INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
PRÓ-REITORIA DE ENSINO
COORDENADORIA DE CIÊNCIA DA COMPUTAÇÃO DO CAMPUS ARACATI
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ADRIANA FERNANDES DA SILVA

ANÁLISE SOBRE A ENGENHARIA DE REQUISITOS EM PROJETOS COM MODELO DE PROCESSO ÁGIL

ARACATI-CE
2017

Adriana Fernandes da Silva

ANÁLISE SOBRE A ENGENHARIA DE REQUISITOS EM PROJETOS COM MODELO DE PROCESSO ÁGIL

Trabalho de conclusão de curso apresentado à Coordenadoria de Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará - Campus Aracati como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Área de pesquisa: Engenharia de Software

Orientadora: Profa Msc. Francisca Raquel de Vasconcelos Silveira.

Aracati-CE
2017

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
COORDENADORIA DE CIÊNCIA DA COMPUTAÇÃO DO CAMPUS ARACATI
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ADRIANA FERNANDES DA SILVA

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção do Grau de Bacharel em Ciência da Computação, sendo aprovado pela Coordenadoria de Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará - Campus Aracati e pela banca examinadora:

Prof. Msc. Francisca Raquel de V. Silveira
Instituto Federal do Ceará - IFCE
Orientadora

Prof. Msc. Paulo Alberto Melo Barbosa
Instituto Federal do Ceará - IFCE

Prof. Msc. Enyo José Tavares Gonçalves
Universidade Federal do Ceará - UFC

Aracati, 25 de Abril de 2017

Agradecimentos

Agradeço primeiramente a DEUS, pois mesmo nas pequenas decisões ele sempre me guiou a tomar o melhor caminho.

A Professora Msc. Raquel Silveira, pela amizade, parceria e orientação. Aprendi muito, recebi lições de dedicação, caráter e paixão pelo ensino e pesquisa que jamais esquecerei.

A Professora Dr. Carina Teixeira por todo acompanhamento dado, críticas e sugestões que muito auxiliaram na evolução desta pesquisa.

Agradeço aos membros da banca examinadora o Prof. Msc. Paulo Alberto e Prof. Msc. Enyo Gonçalves, pela disponibilidade de participar e pelas contribuições pessoais acerca desse trabalho.

Agradeço aos meus pais por toda a atenção, amor, carinho, pelo constante apoio sempre dedicado. Aos meus irmãos, por toda a força e alegria.

Agradeço também com muito carinho ao meu futuro esposo David por todo o amor concedido a mim.

Por fim, agradeço a todos os amigos e professores do curso de Ciência da Computação, pela motivação, pelos bons momentos e conhecimento compartilhados durante o curso.

Dedico este trabalho aos meus maravilhosos pais.

Resumo

A engenharia de requisitos fornece o mecanismo para entender o que o cliente deseja, realizando a elicitação de requisitos, especificação e gerência de requisitos. Porém, pode surgir problema nessa fase, como a falta de compreensão na especificação de requisitos, falta de comunicação entre as partes interessadas no projeto, alto custo para mudanças no decorrer do software, ocasionado o insucesso do projeto. As metodologias ágeis possuem diferentes práticas que tentam lidar com esses problemas, possibilitando a implementação correta dos requisitos e, consequentemente, satisfazendo as necessidades do cliente. Dessa forma, existem práticas que auxiliam na comunicação com o cliente e equipe de desenvolvimento, melhorando a elicitação dos requisitos, entrega do produto no tempo adequado, reduzindo custo, priorizando a qualidade e as funcionalidades. Este trabalho apresenta um comparativo sobre a engenharia de requisitos para a metodologia ágil Scrum e Scrum/*eXtreme Programming* (XP) - (Híbrido), através da aplicação de um questionário em cinco empresas reais, com o intuito de analisar a relação de métodos ágeis e engenharia de requisitos no desenvolvimento de software. Através desta análise comparativa, é possível justificar as principais diferenças das práticas dessas duas metodologias, conhecer os desafios e as boas práticas.

Palavras-chaves: Engenharia de Requisitos. Metodologia ágil. Desenvolvimento de software.

Abstract

The requirements engineering provides the mechanism to understand what the customer wants, realizing the requirements elicitation, specification and requirements management. However, there may be a problem at this stage, as the lack of understanding in the requirements specification, lack of communication between the stakeholders in the project, high cost for changes during the software, raised the failure of the project. Agile methodologies have different practices that attempt to deal with these problems, enabling the correct implementation of the requirements and, consequently, satisfying the customer's needs. Thus, there are practices that assist in communicating with the client and the development team, improving the elicitation of requirements, product delivery in a timely manner, reducing cost, prioritizing the quality and features. This work presents a comparison on requirements engineering for the Scrum methodology and Scrum/eXtreme Programming (XP) (hybrid), through the application of a questionnaire in five real companies, in order to analyze the relationship of agile methods and requirements engineering in software development. Through this comparative analysis, it is possible to justify the main differences of the practices of these two methodologies, to meet the challenges and good practices.

Keywords: Requirements engineering. Agile methodology. Software development.

Sumário

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

XP	*eXtreme Programming*
FDD	*Feature Driven Development*
ASD	*Adaptive Software Development*
CM	*Crystal Methods*
TDD	*Test Driven Development*
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
SEI	*Software Engineering Institute*
DERS	Documentos de Especificações de Requisitos de Sistemas
RUP	*Rational Unified Process*
JAD	*Joint Application Development*
UML	*Unified Modeling Language*
TDA	*Acceptance Testing*

CAPÍTULO 1

INTRODUÇÃO

Engenharia de Software é uma área da computação voltada para todos os aspectos de produção do software. Inclui atividades de especificação, desenvolvimento, criação de sistemas de software, com aplicação de tecnologias e práticas de gerência de projetos. Friedrich Ludwig Bauer, professor da Universidade Técnica de Munique na Alemanha, define Engenharia de Software como "a criação e a utilização de sólidos princípios de engenharia a fim de obter software de maneira econômica, que seja confiável e que trabalhe eficientemente em máquinas reais"(PRESSMAN, 2011).

A Engenharia de Software foca em diversos estudos, metodologias e técnicas que visam aperfeiçoar a produção de software. Entre esses estudos estão: processo de desenvolvimento de software, engenharia de requisitos, planejamento, metodologias ágeis, gerenciamento do desenvolvimento de software, arquitetura de software, padrões de projeto, validação e verificação de sistemas e qualidade de software, entre outros (SOMMERVILLE, 2015).

Neste trabalho, o foco é a Engenharia de Requisitos, processo que fornece o mecanismo apropriado para entender o desejo do cliente, analisando as necessidades, avaliando a viabilidade, especificando a solução sem ambiguidade, validando a especificação e gerenciando as necessidades à medida que são transformadas em um sistema operacional (THAYER; BAILIN; DORFMAN, 1997). Portanto, o papel da engenharia de requisitos é fundamental para a construção e desenvolvimento de software, tendo responsabilidade de garantir qualidade, prazos, custos e funcionalidades.

A preocupação com essa temática não é atual. Estudos realizados em 1995 pelo Standish Group com 350 companhias e 8.000 projetos de software já revelava fatores críticos para sucesso dos projetos de software. Dentre estes, foi possível perceber que três dos principais fatores estavam relacionados às atividades de requisitos: (1) Requisitos Incompletos; (2) Falta de Envolvimento do Usuário; e (3) Mudança de Requisitos e Especificações. Nota-se que esses três fatores estão ligados diretamente ao levantamento de requisitos, usuários e agilidade no processo de mudança. Nos últimos 15 anos não houve nenhuma grande melhoria. Em análise mais detalhada dos projetos de desenvolvimento de software, (HASTIE S.AND WOJEWODA, 2015) analisam o Chaos Report do Standish Group apresentado na Tabela 1, informaram que

nos últimos cinco anos, a porcentagem de projetos de sucesso variou entre 27% a 31%. A porcentagem de projetos desafiados variou entre 49% a 56% e a porcentagem de projetos falhados variou entre 17% e 22%. Eles identificaram em sua análise que os principais fatores que causam projetos a serem desafiados ou que falharam são requisitos e especificações incompletas, bem como mudanças de requisitos e especificações.

Tabela 1: Resolução moderna para todos os projetos. [Fonte: (HASTIE S.AND WOJEWODA, 2015)]

	2011	2012	2013	2014	2015
Sucesso	29%	27%	31%	28%	29%
Desafio	49%	56%	50%	55%	52%
Fracasso	22%	17%	19%	17%	19%

Os mecanismos que a engenharia de requisitos fornece costumam ser muito rigorosos e extensos, sendo compostas de longos períodos de construção de software (SOMMERVILLE, 2015). Com o intuito de entregar o produto mais rápido, com alta qualidade, satisfazer as necessidades do cliente e visando melhorar o andamento dos projetos, foi criada a metodologia ágil, um novo método de gestão e desenvolvimento de software, que usa uma abordagem de execução iterativa e incremental voltado para processos complexos, com mudanças ao longo do processo, dividindo o problema em produtos menores e que visa entregar software funcionando regularmente e entre outras vantagens. Dentre as metodologias ágeis, estão: eXtreme Programming (XP), Scrum, Feature Driven Development (FDD), Adaptive Software Development (ASD), Crystal Methods (CM) (SCHWABER; BEEDLE, 2001) (SCHWABER K.; SURTHERLAND, 2015) (SOMMERVILLE, 2015).

Porém, de acordo com (VERSIONONE, 2015), duas das mais populares metodologias utilizadas no contexto de metodologias ágeis são o Scrum e o Scrum/XP (Híbrido). O Scrum proporciona uma abordagem ágil para a gestão de projetos aumentando a probabilidade de sucesso e o XP é utilizado mais no desenvolvimento de software. Por essa razão, o estudo de caso deverá focar nas duas metodologias ágeis.

O objetivo principal deste trabalho é apresentar uma análise das técnicas da engenharia de requisitos utilizados pelas empresas de desenvolvimento de software que aplicam metodologia ágil localizadas no estado do Ceará.

1.1 Motivação

O sucesso da produção de software depende de n variáveis, tais como manutenção, custo, prazo, qualidade, agilidade e satisfação do cliente. Atualmente, os projetos têm sentido a necessidade de oferecer uma maior atenção aos requisitos, pois estes, no ambiente de negócio e desenvolvimento, tendem a evoluir rapidamente e se

tornarem ultrapassados mesmo antes dos projetos serem finalizados. Dessa forma o mercado de desenvolvimento de software vem sendo desafiado em seus métodos, conceitos e ferramentas.

Assim, surgiram às metodologias ágeis, que têm o intuito de entregar produto mais rápido, com alta qualidade, satisfazer as necessidades do cliente, ser flexível a mudanças, visando um melhor andamento dos projetos. Métodos ágeis não se baseiam em padrões tradicionais de engenharia de requisitos, porém, eles têm adaptado muitas das ideias básicas da engenharia de requisitos ao seu ambiente.

A justificativa deste trabalho vem da visibilidade crescente que as Metodologias Ágeis de desenvolvimento de software têm recebido no mercado. O trabalho apresenta um estudo de como as empresas utilizam a engenharia de requisitos no contexto de metodologia ágil, auxiliando o processo de levantamento de requisitos, análise e construção de um sistema.

1.2 Objetivos e Contribuições

O objetivo principal deste trabalho é apresentar uma análise das técnicas da engenharia de requisitos utilizados pelas empresas de desenvolvimento de software que aplicam metodologia ágil. Para validação deste trabalho foi realizado um estudo de caso com cinco empresas de desenvolvimento de software em atuação no mercado do estado do Ceará, que utilizam em seus projetos modelo de processo ágil. O estudo foi realizado através da análise das respostas de um questionário aplicado para membros das empresas. A avaliação apresentou pontos relacionados a aplicação da engenharia de requisitos no modelo de processo ágil, tais como: as técnicas adotadas na Engenharia de Requisitos do ponto de vista ágil, os processos que são utilizadas na Engenharia de Requisitos e os desafios da Engenharia de Requisitos para a metodologia ágil.

As principais contribuições deste trabalho são:

- Análise das técnicas da engenharia de requisitos utilizadas em cada fase da metodologia ágil.
- Análise das técnicas da engenharia de requisitos utilizadas pelas empresas de desenvolvimento de software que aplicam metodologia ágil.

1.3 Organização do Trabalho

Capítulo 2 – Fundamentação Teórica: Trata os principais conceitos referentes à engenharia de requisitos, destacando os tipos de requisitos, os processos da

engenharia de requisitos e os seus artefatos, além de contextualizar a metodologia ágil Scrum, com as etapas da sprint e seus artefatos, a metodologia ágil XP, seguida de seus valores e princípios.

Capítulo 3 – Trabalhos Relacionados: Apresenta os trabalhos e conceitos da engenharia de requisitos para metodologias ágeis Scrum e XP, destacando suas principais características e técnicas.

Capítulo 4 - Proposta: Dispõe de um estudo de caso que apresenta e detalha os aspectos analisados no Capítulo 5.

Capitulo 5 - Resultados e Discussão: Neste capítulo será apresentado os resultados obtidos a partir da aplicação do questionário e análise na engenharia de requisitos em metodologia ágil Scrum e Scrum/XP (Híbrido).

Capítulo 6 – Conclusões: Relaciona as principais conclusões deste trabalho e apresenta possíveis linhas de pesquisa a serem consideradas em trabalhos futuros.

CAPÍTULO 2

FUNDAMENTAÇÃO TEÓRICA

Nesta seção apresentamos sucintamente os principais conceitos referentes a engenharia de requisitos, destacando os tipos de requisitos, os processos da engenharia de requisitos e os seus artefatos, contextualizando ainda a Metodologia ágil Scrum, com as etapas da *Sprint* e seus artefatos e *eXtreme Programming* com seus valores e práticas.

2.1 Engenharia de Requisitos

A Engenharia de Requisitos é o ramo da Engenharia de Software que envolve as atividades relacionadas a definição dos requisitos de software de um sistema, desenvolvidas ao longo do ciclo de vida de software (SOMMERVILLE, 2015).

