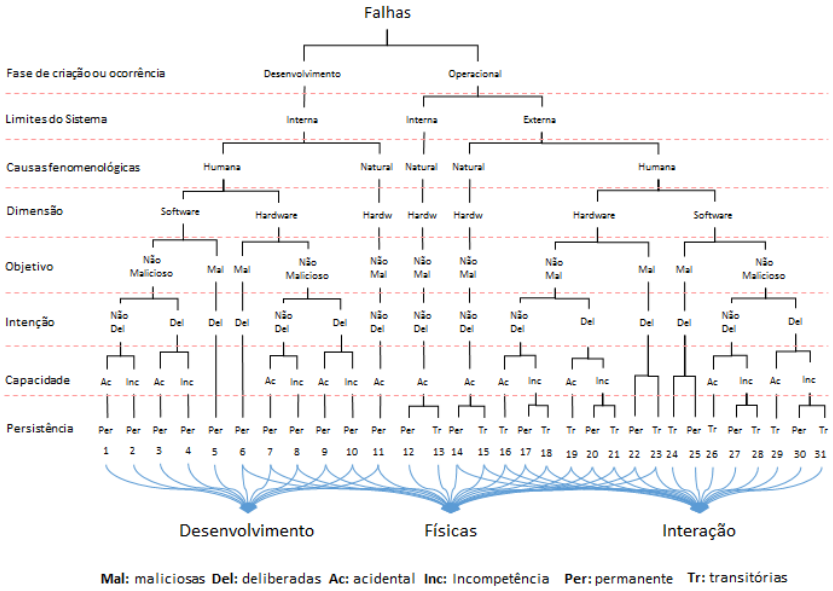

Figura 2.3: Classificação das Falhas. Tradução de Avizienis, retirada de [7]

em que o sistema está inserido, isso inclui desde servidores até o tempo de vida útil dos componentes da máquina utilizada.

Dimensão Aqui as falhas são classificadas em falhas de hardware, que são originadas a partir do hardware ou o afetam, e falhas de software, presente na camada de informação do sistema.

Objetivos Dividas em falhas maliciosas, causadas propositalmente por seres humanos e não maliciosas, causadas por falta de conhecimento ou descuido por exemplo.

Intenção Dividas em falhas deliberadas (resultado de quando humanos tomam decisões prejudiciais ao funcionamento do sistema) e falhas não deliberadas (introduzidas sem conhecimento).

Capacidade Dividas em falhas acidentais e falhas por incompetência (resultado da falta de competência profissional do humano ou inadequação da organização desenvolvedora).

Temporal São divididas em falhas permanente, caracterizada por ser estável e contínua, e transitórias, as quais são resultados causados por condições temporárias no sistema.

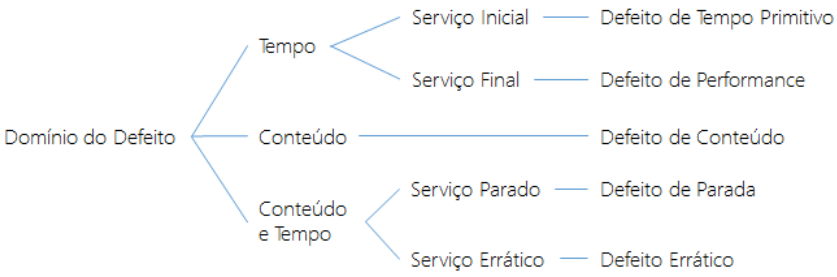

Figura 2.4: Classificação dos defeitos de domínio. Tradução de Avizienis. [4]

Defeitos

São desvios no comportamento correto do sistema que chegam a ser percebidos pelo usuário. Pode se manifestar de diversas formas, sendo esses **modos de defeito** classificados de acordo com quatro pontos de vista (Figura 2.5):

Domínio do defeito Podem ser defeitos de conteúdo e defeitos de tempo. Defeitos de conteúdo ocorrem quando o conteúdo entregue ao usuário após processamento pelo sistema não atende as especificações do sistema e defeitos de tempo aparecem quando o tempo de processamento ou a duração da informação entregue são diferentes da especificada.

A ocorrência simultânea dos dois tipos de defeito causa defeitos de parada (as atividades do sistema não são mais perceptíveis para o usuário) e os defeitos erráticos (quando o sistema não para mas o serviço é entregue de forma errática).

A Figura 2.4 esquematiza os defeitos de domínio.

Detectabilidade do defeito Quando ocorre uma perda de funções para usuários temos um contexto de modos reduzidos de serviço, podendo varias de pequenas perdas de funcionalidade a até serviços de emergencia. Quando essas perdas são encontradas e sinalizadas dizemos que ocorreu um defeito sinalizado, ou, no caso contrário, não sinalizado.

Consistencia dos defeitos Determina a percepção dos defeitos por diferentes usuários, podendo ser divididas para dois ou mais usuários em defeitos consistentes, percebidos da mesma maneira por todos os usuários, e inconsistentes, o contrário do anterior.

Consequências do defeito Classificados de acordo com sua gravidade. No geral podemos definir dois níveis limitantes:

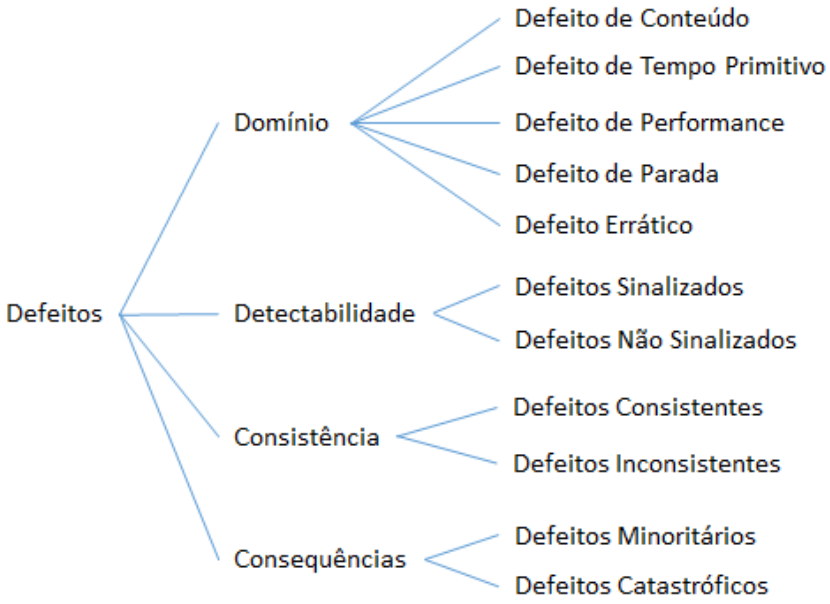

Figura 2.5: Classificação dos defeitos. Tradução de Avizienis. [4]

- Falhas minoritárias: as consequências danosas são de custo similar ao benefício pela entrega do serviço correto.
- Falhas catastróficas : o custo das consequências danosas é muito maior do que o benefício gerado pela entrega do serviço

Outros Defeitos A Figura 2.5 apresenta um resumo dos modos de defeito. Existem outros defeitos quando falamos de dependabilidade. Tais defeitosnão são relacionados ao sistema mas ao processo de desenvolvimento (defeitos no orçamento, no cronograma, na especificação e arquitetura) e a utilização do sistema quando o sistema falha mais frequentemente do que é aceitável pelos usuários).

2.3 Meios

Os meios para se alcançar a dependabilidade são divididos em quatro categorias, sendo elas:

1. Prevenção de Falhas *(Fault Prevention)*: ocorre durante o desenvolvimento. Para ser alcançado devem ser utilizadas técnicas que evitem a ocorrência e a introdução de falhas no sistema.
2. Tolerância a Falhas *(Fault Tolerance)*: Capacidade do sistema de desviar de falhas, apresentando um comportamento correto mesmo quando elas existem. Para isso

é necessário o uso de técnicas de monitoramento de erros (para identificar quando falhas se tornam ativas) para trataá-las e impedir que voltem a ser ativas novamente.

3. Remoção de Falhas *(Fault Removal)*: pode ser feita em qualquer momento do ciclo de vida do software, podendo focar na remoção preventiva ou ma corretiva.

4. Previsão de Falhas *(Fault Forecasting)*: é uma previsão baseada no comportamento do sistema. São dois os principais meios de previsão de falhas: modelagem do sistema e os teste de avaliação da modelagem.

Capítulo 3

Métodos Ágeis

Em fevereiro de 2001, dezessete desenvolvedores se reuniram na cidade de Snowbird, Utah, e escreveram o Manifesto para Desenvolvimento Ágil de Software [16], passando a chamar os métodos leves de "métodos ágeis". Na Tabela 3.1 apresentamos brevemente alguns dos principais métodos ágeis: o XP, o Scrum e o Desenvolvimento Lean.

3.1 Manifesto para Desenvolvimento Ágil de Software

O Manifesto apresentado por Beck et al. [14] traz uma abordagem mais dinâmica para o processo de desenvolvimento de software, colocando iterações, adaptabilidade e colaboração com o cliente como fatores mais importantes do que documentação abrangente, seguir o planejado e negociação de contratos, conforme apresentado na Figura 3.1.

Os métodos ágeis têm como princípios, segundo Beck et al. [14]:

> 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 cara a cara.

Tabela 3.1: Métodos Ágeis. Traduzido de Dybå e Dingsøyr [13].

Método Ágil	Descrição
Programação Extrema (XP)	Focado nas melhores práticas de desenvolvimento. Consiste em 12 práticas: jogo do planejamento, entregas frequentes, metáfora, projeto simples, testes, refatoração, programação em pares, propriedade coletiva, integração contínua, semanas de 40 horas, cliente presente, padrões de codificação. A revisão "XP2"consiste nas seguintes "práticas primárias": trabalhar junto, time coeso, ambiente de trabalho informativo, trabalho energizado, programação em pares, histórias, ciclo semanal, ciclo trimestral, afrouxamento, 10-minute build, integração contínua, testar primeiro, projeto incremental.
Scrum	Focado no gerenciamento de projetos em situações em que é difícil planejar à frente, com mecanismos para "controle de processo empírico"; onde ciclos de feedback consistem no elemento principal. Software é desenvolvido por um time auto-organizado em incrementos (chamados de "*sprints*"), iniciando com o planejamento e finalizando com a revisão. Funcionalidades que serão implementadas no sistema são registradas em um *backlog*. O *Product Owner* decide quais itens do *backlog* devem ser desenvolvidos no próximo *sprint*. Membros da equipe coordenam seu trabalho em uma reunião em pé diária. Um membro da equipe, o *Scrum Master*, é encarregado de resolver problemas que impedem a equipe de trabalhar efetivamente.
Desenvolvimento Lean	Uma adaptação dos princípios da produção enxuta e, em particular, do sistema de produção Toyota, para desenvolvimento de software. Consiste em sete princípios: eliminar desperdício, ampliar o aprendizado, decidir o mais tarde possível, entregar o mais rápido possível, empoderar o time, construir integridade e enxergar o todo.

Figura 3.1: Manifesto para Desenvolvimento Ágil de Software [14]

> 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.

Fatores humanos

O manifesto prega algumas características chave que pessoas envolvidas no processo (tanto clientes quanto desenvolvedores, gerentes, entre outros) devem apresentar para conseguirem seguir os princípios das metodologias ágeis. Antes de elencar essas características é importante frisar o seguinte:

> **O processo de desenvolvimento ágil se molda as necessidades das pessoas e da equipe, e não ao contrário**.

esse fato nos diz que os métodos ágeis põe as pessoas a frente de quaisquer outras questões envolvidas no processo de desevolvimento. Por isso as características chave são:

- Competência
- Foco em comum
- Colaboração
- Habilidade de tomar decisões
- Capacidade de resolver diversos problemas
- Respeito e confiança na equipe
- Auto-organização

A seguir serão apresentadas as práticas dos métodos ágeis do ponto de vista do processo.

3.1.1 Práticas dos Métodos Ágeis

As práticas descritas por Beck em seu livro *Extreme Programming Explained: Embrace Change*, Beck [6] são comuns a metodologias ágeis que são utilizadas no desenvolvimento de software. Ele explica como são aplicadas e sua importância.

Cliente Presente (*On-Site Customer*) - O cliente é considerado parte da equipe de desenvolvimento. Deve está presente sempre que requisitado para esclarecer dúvidas, já que é o usuário final.

Integração Contínua (*Continuous integration*) - Todo novo incremento do deve ser integrado imediatamente ao sistema, tendo que passar por todos os teste e não podendo fazer com que o sistema pare de funcionar, garantindo a integridade do sistema.