Segundo o SEI (*Software Engineering Institute*), requisito é uma condição ou capacidade necessária a um usuário para resolver um problema ou alcançar um objetivo, que deve ser possuído por um sistema ou por um componente do sistema para satisfazer um contrato, padrão, especificação ou outros documentos formalmente impostos (IEEE, 1990).

Na engenharia de requisitos, o requisito representa as descrições do cliente para a construção de um sistema, com o intuito de atender e solucionar um problema. Requisitos são a base de todos os produtos de software e sua elicitação, gerenciamento e entendimento são problemas comuns a todas as metodologias de desenvolvimento de software (AURUM; WOHLIN, 2005).

2.1.1 Tipos de Requisitos

Os requisitos de sistemas de software são classificados em: (i) requisitos funcionais, (ii) requisitos não funcionais ou (iii) requisitos de domínio (SOMMERVILLE, 2015).

Requisitos funcionais são as declarações que o sistema deve fornecer, como o sistema deve reagir a entradas específicas e como o sistema deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais podem também

estabelecer explicitamente o que o sistema não deve fazer (SOMMERVILLE, 2015).

Requisitos não funcionais são restrições sobre os serviços ou as funções oferecidas pelo sistema. Eles podem incluir restrições de *timing* e restrições sobre o processo de desenvolvimento. Os requisitos não funcionais aplicam-se, frequentemente, ao sistema como um todo. Em geral eles não se aplicam às características ou serviços individuais do sistema (SOMMERVILLE, 2015).

Requisitos de domínio são requisitos provenientes do domínio da aplicação do sistema e que refletem as características e as restrições desse domínio. Podem ser classificados como requisitos funcionais ou não funcionais (SOMMERVILLE, 2015).

2.1.2 Processo de Engenharia de Requisitos

A engenharia de requisitos é uma atividade responsável por criar e manter Documentos de Especificações de Requisitos de Sistemas (DERS). Esse processo é dividido em quatro subprocessos: (i) estudo de viabilidade, (ii) elicitação e análise, (iii) especificação e (iv) validação.

A Figura 1 mostra o processo de engenharia de requisitos (SOMMERVILLE, 2015).

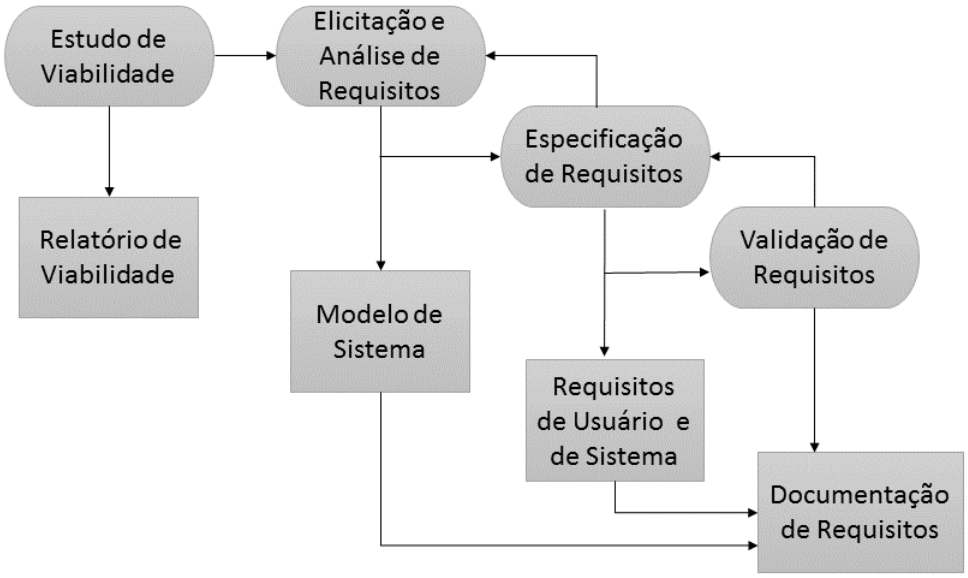

Figura 1: Processo de Engenharia de Requisitos [Fonte: (SOMMERVILLE, 2015)].

Durante o Estudo de Viabilidade, é feita uma estimativa verificar se as necessidades dos usuários podem ser satisfeitas por meio das tecnologias de software e hardware. O estudo considera se o software pode ser desenvolvido dentro de restrições de orçamento existentes (SOMMERVILLE, 2015).

A atividade seguinte, de Elicitação e Análise de Requisitos, corresponde ao processo de obter os requisitos do sistema pela observação de sistemas existentes, pela conversa com potenciais clientes, dentre outros. Isso pode envolver o desenvolvimento de um ou mais modelos de sistemas e protótipos. Esta atividade ajuda o analista a compreender o sistema a ser especificado (SOMMERVILLE, 2015).

A atividade de Especificação de requisitos é responsável por traduzir as informações coletadas em um documento que define um conjunto de requisitos. Devem ser incluídos dois tipos de requisitos neste documento: (i) requisitos de usuário (declarações abstratas dos requerimentos de sistema para o cliente e os seus usuários finais) e (ii) requisitos de sistema (descrição mais detalhadas da funcionalidade a ser fornecida) (SOMMERVILLE, 2015).

A última atividade de Validação de requisitos verifica dos requisitos em relação ao realismo, consistência e abrangência. Durante esse processo, erros no documento de requisitos são inevitavelmente descobertos. Devem então ser feitas modificações para corrigir esses problemas (SOMMERVILLE, 2015).

As atividades no processo de requisitos não são feitas em apenas uma sequência. A análise de requisitos continua durante a definição e especificação, e novos requisitos emergem durante o processo.

2.1.3 Artefatos

Artefatos são produtos de trabalhos finais ou intermediários produzidos e usados durante os projetos para capturar e transmitir informações do projeto. Um artefato pode ser um dos seguintes elementos (CORPORATION, 2001):

- Um documento, como Documento de Caso de Uso;
- Um modelo, como o Modelo de Casos de Uso ou o Modelo de Design;
- Um elemento do modelo, ou seja, um elemento existente em um modelo, como uma classe ou um subsistema.

O *Rational Unified Process* (RUP) é um processo da engenharia de software com o objetivo de assegurar a produção de software de alta qualidade dentro de prazos e orçamentos previsíveis. Derivado dos trabalhos sobre *Unified Modeling Language* (UML) e do Processo Unificado de Desenvolvimento de Software, ele traz elementos de todos os modelos genéricos de processo, apoia a iteração e ilustra boas práticas de especificação e projeto (SOMMERVILLE, 2015).

O RUP descreve um conjunto de artefatos para a engenharia de requisitos que captura e apresenta informações usadas para definir os recursos necessários do sistema, conforme descrito a seguir (CORPORATION, 2001):

- **Plano de gerenciamento de requisitos:** descreve a documentação de requisitos, os tipos de requisitos e seus respectivos atributos de requisitos, especificando as informações e os mecanismos de controle que devem ser coletados e usados para avaliar, relatar e controlar mudanças nos requisitos do produto.
- **Glossário:** define os termos importantes usados no projeto.
- **Visão:** define a visão que os envolvidos têm do produto a ser desenvolvido, em termos das necessidades e características mais importantes.
- **Especificação de requisitos de software:** descreve o sistema com uma visão geral, como também os seus requisitos funcionais e não-funcionais. Fornece aos desenvolvedores as informações necessárias para o projeto e implementação, assim como para a realização dos testes e a implantação do sistema.
- **Documento de caso de uso:** descreve cada caso de uso, deve conter pré-condições e pós-condições, fluxos de eventos, prioridades e relacionamento com outros casos de uso.
- **Modelo de caso de uso:** modelo das funções pretendidas do sistema e seu ambiente; serve como um contrato estabelecido entre o cliente e os desenvolvedores. É usado como fonte de informações para atividades de análise, design e teste.
- **Protótipo da interface do usuário:** representa um protótipo do sistema, podendo ser descrito através de esboço em papel ou figuras, *bitmaps* feitos com uma ferramenta de desenho ou protótipo de executável interativo.

2.2 Metodologia Ágil

Em fevereiro de 2001, um grupo com 17 membros da comunidade de software se reuniram e definiram oficialmente o Desenvolvimento Ágil de Software, criando o manifesto ágil, com o objetivo de propor melhores maneiras de desenvolver softwares. O manifesto ágil afirma valorizar "os indivíduos e interações sobre processos e ferramentas, software funcionando mais que documentação abrangente, colaboração do cliente sobre negociação de contratos, e responder as mudanças seguindo um plano"(SCHWABER; BEEDLE, 2001).

Com intenção de facilitar melhor o entendimento sobre o desenvolvimento ágil de software, os membros da *Agile Alliance* formularam o Manifesto Ágil, criando doze princípios que as metodologias ágeis devem seguir. Estes princípios são:

1. Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor.
2. Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas.
3. Entregar software funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos.
4. Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto.
5. Construir projetos ao redor de indivíduos motivados, dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho.
6. O método mais eficiente e eficaz de transmitir informações para um time de desenvolvimento é através de uma conversa cara a cara.
7. Software funcional é a medida primária de progresso.
8. Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente passos constantes.
9. Contínua atenção à excelência técnica e bom design, aumenta a agilidade.
10. Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.
11. As melhores arquiteturas e requisitos emergem de times auto-organizáveis.
12. Em intervalos regulares, o time reflete em como ficar mais efetivo, então, se ajustam e otimizam seu comportamento.

2.2.1 SCRUM

O Scrum é um método de desenvolvimento ágil de software concebido por Jeff Sutherland e sua equipe de desenvolvimento no início dos anos 1990. O Scrum é um processo para projetos que realizam desenvolvimento de software que seja focado

nas pessoas, no qual o ambiente tenha uma mudança constante nos requisitos, podendo ser também considerado uma técnica utilizada para realizar o gerenciamento de processo para o desenvolvimento de software, além de propor entregar produtos no prazo correto, com alto grau de qualidade e visando satisfação do cliente (SCHWABER; BEEDLE, 2001).

Scrum é fundamentado nas teorias empíricas de controle de processo, ou empirismo. O empirismo afirma que o conhecimento vem da experiência e de tomada de decisões baseadas no que é conhecido. O Scrum emprega uma abordagem iterativa e incremental para aperfeiçoar a previsibilidade e o controle de riscos (SCHWABER K.; SURTHERLAND, 2015).

Três pilares apoiam a implementação de controle de processo empírico: (i) transparência, (ii) inspeção e (iii) adaptação (SCHWABER K.; SURTHERLAND, 2015).

Transparência: São aspectos significativos do processo que devem estar visíveis aos responsáveis pelos resultados. Esta transparência requer aspectos definidos por um padrão comum para que os observadores compartilhem um mesmo entendimento do que está sendo visto.

Inspeção: Os usuários Scrum devem, frequentemente, inspecionar os artefatos Scrum e o progresso em direção a detectar variações. Esta inspeção não deve, no entanto, ser tão frequente que atrapalhe a própria execução das tarefas. As inspeções são mais benéficas quando realizadas de forma diligente por inspetores especializados no trabalho a se verificar.

Adaptação: Se um inspetor determina que um ou mais aspectos de um processo desviou para fora dos limites aceitáveis, e que o produto resultado será inaceitável, o processo ou o material sendo produzido deve ser ajustado. O ajuste deve ser realizado o mais breve possível para minimizar mais desvios.

O time Scrum é composto pelo *Product Owner*, o time de desenvolvimento e o *Scrum Master*. O time Scrum é auto-organizável, escolhe qual a melhor forma para completar seu trabalho, em vez de ser dirigido por outros de fora do time. Também é multifuncional, possuem todas as competências necessárias para completar o trabalho sem depender de outros que não fazem parte da equipe (SCHWABER K.; SURTHERLAND, 2015).

O *Product Owner*, ou dono do produto, é o responsável por maximizar o valor do produto e do trabalho do time de desenvolvimento. O *Product Owner* é a única pessoa responsável por gerenciar o *Backlog* do Produto. Ele tem o dever de expressar claramente os itens do *Backlog* do Produto, ordenar os itens do *Backlog* do Produto para alcançar melhor as metas e missões, garantir o valor do trabalho realizado pelo time de desenvolvimento, garantir que o *Backlog* do Produto seja visível,

transparente, claro para todos, mostrar o que o time Scrum vai trabalhar e garantir que o time de desenvolvimento entenda os itens do *Backlog* do Produto (SCHWABER K.; SURTHERLAND, 2015).

O *Scrum Master* é responsável por garantir que o Scrum seja entendido e aplicado. O *Scrum Master* faz isso para garantir a aderência à teoria, práticas e regras pelo time Scrum. O *Scrum Master* é um servo-líder para o time Scrum. O *Scrum Master* ajuda aqueles que estão fora do time Scrum a entender quais as interações com o time Scrum são úteis e quais não são. O *Scrum Master* ajuda todos a mudarem estas interações para maximizar o valor criado pelo time Scrum (SCHWABER K.; SURTHERLAND, 2015).

O time de desenvolvimento consiste de profissionais que realizam o trabalho de entregar uma versão usável que potencialmente incrementa o produto ao final de cada *Sprint*. Times de desenvolvimento são estruturados e autorizados pela organização para organizar e gerenciar seu próprio trabalho. O tamanho da equipe costuma ter entre 8 a 10 integrantes, podendo variar dependendo do ambiente em que se aplica (SCHWABER; BEEDLE, 2001).

2.2.1.1 Etapas da Sprint

Com a caracterização do *Product Owner*, *Scrum Master* e time Scrum realizada na seção 2.2.1 “SCRUM”, é necessário também conhecer o processo de projeto do Scrum, tais como: *Sprint*, *Sprint Planning Meeting*, *Daily Scrum*, *Sprint Review* e *Sprint Review Meeting* (SCHWABER; BEEDLE, 2001).

De acordo com Ken Schwaber (Schwaber and Beedle 2001), a *Sprint* é definida como um ciclo de desenvolvimento curto em que o time foca em atingir o objetivo proposto pelo *Product Owner*. Durante este período o time deve ter total autoridade para seu gerenciamento, não devendo sofrer interferências externas do *Product Owner* ou mesmo de outras pessoas. A *Sprint* deve ter um tempo de execução de duas a quatro semanas. Esse tempo tem como objetivo fazer com que as funcionalidades sejam entregues e validadas pelo *Product Owner* (SCHWABER; BEEDLE, 2001).