Projeto Simples (*Simple Design*) - Os novos incrementos devem representar somente as funcionalidades planejadas para e nada a mais. É importante manter o projeto do sistema simples para garantir a manutenabilidade, evitando retrabalho inútil.

Desenvolvimento Orientado a Testes (*Test Driven Development* - TDD) - Os testes são desenvolvidos antes dos incrementos, minimizando o esforço gasto com funcionalidade que não são foco no momento. Nem todas as metodologias ágeis utilizam essa prática, entretanto na maioria dos casos realizam teste de seus incrementos antes de o intregar ao sistema.

Programação em Pares (*Pair Programming*) - Como o nome já propoe, o código é construido por dois programadores, situação na qual um tem como função pensar na melhor maneira de implementar o incremento e o outro pensa em uma visão mais macro, se preocupando com o contexto em que está inserido o sistema e se nos casos de teste.

Refatoração (*Refactoring*) - É um complemento à prática de **Projeto Simples**. Visa alterar o código para torná-lo mais simples sem modificar suas funcionalidades.

Padrão de Código (*Coding Standards*) - Prática que permite que várias pessoas trabalhem juntas em um mesmo código. Equipes trabalhando em um mesmo código devem estabelecer um padrão de codificação o qual permita que, conhecendo o padrão, qualquer pessoa possa entender e modificar. Padrões são importantes, mas não universais.

Testes de Aceitação (*Acceptance Tests*) - Teste de aceitação devem ser executados periodicamente pelo cliente para garantir o correto funcionamento do sistema. Dessa maneira evita-se que um requisito mal interpretado seja carregado por vários ciclos de desenvolvimento, por exemplo.

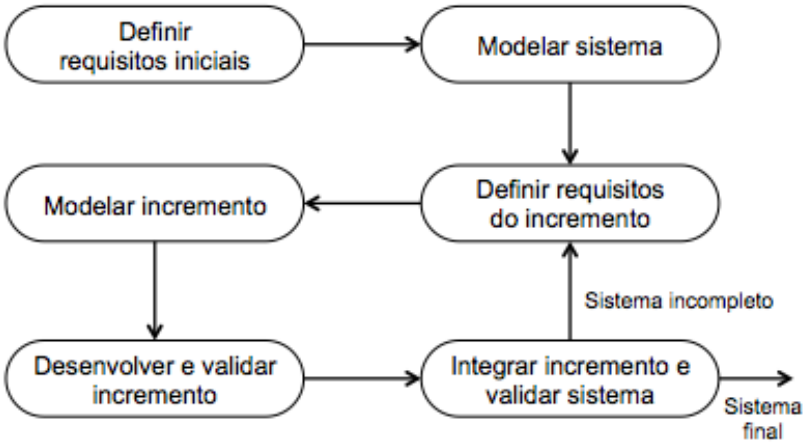

Figura 3.2: Desenvolvimento Ágil

3.1.2 Etapas do Desenvolvimento Ágil

Segundo Sommerville [42], é no processo de Desenvolvimento Incremental que as etapas de desenvolvimento das metodologias ágeis se baseiam. Os dois modelos, ágil e incremental, são bem semelhantes, porém não são a mesma coisa. O processo de Desenvolvimento Ágil de Software 3.2 é um modelo simplificado do modelo de Desenvolvimento Incremental (Figura 3.3)

No Desenvolvimento Incremental os clientes inicialmente identificam, de forma simplificada, quais os requisitos do sistema e quais são os mais e os menos importantes. Em seguida são definidas as iterações de entrega, sendo cada iteração um novo incremento que trará novas funcionalidades de acordo com a priorização dos requisitos. Os incrementos devem ser validados e integrados ao sistema, validando-se então o sistema com o novo incremento, até que todas as iterações sejam finalizadas.

No Desenvolvimento Ágil são definidos os requisitos iniciais, é feita a modelagem do sistema e a cada iteração é feita a definição, modelagem, teste e integração dos novos incrementos. As etapas do Desenvolvimento Ágil são descritas com mais detalhes a seguir:

Definir requisitos iniciais - São levantados os requisitos gerais do sistema, que serão usados para realizar a modelagem.

Modelar sistema - É feita a modelagem mais simples possível para que o sistema comece a funcionar. Desta maneira é possível manter a integridade do sistema, pois desde o inicio do desenvolvimento é possível realizar integração dos novos incrementos ao sistema.

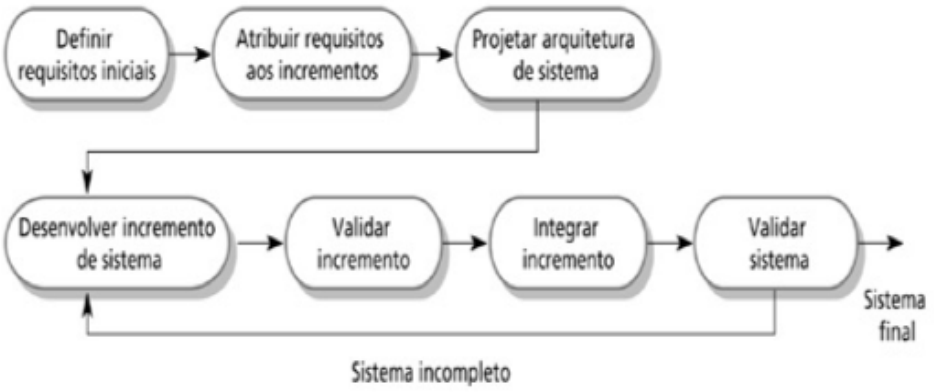

Figura 3.3: Desenvolvimento incremental [42]

Definir requisitos do incremento - Definição de todos os requisitos do incremento que será desenvolvido, quais funcionalidades deve apresentar e quando deve ser entregue.

Modelar incremento - Seguindo o princípio **projeto simples** o incremento deve ser modelado da forma mais simples possível, pensando em como implementar, integrar e testar, tanto antes quanto depois de integrar.

Desenvolver e validar incremento - As funcionalidades do incremento são desenvolvidas e testadas de acordo com a modelagem. Os testes podem ser escritos antes ou depois do código do incremento, de acordo com a metodologia utilizada.

Integrar incremento e validar sistema - O novo incremento é integrado ao sistema e então são realizados os testes de integração e aceitação. Caso o incremento passe em todos os testes o ciclo é finalizado e se inicia o próximo.

3.1.3 Limitações

Ainda não há um consenso sobre a utilização dos métodos ágeis. Tanto o XP [6] quanto o Scrum [39] são apresentados como metodologias para equipes pequenas (usualmente variando entre sete e quinze participantes). Sommerville [42] defende que as metodologias ágeis não são indicadas para sistemas muito complexos ou, principalmente, para sistemas críticos, pois estes carecem de documentação formalizada e extensa para garantir seu correto funcionamento.

Em seus estudos, Petersen e Wohlin [33] [34] apresentam várias dificuldades oriundas da implementação de metodologias ágeis em grandes empresas, como a coordenação de grandes equipes, realização e controle contínuos dos testes e dificuldades de integração

devida à falta de foco na arquitetura. No estudo publicado por Rao [37] são apresentadas também dificuldades na implantação de métodos ágeis na indústria, porém nesse mesmo estudo o autor afirma que existe a dificuldade de aplicação do método completo, mas as práticas podem ser aplicadas de acordo com o contexto das empresas para aumentar eficiência e qualidade.

Capítulo 4

Estudo de Mapeamento Sistemático

A revisão sistemática de literatura, apesar das suas diversas vantagens apresentadas [21] é um processo que necessita de muito tempo e dedicação para ser realizado. Como já iniciado no capítulo anterior, outra forma de estudo secundário bastante utilizada que vem ganhado cada vez mais destaque em áreas de engenharia de software são os mapeamentos sistemáticos. Uma pesquisa rápida com o termo *"mapping study"* na base do IEEE Xplore mostra que só nos anos de 2011 a 2015, 5 anos, foram publicados 10.812 estudos desse tipo enquanto que de 2003 a 2010, 8 anos, foram publicados 10.064.

Os estudos desse tipo propõem uma estrutura de modo a categorizar os tipos de pesquisa e os resultados publicados de estudos realizados sobre um tema específico e depois apresentá-los de forma visual para tornar mais fácil a vizualização dos resultados e possibilitar a identificação de possíveis lacunas na área estudada.

Nos próximos capítulos apresentamos a teoria para a realização de estudos de mapeamento sistemático *(systematic mapping studies)*, como o mapeamento do Estudo Base realizado por Braga et al. [7] foi feito e futuramente como o mapeamento desse trabalho será realizado. As etapas do estudo base dessa monografia foram realizada baseadas nos estudos de [32] e [21], com adaptações mostradas na Figura 4.1. Serão mostradas todos os resultados dessa monografia sendo comparados com os resultados do estudo base [7].

4.1 Necessidade do Estudo

O primeiro passo realizado no estudo base [7] foi realizar uma pesquisa para saber se trabalhos similares ou iguais já não haviam sido realizados em algum lugar no passado. Para isso foi realizada uma busca manual nas bases de artigos *IEEE Xplore Digital Library, SpringerLink e ScienceDirect* utilizando uma string de busca na qual foi relacionada as principais metodologias ágeis encontradas na literatura, estudos que são bastante extensos como artigos, livros, periódicos, monografias, teses de mestrado, doutorado, entre outras;

Tabela 4.1: *String* de Busca da pesquisa de Necessidade

"((systematic review"OR "mapping study"OR "research review"OR "research synthesis"OR "research integration"OR "systematic overview") AND (("Agile"OR "extreme programming"OR "scrum"OR "lean") AND ("Dependability"OR "Reliability")))

com conceitos de dependabilidade e confiabilidade, os quais foram os temas de estudos deles. Para tornar a pesquisa ainda mais abrangente foram utilizados sinônimos de revisão sistemática *(systematic review)* definidos por [11]. A string de busca é descrita na Tabela 4.1.

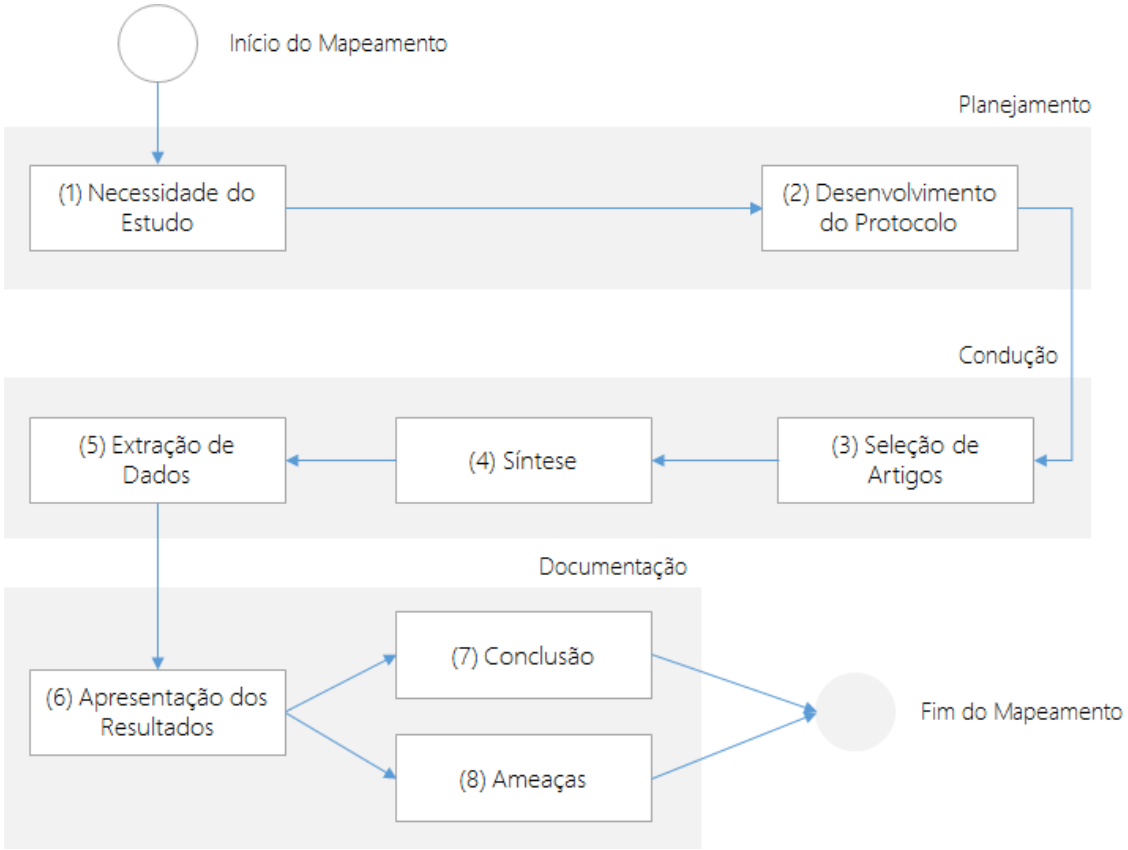

Figura 4.1: Processo para realização de Estudos de Mapeamento Sistemático. Retirada de [7]

Após a busca foram encontrados 449 trabalhos, já excluindo os que não se encontravam no domínio de Ciência da Computação ou não eram escritos em inglês ou português. Após foi feita a leitura dos títulos e abstracts, o que fez com que mais 354 artigos fossem excluidos. A ultima etapa foi a leitura da introdução e da conclusão, o que os levou a concluir que não existiam estudos de mapeamento sistemático que abordassem os dois temas simultaneamente da maneira desejada, apenas dois trabalhos tinham objetivos

Tabela 4.2: Quantidade de resultados retornados em cada etapa da pesquisa de necessidade no Estudo Base.

	IEEExplore	SpringerLink	ScienceDirect
String de busca	193	82	174
Título + Abstracts	15	6	8
Introdução + Conclusão	9	1	6

semelhantes. A Tabela 4.2 mostra a quantidade de resultados retornados em cada etapa da pesquisa.

Os dois estudos que se assemelharam foram:

- Estudo realizado por Mitchel e Seaman [28] em 2009 que é a primeira revisão existente na literatura sobre dependabilidade e métodos ágeis.
- Revisão feita por Sfetsos e Stamelos [40] em 2010, que teve foco na qualidade nos métodos ágeis , mas não especifica a mesma em conceitos de dependabilidade.

4.2 Desenvolvimento do Protocolo

O desenvolvimento do protocolo é a parte do trabalho que possibilita que o estudo seja replicado futuramente por outros pesquisadores. Utilizando-o será possível entender melhor o que foi feito no trabalho base, para então identificar as técnicas não utilizadas e o que pode ser melhorado.

4.2.1 Questões de pesquisa

O primeiro passo do processo para a realização do estudo de mapeamento sistemático é definir quais questões de pesquisa nortearam o estudo. Essas questões formam o escopo que será utilizado em todas as etapas. Para criação de tais questões é preciso ter em mente que o objetivo do estudo é propiciar uma ampla visão sobre a área, classificando tópicos contidos nela e quantificando-os em categorias específicas. Para enriquecer a análise e contribuição feita por esse trabalho, foram propostas as seguintes questões:

- *QP1.* Quais lacunas os métodos ágeis apresentam quando são aplicados ao desenvolvimento de sistemas fidedignos e quais métodos tradicionais podem ser combinados para suprir essa lacuna?
- *QP2.* Em quais áreas são mais aplicados os métodos ágeis?
- *QP3.* Quais atributos são citados como os mais importantes?

Tabela 4.3: Critérios de Inclusão e Exclusão de artigos

Critérios de Inclusão	Trabalhos (capítulos de livros e artigos) que possuam os temas dependabilidade e métodos ágeis como foco escritos até julho de 2013.Caso vários artigos se refiram ao mesmo estudo, o trabalho mais completo será considerado
Critérios de Exclusão	Keynotes, White Paper, trabalhos apresentados apenas na forma de abstract ou apresentação, trabalhos em outras línguas que não inglês e português , trabalhos cujo domínio não seja o da ciência da computação, trabalhos em que o tema principal não é dependabilidade e métodos ágeis e aqueles não disponíveis ao nosso acesso através do login institucional propiciado pela rede da Universidade de Brasília.Como o foco do estudo era metodologias ágeis como um todo, os estudos que focavam

- *QP4.* Quais são os pesquisadores e centros de pesquisa de maior relevância quando falamos sobre dependabilidade e métodos ágeis?

Além da resposta das questões 1 e 2 propostas no estudo base [7]:

- *QP1.EB* Em quais estágios do ciclo de desenvolvimento ágil de um software as falhas e defeitos estão sendo tratados?
- *QP2.EB* Quais são as principais práticas ágeis que garantem a entrega de um produto confiável?

4.2.2 Critérios de Inclusão e Exclusão

O critérios de inclusão e exclusão são necessários para que na etapa de seleção de artigos seja possível uma seleção de maneira sistemática, justificável e reaplicável. Os critérios escolhidos no trabalho estão elencados na Tabela 4.3.

A partir dos critérios foi elaborada uma lista para marcar o trabalho como válido ou não:

- O trabalho foi escrito em inglês ou português? (Sim/Não)
- O trabalho está disponível na integra? (Sim/Não)
- O contexto do trabalho é referente a métodos ágeis? (Sim/Não)
- O tema dependabilidade é abordado no trabalho? (Sim/Não)

Foram elaboradas também 4 questões que deveriam ser respondidas posistivamente para o artigo ser considerado válido para a pesquisa:

1. Os objetivos da pesquisa estão claros?
2. A pesquisa foi realizada em um contexto apropriado para a observação dos resultados?
3. Responde em qual etapa do processo de desenvolvimento de software ocorrem falhas e defeitos?
4. Apresenta o impacto das práticas ágeis na ocorrência de falhas e defeitos?

4.2.3 Bancos de Artigos

Para a realização da pesquisa eletrônica foram utilizada quatro bases de dados,selecionadas levando em conta a permissão de acesso e relevância da base nos campos da ciência da computação e engenharia de software.

- IEEE Explore
- ACM Digital
- Springer Link
- Science Direct

4.2.4 String de busca

A busca foi feita de forma automatizada utilizando uma *String* de pesquisa nos bancos de dados citados anteriormente. A *String* foi elaborada levando em conta o tema do estudo e as questões proposta a serem respondidas, traduzindo os termos para a língua inglesa. A string resultante foi montada da seguinte forma:

1. (dependable OR dependability OR reliability)
2. ("agile methods"OR "agile methodology"OR agile method"OR "agile methodologies"OR scrum OR xp OR "extreme programming"OR "agile development"OR "lean development"Or "agile principles")
3. (correctness OR failure OR fault)

$$\textbf{(1) AND (2) AND (3)} \tag{4.1}$$

Tabela 4.4: *String*Final de Busca da Pesquisa Eletrônica

"((dependable OR dependability OR reliability) AND ("agile methods"OR "agile methodology"OR agile method"OR "agile methodologies"OR scrum OR xp OR "extreme programming"OR "agile development"OR "lean development"Or "agile principles") AND (correctness OR failure OR fault)

Não foram incluidos termos como *reliable* ou *values* pois retornavam muitos resultados totalmente fora do contexto do estudo. A *String* final utilizada para busca está descrita na Tabela 4.4

Capítulo 5

Reprodução do estudo

Neste Capítulo serão mostradas as etapas realizadas na reprodução e atualização do estudo base com seus respectivos resultados, comparando-os com os resultados obtidos no estudo base. Ao final são mostradas as ferramentas que foram utilizadas para a classificação dos trabalhos.

5.1 A Reprodução do Estudo

A reprodução do estudo foi iniciada em maio de 2015. Todos os passos do estudo base foram seguidos na íntegra exceto a pesquisa de necessidade, o qual foi realizado somente a pesquisa automatizada utilizando a mesma *string* de busca. Para esse estudo, foi alterando o intervalo pesquisado para entre 2013 até maio de 2015. A seguir serão descritos os passos e quais foram os resultados obtidos no estudo base e quais foram os encontrados nesta monografia.

5.2 Resultado da Busca Automatizada

A busca automatizada foi feita utilizando a mesma *String* descrita na Tabela 4.4. A partir da lista de todos os artigos resultantes da busca foi feita a leitura dos títulos e dos resumos e, caso o título possuísse algum dos termos relacionados com o trabalho a leitura do resumo era feita. Após a leitura e avaliação de todos os títulos foi feita a leitura dos resumos dos trabalhos que tiveram os títulos selecionados. Caso o resumo lido apresentasse contexto dentro do tema proposto, a leitura da introdução e conclusão era feita. Ao final da etapa anterior foram selecionados os artigos para leitura completa. Os resultados obtidos no estudo base estão apresentados na Tabela 5.1 e os obtidos nessa monografia estão apresentados na Tabela 5.2

Os dados apresentados seguem os mesmo critérios de inclusão e exclusão.

Tabela 5.1: Quantidade de resultados retornados em cada etapa da pesquisa automatizada do estudo base.

	IEEExplore	SpringerLink	ScienceDirect	ACM DL	TOTAL
String de busca	721	404	392	278	1795
Título + Abstracts	104	69	75	59	307
Introdução + Conclusão	10	3	2	1	18
Leitura Completa	5	1	2	1	9

Tabela 5.2: Quantidade de resultados retornados em cada etapa da pesquisa automatizada dessa Monografia.

	IEEExplore	SpringerLink	ScienceDirect	ACM DL	TOTAL
String de busca	446	658	302	275	1681
Título + Abstracts	10	0	1	2	13
Introdução + Conclusão	10	0	0	1	11
Leitura Completa	7	0	0	0	0

5.3 Seleção de Artigos

5.3.1 Resultado da Pesquisa Manual

Foi realizada ainda pesquisa manual nas principais conferências, workshops e periódicos dos temas de confiabilidade (*reliability*) e metodologias ágeis, além de pesquisa no *website* DBPL (`http://dblp.uni-trier.de/`) com os nomes dos autores citados como os principais estudiosos na área no trabalho base. Foram eles:

- Claes Wohlin
- Kai Petersen
- Kirsi Korhonen
- Laurie Williams
- Letha H. Etzkorn
- Lucas LaymanNorth

Foram encontrados, analisando somente o título, 4 artigos, sendo 3 deles não estavam disponíveis para acesso utilizando o login institucional da Universidade de Brasília e 1 foi descartado após a análise da introdução. Foi aplicada tambem a técnica de *snowballing*.

Na busca manual foram utilizados os termos *agile, lean, extreme programming* e *scrum* nos eventos relativos a dependabilidade e *reliability* e *dependability* nos eventos sobre métodos ágeis.

As conferências, workshops e periódicos utilizados para busca foram:

1. Agile Development Conference - 2 resultados
2. IEEE International Conference on Software Reliability Engineering Workshops (ISSRE Wksp) - 4 resultados
3. International Conference on Availability, Reliability and Security (ARES) - 8 resultados
4. International Symposium on Software Reliability Engineering (ISSRE) - 2 resultados
5. Annual Symposium Reliability and Maintainability (RAMS) - 3 resultados
6. International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE) - 3 resultados
7. International Conference on Reliability, Maintainability and Safety (ICRMS) - 1 resultado
8. International Conference on Dependable Systems and Networks (DSN) - 1 resultado
9. International Conference on Dependable Systems and Networks Workshops (DSN-W) - 1 resultado
10. International Conference on Software Engineering (ICSE) - 14 resultados
11. IEEE/ACM International Conference on Automated Software Engineering (ASE) - 1 resultado
12. International Symposium on Software Testing and Analysis (ISSTA) - 3 resultados
13. IEEE Transactions on Reliability - 1 resultado
14. IEEE Transactions on Software Engineering - 7 resultados
15. Outros eventos e workshops sobre dependabilidade que não retornaram resultados nesta etapa: IEEE International Conference on Quality and Reliability; International Conference on Secure System Integration and Reliability Improvement (SSIRI); European Dependable Computing Conference (EDCC); Latin-American Symposium on Dependable Computing (LADC); Pacific Rim International Symposium on Dependable Computing;IEEE International Symposium on Dependable, Autonomic and Secure Computing (DASC); International Conference on Secure System Integration and Reliability Improvement (SSIRI,SERE).
16. Conferências que abordam temas em geral sobre engenharia de software, que são reconhecidas e não retornaram resultados nesta etapa: FSE (Foundations of Software Engineering); SPLASH (Systems, Programming, Languages and Applications:

Software for Humanity); ECOOP (European Conference on Object-Oriented Programming) e FASE (Fundamental Approaches to Software Engineering).

17. Periódicos selecionados para nossa pesquisa e que não retornaram resultados nesta etapa: ACM Transactions on Software Engineering, Methodology; IEEE Software; Wiley Software Testing, Verification and Reliability; Springer Empirical Software Engineering; Springer Software and Systems Modelin;, Wiley Journal of Software Maintenance and Evolution; Journal of Systems and Software, Software Practice and Experience, Information and Software Technology.