A *Sprint Planning Meeting* é a reunião realizada na qual devem estar presentes o *Product Owner*, o *Scrum Master* e todo a equipe do Scrum. O *Product Owner* debate o objetivo que a *Sprint* deve realizar e os itens de *Backlog* do Produto que, se completados na *Sprint*, atingirão o objetivo da *Sprint*. Todo o time Scrum colabora com o entendimento do trabalho da *Sprint* (SCHWABER; BEEDLE, 2001).

A entrada da reunião de planejamento da *Sprint* é o *Backlog* do Produto, o mais recente incremento do produto, a capacidade projetada do time de desenvolvimento durante a *Sprint* e o desempenho do time de desenvolvimento. O número de

itens selecionados do *Backlog* do Produto para a *Sprint* é o único trabalho do time de desenvolvimento. Somente o time de desenvolvimento pode avaliar o que pode ser completado ao longo da próxima *Sprint* (SCHWABER K.; SURTHERLAND, 2015).

Após o time de desenvolvimento prever os itens de *Backlog* do Produto que irá entregar na *Sprint*, o time Scrum determina a meta da *Sprint*, ou seja, o objetivo que será conhecido dentro da *Sprint* através da implementação do *Backlog* do Produto, fornecendo a orientação para o time de desenvolvimento sobre o porquê dele estar construindo o incremento (SCHWABER K.; SURTHERLAND, 2015).

Daily Scrum é uma reunião diária que acontece precisamente no tempo 15 minutos, para que o time de desenvolvimento possa sincronizar as atividades e criar um plano para as próximas 24 horas. Esta reunião é feita para inspecionar o trabalho desde a última Reunião Diária e prever o trabalho que deverá ser feito antes da próxima Reunião Diária. O *Scrum Master* assegura que o time de desenvolvimento tenha a reunião, mas o time de desenvolvimento é responsável por conduzir a Reunião Diária (SCHWABER K.; SURTHERLAND, 2015).

Durante a reunião, os membros do time de desenvolvimento esclarecem:

- O que eu fiz ontem que ajudou o time de desenvolvimento a atender a meta da *Sprint*?
- O que eu farei hoje para ajudar o time de desenvolvimento atender a meta da *Sprint*?
- Eu vejo algum obstáculo que impeça a mim ou o time de desenvolvimento no atendimento da meta da *Sprint*?

A Revisão da *Sprint* é executada no final da *Sprint* para inspecionar o incremento e adaptar o *Backlog* do Produto se necessário. Durante a reunião de Revisão da *Sprint*, o time Scrum e as partes interessadas colaboram sobre o que foi feito na *Sprint*. Com base nisso e em qualquer mudança no *Backlog* do Produto durante a Sprint, os participantes colaboram nas próximas coisas que podem ser feitas para otimizar valor. Esta é uma reunião informal e a apresentação do incremento destina-se a motivar e obter comentários e promover a colaboração (SCHWABER K.; SURTHERLAND, 2015).

A Retrospectiva da *Sprint* ocorre depois da Revisão da *Sprint* e antes da reunião de planejamento da próxima *Sprint*. Esta é uma reunião com tempo de três horas para uma *Sprint* de um mês. Para *Sprint* menores, este evento é usualmente menor. O *Scrum Master* garante que o evento ocorra e que os participantes entendam seu propósito. O *Scrum Master* participa da reunião como um membro auxiliar do time de-

vido a sua responsabilidade pelo processo Scrum (SCHWABER K.; SURTHERLAND, 2015).

O *Scrum Master* encoraja o time Scrum a melhorar o processo de desenvolvimento e as práticas para fazê-lo mais efetivo e agradável para a próxima *Sprint* (SCHWABER K.; SURTHERLAND, 2015).

Ao final da Retrospectiva da *Sprint*, o time Scrum deverá ter identificado melhorias que serão implementadas na próxima *Sprint*. A implementação destas melhorias na próxima *Sprint* é a forma de adaptação à inspeção que o time Scrum faz a si próprio. A Retrospectiva da *Sprint* fornece um evento dedicado e focado na inspeção e adaptação, no entanto, as melhorias podem ser adotadas a qualquer momento (SCHWABER K.; SURTHERLAND, 2015).

2.2.1.2 Artefatos

Os artefatos do Scrum representam o trabalho ou o valor para o fornecimento de transparência e oportunidades para inspeção e adaptação. Os artefatos definidos para o Scrum são especificamente projetados para maximizar a transparência das informações chave de modo que todos tenham o mesmo entendimento dos artefatos. Podem ser representados pelo *Backlog* do Produto, *Backlog* da *Sprint* e interação/incremento (SCHWABER K.; SURTHERLAND, 2015).

O *Backlog* do Produto é uma lista ordenada de tudo que deve ser necessário no produto e é uma origem única dos requisitos para qualquer mudança a ser feita no produto. Os itens do *Backlog* do Produto de ordem mais alta (topo da lista) devem ser mais claros e mais detalhados que os itens de ordem mais baixa. Estimativas mais precisas são feitas baseadas em maior clareza e maior detalhamento; quanto menor a ordem na lista, menos detalhes (SCHWABER K.; SURTHERLAND, 2015).

O *Backlog* da *Sprint* é um conjunto de itens do *Backlog* do Produto selecionados para a Sprint, juntamente com o plano para entregar o incremento do produto e atingir o objetivo da *Sprint*. O *Backlog* da *Sprint* é a previsão do time de desenvolvimento sobre qual funcionalidade estará no próximo incremento e sobre o trabalho necessário para entregar essa funcionalidade em um incremento (SCHWABER K.; SURTHERLAND, 2015).

O incremento é a soma de todos os itens do *Backlog* do Produto completados durante a *Sprint* e o valor dos incrementos de todas as *Sprints* anteriores. Ao final da *Sprint* um novo incremento deve estar concluído, o que significa que deve estar na condição utilizável e atender a definição do *Time Scrum* (SCHWABER K.; SURTHERLAND, 2015). A Figura 2 apresenta o ciclo de vida Scrum, onde é relacionado as etapas do Scrum e artefatos que foram gerados.

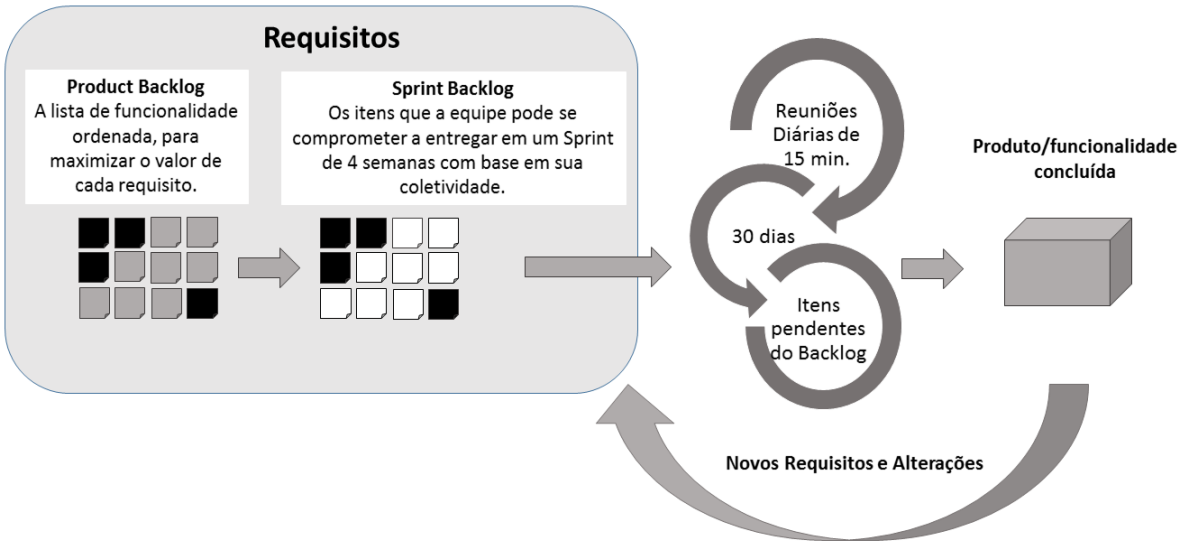

Figura 2: Ciclo de vida Scrum [Adaptado (SCHWABER; BEEDLE, 2001)].

2.2.2 Extreme Programming – XP

Na década de 1990, foi proposto um novo método ágil o Extreme Programming (XP) criado por Kent Beck, Ward Cunningham e sua equipe, tem como objetivo resolver problemas básicos no desenvolvimento de software, incluindo atrasos no cronograma, cancelamento antecipado do projeto, sistema em produção com alta taxa de erros, funcionalidades entregues sem atender as necessidades de negócio, sistema com baixo valor agregado e de difícil manutenção (BECK K., 2001).

BECK K. (2001) define Extreme Programming como sendo uma metodologia leve, eficiente e de baixo risco, com forma flexível, previsível, científica e apropriada para pequenas e médias equipes de desenvolvimento de software, onde os requisitos são vagos ou mudam rapidamente.

O XP foi concebido para trabalhar com projetos que podem ser construídos por equipes pequenas, constituídas por dois a dez programadores, e onde um trabalho razoável de execução de testes pode ser feito em pelo menos um dia.

Em XP, os requisitos são expressos como cenários (chamados de estórias de usuário), que são implementados diretamente como uma série de tarefas. Os programadores trabalham em pares e desenvolvem testes para cada tarefa antes de escreverem o código. Quando o novo código é integrado ao sistema, todos os testes devem ser executados com sucesso. Há um curto intervalo entre os *releases* do sistema (SOMMERVILLE, 2015).

2.2.2.1 Valores

BECK K. (2001) definiu um conjunto de cinco valores que estabelecem as bases para todo trabalho realizado como parte do XP — (i) comunicação, (ii) simplicidade, (iii) *feedback*, (iv) coragem e (v) respeito. Cada um desses valores é usado como um direcionador das atividades, ações e tarefas específicas do XP.

- **Comunicação:** A comunicação deve ocorrer de maneira contínua entre os próprios desenvolvedores. Problemas em projetos pode acontecer porque alguém deixou de comunicar algo a outra pessoa. XP tem por objetivo manter a comunicação fluindo de forma rápida e eficaz entre a equipe de desenvolvimento e o cliente para criar um senso de cooperação eficiente, sendo que o XP emprega muitas práticas que não podem ser feitas sem comunicação.
- **Simplicidade:** XP aposta no conceito de simplicidade onde fazer o mais simples e futuramente gastar um pouco mais para fazer mudanças é mais vantajoso que fazer algo mais complexo, mas que pode nunca ser utilizado.
- **Feedback:** O *feedback* contribui para o aprendizado no sentido de aprender e melhorar. Quanto mais rápido for o *feedback* do cliente, mais rápido será identificado se o projeto está seguindo na direção correta e, caso não esteja, o esforço para efetuar correções será menor.
- **Coragem:** Combinada com os três valores anteriores, a coragem se torna extremamente valiosa. É preciso coragem para inovar, para propor mudanças, para pedir ajuda quando necessário, para comunicar problemas ao cliente e encarar mudanças no projeto. Porém, em contrapartida a coragem utilizada de forma isolada, sem contrabalancear os demais valores é perigosa. Fazer algo sem levar em conta as suas consequências não é um exemplo de equipe de trabalho eficiente.
- **Respeito:** O respeito entre as pessoas é a parte fundamental para que os demais valores colaborem entre si. Respeitar os outros membros da equipe é essencial para o sucesso de um projeto de software, e somente com o respeito os outros valores poderão ser amplamente seguidos.

Em um processo XP, os clientes estão envolvidos na especificação e priorização dos requisitos do sistema. Os requisitos não estão especificados como uma lista de funções requeridas do sistema. Pelo contrário, o cliente do sistema é parte da equipe de desenvolvimento e discute cenários com outros membros da equipe (SOMMERVILLE, 2015).

2.2.2.2 As práticas do XP

O Extreme Programming emprega uma abordagem orientada a objetos como seu paradigma de desenvolvimento preferido e envolve um conjunto de regras e práticas constantes no contexto de quatro atividades metodológicas: planejamento, projeto, codificação e testes (PRESSMAN, 2011).

Na figura 3 é apresentada uma visão geral do processo das atividades do XP.

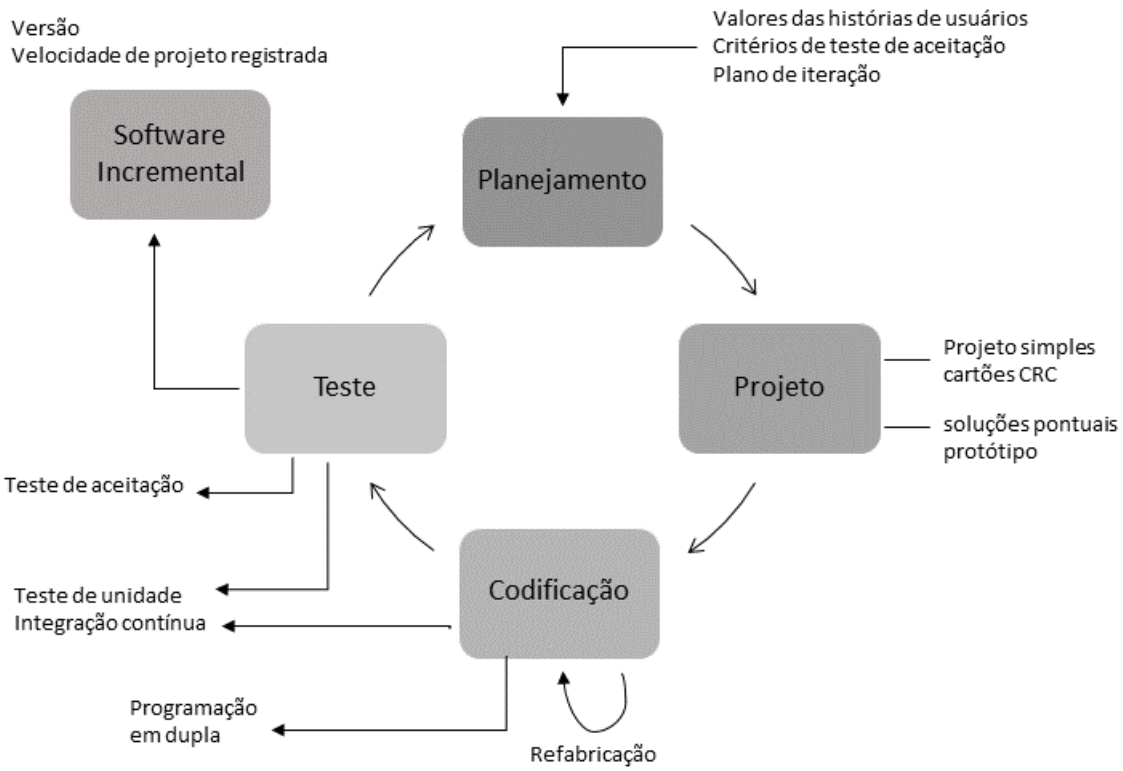

Figura 3: Visão geral do processo de desenvolvimento Extreme Programming [Fonte: (PRESSMAN, 2011)].