5.3.2 Resultado do *Snowballing*

Segundo Jalai e Wohlin [18] o *snowballing* é a técnica utilizada para encontrar trabalhos relacionados ao estudo nas referências dos estudos retornados pela busca automatizada. Ele tende a retornar uma menor quantidade de "ruído". Ele pode ser aplicado na forma de:

1. *Forward Snowballing* busca trabalhos que utilizam como referências nossa lista inicial de estudos.
2. *Backward Snowballing* busca novos trabalhos nas referências da nossa lista.

O estudo base optou pela utilização do backward snowballing. O forward snowballing não faria muito sentido no contexto dessa monografia pois o intervalo de tempo analisado é curto, sendo a chance de serem identificados estudos que utilizam algum trabalho selecionado na busca eletrônica como referência é pequena. Os resultados obtidos no *snowballing* do estudo base e dessa monografia estão apresentados na Tabela 5.3. Muitos estudos foram descartados nesta fase durante a reprodução do estudo pois tinham data de publicação anterior ao intervalo de datas utilizado neste trabalho.

Tabela 5.3: Quantidade de resultados retornados no *Snowballing* do estudo base e dessa monografia

	Estudo Base	Esse estudo
Pesquisa Inicial	616	14
Título + Abstracts	64	6
Introdução + Conclusão	8	4
Leitura completa	*	0

*Não há no estudo de Braga et al. [7] dados sobre quantos trabalhos selecionados no *snowballing* foram selecionados após a leitura completa.

Não foram apresentados no trabalho base a quantidade de artigos que foram selecionados na etapa da leitura completa provenientes do *snowballing*. A quantidade de artigos encontrados durante esta fase foi consideravelmente menor do que no estudo base pois o intervalo de datas utilizado foi restringido para 2 anos e meio.

A Figura 5.1 mostra as etapas e quantidades de trabalhos retornados em todo o processo de busca deste estudo.

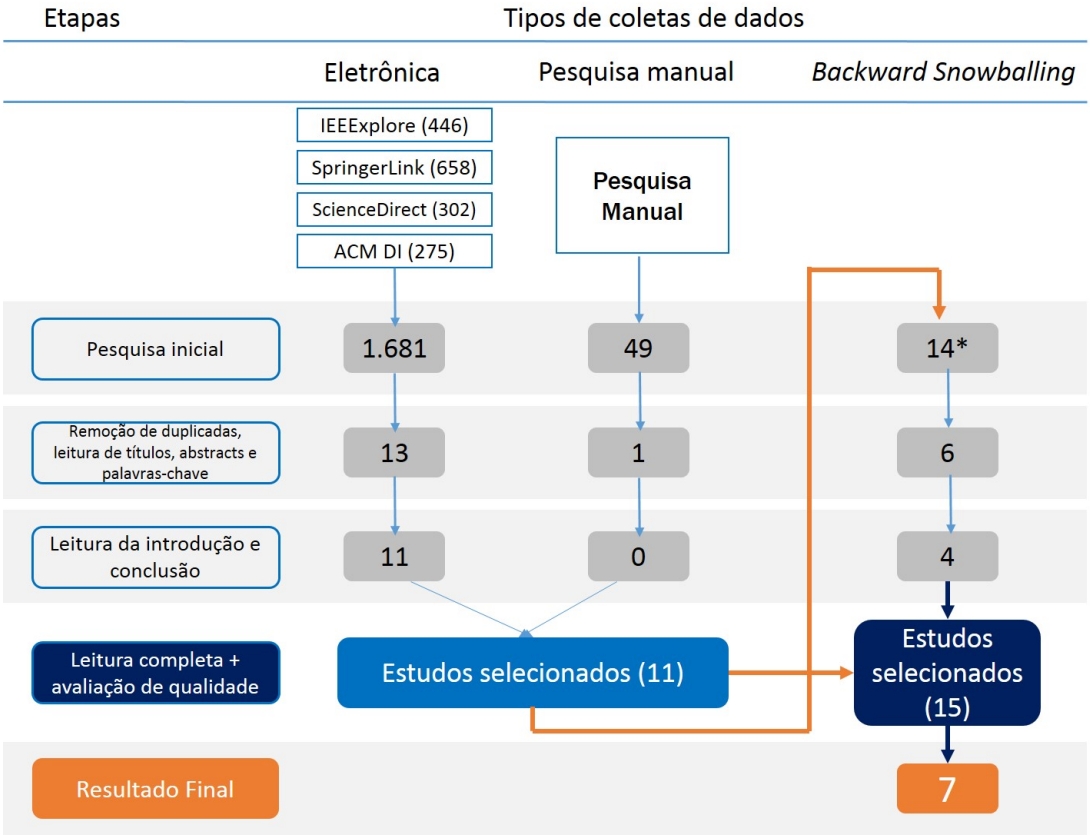

Figura 5.1: Etapas do processo de seleção e extração de dados. Adaptação de Braga e Leal [7]

*Foram contabilizados na etapa da pesquisa inicial do *backward snowballing* apenas os artigos que estavam dentro do intervalo de anos, entre 2013 e 2015, e que possuiam título relacionado com o contexto dessa monografia.

5.4 Classificação

Para realizar a classificação dos artigos foi utilizada a proposta de Petersen et al. [32], que a estrutura em facetas, das quais foram escolhidas 3: faceta de pesquisa, faceta de contribuição e faceta de contexto.

Tabela 5.4: Faceta de Pesquisa. Traduzido de Wieringa [44]

Tipo	Descrição
Pesquisa de Validação	As técnicas investigadas são novas e ainda não foram implementadas. As técnicas usadas são, por exemplo, experimentos realizados em laboratório.
Pesquisa de Avaliação	Técnicas são implementadas na prática e uma avaliação da técnica é conduzida. Ou seja, é mostrado como a técnica é implementada na prática (solução de implementação) e quais são as consequências da implementação em termos de benefícios e perdas (avaliação da implementação). Isso também inclui identificar problemas na indústria.
Proposta de Solução	A solução para o problema é proposta, a solução pode ser nova ou uma extensão significativa de uma técnica existente. Os potenciais benefícios e a aplicabilidade da solução são mostrados por pequenos exemplos ou uma boa linha de argumentação.
Artigos Filosóficos	Esses artigos esboçam uma nova forma de visualizar coisas existentes pela estruturação do campo de pesquisa em termos de taxonomia ou framework conceitual.
Artigos de Opinião	Esses artigos expressam uma opinião pessoal de alguém sobre alguma técnica afirmando se ela é boa ou ruim, ou como as coisas deveriam ser feitas. Eles não dependem de trabalhos relacionados ou metodologias de pesquisa.
Artigos de Experiência	Artigos de experiência explicam o que e como algo vem sendo feito na prática. Deve ser uma experiência pessoal do autor.

1. **Faceta de pesquisa:** foi utilizada a definição dada por Wieringa [44], descrita na Tabela 5.4

2. **Faceta de contexto:** diz respeito ao tema estudado (dependabilidade), logo foram postos aqui estudos que tratavam sobre defeitos e falhas como explicado anteriormente.

3. **Faceta de contribuição:** foram consideradas as melhores práticas presentes nos métodos ágeis, como *cliente on-site, refactoring, test driven development.*

5.5 Ferramentas Utilizadas

Para análise e classificação dos estudos retornados foram utilizadas 2 planilhas. Na primeira foram listados todos os estudos retornados na busca automatizada das 4 bases utilizadas, todos os estudos retornados na busca manual e todos os estudos os quais o título foi selecionado na fase do snowballing.

Foi utilizada a seguinte legenda para classificação dos trabalhos:

- **N**: Resultado rejeitado pelo título
- **Ns**: Resultado aceito pelo título e rejeitado pelo *abstract*
- **S**: Resultado aceito
- **0**: Nenhum resultado retornado, utilizada no caso da aba de busca manual
- **SS**: Resultado já retornado pela busca automatizada e já aceito, no caso do snowballing
- **NN**: Resultado já retornado pela busca automatizada e já recusado, no caso do snowballing

Além da classificação utilizando a legenda acima, foram justificadas todas as classificações aceitando ou rejeitando trabalhos. A segunda planilha segue o mesmo modelo da utilizada no trabalho base para avaliar os artigos que foram selecionados para leitura completa.

Capítulo 6

Resultados Obtidos e Análise

6.1 Síntese dos Resultados

6.1.1 Dados Gerais

Após o término da reprodução do estudo foi constatado o aumento de 50% na quantidade de publicações por ano sobre o tema, como mostra a Figura 6.1. Mesmo aplicando o filtro dos anos, foram retornados os estudos de Webster et al. [43] e Wolff [45], publicados em 2012. Devido a relevância das informação apresentadas em cada um deles foi decidido mantê-los, pois o maior intuito do trabalho é atualização da lista de estudos relevantes na área.

Na Figura 6.1 as barras azuis representam o resultado do estudo Base [7] e as barras laranjas o resultado dessa monografia.

Foi observado tambem que a metodologia mais frequente nos estudos foi o Scrum, sendo utilizado 40% das vezes sozinho, e 20% em comparação com o XP. Foram encontrados tambem estudos que utilizavam Lean, XP e que não falavam sobre nenhuma metodologia específica, tratando apenas como "métodos ágeis". Em alguns estudos foram tiradas conclusão bastante interessante sobre a combinação de algum método ágil com práticas tradicionais de desenvolvimento de software. Essa análise será feita na seção seguinte. A seguir serão feitas as mesmas análises realizadas no estudo base para possibilitar a comparação entre os resultados.

6.1.2 Metodologia de pesquisa e Validade

Na Tabela 6.1 é apresentada a distribuição do tipo de método utilizado nos estudos selecionados nessa monografia e na Tabela 6.2 é apresentada a distribuição dos métodos no estudo base [7]. Durante a atualização foram identificados estudos sobre metodologias que não estavam presentes nos trabalhos do estudo base, o que pode ter ocorrido pois a

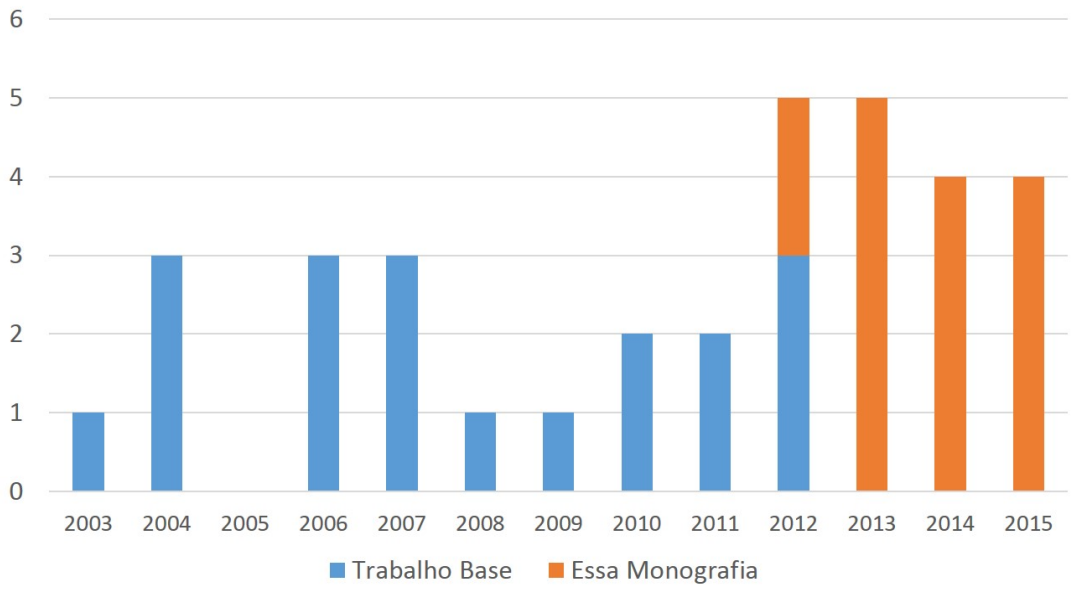

Figura 6.1: Número de resultados por ano de publicação.

Tabela 6.1: Tipo de Metodologia Ágil Utilizada - Monografia

Metodologia	Quantidade	Porcentagem
Scrum	6	40%
XP	2	13%
Lean	1	7%
Scrum e XP	3	20%
Geral*	3	20%
Total	15	100%

*"Geral"se refere a estudos que citam um método ágil sem especificar qual

Tabela 6.2: Tipo de Metodologia Ágil Utilizada - Estudo Base

Metodologia	Quantidade	Porcentagem
Geral*	8	44%
Scrum	5	28%
XP	5	28%
Total	18	100%

*"Geral"se refere a estudos que citam um método ágil sem especificar qual

confiança na utilização dos métodos ágeis vem aumentando nos últimos anos [19], logo é natural que surja o interesse e pesquisa sobre metodologias menos disseminadas.