Planejamento: A atividade de planejamento se inicia com a atividade de ouvir, uma atividade de levantamento de requisitos que capacita os membros técnicos da equipe XP a entender o ambiente de negócios do software e possibilita que se consiga ter uma percepção ampla sobre os resultados solicitados, fatores principais e funcionalidade.

Cada história escrita pelos usuários é colocada em uma ficha, o cliente atribui um valor a essa ficha de acordo com o valor de negócio global do recurso ou função, em seguida os membros da equipe avaliam as fichas e atribuem um custo (medido em semanas para desenvolvimento), se esse custo for maior que 3 semanas, então é solicitado aos clientes que realizam a decomposição das histórias em partes menores, novas histórias podem ser escritas a qualquer momento.

Projeto: O projeto XP segue rigorosamente o princípio KIS (*keep it simple*, ou seja, preserve a simplicidade). É preferível sempre um projeto simples a uma representação mais complexa. Nessa fase é recomendável a utilização de cartões CRC (classe-responsabilidade-colaborador), identificam e organizam as classes orientadas a objetos relevantes para o incremento de software corrente.

Codificação: Depois de desenvolvidas as histórias e o trabalho preliminar de elaboração do projeto ter sido feito, a equipe não passa para a codificação, mas sim, desenvolve uma série de testes de unidades que exercitam cada uma das histórias a ser incluídas na versão corrente (incremento de software). Nessa fase é executada a programação em pares.

Testes: Os testes de unidade criados devem ser implementados usando-se uma metodologia que os capacite a ser automatizados. Isso encoraja uma estratégia de testes de regressão, toda vez em que o código for modificado. Os testes de aceitação (teste de cliente) são obtidos a partir das histórias dos usuários. Um projeto XP ideal contém 6 fases: (i) exploração, (ii) planejamento, (iii) iterações para as versões, (iv) produção, (v) manutenção e (vi) morte. Equipes de desenvolvimento que utilizam XP executam quase todas as atividades de desenvolvimento de software simultaneamente, incluindo análise, projeto, codificação, testes e implantação (BECK K., 2001).

As práticas apresentadas pelo método XP correspondem a uma estrutura genérica. Sua adoção integral pode não ser necessária para todos os projetos de software. Adotar integralmente tais técnicas pode acarretar em sobrecarga desnecessária ao processo de desenvolvimento. Para cada projeto e contexto do time de desenvolvimento, é importante que o engenheiro de software analise suas demandas e adapte os recursos oferecidos pelo método XP ao projeto que será desenvolvido, almejando um equilíbrio na relação custo/benefício, sendo necessário adaptar um processo de software maduro de forma apropriada aos produtos e às demandas do mercado (PRESSMAN, 2011).

CAPÍTULO 3

TRABALHOS RELACIONADOS

Neste capítulo são apresentados os trabalhos que estão relacionados com a ideia desta pesquisa. Na Seção 3.1 será apresentada a metodologia de busca utilizada para identificar os trabalhos relacionados; na Seção 3.2 são apresentados os trabalhos encontrados para especificação de casos de uso para linhas de produto de software; na Seção 3.3 apresenta alguns métodos da engenharia de requisitos que podem ser aplicados nas metodologias ágeis Scrum e XP.

3.1 Metodologia de Busca

Para encontrar os trabalhos relacionados foi realizada uma busca na literatura por publicações que estivessem relacionados a pelo menos um dos seguintes tópicos: i) Engenharia de requisitos em metodologias ágeis Scrum e XP, procurando identificar como o estudo desse modelo está se comportando para a área de desenvolvimento de software; ii) Identificação da prática/técnicas utilizadas para levantar o impacto de aplicação.

Portanto, em relação as buscas não foram executadas uma revisão sistemática (KITCHENHAM, 2004), alguns princípios desse processo de pesquisa bibliográfica, tais como a definição de strings de busca e locais de pesquisa, foram adotados neste trabalho. Dessa forma, para cada um dos dois tópicos foram definidas duas strings de busca, uma em inglês e outra em português. Como locais de buscas utilizou-se as bases ACM DL Library , o IEEE Explorer, ScienceDirect e no Google Scholar.

Nas seções a seguir são apresentados mais detalhes sobre os trabalhos relacionados identificados.

3.2 Engenharia de Requisitos em Metodologia Ágil

LUCIA A.; QUSEF (2010) e PAETSCH; EBERLEIN; MAURER (2003) discutem problemas relacionados com a utilização da engenharia de Requisitos em metodologias ágeis, identificando as técnicas e os processos da Engenharia de requisitos que podem ser combinados com práticas ágeis e encontrar soluções para algumas das

dificuldades relacionadas a engenharia de requisitos.

LUCIA A.; QUSEF (2010) cita estudo de viabilidade, elicitação, análise, documentação, validação e gestão de requisitos como atividades necessárias para realizar Engenharia de requisitos ágil. A metodologia ágil suporta a engenharia de requisitos de forma muito eficiente. O segredo do sucesso da Engenharia de requisitos ágil é a colaboração do cliente, bons desenvolvedores e gerentes de projeto experientes. Os autores fornecem algumas recomendações para (i) resolver os problemas de documentação em projetos ágeis, (ii) adequar a metodologia ágil a lidar com projetos com requisitos críticos, e (iii) permitir que equipes ágeis envolvidas em grandes projetos de software possam trabalhar em paralelo com comunicação frequente entre elas.

PAETSCH; EBERLEIN; MAURER (2003) apresentam que o processo de engenharia de requisitos especialmente as fases de elicitação, análise e validação estão presentes em todos os processos ágeis. As técnicas utilizadas variam nas diferentes abordagens e as fases não são tão claramente separadas.

Como todas as abordagens ágeis, incluir pelo menos um mínimo de documentação é de responsabilidade da equipe de desenvolvimento, garantindo a existência de documentação suficiente disponível para manutenção futura. No geral, metodologias ágeis e engenharia de requisitos estão perseguindo objetivos semelhantes. A principal diferença é a ênfase na quantidade de documentação necessária em um projeto eficaz (PAETSCH; EBERLEIN; MAURER, 2003).

Muitos esforços têm sido empenhados em pesquisas que discutem a relação entre Engenharia de Requisitos com Metodologia Ágil. Para PAETSCH; EBERLEIN; MAURER (2003) é necessário saber o que construir antes de começar o desenvolvimento do sistema, a fim de evitar um grande retrabalho. (SCHWABER; BEEDLE, 2001) alertam que quanto mais tarde os erros são descobertos, mais caro será para corrigi-los .

A Tabela 2 expõe como ocorre a atividade de Engenharia de Requisitos na implementação do Scrum, de forma que os papéis e eventos do Scrum são empregados em cada fase da Engenharia de uma maneira ágil.

Tabela 2: Aplicação da Engenharia de requisitos no Scrum [Fonte: (LUCIA A.; QUSEF, 2010)]

ATIVIDADE DA ENGENHARIA DE REQUISITO	**SCRUM**
Elicitação de requisitos	• Product Owner formula o Product Backlog. • Todas as partes interessadas podem participar no Product Backlog.
Análise de requisitos	• Reunião de refinamento do Backlog. • Product Owner prioriza o Product Backlog. • Product Owner analisa a viabilidade de requisitos.
Documentação de requisitos	• Comunicação face a face
Validação de requisitos	• Reuniões de revisão

LUCIA A.; QUSEF (2010) procuram estabelecer o trabalho da Engenharia de Requisitos do ponto de vista do desenvolvimento ágil, apresentando atividades de Engenharia de requisitos na implementação Scrum e algumas recomendações para ajudar a gestão e execução das equipes de desenvolvimento. Os autores salientam que a Engenharia de Requisitos está preocupada com a descoberta, análise, especificação e documentação dos requisitos do sistema e merece o maior cuidado, pois os problemas inseridos no sistema durante a fase de Engenharia de requisitos são os mais caros para remover.

A Tabela 3 expõe como ocorre a atividade de Engenharia de Requisitos na implementação do XP, de forma que as práticas são empregados em cada fase de uma maneira ágil.

Tabela 3: Aplicação da Engenharia de requisitos no XP [Fonte: (LUCIA A.; QUSEF, 2010)]

ATIVIDADE DA ENGENHARIA DE REQUISITO	**XP**
Elicitação de requisitos	• Os clientes escrevem as histórias do usuário. • Requisitos como histórias.
Análise de requisitos	• Não existe fase separada. • Análise durante o desenvolvimento. • Cliente prioriza as histórias do usuário.
Documentação de requisitos	• Histórias de usuários e testes de aceitação, como documentos de requisitos. • Comunicação face a face.
Validação de requisitos	• Test Driven Development (TDD). • Executar testes de aceitação. • Feedback frequente.

MUSHTAQ (2012), propõe um modelo híbrido com as metodologias Scrum e XP. Desse modo, o Scrum favorece em paradigma enriquecido de gerenciamento de projetos e o XP aplicando as práticas durante o planejamento, criação, codificação, testes e fases de integração.

SILLITTI; SUCCI (2005), afirmam que as necessidades dos requisitos podem evoluir com o tempo, o cliente pode mudar de opinião ou o ambiente técnico e socioeconômico global pode evoluir. Todavia, a metodologia ágil requer um alto nível de interação entre clientes, gerentes e desenvolvedores. Papéis e responsabilidades dos clientes, gerentes e desenvolvedores são de suma importância e tem um amplo impacto sobre a evolução de um software projeto.

BJARNASON; WNUK; REGNELL (2011), relatam um estudo de caso com nove profissionais de uma grande empresa de desenvolvimento de software, com objetivo de investigar como a Engenharia Requisitos Ágil pode resolver os desafios da Engenharia Requisitos tradicional e quais os novos desafios que a engenharia de requisitos ágil pode representar. A empresa estudada estava em processo de transição do processo de engenharia de requisitos tradicional para a engenharia de requisitos ágil.

Nesse processo de transição, cinco práticas ágeis relacionadas à engenharia de requisitos foram introduzidas na empresa:

- Histórias de usuário.
- Critérios de aceitação das histórias de usuário.
- Detalhamento de requisitos iterativo.
- Equipes de desenvolvimento multifuncionais.
- Escopo contínuo.

Os resultados mostram que as práticas ágeis enfrentam alguns desafios da engenharia de requisitos como falhas de comunicação e escopo, mas também fazem com que os novos desafios, como um bom equilíbrio entre agilidade e estabilidade, e garantir a competência suficiente em equipes de desenvolvimento multifuncionais.

A Tabela 4 mostra uma comparação entre os trabalhos relacionados e o trabalho proposto. Observa-se que um diferencial deste trabalho é o estudo de caso que apresenta uma comparação da engenharia de requisitos em diferentes metodologias ágeis.

Tabela 4: Comparativo entre os Trabalhos Relacionados e o Trabalho Proposto.

	(EBERLEIN A., 2002)	(LUCIA A.; QUSEF, 2010)	(BJARNASON; WNUK; REGNELL, 2011)	(MUSHTAQ, 2012)	Trabalho Proposto
Estudo Realizado em empresas			X		X
Aplicação Scrum	X	X		X	X
Aplicação Scrum/XP (Híbrido)				X	X
Estudo de técnicas da Engenharia de Requisitos ágil	X	X			X
Análise comparativa					X
Apresentação de boas práticas		X	X		X
Apresentação do desafios			X		X

3.3 Técnicas da Engenharia de Requisitos em Metodologias ágeis

A seguir são apresentadas técnicas relacionadas aos requisitos, o processo de identificação das técnicas foram obtidas a partir de uma revisão onde foi detectado as técnicas da engenharia de requisitos que estão sendo utilizadas em comum, como por exemplo como entrevistas, casos de uso/cenários *brainstorming*, prototipagem, entre outros.

PAETSCH; EBERLEIN; MAURER (2003) e LUCIA A.; QUSEF (2010) descrevem que o processo de engenharia de requisitos consiste em quatro atividades principais: elicitação, análise e negociação, documentação e validação. As técnicas utilizadas nessas atividades variam nas diferentes abordagens.

Elicitação de Requisitos

Entrevistas: "Entrevistar é um método para descobrir fatos e opiniões detidos por potenciais interessados do sistema em desenvolvimento"(SCHWABER; BEEDLE, 2001). Existem dois tipos de entrevistas: (i) entrevistas fechadas, onde perguntas predefinidas são respondidas, e (ii) entrevistas abertas, onde não há nenhuma agenda predefinida e uma série de questões são exploradas com as partes interessadas. Na verdade, as entrevistas são boas para a obtenção de um entendimento geral do que os *stakeholders* fazem e como eles podem interagir com o sistema, mas elas não são boas para o entendimento do domínio dos requisitos. As metodologias ágeis afirmam que as entrevistas são uma maneira eficiente de se comunicar com os clientes e de aumentar a confiança entre os dois lados (LUCIA A.; QUSEF, 2010).