Em relação a distribuição dos resultados por veículo de publicação, foram 11 trabalhos publicados em conferências e 3 em periódicos como mostra a Tabela 6.3. Foram selecionados 3 estudos em conferências e 2 em periódico nos quais o estudo base tambem havia selecionado trabalhos, porém em outros anos. Foram eles *ICSE*, *Jounal of Systems*

and Software e *IEEE Software*. A Tabela 6.4 traz os resultados selecionados no estudo base [7].

Tabela 6.3: Periódicos e Conferências em que os resultados foram publicados - Monografia

Veículo	Tipo	Quantidade
ISSRE	Conferência	1
ITNG	Conferência	2
CompSysTech'15	Conferência	1
SAC'13	Periódico	1
e-Challenges	Conferência	1
ICCUBEA	Conferência	1
AeroConf	Conferência	1
CSMR-WCRE	Conferência	1
SMC-IT	Conferência	1
RCoSE - ICSE	Conferência	1
ICSE	Conferência	1
FormSERA - ICSE	Conferência	1
IEEE software	Periódico	1
Journal of Systems and Software	Periódico	1
Total		15

Tabela 6.4: Periódicos e Conferências em que os resultados foram publicados - Estudo Base

Veículo	Tipo	Quantidade
ESEM Conferência	Conferência	2
ICSE	Conferência	1
FIT	Conferência	1
ICCSA	Conferência	1
Journal of Systems and Software	Periódico	1
Software Quality Journal	Periódico	1
Empirical Software Engeneering	Periódico	1
Software Quality Professional	Periódico	1
ADC	Conferência	1
IEEE Software	Periódico	1
IEEE Transactions on Software Engineering	Periódico	1
Journal of Software Maintenance and Evolution	Periódico	1
Agile Conference	Conferência	1
Euromicro Conference	Conferência	1
COMPSAC	Conferência	1
ICSEA	Conferência	1
CCECE	Conferência	1
Total		18

Na classificação feita de acordo com as facetas de pesquisa mostradas no capítulo 5 foram identificados estudos em 6 facetas, contra 4 do estudos anterior. As Pesquisas

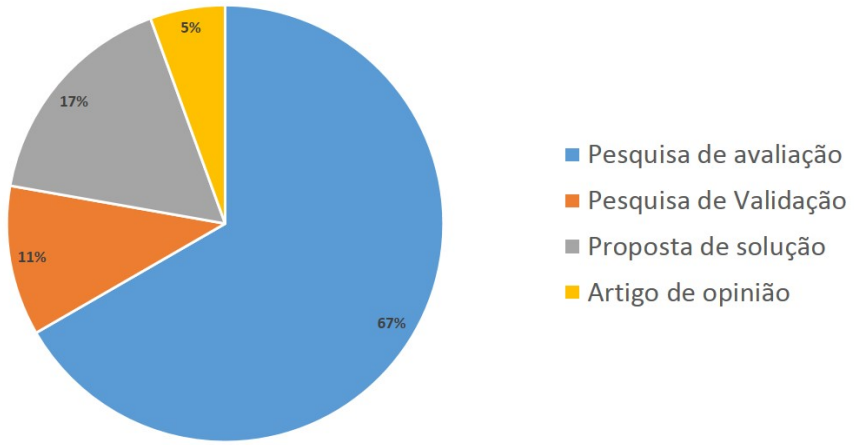

Figura 6.2: Classificação de Acordo com a Faceta de Pesquisa - Estudo Base

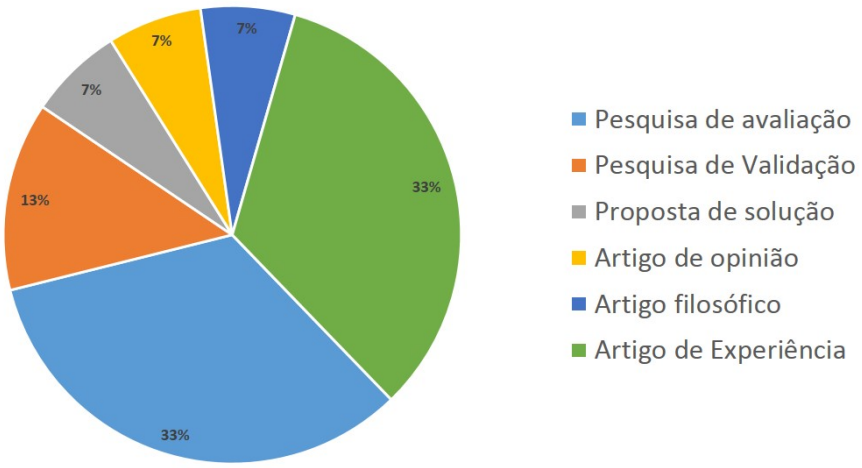

Figura 6.3: Classificação de Acordo com a Faceta de Pesquisa - Monografia

de Avaliação continuaram sendo frequentes como no estudo base, porém os Artigos de Experiência ficaram empatados, cada um apresentando 5 estudos, 33% do total. Esse aumento na quantidade de artigos de experiência encontrados pode estar refletindo o aumento da adesão aos métodos ágeis. A segunda faceta em número de resultados foram as Pesquisas de Validação, 2 estudos, 13% do total, enquanto no estudo base a segunda faceta havia sido as Propostas de solução. Aqui podemos interpretar o aumento na quantidade de Pesquisas de Validação como uma evolução natural da Propostas de Solução, pois a segunda apresenta uma proposta baseada em uma boa linha de argumentação, e a segunda seria a aplicação das soluções propostas. Em seguida temos Propostas de solução, Artigos de Opinião e Artigos Filosóficos empatados com 1 estudos cada um, 7% do total cada. No estudo Base tivemos 2 Pesquisas de Validação, 11% do total, e 1 artigo de opinião, 5% do total. A distribuição das facetas de pesquisa nessa monografia e no estudo base são mostradas nas Figuras 6.3 e 6.2.

A distribuição de acordo com os métodos de pesquisa utilizados são apresentados nas Tabelas 6.5 e 6.6. Dentre os estudos selecionados neste trabalho foi encontrada maior variedade de métodos de pesquisa, o que pode ser uma consequência da maior aderência aos métodos ágeis.

Tabela 6.5: Tipo de Método de Pesquisa Utilizado - Monografia

Metodologia	Quantidade	Porcentagem
Experimento	3	20%
Estudo de Caso	6	40%
Pesquisa	3	20%
Múltiplos Estudos de Caso	2	13%
Sem Pesquisa	1	7%
Total	15	100%

Tabela 6.6: Tipo de Método de Pesquisa Utilizado - Estudo Base

Metodologia	Quantidade	Porcentagem
Experimento	1	5,5%
Estudo de Caso	11	61,1%
Múltiplos Estudos de Caso	3	16,6%
Sem Pesquisa	3	16,6%
Total	18	100%

6.1.3 Faltas, erros e defeitos

Nesta análise foi identificada a mesma dificuldade que os autores do estudo base para separar as definições de falta, erro e defeitos. A maioria dos estudos analisados utiliza o

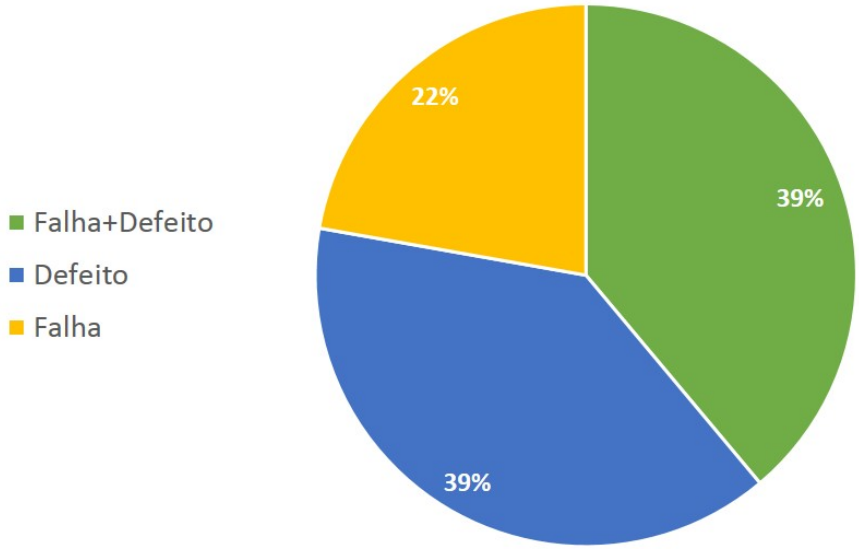

Figura 6.4: Distribuição dos termos Falha, erro e defeito - Estudo Base

termo *failure* tanto para defeitos quanto para erros, além do uso de outros termos como *issue* e *defects*. O termo Falta tem um uso mais específico e na maioria das vezes em que aparece segue a definição de Avizienis [4].

As distribuições apresentadas pelos estudos selecionados neste trabalho e no estudo base são apresentadas nas Figuras 6.5 e 6.4. A maioria dos estudos aborda defeitos ou fala de maneira geral dos problemas que podem ocorrer durante o processo de produção do software, não especificando qual o tipo de atributo está sendo tratado. Dos estudos selecionados nessa monografia, 27%, 4 dos 15 estudos, não falaram sobre nenhum dos termos, mas trata sobre dependabilidade a partir de outros atributos.

6.2 Análise dos Resultados

Para enriquecer a análise e a contribuição deste trabalho para a comunidade acadêmica, nesta seção são respondidas as questões de pesquisa propostas no Capítulo 1 além da realização de uma análise complementar as questões, feita a partir de dados encontrados nos estudos selecionados.

Após a leitura completa do 15 artigos selecionados foram rejeitados para análise final os artigos que não traziam dados de como os métodos ágeis influenciavam nos atributos de dependabilidade do software. Essa abordagem permitiu que fosse feita uma análise macro

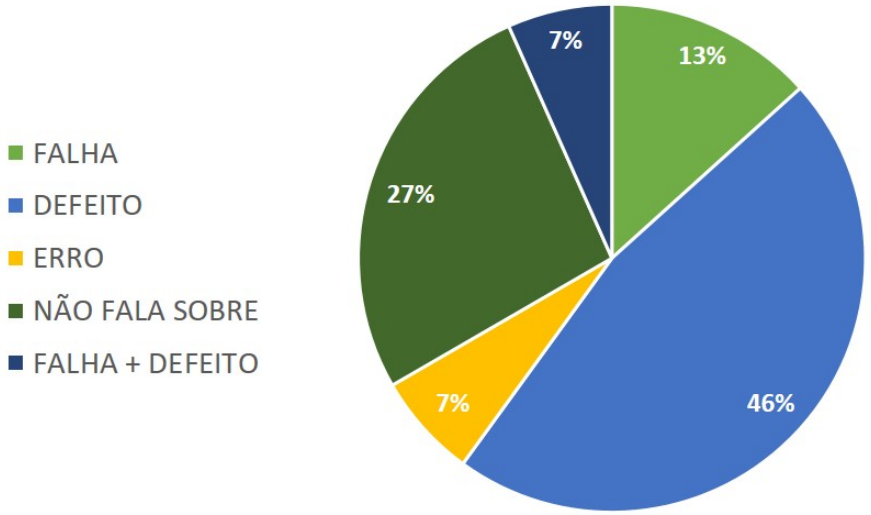

Figura 6.5: Distribuição dos termos Falha, erro e defeito - Monografia

de como os métodos influenciam na produção de software. Ao final faremos a análise baseada em 7 estudos aceitos, como detalhado na Tabela 5.2.

6.2.1 Respostas as questões de pesquisa

QP1.EB. Em quais estágios do ciclo de desenvolvimento ágil de um software as falhas e defeitos estão sendo tratados?

Para responder à essa pergunta, utilizamos as quatro etapas descritas no Capítulo 2 como foi feito no estudo base, para possibilitar a comparação dos resultados. Foram encontradas referências as estapas descritas na Tabela 6.7. Foi observado que a 71% dos artigos ([S8][S10][S12][S13][S15]) abordaram o processo de desenvolvimento como um todo, os estudos [S6][S7][S8][S12][S15]defendem a importância de uma boa definição de requisitos do incremento e do incentivo a boa convivência e comunicação da equipe e o [S6] fala sobre a aplicação de boas práticas no desenvolvimento para evitar retrabalho. O estudo de Carpenter e Dagnino [S10], afirma que de 80 a 95% dos defeitos são causados por erro humano ou problemas de comunicação, tanto entre a equipe quanto entre a equipe e o cliente. Essa afirmação engloba todos os estágios de desenvolvimento, pois os humanos estão envolvido em todos eles.

No estudo base [7] não foram encontradas menções à modelagem dos incrementos, o que o autor colocou como um fator agravante para a não utilização dos métodos ágies no desenvolvimetno de sistemas críticos. Nesta monografia encontramos menções a mo-

Tabela 6.7: Estágio do ciclo de desenvolvimento - Estudo Base *versus* Monografia

	Quantidade	
Estágio	Estudo Base	Monografia
Definir requisitos do incremento	2	7
Modelar Incremento	0	5
Desenvolver e validar incremento	10	6
Integrar e validar sistema	7	7

delagem dos incrementos em 5 dos 7 artigos selecionados, além de que todos abordarem a utilização de métodos ágeis em sistema críticos, trazendo resultados positivos, o que mostra uma mudança de percepção.

QP2.EB. Quais são as principais práticas ágeis que garantem a entrega de um produto confiável?

Após a leitura dos trabalhos selecionados, foram identificadas 4 práticas ágeis que foram mencionadas por impactar a dependabilidade do software, contra 8 encontradas no estudo base. Dos 7 estudos selecionados, 5 trouxeram dados sobre a redução do defeitos, 1 sobre redução de erro e 1 sobre a redução de falhas e defeitos. A Tabela 6.8 traz a comparação dos resultados obtidos nesta monografia com os obtidos no estudo base e as Figuras 6.6 e 6.7 trazem as distribuições das práticas de acordo com os termos de dependabilidade apresentados.

Dos resultados apresentados os estudos [S6][S7] apresenta resultados sobre refatoração, [S6][S7][S8][S12][S13[S15] sobre integração contínua e [S10] sobre TDD e programação em pares.

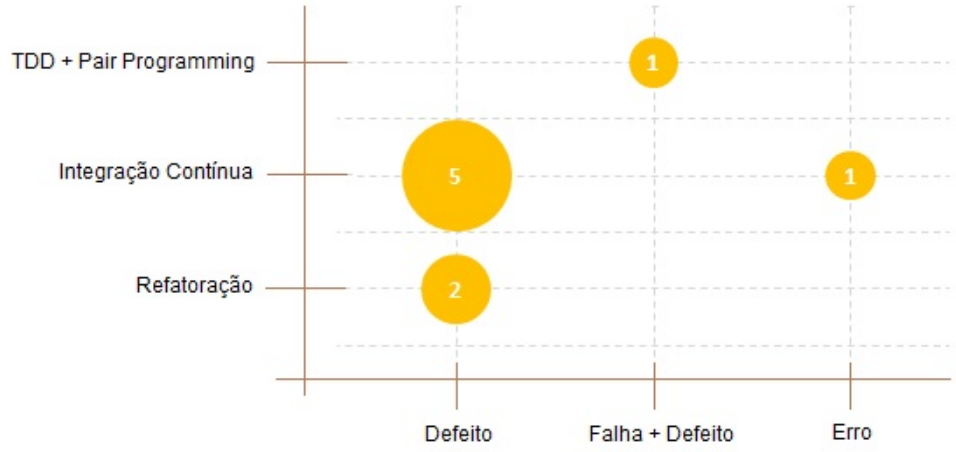

Figura 6.6: Distribuição de práticas ágeis - Monografia

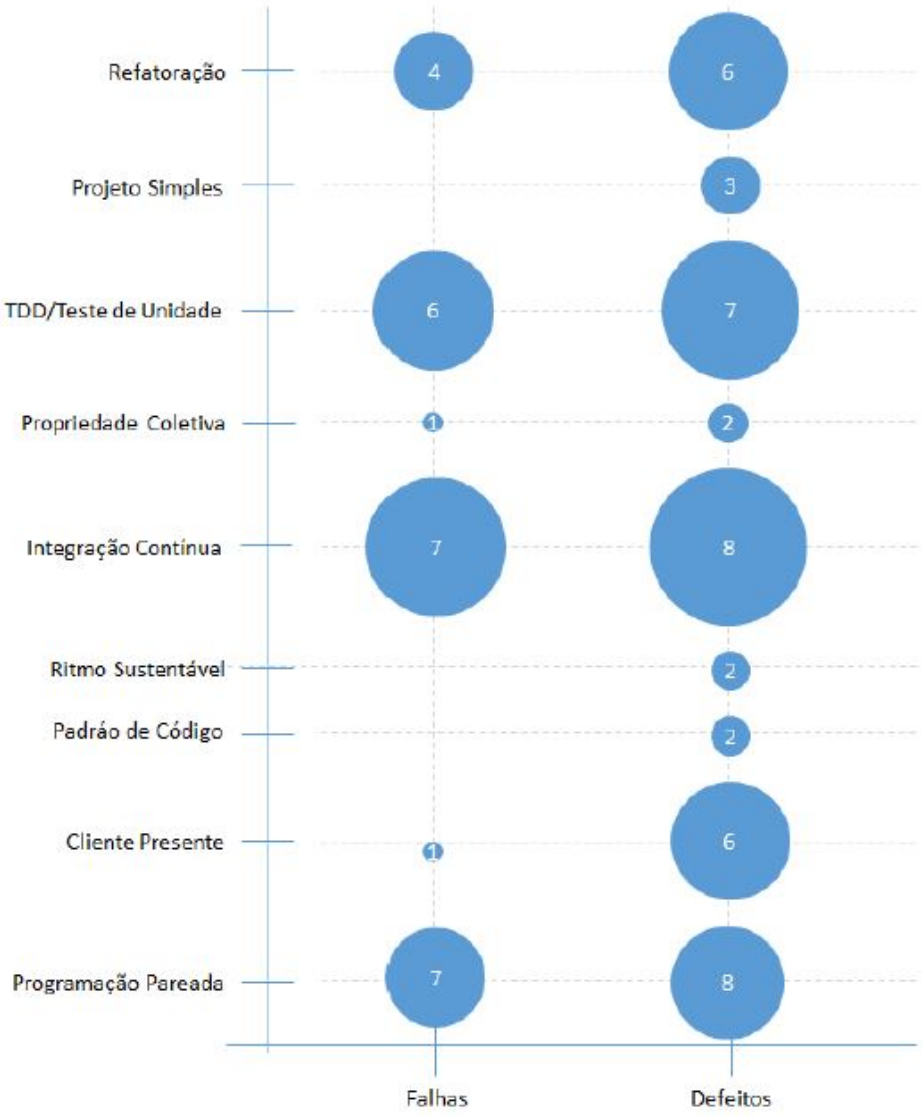

Figura 6.7: Distribuição de práticas ágeis - estudo base [7]

Tabela 6.8: Redução de falha, defeito e erro - Estudo Base *versus* Monografia

	Quantidade	
Redução	Estudo Base	Monografia
Defeitos	9	5
Falhas	5	0
Erros	0	1
Defeitos e Falhas	0	1
Não relaciona	4	0

QP1. Quais são as lacunas que os métodos ágeis apresentam quando são aplicados ao desenvolvimento de sistemas fidedignos e quais métodos tradicionais podem ser combinados para suprir essa lacuna?

De maneira geral, cada estudo identificou diferentes lacunas de acordo com o projeto que estava sendo desenvolvido, mas não foram necessariamente ligados ao software, podendo ser ligados ao processo. Isso é natural já que métodos ágeis são técnicas de desenvolvimento que dão grande importância ao aspecto humano do desenvolvimento [14]. No estudo de Koski e Mikkonen([S12]), por exemplo, é relatado que a maior dificuldade na aplicação do método ágil no projeto foi a ausência prolongada do cliente durante o processo e falta de adaptação da própria equipe às práticas ágeis. Os demais estudos também trouxeram dados sobre esses aspectos, mas não foram investigados pelo foco da questão ser outro.

Do ponto de vista de softwares críticos, os estudos trouxeram dados sobre qual seriam os maiores problemas em aplicar métodos ágeis sozinhos. Todos os estudos apontam a falta de documentação formal e da rotina de verificação e validação do software como o problema na aplicação dos métodos a sistemas críticos e relatam, para cada contexto, que a combinação do método com técnicas formais de desenvolvimento resolvem o gap. Os métodos tradicionais mencionados são o framework QUMAS ([S13]), CMMI ([S10]), Jira([S8]), API de documentação ([S8]), *Good Clinical Practice - GCP* ([S6]), *Software-in-Simulation (SiS)* ([S15]). Os demais estudos não falam de uma técnica, mas o que deve ser feito para suprir a necessidade, como adição de um *sprint* específico para a validação e verificação formal [S6], a adoção de sistemática de testes em cada *sprint* equilibrada entre teste manuais e teste automatizados ([S7]) e ter acesso direto ao cliente real, manter o ambiente colaborativo entre a equipe e realizar o maximo de iterações com o menor tamanho possível ([S12]) .

QP2. Em quais áreas são mais aplicados os métodos ágeis?

Todos os estudos tratavam de softwares comerciais, e 6 dos 7 selecionados([S6][S8][S10][S12][S13][S15]) abordaram a utilização de métodos ágeis em sistemas críticos, desde de sistemas clínicos([S6]), e sistema utilizados em missões espaciais classificados como *Space-Based Systems* ([S8][S10]) até *Mission Critical Systems* ([S12]), *safety-critical systems* utilizados em ambientes controlados ([S13]) e software controlador simulador de vôo([S15]). O fato de termos encontrado tantos estudos sobre essa categoria de software demonstrou uma tendência de pesquisa que pode aumentar nos próximos anos.

QP3. Quais atributos são citados como os mais importantes?

Dos estudos selecionados, foi identificado que a maioria aborda confiabilidade combinado com segurança, 29% ([S6][S10]), porém o atributo mais presente é segurança (*Safety*), aparecendo em 4 dos 7 estudos ([S6][S8][S10][S15]). O segundo mais frequente é confiabilidade, estando em 3 dos 7 trabalhos, tanto da maneira já citada, quanto combinado com Usabilidade e Manutenabilidade ([S6][S10][S12]). Foram encontrados tambem estudo que aborda sobre manutenabilidade ([S7]) e que não aborda sobre um atributo específico, mas sobre dependabilidade de maneira geral([S13]). A distribuição está sendo mostrada na Figura 6.8.

QP4. Quais são os pesquisadores e centros de pesquisa de maior relevância quando falamos sobre dependabilidade e métodos ágeis?

19 autores escreveram os 7 artigos selecionados, sem repetição de autores. Para escolher os principais foi feita uma pesquisa no Google Scholar e no próprio Google para identificar qual a principal linha de pesquisa do autor e quantas citações ele possue. Nesse trabalho foram identificados e 3 autores. O resumo sobre cada um deles é apresentado a seguir e a lista completa com os Autores identificados nesse estudo encontra-se no Apêndice C.

Nos estudos selecionados não identificamos novamente nenhum dos autores citados no estudo base como principal. Um estudo de Laurie Williams estava entre os 15 estudos selecionados para leitura completa, mas esse estudo foi rejeitado após a etapa citada por analisar métodos iterativos.

Christopher Webster

Pesquisador americano, membro da *Mission Control Tecnologies - MCT*, ja foi membro da Intel, *NASA Ames Research Center, Sun Microsystems* e é atual membro da *Verizon e JExamples.com*. Seu trabalho tem foco em métodos ágeis e no estudo selecionado relata experiência do uso dos métodos ágeis para melhorar a usabilidade, a confiabilidade e a segurança de software crítico ([S8]).