Casos de uso / Cenários: Casos de uso descrevem as interações entre usuários e sistema, incidindo sobre o que os usuários precisam fazer com o sistema. Um caso de uso especifica uma sequência de interação entre o sistema e um agente externo (por exemplo, uma pessoa, uma peça de hardware, software de outro produto), incluindo variantes e extensões, que o sistema pode realizar. Os casos de uso representam requisitos funcionais do sistema de software e podem ser utilizados durante as fases iniciais do processo de desenvolvimento. Os analistas e os clientes devem examinar cada caso de uso proposto para validá-lo (PAETSCH; EBERLEIN; MAURER, 2003).

Esta é uma técnica baseada em cenário, ou seja, exemplos de interação em que um único tipo de interação entre o usuário e o sistema é simulado. Cenários devem incluir uma descrição do estado do sistema antes de entrar e após a conclusão do cenário, quais as atividades podem ser simultâneas, o fluxo normal de eventos e exceções aos acontecimentos (PAETSCH; EBERLEIN; MAURER, 2003).

Brainstorming É uma técnica de grupo para gerar novas ideias úteis e promover o pensamento criativo. *Brainstorming* pode ser usado para obter novas ideias e recursos para a aplicação, definir o projeto ou problema para trabalhar e para diagnosticar problemas em um curto espaço de tempo. O gerente de projeto desempenha um papel importante, determinando o tempo da sessão, certificando-se de que não haja discussões crescentes sobre determinados temas, e de que cada corpo expressa sua opinião livremente. Após a finalização da sessão, os tópicos são avaliados pela equipe. Além disso, as conexões e dependências entre as ideias discutidas são representados através da visualização de gráficos, por exemplo, de modo que os conflitos com outros requisitos são encontrados e avaliados (LUCIA A.; QUSEF, 2010).

Análise de Requisitos:

***Joint Application Development* (JAD)** É uma oficina usada para coletar os

requisitos de negócios durante o desenvolvimento de um sistema. As sessões JAD também incluem abordagens para aumentar a participação do usuário, acelerar o desenvolvimento e melhoria da qualidade de especificações (LUCIA A.; QUSEF, 2010).

No ambiente ágil, em caso de conflitos entre os requisitos das partes interessadas, o uso de JAD pode ajudar a promover o uso de um facilitador profissional que pode ajudar a resolver conflitos. Além disso, as sessões JAD incentivam o envolvimento do cliente e a confiança no sistema desenvolvido (LUCIA A.; QUSEF, 2010).

Modelagem Modelos do sistema são importantes para análise e o processo de design (PAETSCH; EBERLEIN; MAURER, 2003). Em ambiente ágil, a modelagem é dividida em três seções: (i) modelos a serem implementados, (ii) modelos em fase de implementação, e (ii) modelos concluídos. Este esquema deve ser documentados e fornece uma representação visual do *status* do projeto (SILLITTI; SUCCI, 2005).

Priorização As metodologias ágeis especificam que os requisitos devem ser organizados pela lista de priorização de cada funcionalidade. Os recursos são priorizados pelos clientes com base no seu valor de negócio, de modo que as equipes ágeis estimam o tempo necessário para implementar cada requisito. Essa abordagem ajuda a identificar os recursos mais importantes dentro do projeto em andamento. Normalmente, se um requisito é muito importante é programado para a implementação na próxima iteração, caso contrário, é mantido em espera. Na iteração seguinte, são avaliados os requisitos em espera e, se ainda estiverem válidos, são incluídos na lista de requisitos candidatos juntamente com os novos. Então, a nova lista é priorizada para identificar os recursos que serão implementados, se um requisito não é suficientemente importante, ele é mantido em espera indefinidamente (SILLITTI; SUCCI, 2005).

Validação de Requisitos:

Revisão de Requisitos Trata-se de um processo manual que envolve múltiplos leitores, tanto da equipe ágil quanto das partes interessadas, verificando os requisitos de acordo com os padrões organizacionais para anomalias e omissões. Em projetos ágeis, as revisões de requisitos devem ser avaliações formais: queremos dizer que a equipe ágil deve caminhar com os clientes através de cada requisito. Conflitos, erros, extra e omissões nos requisitos devem ser formalmente registrados (LUCIA A.; QUSEF, 2010).

Teste de Unidade Na metodologia ágil XP, os requisitos são implementados e testados usando a técnica *Test Driven Development* (TDD). Ao aplicar esta técnica, os desenvolvedores criam testes antes de escrever código. O código desenvolvido é então refatorado para melhorar sua estrutura. A regra aqui é escrever um código se e somente se um teste falhar. Esta técnica tem algumas vantagens. É a maior

vantagem para definir casos de teste que testar sua exigência com muita precisão. O requisito do qual o caso de teste foi criado é agora apresentado sob uma forma em que é completamente validado, no sentido de que pode ser automaticamente (após cada iteração) determinado se um requisito é implementado pelo software ou não. Isso torna os desenvolvedores cientes para o progresso do projeto e o estado da iteração atual do projeto. Além disso, suporta o processo de refatoração para obter um design melhorado por acoplamento reduzido e forte coesão.

O TDD contém pequenas iterações que fornecem *feedback* rápido. Refatoração de código e testes de unidade garantem que o código emergente é mais simples e legível. De fato, os testes unitários podem ser considerados uma documentação ao vivo e atualizada: eles representam um repositório excelente para os desenvolvedores que tentam entender o sistema, uma vez que eles mostram como partes de um sistema são executadas (LUCIA A.; QUSEF, 2010).

Prototipagem Um protótipo de um sistema é uma versão inicial do sistema, que está disponível no início do processo de desenvolvimento. Protótipos de sistemas de software são muitas vezes utilizados para ajudar a obter e validar requisitos do sistema. Existem dois diferentes tipos de protótipos: (i) o protótipo descartável, que ajuda a descobrir requisitos que causam dificuldades de compreensão, e (ii) o protótipo evolutivo, que oferece um sistema usável para o cliente e pode se tornar uma parte do sistema final (PAETSCH; EBERLEIN; MAURER, 2003).

Teste de aceitação O teste de aceitação é um teste formal conduzido pelo cliente para que o sistema satisfaz os critérios de aceitação contratuais. Os testes de aceitação não são diferentes dos testes automatizados do sistema, mas são realizados pelo cliente. Os clientes criam critérios de aceitação para os requisitos e testam os requisitos em relação a esses critérios. Os clientes podem dar *feedback* para o Desenvolvedores para melhorar o desenvolvimento de incrementos futuros do sistema (LUCIA A.; QUSEF, 2010).

No processo de levantamento de requisitos, nem sempre irá existir uma técnica padrão para alcançar um levantamento de requisitos desejado. Sendo, portanto, necessário conhecer diversas técnicas para saber qual a melhor a ser aplicada em cada situação.

CAPÍTULO 4

PROPOSTA

Na engenharia de requisito tradicional a atividade de obtenção de requisitos é uma preocupação constante dos desenvolvedores de software, pois muitos problemas que acontecem na fase final do software é resultado do mau uso de técnicas realizadas nas fases da engenharia de requisitos, sendo que a correção desses problemas se torna mais complexa e cara uma vez que se avança o processo de desenvolvimento.

No desenvolvimento ágil o envolvimento do cliente tem um maior valor, sendo responsável por participar junto com o time de desenvolvimento promovendo comunicação aumentado o nível do requisito e diminuindo as mudanças.

Desta forma, a motivação deste trabalho é estudar as técnicas da engenharia de requisitos em projetos que utilizam modelo de processo Scrum e Scrum/XP (Híbrido), pois ainda existe a necessidade de oferecer uma maior atenção aos requisitos, sendo estes essenciais para alcançar o sucesso do desenvolvimento de software.

Este capítulo apresenta um estudo de caso com empresas de desenvolvimento de software que utilizam modelo de processo Scrum e Scrum/XP. A seção 4.1 apresenta a avaliação experimental da realização do estudo de caso em maiores detalhes; a seção 4.2 encontra-se o detalhamento das empresas participantes do estudo de caso; a seção 4.3 apresenta os artefatos de entrada, por fim, a seção 4.5 esclarece como foi realizado a coleta de dados através da aplicação do questionário.

4.1 Visão Geral

Aplicamos um estudo de caso em empresas de desenvolvimento de software com o intuito de realizar uma análise comparativa sobre a engenharia de requisitos em projetos com metodologias ágeis Scrum e Scrum/XP(Híbrido), para entender a utilização da Engenharia de Requisitos nas metodologias ágeis, observando as vantagens e desvantagens, além do que há em comum entre as atividades. Ainda é importante salientar que o objetivo não é estudar a engenharia de requisitos tradicional, mas sim a engenharia de requisitos em metodologia ágil.

O estudo de caso analisa critérios associados à fase da atividade da engenharia de requisitos, tais como: elicitação, análise, validação e documentação em

comparação as metodologias ágeis Scrum e Scrum/XP, com o objetivo de conhecer as técnicas que são utilizadas no desenvolvimento de software ágil e de analisar as etapas utilizadas na sprint e os artefatos gerados em cada atividade.

4.2 Participantes

O questionário foi aplicado em cinco empresas reais de desenvolvimento de software, cujos produtos são direcionados às seguintes áreas: escritório/negócios, científico, internet, mobile, mercado financeiro, desenvolvimento em sistema bancário e governo. A Figura 4 apresenta a distribuição dos métodos nas empresas estudadas, onde 60% usam metodologia ágil Scrum e 40% usam metodologia ágil Scrum/XP (Híbrido).

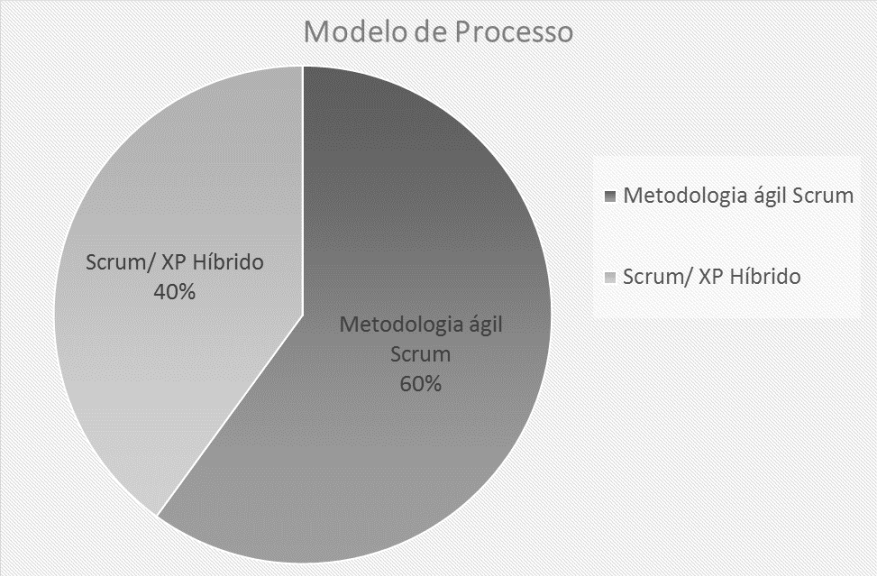

Figura 4: Modelo de processo das empresas estudadas.

Apresentamos a seguir uma breve descrição de cada uma das empresas participantes desta pesquisa, descrevendo o modelo de processo utilizado e a quantidade de pessoas alocadas nas atividades de engenharia de software, incluindo todos os processos, desde a implementação até a implantação. Todas as empresas pesquisadas estão localizadas em Fortaleza - Ceará, as empresas possuem tamanhos variados de até 5 a mais de 120 funcionários. O nome das empresas é mantido em sigilo e, com isso, apresentamos apenas um literal para distinguir cada uma das empresas participantes. Escolhemos essas empresas por possuir portes diferentes, atuarem em mercado diferente, para que possa avaliar como é que requisitos atuam nas empresas.

A empresa X com tamanho variando de 51 a 100 pessoas alocadas em atividades de engenharia de software, utiliza Metodologia ágil Scrum como modelo de processo nos projetos.

A empresa Z com tamanho variado de até 5 pessoas alocadas em atividades de engenharia de software utiliza Metodologia ágil Scrum como modelo de processo nos projetos.

A empresa V com tamanho de mais de 100 pessoas alocadas em atividades de engenharia de software, utiliza Metodologia ágil Scrum como modelo de processo nos projetos.

A empresa Y com tamanho variando entre 21 a 50 pessoas alocadas em atividades de engenharia de software, utiliza Metodologia ágil Scrum/XP (Híbrido) como modelo de processo nos projetos.

A empresa W com tamanho variando de 21 a 50 pessoas alocadas em atividades de engenharia de software, utiliza Metodologia ágil Scrum/XP (Híbrido) como modelo de processo nos projetos.

4.3 Artefatos de Entradas

Para o estudo de caso utilizamos um questionário online (disponibilizado no apêndice deste trabalho), garantindo o anonimato da empresa, além de economizar tempo de deslocamento e por conceder flexibilidade quanto ao horário de resposta pelas empresas.

A avaliação apresentada pelo questionário foi realizada levando em consideração os seguintes aspectos: (i) tamanho do time de desenvolvimento de software (ii) processos utilizados nas fases da engenharia de requisitos ágil no desenvolvimento de software, (iii) técnicas adotadas na engenharia de requisitos do ponto de vista ágil, (iv) artefatos de requisitos produzidos no projeto de desenvolvimento de software, (v) Quais as vantagens e desvantagens da engenharia de requisitos para a metodologia ágil e (vi) Quais são os maiores desafios da aplicação da engenharia de requisitos com metodologia ágil.

A avaliação desses critérios nos permite conhecer os pontos de dificuldade, descobrir problemas e apresentar possíveis melhorias. Com auxílio do questionário ainda foi possível identificar os principais desafios enfrentados pelas empresas no desenvolvimento de software, além de recomendar uma série de boas práticas para estimular as atividades de desenvolvimento.

4.4 Metodologia

A metodologia de pesquisa foi realizada em empresas através de um questionário, sendo este o método de coleta de dados, o questionário é quantitativo, mas

há algumas questões que serão analisadas de forma qualitativa este é um trabalho exploratório junto às empresas do estado do Ceará.