Scott E. Carpenter

Tem mais de 20 anos de experiência na utilização de técnicas de engenharia de software em variados campos de aplicação como aviação espacial, assunto no qual o estudo selecionado está focado ([S10]). Nesse artigo de opinião é feita uma análise sobre quais métodos ágeis são desejáveis e/ou adaptávies para o desenvolvimento de software utilizados em missões espaciais, os quais requerem elevado grau de confiabilidade, usabilidade e

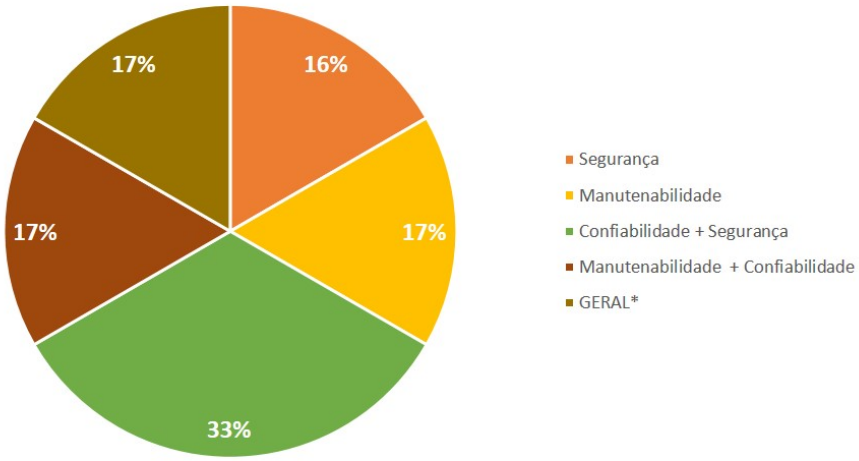

Figura 6.8: Distribuição dos atributos de dependabilidade encontrados

segurança (*safety*).

Brian Fitzgerald

É cientista chefe no *LERO - the Irish Research Center*. Antes de se dedicar efetivamente à academia trabalhou na industria de software por 12 anos em uma variedade de setores, incluindo finanças, telecomunicações e desenvolvimento de software sob medida. Seus estudos seguem 3 linhas, sendo uma delas métodos ágeis. O estudo selecionado fala sobre quais são os pontos de tensão quando aplicamos métodos ágeis a desenvolvimento em ambientes controlados ([S13]).

6.3 Conclusões da Análise

Para facilitar o entedimento a seção foi dividida entre conclusões sobre a reprodução e atualização do estudo e conclusões sobre as novas informações verificadas ao responder às questões de pesquisa desta monografia.

6.3.1 Reprodução e Atualização do Estudo

Foi verificado um aumento no número de estudos por ano que relacionam métodos ágeis com dependabilidade, resultado esperado tendo em vista o aumento na quantidade de

estudos com bons resultados sobre essa prática, o que leva ao aumento da confiança das equipes em adotar os métodos em seus projetos. O Scrum e o XP continuam sendo os métodos mais adotados nos projetos, e não foi encontrado nenhum estudo que relatasse a aplicação dos métodos em seu formato puro. Todos sofrem adaptações de acordo com o contexto em que estão sendo aplicados. Dentre as práticas ágeis apontadas como desejáveis para aumentar a dependabilidade do software foram citadas programação em pares e TDD ([S10]), refatoração([S7][S6]) e Integração contínua ([S6][S7][S8][S12][S13][s15]) .

Ainda existe uma falta de padronização na utilização dos termos falha, defeito e erro, sendo que na maioria das vezes são aplicados como sinônimos, ou em seus lugares são utilizadas palavras como problemas e *bug*.

Diferente do estudo base, todos os estudos selecionados analisam os efeitos dos métodos ágeis no processo de desenvolvimento e no software. Nos apêndices A, B e C são apresentadas as tabelas atualizadas com os resultados encontrados nesta monografia.

6.3.2 Informações Novas

Os maiores ganhos verificados na utilização dos métodos foram a redução do custo e tempo para identificação de problemas com o software, a entrega contínua de versões completas e a melhora na usabilidade, devido a aproximação do *product owner*. Entre os gargalos foi identificado de modo geral a falta de documentação formal para validação e verificação do software. Esse dado pode ter sido generalizado pois 6 dos 7 estudos selecionados são sobre aplicação de métodos ágeis em software crítico.

Foi identificada uma tendência na realização de estudos sobre a aplicabilidade dos métodos ágeis em sistemas críticos como *Spaced-Based Systems*([S8][S10]), *Mission critical*([S12]), Sistemas clínicos([S6]), Controlador de simulador de vôo ([S15]) e *Safety-Critical systems* utilizado em ambientes controlados ([S13]) . Foi ainda verificada uma resposta positiva sobre essa adoção, o que pode significar que nos próximos anos iremos nos deparar cada vez mais com estudos que relatem a utilização de métodos ágeis combinados a outras técnicas.

Sobre a dependabilidade, os atributos são considerados mais ou menos importantes de acordo com o contexto do projeto. No casos dos estudos selecionados a confiabilidade *(reliability)* e a segurança *(safety)* foram considerados os mais importantes ([S6][S10][S12] e [S6][S8][S10][S15]), seguidos da manutenabilidade *(maintainability)* ([S7][S12]). A integridade *(Integrity)* e a confidencialidade *(Confidentiality)* não foram mencionada diretamente em nenhum dos estudos selecionados, porém nos contextos estudados deve haver preocupação com tais atributos.

Capítulo 7

Considerações Finais

Neste capítulo estão descritas as considerações finais, as limitações e ameaças à validade do trabalho, assim como são apresentadas propostas de trabalhos futuros a partir dos resultados encontrados e finaliza-se com as conclusões do trabalho.

7.1 Ameaças a Validade e Limitações

- Metodologia utilizada: A metodologia utilizada foi validada por autores que são referência na área e são encontradas em uma grande quantidade de trabalhos de mapeamento sistemático, o que diminui a possibilidade da existência de erros no processo
- Questões de Pesquisa: as questões definidas nesta monografia não englobam todo o universo comprendido por dependabilidade e métodos ágeis, logo algumas questões sobre o assunto não foram respondidas. Entretanto, as questões propostas aqui foram escolhidas em conjunto, autora e orientadora, para se chegar a um resultado que trouxesse contribuição relevante à literatura.
- *String* de busca e seleção dos estudos: a *String* utilizada foi a mesma do trabalho base, pois um dos objetivos desse trabalho foi a atualização dos resultados. A busca da primeira fase, por ser automatica, pode não ter retornado todos os estudos publicados nas bases nesses 2 anos. Para mitigar possiveis falhas foram feitos o *backward snowballing* e busca manual, tanto nos eventos sobre os assuntos quanto no dbpl dos principais autores na área. Foi utilizado os mesmo critérios de inclusão e exclusão.
- Fonte de dados: as buscas foram feitas em 4 bases de dados, então existe a possibilidade de trabalhos relevantes no assunto não terem sido selecionados por estarem

em outras bases. Houve uma dificuldade na busca em cada uma das bases pois as ferramentas de busca executam a *string* de maneiras diferentes.

- Extração dos dados: apesar de se tratar de processo sistemático, o gargalo dessa metodologia é a análise dos estudos retornados, pois mesmo sendo seguidas todos os critérios ainda sim é um julgamento subjetivo. O estudos base foi realizado em dupla e esta monografia foi realizada somente pela autora. Para mitigar possíveis equívocos na seleção os dados foram analisados tambem pela orientadora.
- Análise: foi feita análise sobre como os métodos ágeis influenciam em software críticos, porém as buscas não foram focadas nesse assunto desde o princípio. Por isso pode haver mais trabalhos relevantes sobre o assunto que não foram selecionados nesta monografia.

7.2 Conclusão

Foram identificados na literatura 1789 estudos, dos quais foram selecionados 15 utilizando os mesmo critérios do trabalho base e para reponder as questões de pesquisa desta monografia foram selecionados 7 estudos. Como já concluído no estudo base, o tema de dependabilidade relacionado com a utilização de métodos ágeis ainda é bastante inexplorado, porém foi identificada um aumento significativo na quantidade de estudos, principalmente quando falamos na adoção dos métodos para desenvolvimento de software crítico.

Ainda como resultado deve-se destacar a importância de uma melhor definição do uso dos termos de dependabilidade. Os atributos são bem aplicados, porém ainda existe a dificuldade de análise de como os métodos ágeis influenciam em falha, erro e defeito por causa da falta de padrão na utilização dos termos. Mantemos a sujestão de serem utilizadas as definições dadas por Avizienis et al. [4].

Por fim abaixo é apresentado um resumo dos principais resultados deste trabalho:

- A quantidade de estudos relevantes para o tema aumentou, porém ainda é baixa a quantidade de trabalhos e os estudos existentes não cobrem toda a gama de assuntos relacionados sobre o assunto.
- Baseado nos resultados pode-se concluir que metodologias ágeis tendem a ser benéficas a dependabilidade do software, mas a sua contribuição é melhorada quando são combinadas com outros métodos, de acordo com o contexto de cada projeto.
- Foi identificado um estudo que utiliza as mesmas métricas aplicadas à métodos tradicionais para avaliação dos métodos ágeis [19].

- Continua a necessidade de melhor definição e utilização dos termos de qualidade, falta e defeito.

7.3 Trabalhos Futuros

Após a realização do estudos foi verificado que algumas das sugestões apresentadas no estudo base permanecem. Foram tambem identificados novos campos a serem explorados. Ambas as sugestões estão listadas a seguir:

Sugestões do Estudo Base

- É necessária a realização de mais estudos relevantes para definir "qualidade do software"quando utilizamos termos de dependabilidade.
- Pode-se aplicar a mesma metodologia desse estudo para averiguar estudos específicos sobre práticas ágeis que tratem do tema dependabilidade.Muitos estudos voltaram a ser retornados sobre práticas como TDD, pair programming e refactoring, porem foram retirados de acordo com o critério deste estudo.
- Ainda existe uma deficiência na quantidade de estudos que tratem sobre métricas de qualidade aplicadas a métodos ágeis. É preciso realizar mais estudos sobre o assunto, tanto estudos da literatura quanto casos de estudo.

Sugestões desse Estudo

- Realização de estudo da literatura e estudos de caso sobre a adoção dos métodos ágeis no desenvolvimento de sistemas críticos, como *mission critical* e *clinical software*, por exemplo.
- Estudos sobre quais são as técnicas utilizadas em métodos tradicionais de desenvolvimento de software que podem ser combinadas ao métodos ágil para suprir os gaps que os métodos ágeis apresentam em certos contextos
- Levantamento sobre quais são os fatores do processo dos métodos ágeis, do ponto de vista da equipe, que influenciam na dependabilidade do software.

Referências

[1] *ESEC/FSE-9: Proceedings of the 8th European Software Engineering Conference Held Jointly with 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, New York, NY, USA, 2001. ACM. 594010. 1

[2] V. Antinyan, M. Staron, W. Meding, P. Osterstrom, E. Wikstrom, J. Wranker, A. Henriksson, e J. Hansson. Identifying risky areas of software code in agile/lean software development: An industrial experience report. In *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week - IEEE Conference on*, pages 154–163, Feb 2014. 56

[3] J. Ard, K. Davidsen, e T. Hurst. Simulation-based embedded agile development. *Software, IEEE*, 31(2):97–101, Mar 2014. 56

[4] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, e Carl Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Secur. Comput.*, 1(1):11–33, January 2004. x, 5, 6, 7, 8, 10, 11, 39, 48

[5] V. R. Basili, F. Shull, e F. Lanubile. Building knowledge through families of experiments. *IEEE Transactions on Software Engineering*, 25(4):456–473, 1999. Cited By (since 1996):326. 1

[6] Kent Beck e Cynthia Andres. *Extreme Programming Explained: Embrace Change (2nd Edition)*. Addison-Wesley Professional, 2004. 4, 16, 19

[7] Artur de Azevedo Braga e Renato dos Santos Leal. Estudo sistemático em dependabilidade e métodos ágeis: uma análise de falhas e defeitos. 2013. vi, vii, x, 2, 3, 4, 5, 6, 8, 9, 21, 22, 24, 30, 31, 34, 36, 40, 42

[8] S.E. Carpenter e A. Dagnino. Is agile too fragile for space-based systems engineering? In *Space Mission Challenges for Information Technology (SMC-IT), 2014 IEEE International Conference on*, pages 38–45, Sept 2014. 56

[9] Harris Cooper, Larry V Hedges, e Jeffrey C Valentine. *The handbook of research synthesis and meta-analysis*. Russell Sage Foundation, 2009. 1

[10] Mario Dantas. *Computação Distribuída de Alto Desempenho*. Axcel Books, Brasília, 2005. 7

[11] Jorge Calmon de Almeida Biolchini, Paula Gomes Mian, Ana Candida Cruz Natali, Tayana Uchôa Conte, e Guilherme Horta Travassos. Scientific research ontology to

support systematic review in software engineering. *Adv. Eng. Inform.*, 21(2):133–151, April 2007. 1, 2, 22

[12] Tore Dybå, Vigdis By Kampenes, e Dag Sjøberg. A systematic review of statistical power in software engineering experiments. *Information and Software Technology*, 48(8):745–755, August 2006. 2

[13] Tore Dybå e Torgeir Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9–10):833 – 859, 2008. xi, 14

[14] Kent Beck et al. *Manifesto for Agile Software Development.* Disponível em: http://agilemanifesto.org/, 2001. x, 4, 13, 15, 43