A metodologia de pesquisa busca avaliar como os requisitos atuam em empresas que utilizam metodologia ágil, realizando a análise de quais são as técnicas utilizadas em cada uma das empresas, em cada um dos processos da engenharia de requisitos e a partir identificar quais são as atividades que são realizadas em comum e as técnicas que são utilizadas em determinada metodologia.

4.5 Coleta de Dados

O estudo de caso foi aplicado separadamente com cada empresa, utilizando a ferramenta de pesquisa online gratuita Google Forms [1]. O tempo de resposta ocorreu em um prazo máximo de duas semanas. Após a aplicação do questionário, as respostas foram coletadas para serem analisadas e apresentadas no capítulo a seguir.

[1] Ferramenta online Google Form: https://www.google.com/forms/about/

CAPÍTULO 5

RESULTADOS E DISCUSSÃO

Este capítulo apresenta o resultado da análise comparativa realizada no estudo de caso que avalia a Engenharia de Requisitos em Metodologias Ágeis Scrum e Scrum/XP (Híbrido), para demonstrar a relação entre os processos da Engenharia de Requisitos em diferentes metodologias ágeis. A seção 5.1 descreve a análise comparativa entre os processos da engenharia de requisitos em metodologias ágeis, visando esclarecer os objetivos dos experimentos. A seção 5.2 apresenta os desafios a serem superados entre a engenharia de requisitos em metodologia ágeis. A seção 5.3 caracteriza as boas práticas a serem seguidas no desenvolvimento do software.

5.1 Análise Comparativa entre a Engenharia de Requisitos em Metodologias Ágeis

Na engenharia de requisitos, o requisito representa as descrições do cliente para a construção de um sistema, com o intuito de atender e solucionar um problema. Requisitos são a base de todos os produtos de software e sua elicitação, gerenciamento e entendimento são problemas comuns a todas as metodologias de desenvolvimento de software (AURUM; WOHLIN, 2005).

As empresas estudadas utilizam os seguintes processos de engenharia de requisitos: Elicitação de requisitos, Análise de requisitos, Documentação de requisitos e Validação de requisitos. A Figura 5 apresenta os processos que são importantes na engenharia de requisitos, ordenadas por relevância, conforme respostas das empresas analisadas. É possível visualizar que para as empresas o processo de Análise de requisitos tem um maior valor de relevância, seguido de Elicitação de requisitos, Validação de requisitos e Documentação de requisitos.

Analisando o gráfico, a análise de requisitos tem 100%, isso se deve ao ambiente de desenvolvimento ter a análise de requisitos como uma forma de descobrir possíveis problemas, ambiguidades, inconsistência nos requisitos que acabaram de ser elicitados através dos *user stories*. Portanto, a análise auxilia no andamento de todas as fases posteriores do projeto, permitindo assim, que o seu cliente fique satisfeito e o projeto obtenha o sucesso.

Na documentação de requisitos tem 40%, isso se deve ao fato de que metodologia ágil segue o princípio de software funcionando ao invés de documentação abrangente, deste modo utiliza a documentação mínima sendo suficiente para validar cada requisito.

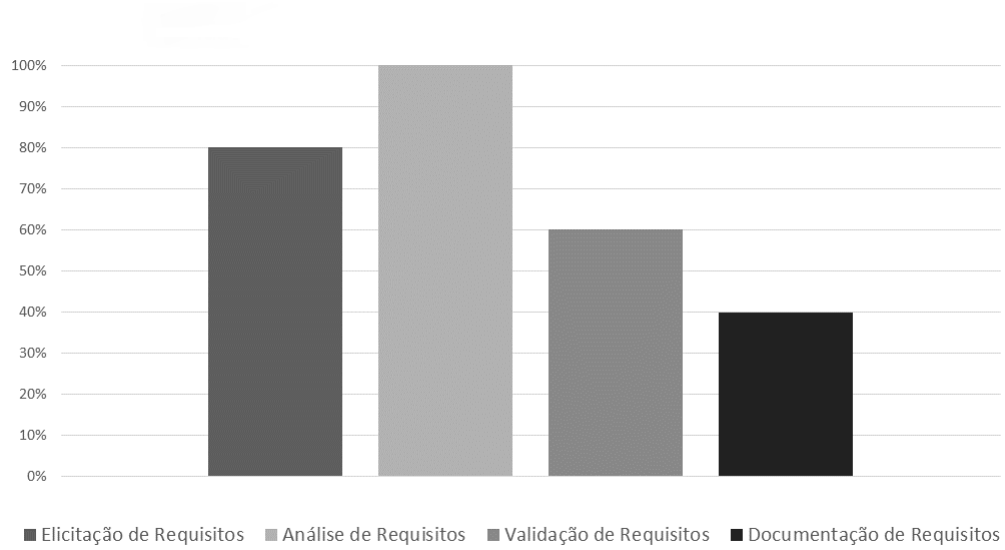

Figura 5: Atividades da Engenharia de requisitos ordenada por relevância segundo empresas.

Sobre as técnicas de elicitação de requisitos, descritas no Capítulo 2, foi questionado quais técnicas são adotadas pelas empresas na engenharia de requisitos do ponto de vista ágil. As técnicas avaliadas foram: entrevista, etnografia, *brainstorming*, caso de uso, *joint application*, modelagem, priorização, revisões de requisitos, testes de unidade, prototipação, e teste de aceitação. Essas técnicas são consideradas imprescindíveis para obtenção de requisitos mais precisos, proporcionando um alto nível de interação e entendimento entre as partes envolvidas no projeto (*Product Owner*, *Scrum Master* e Time Scrum). Quanto maior o entendimento do projeto a nível de requisitos, menor será o risco de mudança e maior será a confiança da equipe em atender as necessidades do projeto de software solicitado pelo cliente.

A empresa **X** utiliza metodologia ágil Scrum, o tamanho do time de desenvolvimento de software é igual a 6, para essa metodologia esse número é considerado razoável, pois a equipe tem um foco melhor com grupos pequenos, simplificando assim o processo de desenvolvimento, minimizando e dinamizando tarefas e artefatos. Os processos aplicados para obtenção dos requisitos do software são Análise de requisitos de software, Elicitação de requisitos e Validação de requisitos. As técnicas adotadas no desenvolvimento de requisitos são: *Brainstorming*, Casos de Uso, Modelagem, Priorização, Revisões de requisitos, Teste de unidade e Prototipação.

Nos projetos de software é necessário que os requisitos sejam documentados, produzindo artefatos sobre requisitos funcionais, *user stories* e *test cases*. A empresa

relata que os requisitos são documentados através de *user stories* e distribuídos em um quadro, conseguindo ajudar a planejar o projeto, priorizar as atividades, assim como distribuí-las para o time Scrum. As *user stories* facilitam o entendimento da funcionalidade a ser desenvolvida por cada integrante do time. No entanto, algumas vezes as *user stories* dificultam um pouco o processo de comunicação e desenvolvimento quando se trabalha com equipes remotas, ou seja, distribuídas geograficamente, uma vez que as histórias são curtas e não trazem muitos detalhes do requisito a ser desenvolvido. A empresa relata que o engajamento de toda a equipe é um desafio para construir o processo Scrum, assim como para cumprir os prazos e atividades previstas pelas *Sprints*.

Uma das características do Scrum é ter uma equipe auto organizável e multidisciplinar, ou seja, uma equipe que organiza e gerencia seu próprio trabalho e possui as competências necessárias para completar o trabalho. A comunicação face a face entre o time Scrum, assim como a liderança e o encorajamento dado pelo *Scrum Master*, são pontos muito relevantes para melhorar o processo de desenvolvimento e as práticas para fazê-lo mais efetivo e agradável. A compreensão de forma clara dos requisitos e a motivação da equipe ajudam a somar confiança entre os membros da equipe, minimizando a dificuldade de engajamento de toda a equipe.

A empresa **Z** utiliza metodologia ágil Scrum, o tamanho do time de desenvolvimento de software é igual a 4, os processos aplicados para obtenção dos requisitos do software são elicitação de requisitos, análise de requisitos e documentação. As técnicas adotadas no desenvolvimento de requisitos são: Entrevista, *Brainstorming* e Teste de aceitação. Os *stakeholders* e *Scrum Master* participam da fase de Elicitação da engenharia de requisitos e são criados os seguintes artefatos: *Product Backlog*, *Sprint Backlog* e tarefas.

Nos projetos de software os requisitos são documentados, produzindo apenas um documento descrevendo os requisitos, facilitando assim a planejamento do produto. A empresa relata que um dos maiores desafios é o gerenciamento de tempo. No Scrum é necessário definir quais as prioridades do negócio e da equipe, para uma gestão de tempo eficiente. A fim de conseguir vencer esse desafio é indicado para empresa a utilização da prática de reunião diária, já que a empresa relata que não se aplica essa técnica.

A empresa **V** utiliza metodologia ágil Scrum, o tamanho do time de desenvolvimento de software é igual a 120. A empresa obtém os requisitos do software através da elicitação de requisitos e análise de requisitos. As técnicas adotadas no desenvolvimento de requisitos são: Entrevista, Teste de unidade, Prototipação e Teste de aceitação. Na fase de Elicitação da dos requisitos os *stakeholders* e *Scrum Master* participam dessa fase e são criados os seguintes artefatos *Product Backlog*, *Sprint*

Backlog e tarefas.

Nos projetos de software os requisitos são documentados, através de uma ferramenta de monitoramento de tarefas e acompanhamento de projetos garantindo o gerenciamento de todas as atividades em um único lugar, com a descrição dos requisitos. Um dos maiores desafios apresentados pela empresa é o maior envolvimento do time de desenvolvimento, consequência do tamanho da equipe.

A empresa **Y** utiliza metodologia ágil Scrum/XP (Híbrido), o tamanho do time de desenvolvimento de software é de 20. Os processos aplicados para obtenção dos requisitos do software são elicitação de requisitos e análise de requisitos. As técnicas adotadas no desenvolvimento de requisitos são: Entrevistas, Casos de Uso, Priorização, Prototipação e Teste de aceitação. Na fase de Elicitação da engenharia de requisitos participam o Dono do produto, *stakeholders*, membros do time e Scrum Master. Os artefatos criados são *Product Backlog*, *Sprint Backlog*, Histórias, cartões simples e *Scrum board*.

Nos projetos de software os requisitos são documentados, produzindo Caso de Uso e Especificação Complementar. A empresa relata que é perceptível a relevância em ter a proximidade do cliente com o projeto, tornando fácil a elucidação de dúvidas e enriquecendo o produto. Um dos maiores desafios apresentados pela empresa é o prazo reduzido imposto pelo cliente, afetando diretamente a qualidade do produto.

A empresa **W** utiliza metodologia ágil Scrum/XP (Híbrido), o tamanho do time de desenvolvimento de software é de 20. Os processos aplicados para obtenção dos requisitos do software são elicitação de requisitos, análise de requisitos, documentação de requisitos e validação de requisitos. As técnicas adotadas no desenvolvimento de requisitos são: Priorização, Prototipação e Teste de aceitação. Na fase de Elicitação da engenharia de requisitos participam o Dono do produto, *stakeholders*, membros do time e Scrum Master, os artefatos criados são *Product Backlog*, *Sprint Backlog*, *Burn down chart*, Histórias, cartões simples, Tarefas e *Scrum board*. O dono do levanta todas as características do *Product Backlog*. Nos projetos os requisitos são documentados, produzindo artefatos como histórias de usuários. Um dos maiores desafios da empresa é conseguir manter a agilidade nos projetos.

A Figura 6 apresenta uma análise comparativa entre as fases da engenharia de requisitos na metodologia ágil Scrum e Scrum/XP (Híbrido), utilizadas nas empresas avaliadas, relacionando as práticas/técnicas presentes nas atividades da engenharia de requisitos de cada modelo de processo. A seguir será apresentado os pontos da relação dos processos da engenharia de requisitos em metodologias ágeis Scrum e Scrum/XP de acordo com a Figura 6, ainda é importante salientar que a figura apresenta o quantitativo de cada técnica utilizada pelas empresas.

PROCESSO DA ENGENHARIA DE REQUISITOS	SCRUM	SCRUM/XP (HÍBRIDO)
ELICITAÇÃO DE REQUISITOS	• Histórias de usuários (3) • Entrevista (2) • Brainstorming (2)	• Histórias de usuários (2) • Entrevista (2)
ANÁLISE DE REQUISITOS	• Prototipação (2) • Priorização (2) • Modelagem (1) • Cenário (2)	• Prototipação (2) • Priorização (1)
DOCUMENTAÇÃO DE REQUISITOS	• Documento descrevendo os requisitos (3) • Histórias de usuários (3) • Teste de casos (1) • Ferramenta de monitoramento de tarefas (2)	• Histórias de usuários (2) • Caso de uso (1) • Especificação complementar (1)
VALIDAÇÃO DE REQUISITOS	• Teste de aceitação (2) • Revisão de requisitos (1)	• Teste de aceitação (2) • Feedback frequente (1)

Figura 6: Análise comparativa entre os processos da engenharia de requisitos no Scrum e Scrum/XP (Híbrido)

Fase de Elicitação de Requisitos:

Na fase de elicitação de requisitos na metodologia ágil Scrum a técnica utilizada foi a entrevista e *brainstorming*, sendo que os requisitos são elicitados pelo o cliente/*product owner* a lista de funcionalidade de requisitos, para maximizar o valor do mesmo. Essa lista de requisitos, pode ainda não estar necessariamente refinados. Portanto a elicitação de requisitos utiliza as histórias dos usuário uma prática disponível pela a metodologia Scrum, essa por sua vez estar sendo mais utilizada na engenharia de requisitos ágil.

Na fase de elicitação de requisitos na metodologia ágil Scrum/XP, caracterizou que utiliza apenas a técnica de entrevista. Para a engenharia de requisitos as histórias do usuário são transformadas em funcionalidades do sistema.

Em comparação dessa fase, pode-se perceber que a metodologia Scrum está utilizando histórias de usuários como forma de obtenção de requisitos e idem para Scrum/XP que usam as histórias de usuários com o mesmo propósito.

Fase de Análise de Requisitos:

Na fase de análise de requisitos na metodologia Scrum tem como objetivo identificar as inconsistências, ambiguidades e possíveis problemas, a principal técnica utilizada é priorização seguida de prototipação, cenário e modelagem.

Na fase de análise de requisitos na metodologia Scrum/XP, as principais técnicas são prototipação e priorização.

Em comparação identificamos que tanto a metodologia Scrum quanto a metodologia Scrum/XP, podem continuar analisando os requisitos durante as fases de desenvolvimento do software. As empresas que utiliza que utilizam a metodologia ágil Scrum/ XP, não tem a necessidade de utilizar as técnicas de modelagem e cenário em seu ambiente de desenvolvimento.

Fase de Documentação de Requisitos:

Nessa fase foi relatado pelas empresas que é documentado apenas o necessário para validar os requisitos. A engenharia de requisitos ágil não predefine nenhuma técnica. Portanto, é de responsabilidade do time de desenvolvimento criar um documento que descreve os requisitos.

Na fase de documentação de requisitos da metodologia Scrum, foi relatado que produz um documento com as histórias dos usuários, outra forma de manter essa documentação é através de ferramentas de monitoramento de tarefas.

Na fase de documentação de requisitos da metodologia Scrum/XP, foi relatado que produz as histórias de usuários como documentação, caso de uso e especificação complementar.

Fase de Validação de Requisitos:

Na fase de validação de requisitos na metodologia Scrum no final de cada sprint tem como objetivo verificar e validar se o produto desenvolvido está de acordo com o que foi estabelecido nos requisitos, a principal técnica utilizada é teste de aceitação (TDA) e revisão de requisitos iniciais. Os testes de aceitação envolvem testar uma funcionalidade do início ao fim, dando entradas para o sistema e observando o comportamento de saída. A revisão de requisitos é o processo de verificação que a metodologia Scrum oferece, é uma reunião onde se a equipe apresenta o que foi desenvolvido para o *product owner* e partes interessadas. Uma reunião que ajuda a equipe obter *feedback* e determinar quais são os objetivos das próximas iterações.

Na fase de validação da metodologia ágil Scrum/XP, a principal técnica utilizada foi o teste de aceitação, que consiste em testar o sistema do ponto de vista do usuário, sendo suscetíveis a alterações e *feedback* frequente.

Em comparação observou que o Scrum válida a integração dos incrementos no final de cada sprint, já o Scrum/XP pode validar suas integrações paralelamente às implementações das histórias do usuário.

As vantagens segundos as empresas entre a engenharia de requisitos ágil estão diretamente relacionadas em manter a agilidade do processo com alta qualidade, facilitando o planejamento dos requisitos e mantendo um maior entendimento do que deve ser desenvolvido. Desse modo, a engenharia de requisitos ajuda principalmente

a elaborar melhor os requisitos tornando mais consistentes evitando assim o retrabalho. Como desvantagens as empresas apontaram que o prazo reduzido imposto pelo cliente pode às vezes diminuir a qualidade do produto.

Portanto, pode-se destacar como impacto para a engenharia de requisito em ambiente ágil os seguintes fatores:

(i) No processo da engenharia de requisitos existem diferentes técnicas que podem ser usadas, não significa dizer que ao adotar uma determinada metodologia ágil o projeto terá que utilizar todas as técnicas existentes para um determinado processo da engenharia de requisitos, então cabe a equipe de desenvolvimento conhecer seu ambiente e suas prioridades e escolher qual técnicas se encaixa melhor em determinada atividade.

(ii) Os requisitos devem ser elicitados utilizando uma linguagem mais próxima do cliente para melhorar a comunicação e o entendimento do que se deve ser realizado no desenvolvimento de software.

(iii) Os documentos produzidos nas fases iniciais podem se tornar irrelevantes, a equipe de desenvolvimento deve atualizar os documentos.

5.2 Desafios da Engenharia de Requisitos em Metodologias Ágeis

O desenvolvimento de um produto pode ser prejudicado quando não há um ambiente adequado, portanto é necessário descobrir os desafios em uma organização e solucioná-los para melhorar o desenvolvimento de software.

Nesta seção, identificamos os desafios comuns, ou pelo menos as questões que são percebidas como desafios, que as equipes de desenvolvimento enfrentam quando se trata da engenharia de requisitos ágil. Discutimos ainda possíveis soluções para lidar com esses desafios. Os itens a seguir foram identificados a partir das respostas das empresas.

1. Envolvimento do Cliente e Stakeholders

O envolvimento do cliente e *stakeholders* desempenha um papel importante em metodologia ágil, pois promove a interação constante com o cliente e a equipe. A frequência de comunicação depende da disponibilidade e disposição dos membros da equipe. Quando a comunicação presencial não é viável, uma boa prática pode ser o uso da tecnologia disponível para se aproximar da comunicação face a face.

Desse modo, é importante salientar que quando não há o envolvimento do cliente e *stakeholders* no projeto, pode-se acontecer o insucesso do mesmo ou não atin-

gir o resultado esperado pelo cliente. Quanto maior for o comprometimento do cliente com o projeto maior deverá ser o nível de requisitos, menor será o risco e mudanças, fornecendo uma maior confiança entre a equipe e o atendimento às necessidades do produto.

2. Falta de Comunicação

A falta de comunicação ocorre quando há uma ausência no fornecimento de informações necessárias entre os integrantes do projeto. A metodologia ágil tem a capacidade de lidar com esse desafio de falta de comunicação, sendo superados pela comunicação face a face frequente.

As desvantagens para a falta de comunicação, se emprega em relação a troca de informações entre os integrantes da equipe. A comunicação precisa ser clara, objetiva e direta. Sem a comunicação entre todos os membros da equipe não haverá o sucesso do projeto, ocasionando o não cumprimento de prazos, ocultação de informações relevantes, podendo resultar no levantamento de requisitos inconsistentes.

3. Validação de Requisitos

A validação de requisitos é um dos maiores desafios da Engenharia tradicional. As metodologias ágeis lidam com a validação de requisitos através de priorização contínua de requisitos. Ao final de cada iteração, a história do usuário é demonstrada ao proprietário do produto e aos representantes do cliente. Desta forma, as mudanças são sugeridas e analisadas para garantir que a história do usuário satisfaça o que foi pretendido. A decisão de priorizar as necessidades de um projeto deve ser tomada a partir de uma parceria entre cliente e desenvolvedor. No caso de não fazer a validação de requisitos corretamente poderá gerar custos altos por erros de requisitos. Uma técnica que ajuda a fornecer ao cliente um modelo do produto inicial é a prototipagem.

4. Documentação de Requisitos

A documentação de requisitos é um desafio por causa da constante mudança dos requisitos no desenvolvimento ágil. Sendo que devem ser descritos e documentados apenas o necessário para validar cada requisito. Sugere começar com um documento que descreva o suficiente para as necessidades de seus clientes. O Manifesto para Desenvolvimento Ágil de Software valoriza "software trabalhando mais que documentação abrangente". Isto é, o modelo ágil dá preferência a comunicação verbal do que a comunicação escrita. Sua vantagem está relacionada a eficiência do tempo, onde ambas as partes podem esclarecer um assunto de forma clara e rápida sem deixar dúvidas ou espaços para desentendimento.

Desse modo, nenhuma especificação de requisitos formal é produzida no desenvolvimento ágil, já que ágil se concentra em documentação mínima. Os recursos e os requisitos são registrados em *story boards*, cartões de índice e protótipos de papel,

como casos de uso e diagramas de fluxo de dados (LUCIA A.; QUSEF, 2010).

5.3 Boas práticas da Engenharia de Requisitos em Metodologia Ágeis

Desenvolver software que atende as necessidades e a expectativa das partes interessadas do projeto é o objetivo principal de qualquer metodologia de desenvolvimento de software. O sucesso geral e as falhas do projeto dependem dos requisitos. As boas práticas não estabelecem sua utilização como única forma de se obter bons resultados, pelo contrário, sugere que o sucesso de um projeto é favorecido por sua adoção.

Nesta seção, será apresentada uma lista de boas práticas, que são essenciais para a melhoria da qualidade no processo de desenvolvimento.

1. User Stories

User Stories ou histórias de usuário são criadas como especificações dos requisitos do cliente. As histórias de usuários facilitam a comunicação e uma melhor compreensão geral entre as partes interessadas (DANEVA et al., 2013).

As histórias de usuários deslocam a concentração da documentação escrita para a comunicação (CARLSON D., 2010). Acredita-se que as histórias de usuários sejam capazes de acabar com o desafio da constante atualização de documentos de especificação de requisitos na engenharia de requisitos tradicional (BJARNASON; WNUK; REGNELL, 2011), mantendo os membros da equipe atualizados. As histórias de usuários enfatizam os "objetivos do usuário", explicam brevemente a percepção do usuário, focalizam o que é necessário fazer e apoiam o desenvolvimento colaborativo e iterativo (CARLSON D., 2010).

2. Comunicação face a face

Uma característica da engenharia de requisitos ágil é a comunicação face a face entre os clientes, *stakeholders* e a equipe de desenvolvimento. Essa prática defende uma documentação mínima, visto que na engenharia de requisitos tradicional a especificação de documentos é longa e complexa.

A comunicação face a face frequente ajuda o cliente a direcionar melhor o projeto de acordo com sua própria compreensão do produto a ser desenvolvido. As reuniões frequentes levam à comunicação para as partes interessadas, o que auxilia na evolução dos requisitos. A frequência de comunicação depende da disponibilidade e disposição dos membros da equipe. Quando a comunicação presencial não é viável, uma boa prática deve ser o uso da tecnologia disponível para se aproximar da

comunicação face a face (LUCIA A.; QUSEF, 2010).

3. Participação do cliente

O envolvimento do cliente foi declarado uma das principais razões para o sucesso do projeto (EBERLEIN A., 2002). Em metodologia ágil existe a necessidade de identificação de clientes ou representantes de grupos empresariais para assegurar que os requisitos sejam adequadamente definidos, esclarecidos e priorizados (DANEVA et al., 2013). A incapacidade de identificar representantes de clientes pode levar a discordâncias e pontos de vista diferentes sobre uma variedade de questões. Portanto, dependem de colaboração frequente (por exemplo, face a face, (CAO; RAMESH, 2008)) com um cliente disponível (BECK K., 2001).

4. Requisitos Iterativos

São requisitos que emergem ao longo do tempo em metodologia ágil (RAMESH; CAO; BASKERVILLE, 2010). A interação frequente entre as partes interessadas acaba tornando os requisitos mais claros ao longo do tempo, fortalecendo as relações com o cliente, e permite que os requisitos evoluem com menos investimento de tempo (CAO; RAMESH, 2008). Esse detalhamento gradual dos requisitos (BJARNASON; WNUK; REGNELL, 2011) produz um documento que está mais próximo do desenvolvimento, o que o torna mais sutil e menos frágil.

5. Priorização dos Requisitos

Na Engenharia de requisitos tradicional, a priorização é realizada apenas uma vez antes do início do desenvolvimento. Em metodologia ágil, a priorização dos requisitos é parte de cada iteração, ou seja, os requisitos são priorizados continuamente em cada ciclo de desenvolvimento por clientes que se concentram no valor do negócio (CAO; RAMESH, 2008) ou em risco (DANEVA et al., 2013).

6. Planejamento contínuo

O planejamento contínuo é uma tarefa de rotina para equipes ágeis (JUN; QIUZHEN; LIN, 2010). A equipe nunca adere a planos fixos, sendo necessário adaptar às próximas mudanças dos clientes conforme o projeto progride. Essa flexibilidade facilita a mudança de requisitos nas fases posteriores dos projetos.

7. Gerenciamento de requisitos

O gerenciamento de requisitos é realizado através da manutenção de listas de atraso de produtos / listas de recursos e cartões de índice (CAO; RAMESH, 2008); (RAMESH; CAO; BASKERVILLE, 2010). No método Scrum, o *backlog* do produto pode ser usado para acompanhar as alterações de requisitos. No XP, o *feedback* é constante podendo adaptar a eventuais mudanças com a utilização de ferramentas de gerenciamento de requisitos.

8. Testar antes de codificar

Testar antes da codificação significa escrever testes antes de escrever códigos funcionais para requisitos, promovendo um *feedback* no caso de falhas de teste. Outra abordagem proposta é o desenvolvimento automatizado de teste de aceitação (HAUGSET; STALHANE, 2012). Essa prática é muito utilizada na metodologia ágil XP.

9. Retrospectivas

Retrospectivas são as reuniões realizadas após a conclusão de uma iteração (CARLSON R., 2012). Essas reuniões muitas vezes analisam o trabalho concluído e determinam as etapas futuras e retrabalho. O cliente propõe novos requisitos sob a forma de mudanças nas entregas.

10. Equipe multifuncional

Equipes multifuncionais são compostas por membros de áreas funcionais distintas que tem por objetivo propor medidas para a melhoria da qualidade e resolver problemas que afetam mais de uma área funcional do projeto. Em metodologia ágil, desenvolvedores, testadores, *designers* e gerentes sentam e trabalham juntos. Uma das vantagens é que esse conceito ajuda a reduzir desentendimentos entre os membros, visto que os membros se unem uns com os outros e compartilham seus conhecimentos, o que aumenta seu nível de confiança e, finalmente, gera mais comunicação.

CAPÍTULO 6

CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresentou técnicas e abordagens da Engenharia de Requisitos utilizadas durante os processos de desenvolvimento ágil, incluindo o estudo de viabilidade, elicitação, análise, documentação e validação. Da mesma forma, expõe a metodologia ágil Scrum, definindo os papéis do Scrum, as etapas dos *Sprints* e os artefatos, além de apresentar a metodologia XP com seus valores e princípios.

Para a construção do processo de engenharia de requisito de uma forma ágil, alguns aspectos são imprescindíveis, tais como: colaboração do cliente, bons desenvolvedores ágeis e gerentes de projeto experientes. Além disso, a comunicação entre os membros da equipe e as partes interessadas podem ser o segredo do sucesso na entrega final do produto, focando na qualidade, prazos, custo, agilidade e satisfação do cliente.

Os desafios dentro da organização devem ser encontrados e tratados para melhorar o desenvolvimento de software alcançando os objetivos dos projetos. As boas práticas podem ser utilizadas para auxiliar no processo de desenvolvimento.