[15] B. Fitzgerald, K.-J. Stol, R. O'Sullivan, e D. O'Brien. Scaling agile methods to regulated environments: An industry case study. In *Software Engineering (ICSE), 2013 35th International Conference on*, pages 863–872, May 2013. 56

[16] Martin Fowler e Jim Highsmith. The agile manifesto. *Software Development Magazine*, 9(8):29–30, 2001. 13

[17] A. Ghazarian. Reliability in agile software engineering: A dilemma. Technical report, IEEE Reliability Society, East Lansing, Michigan, 2011. 2

[18] Samireh Jalali e Claes Wohlin. Systematic literature studies: database searches vs. backward snowballing. In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*, ESEM '12, pages 29–38, New York, NY, USA, 2012. ACM. 30

[19] K. Jinzenji, T. Hoshino, L. Williams, e K. Takahashi. An experience report for software quality evaluation in highly iterative development methodology using traditional metrics. In *Software Reliability Engineering (ISSRE), 2013 IEEE 24th International Symposium on*, pages 310–319, Nov 2013. 35, 48, 56

[20] K. Kaur, A. Jajoo, e Manisha. Applying agile methodologies in industry projects: Benefits and challenges. In *Computing Communication Control and Automation (ICCUBEA), 2015 International Conference on*, pages 832–836, Feb 2015. 56

[21] Barbara Kitchenham e Stuart Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 2007. 21

[22] Barbara A. Kitchenham, Tore Dyba, e Magne Jorgensen. Evidence-based software engineering. In *Proceedings of the 26th International Conference on Software Engineering*, ICSE '04, pages 273–281, Washington, DC, USA, 2004. IEEE Computer Society. 1

[23] Aapo Koski e Tommi Mikkonen. Rolling out a mission critical system in an agilish way. reflections on building a large-scale dependable information system for public sector. In *Rapid Continuous Software Engineering (RCoSE), 2015 IEEE/ACM 2nd International Workshop on*, pages 41–44, May 2015. 56

[24] W. Kuchinke, C. Krauth, e T. Karakoyun. Agile software development requires an agile approach for computer system validation of clinical trials software products. In *eChallenges e-2014, 2014 Conference*, pages 1–8, Oct 2014. 56

[25] Lawrence Leemis. *Probabilistic Models and Statistical Methods*. Prentice Hall, 1995. 6

[26] Michael R. Lyu, editor. *Handbook of software reliability engineering*. McGraw-Hill, Inc., Hightstown, NJ, USA, 1996. 6

[27] Yuan-Shun Dai Min Xie, Kim-Leng Poh. *Computing System Reliability: Models and Analysis*. Springer, US, 2004. 6

[28] S.M. Mitchell e C.B. Seaman. A comparison of software cost, duration, and quality for waterfall vs. iterative and incremental development: A systematic review. In *Empirical Software Engineering and Measurement, 2009. ESEM 2009. 3rd International Symposium on*, pages 511–515, 2009. 23

[29] Jesper Pedersen Notander, Per Runeson, e Martin Höst. A model-based framework for flexible safety-critical software development: A design study. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 1137–1144, New York, NY, USA, 2013. ACM. 56

[30] M. Paiva Ramos, G. Ravanhani Matuck, C. Fernandes Matrigrani, S. Mirachi, E. Segeti, M. Leite, A.M. Da Cunha, e L.A. Vieira Dias. Applying interdisciplinarity and agile methods in the development of a smart grids system. In *Information Technology: New Generations (ITNG), 2013 Tenth International Conference on*, pages 103–110, April 2013. 56

[31] Rivalino Mathias Paulo Maciel, Kishor Trivedi e Dong Kim. *Dependability Modeling In: Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions.* Ed. Hershey: IGI Global, Pennsylvania, USA, 2010. 6, 7

[32] Kai Petersen, Robert Feldt, Shahid Mujtaba, e Michael Mattsson. Systematic mapping studies in software engineering. In *Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering*, EASE'08, pages 68–77, Swinton, UK, UK, 2008. British Computer Society. 21, 31

[33] Kai Petersen e Claes Wohlin. A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. *J. Syst. Softw.*, 82(9):1479–1490, September 2009. 19

[34] Kai Petersen e Claes Wohlin. The effect of moving from a plan-driven to an incremental software development approach with agile practices. *Empirical Softw. Engg.*, 15(6):654–693, December 2010. 19

[35] A.L. Pierre Mattei, A. Marques da Cunha, L.A. Vieira Dias, E. Fonseca, O. Saotome, P. Takachi, G. Sousa Goncalves, T.A. Pivetta, V. da Silva Montalvao, C. Kendi, F. Lopes de Freitas, M. Alves Ferreira, M. Andrade Almeida, e G. Goncalves de Oliveira Rodrigues. Nanosatellite event simulator development using scrum agile

method and safety-critical application development environment. In *Information Technology - New Generations (ITNG), 2015 12th International Conference on*, pages 101–106, April 2015. 56

[36] Tom Poppendieck e Mary Poppendieck. *Lean Software Development: An Agile Toolkit for Software Development Managers.* Addison-Wesley Professional, 1 edition, May 2003. 4

[37] Kuda Nageswara Rao, G Kavita Naidu, e Praneeth Chakka. A study of the agile software development methods, applicability and implications in industry. *International Journal of Software Engineering and its applications*, 5(2):35–45, 2011. 20

[38] Kalle Rindell, Sami Hyrynsalmi, e Ville Leppänen. A comparison of security assurance support of agile software development methods. In *Proceedings of the 16th International Conference on Computer Systems and Technologies*, CompSysTech '15, pages 61–68, New York, NY, USA, 2015. ACM. 56

[39] Ken Schwaber. *Agile Project Management With Scrum.* Microsoft Press, Redmond, WA, USA, 2004. 4, 19

[40] Panagiotis Sfetsos e I. Stamelos. Empirical studies on quality in agile practices: A systematic literature review. In *Quality of Information and Communications Technology (QUATIC), 2010 Seventh International Conference on the*, pages 44–53, 2010. 23

[41] Forrest Shull, Vic Basili, Barry Boehm, A. Winsor Brown, Patricia Costa, Mikael Lindvall, Dan Port, Ioana Rus, Roseanne Tesoriero, e Marvin Zelkowitz. What we have learned about fighting defects. In *Proceedings of the 8th International Symposium on Software Metrics*, METRICS '02, pages 249–, Washington, DC, USA, 2002. IEEE Computer Society. 2

[42] Ian Sommerville. *Software engineering (8th ed.).* Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2007. x, 18, 19

[43] C. Webster, Nija Shi, e I.S. Smith. Delivering software into nasa's mission control center using agile development techniques. In *Aerospace Conference, 2012 IEEE*, pages 1–7, March 2012. 34, 56

[44] Roel Wieringa, Neil Maiden, Nancy Mead, e Colette Rolland. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requir. Eng.*, 11(1):102–107, December 2005. xi, 32

[45] Sune Wolff. Scrum goes formal: Agile methods for safety-critical systems. In *Proceedings of the First International Workshop on Formal Methods in Software Engineering: Rigorous and Agile Approaches*, FormSERA '12, pages 23–29, Piscataway, NJ, USA, 2012. IEEE Press. 34, 56

Apêndice A

Listas de Artigos

A.1 Busca Eletrônica

NO	Título
SSQ1	An Experience Report for Software Quality Evaluation in Highly Iterative Development Methodology Using Traditional Metrics
SSQ2	Applying Agile Methodologies in Industry Projects: Benefits and Challenges
SSQ3	Applying Interdisciplinarity and Agile Methods in,the Development of a Smart Grids System
SSQ4	Delivering software into NASA's Mission Control Center using agile development techniques
SSQ5	Is Agile Too Fragile for Space-Based Systems Engineering?
SSQ6	Nanosatellite Event Simulator Development Using Scrum Agile Method and Safety Critical Application Development Environment
SSQ7	Simulation-Based Embedded Agile Development
SSQ8	Rolling Out a Mission Critical System in an Agilish Way. Reflections on Building a Large-Scale Dependable Information System for Public Sector
SSQ9	Scaling agile methods to regulated environments: An industry case study
SSQ10	Scrum goes formal: Agile methods for safety-critical systems
SSQ11	Selecting software reliability growth models and improving their predictive accuracy using historical projects data
SSQ12	Amodel-based framework for flexible safety-critical software development: a design study
SSQ13	Agile development with software process mining
SSQ14	An experience report for software quality evaluation in highly iterative development methodology using traditional metrics

Tabela A.1: Busca Eletrônica

A.2 *Snowballing*

No.	Artigo
SSB1	A survey study of critical success factors in agile software projects
SSB2	Identifying risky areas of software code in Agile/Lean software development: An industrial experience report
SSB3	Metric-based quality evaluations for iterative software development approaches like Agile
SSB4	Software Quality Control via Exit Criteria Methodology: An Industrial Experience Report
SSB5	Agile software development requires an agile approach for computer system validation of clinical trials software products
SSB6	A Comparison of Security Assurance Support of Agile Software Development Methods

Tabela A.2: Snowballing

Apêndice B

Estudos Selecionados

No.	Título
[S1]	An Experience Report for Software Quality Evaluation in Highly Iterative Development Methodology Using Traditional Metrics [19]
[S2]	Applying Interdisciplinarity and Agile Methods in the Development of a Smart Grids System [30]
[S3]	A Comparison of Security Assurance Support of Agile Software Development Methods [38]
[S4]	A Model-Based Framework for Flexible Safety-Critical Software Development – A Design Study []
[S6]*	Agile Software Development Requires an Agile Approach for Computer System Validation of Clinical Trials Software Products [24]
[S7]*	Applying Agile Methodologies in Industry Projects: Benefits and Challenges [20]
[S8]*	Delivering Software into NASA's Mission Control Center Using Agile Development Techniques [43]
[S9]	Identifying Risky Areas of Software Code in Agile/Lean Software Development: An Industrial Experience Report [2]
[S10]*	Is Agile too Fragile for Space-Based Systems Engineering? [8]
[S11]	Nanosatellite Event Simulator Development Using Scrum Agile Method and Safety-Critical Application Development Environment [35]
[S12]*	Rolling out a mission critical system in an agilish way [23]
[S13]*	Scaling Agile Methods to Regulated Environments: An Industry Case Study [15]
[S14]	Scrum Goes Formal: Agile Methods for Safety-Critical Systems [45]
[S15]*	Simulation-Based Embedded Agile Development [3]
[S16]	A survey study of critical success factors in agile software projects in former Yugoslavia IT companies [29]

Tabela B.1: Estudos Selecionados Para leitura final

Estudos com *: Estudos selecionados para responder às perguntas desta monografia

Apêndice C

Autores

Nome	GS*	Instituição	Página
Scott E. Carpenter	241000	NorthCarolina State University	[1]
Christopher Webster	8		[2]
Brian Fitzgerald	4323	University of Limerick	[3]
Ryan O'Sullivan	31900		
Nija Shi	11900	UC Davis	[4]
Christian Krauth	2630		
Aldo Dagnino	1430		[5]
Kamaljeet Kaur	1220		
Tommi Mikkonen	1190	Tampere University of Technology	[6]
Terril Hurst	823		
Wolfgang Kuchinke	465	Heinrich-Heine-Universität Düsseldorf	[7]
Aapo Koski	313	Tampere University of Technology	[8]
Klaas-Jan Stol	167	University of Limerick	[9]
Töresin Karakoyun	17		[10]
Irene Skupniewicz Smith	3		[11]
Manisha Anuj Jajoo	x		
Donal O'Brien	x		
Jason Ard	x		
Kristine Davidsen	x		[12]

Tabela C.1: Autores

* Número de citações na página do autor do Google Scholar(`scholar.google.com`).

[1]`http://www4.ncsu.edu/~scarpen/index.html`

[2]`https://www.researchgate.net/profile/Christopher_Webster3`

[3]`https://www.researchgate.net/profile/Brian_Fitzgerald`

[4]`https://www.linkedin.com/in/nijashi`

[5]`http://www.agileteams.com/aboutAD.html`

[6]`http://www.cs.tut.fi/~tjm/index-international.html`

[7]https://www.researchgate.net/profile/Wolfgang_Kuchinke2
[8]https://www.linkedin.com/in/aapokoski
[9]http://staff.lero.ie/stol/
[10]https://www.xing.com/profile/Toeresin_Karakoyun
[11]https://www.linkedin.com/in/ireneskupniewiczsmith
[12]https://www.linkedin.com/in/kldavidsen