As fases do processo da Engenharia de requisitos como a elicitação, análise e validação estão presentes no processo ágil. Em metodologia ágil as fases não se separam como na engenharia de requisitos tradicional e as técnicas utilizadas variam de acordo com a abordagem. Sendo assim, cada prática, como por exemplo no jogo do planejamento em XP, incorpora as fases de elicitação e análise de requisitos.

O presente trabalho buscou realizar comparativo da engenharia de requisitos entre Scrum e Scrum/XP, de modo que obtivesse um melhor entendimento de como estar ocorrendo o processo das atividades da engenharia de requisitos. O requisito é a parte essencial do desenvolvimento de software e merece uma maior atenção. Realizar um bom levantamento de requisitos e análise de requisitos deve ser algo fundamental para o desenvolvimento de software, pois são exatamente nessas fases que acontecem a obtenção de requisitos e priorização dos requisitos que auxiliam no andamento de todas as fases posteriores do projeto. A validação de requisitos ocorre de maneira verifica se o sistema atende as necessidades e expectativas do cliente.

O desenvolvimento ágil se concentra em documentação mínima no desenvolvimento de software. A equipe de desenvolvimento é responsável por assegurar que

há documentação suficiente para entendimento do software e manutenção futura.

O modelo de processo deve ser escolhido de acordo com as necessidades da área de aplicação da empresa, sendo que cada empresa atende a uma necessidade diferente. No processo da engenharia de requisitos existem diferentes técnicas que podem ser usadas, não significa dizer que ao adotar uma determinada metodologia ágil o projeto terá que utilizar todas as técnicas existentes para um determinado processo da engenharia de requisitos, então cabe a equipe de desenvolvimento conhecer seu ambiente e suas prioridades e escolher qual técnicas se encaixa melhor em determinada atividade.

A vantagem em utilizar os processos, técnicas e os desafios da engenharia de requisitos é que ajuda as empresas a elaborar melhor seus requisitos, conhecendo as necessidades do sistema, tornando os requisitos mais consistentes e evitando assim o retrabalho.

Como trabalhos futuros, sugere-se aprofundar a avaliação do estudo de caso para engenharia de requisitos em metodologias ágeis diferentes das estudadas aqui, a fim de identificar novas vantagens e desvantagens em utilizá-lo e encontrar o que há de incomum.

Referências

AURUM, A.; WOHLIN, C. *Engineering and Managing Software Requirements*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005. ISBN 3540250433. Citado 2 vezes nas páginas 17 e 41.

BECK K., B. M. v. B. A. C. A. C. W. F. M. e. a. Manifesto for agile software development. 2001. Disponível em: <http:// agilemanifesto.org/>. Acesso em: 08/11/2016. Citado 4 vezes nas páginas 26, 27, 29 e 50.

BJARNASON, E.; WNUK, K.; REGNELL, B. A case study on benefits and side-effects of agile practices in large-scale requirements engineering. In: *Proceedings of the 1st Workshop on Agile Requirements Engineering*. New York, NY, USA: ACM, 2011. (AREW '11), p. 3:1–3:5. ISBN 978-1-4503-0890-8. Citado 4 vezes nas páginas 32, 33, 49 e 50.

CAO, L.; RAMESH, B. Agile requirements engineering practices: An empirical study. *IEEE Software*, v. 25, n. 1, p. 60–67, Jan 2008. ISSN 0740-7459. Citado na página 50.

CARLSON D., . M. P. Practical agile requirements engineering. 2010. Citado na página 49.

CARLSON R., M. P. J. . S. R. L. Applying scrum to stabilize systems engineering execution. 2012. Citado na página 51.

CORPORATION, R. S. Rational unified process: Artefatos, 1987 - 2001. 2001. Citado 2 vezes nas páginas 19 e 20.

DANEVA, M. et al. Agile requirements prioritization in large-scale outsourced system projects: an empirical study. *Journal of systems and software*, Elsevier, Amsterdam, v. 86, n. 5, p. 1333 –1353, May 2013. ISSN 0164-1212. Citado 2 vezes nas páginas 49 e 50.

EBERLEIN A., . J. C. S. d. P. L. Agile requirements definition: A view from requirements engineering. In: *In Proceedings of the international workshop on time constrained requirements engineering*. [S.l.: s.n.], 2002. Citado 2 vezes nas páginas 33 e 50.

HASTIE S.AND WOJEWODA, S. Standish group 2015 chaos report. 2015. Disponível em: <https://www.infoq.com/articles/standish-chaos-2015>. Acesso em: 15/10/2016. Citado 3 vezes nas páginas 11, 13 e 14.

HAUGSET, B.; STALHANE, T. Automated acceptance testing as an agile requirements engineering practice. In: *Proceedings of the 2012 45th Hawaii International Conference on System Sciences*. Washington, DC, USA: IEEE Computer Society, 2012. (HICSS '12), p. 5289–5298. ISBN 978-0-7695-4525-7. Citado na página 51.

IEEE. Ieee standard glossary of software engineering terminology. *IEEE Std 610.12-1990*, p. 1–84, Dec 1990. Citado na página 17.

JUN, L.; QIUZHEN, W.; LIN, G. Application of agile requirement engineering in modest-sized information systems development. In: *2010 Second World Congress on Software Engineering*. [S.l.: s.n.], 2010. v. 2, p. 207–210. Citado na página 50.

KITCHENHAM, B. *Kitchenham, 2004 Procedures for Performing Systematic Reviews*. 2004. Citado na página 30.

LUCIA A.; QUSEF, A. Requirements engineering in agile software development. In: . [S.l.]: Journal of Emerging Technologies in Web Intelligence, 2010. Citado 10 vezes nas páginas 11, 30, 31, 32, 33, 34, 35, 36, 49 e 50.

MUSHTAQ, Z. Novel hybrid model: Integrating scrum and xp. International Journal of Information Technology and Computer Science (IJITCS)., v. 4, 2012. Citado 2 vezes nas páginas 32 e 33.

PAETSCH, F.; EBERLEIN, A.; MAURER, F. Requirements engineering and agile software development. In: *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on*. [S.l.: s.n.], 2003. p. 308–313. ISSN 1080-1383. Citado 6 vezes nas páginas 30, 31, 33, 34, 35 e 36.

PRESSMAN, R. S. *Engenharia de Software: uma abordagem profissional.* [S.l.]: Pearson Makron Books, 2011. Citado 4 vezes nas páginas 10, 13, 28 e 29.

RAMESH, B.; CAO, L.; BASKERVILLE, R. Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal*, Blackwell Publishing Ltd, v. 20, n. 5, p. 449–480, 2010. ISSN 1365-2575. Citado na página 50.

SCHWABER, K.; BEEDLE, M. *Agile Software Development with Scrum*. 1st. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001. ISBN 0130676349. Citado 8 vezes nas páginas 10, 14, 20, 22, 23, 26, 31 e 34.

SCHWABER K.; SURTHERLAND, J. Guia do scrum. um guia definitivo para o scrum: As regras do jogo. 2015. Citado 5 vezes nas páginas 14, 22, 23, 24 e 25.

SILLITTI, A.; SUCCI, G. Engineering and managing software requirements. In: ____. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. cap. Requirements Engineering for Agile Methods, p. 309–326. ISBN 978-3-540-28244-0. Citado 2 vezes nas páginas 32 e 35.

SOMMERVILLE, I. *Software Engineering*. 10th. ed. [S.l.]: Pearson, 2015. ISBN 0133943038, 9780133943030. Citado 8 vezes nas páginas 10, 13, 14, 17, 18, 19, 26 e 27.

THAYER, R. H.; BAILIN, S. C.; DORFMAN, M. *Software Requirements Engineerings, 2Nd Edition*. 2nd. ed. Los Alamitos, CA, USA: IEEE Computer Society Press, 1997. ISBN 0818677384. Citado na página 13.

VERSIONONE. 9th annual state of agile survey. 2015. Disponível em: <https:/-/www.versionone.com/pdf/state-of-agile-development-survey-ninth.pdf>. Acesso em: 15/10/2016. Citado na página 14.

Apêndices

Apêndice A - QUESTIONÁRIO DE AVALIAÇÃO

O questionário apresentado faz parte da pesquisa para extrair os dados apresentado no capítulo 5 e utilizá-lo como referência para as entrevistas das cinco empresas.

UM COMPARATIVO SOBRE A ENGENHARIA DE REQUISITOS EM PROJETOS COM METODOLOGIA ÁGIL

Olá, sou aluna do IFCE - campus Aracati, faço o curso de Bacharelado em Ciência da Computação. Estou enviando o questionário, composto por perguntas referentes a Engenharia de Requisitos e Metodologias ágeis Scrum, XP e o modelo hibrido (Scrum e XP). Qualquer dúvida ou questionamento entrar em contato por email: adrica.contato@gmail.com

*Obrigatório

1. **Nome da Empresa:** *

2. **Onde sua organização está localizada (Cidade / Estado / País)?** *

3. **Qual a principal área de atuação da sua organização?** *
   Marque todas que se aplicam.
   - [ ] Armazanamento
   - [ ] Científico
   - [ ] Comunicação
   - [ ] Educação
   - [ ] Escritório / Negócios
   - [ ] Internet
   - [ ] Governo
   - [ ] Jogos / Entretenimento
   - [ ] Mobile
   - [ ] Multimídea
   - [ ] Segurança
   - [ ] Outro:

4. **Qual o tamanho total da equipe de software em sua organização? (incluindo todos os aspectos de desenvolvimento de software e de entrega)** *
   Marque todas que se aplicam.
   - [ ] Até 5 pessoas
   - [ ] De 6 a 20 pessoas
   - [ ] De 21 a 50 pessoas
   - [ ] De 50 a 100 pessoas
   - [ ] Mais de 100

5. **Qual o tamanho do time de desenvolvimento de software?** *

6. **Qual modelo processo é usada nos projetos da organização?** *
Marcar apenas uma oval.

- ◯ Metodologia ágil Scrum
- ◯ Metodologia ágil XP
- ◯ Scrum e XP
- ◯ Modelos tradicionais da engenharia de requisitos
- ◯ Scrum/ XP Híbrido
- ◯ Outro: _____

7. **Quais atividades são utilizadas na engenharia de requisitos?** *
Marque todas que se aplicam.

- ☐ Elicitação de Requisitos
- ☐ Análise de Requisitos
- ☐ Documentação de Requisitos
- ☐ Validação de Requisitos
- ☐ Outro: _____

8. **Quais são as práticas/técnicas adotadas no desenvolvimento?**
Marque todas que se aplicam.

- ☐ Entrevistas
- ☐ Etnografia
- ☐ Brainstorming
- ☐ Casos de Uso
- ☐ Joint Application Development (JAD)
- ☐ Modelagem
- ☐ Priorização
- ☐ Revisões de requisitos
- ☐ Teste de unidade
- ☐ Prototipação
- ☐ Teste de aceitação
- ☐ Outro: _____

9. **Quem participa da fase de Elicitação da atividade da engenharia de requisito nos projetos que você trabalha?** *
Marque todas que se aplicam.

- ☐ Dono do produto
- ☐ Stakeholders
- ☐ Membros do time
- ☐ Scrum Master
- ☐ Outro: _____

10. **Quais artefatos são criados nas atividades do projeto?** *
Marque todas que se aplicam.

- [ ] Product Backlog
- [ ] Sprint Backlog
- [ ] Burn down chart
- [ ] Histórias, cartões simples.
- [ ] Tarefas
- [ ] Módulos funcionais
- [ ] Scrum board
- [ ] Outro: _____

11. **Quais atividades são utilizadas nas etapas da sprint?**
Marque todas que se aplicam.

- [ ] Iteração / Sprint
- [ ] Sprint Planning
- [ ] Sprint Daily
- [ ] Stand up meeting
- [ ] Sprint Review
- [ ] Sprint Retrospective
- [ ] Iteração semanal/trimestral
- [ ] Colaboração
- [ ] Programação em par
- [ ] Envolvimento real com o cliente
- [ ] Código compartilhado
- [ ] Repositório de código único
- [ ] Padronizar código fonte
- [ ] Manter apenas código e testes
- [ ] Evoluir apenas código e teste

12. **Qual práticas/técnicas usam na atividade da RE na fase de Elicitação de requisitos?** *
Marque todas que se aplicam.

- [ ] Entrevista
- [ ] Brainstorming
- [ ] Caso de uso
- [ ] Product Owner levanta todas as características do Product Backlog.
- [ ] Os clientes escrevem as histórias do usuário.

13. **Como é realizada a fase da Análise da atividade da engenharia de requisito nos projetos que você trabalha?**
Marque todas que se aplicam.

- [ ] Reunião de refinamento do Backlog
- [ ] Product Owner prioriza o Product Backlog.
- [ ] Product Owner analisa a viabilidade de requisitos.
- [ ] Não existe fase separada.
- [ ] Análise durante o desenvolvimento.
- [ ] Usa Modelagem
- [ ] Usa priorização
- [ ] Outro: _____

14. **Como é realizada a fase de Documentação da atividade da engenharia de requisitos nos projetos que você trabalha?** *
Marque todas que se aplicam.

- [ ] Histórias de usuários e testes de aceitação, como documentos de requisitos.
- [ ] Comunicação face-a-face
- [ ] Os produtos de software como, a persistência em formação.
- [ ] Outro: _____

15. **Como é realizada a fase de Validação da atividade da engenharia de requisitos nos projetos que você trabalha?** *
Marque todas que se aplicam.

- [ ] Reuniões de revisão
- [ ] Teste de aceitação
- [ ] Test Driven Development (TDD).
- [ ] Feedback frequente.
- [ ] Teste de unidade
- [ ] Prototipação
- [ ] Outro: _____

16. **Nos projetos de software que você trabalha, os requisitos são documentados? Se sim, quais artefatos são produzidos?** *

_____

_____

_____

_____

_____

17. **Quais as vantagens e desvantagens da engenharia de requisitos para a metodologia ágil que seu projeto utiliza?** *

18. **Quais são os maiores desafios da aplicação da engenharia de requisitos com metodologia ágil? E quais os principais problemas encontrados na engenharia de requisitos atrelados a metodologia ágil?** *