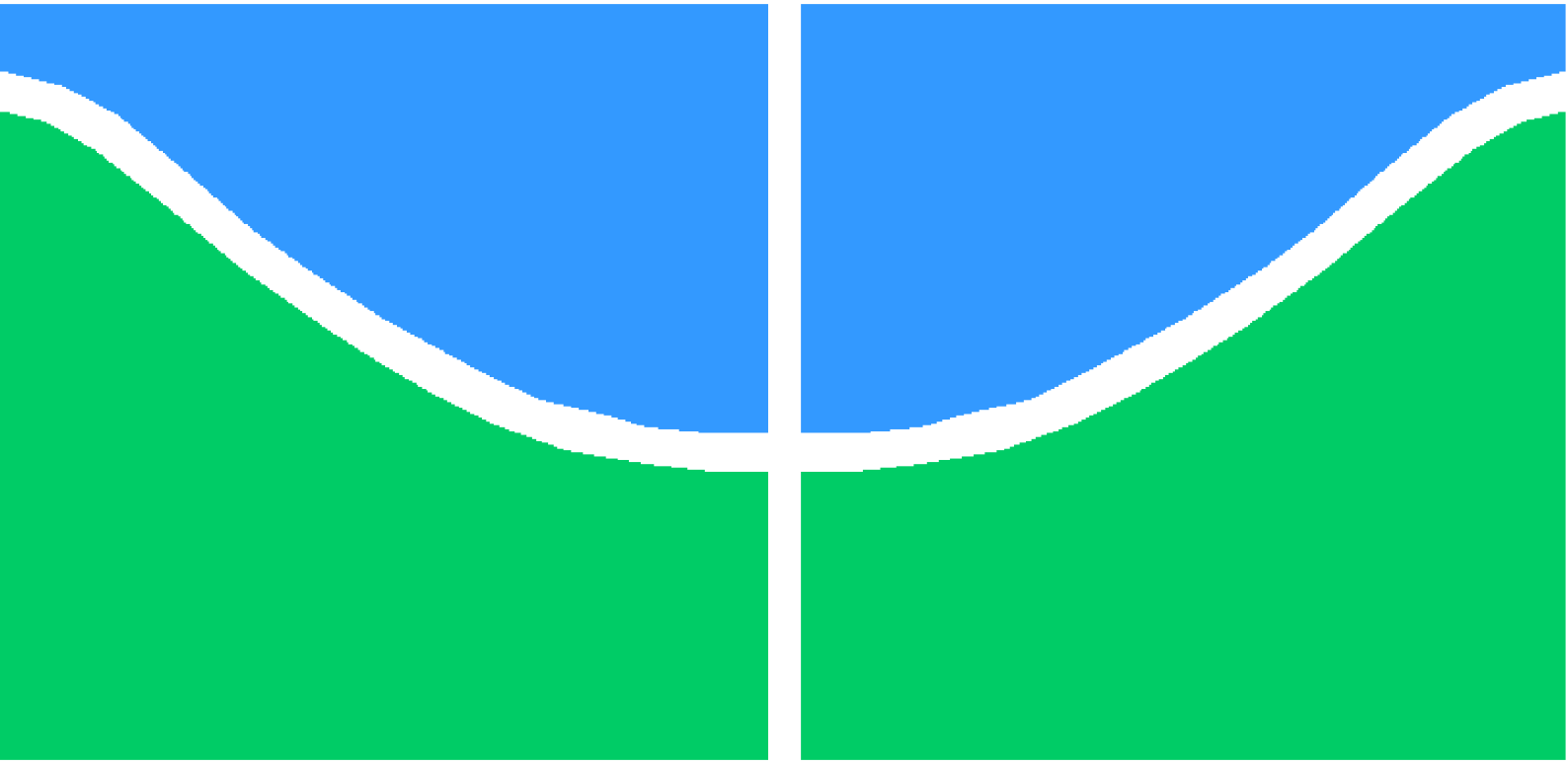

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Métodos e Métricas para Estimativas e Planejamento de Projetos Ágeis de Software: Estudo Comparativo entre Pontos de Função e Story Points

Autor: Dandara Pereira Aranha e Maxwell de Oliveira Cardoso

Orientador: Professora Dra. Edna Dias Canedo

Brasília, DF

2017

Dandara Pereira Aranha e Maxwell de Oliveira Cardoso

Métodos e Métricas para Estimativas e Planejamento de Projetos Ágeis de Software: Estudo Comparativo entre Pontos de Função e Story Points

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Professora Dra. Edna Dias Canedo

Brasília, DF

2017

Dandara Pereira Aranha e Maxwell de Oliveira Cardoso

Métodos e Métricas para Estimativas e Planejamento de Projetos Ágeis de Software: Estudo Comparativo entre Pontos de Função e Story Points/ Dandara Pereira Aranha e Maxwell de Oliveira Cardoso. – Brasília, DF, 2017-

64 p. : il. (algumas color.) ; 30 cm.

Orientador: Professora Dra. Edna Dias Canedo

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2017.

Métodos e Métricas para Estimativas e Planejamento de Projetos Ágeis de Software: Estudo Comparativo entre Pontos de Função e Story Points

CDU 02:141:005.6

Dandara Pereira Aranha e Maxwell de Oliveira Cardoso

Métodos e Métricas para Estimativas e Planejamento de Projetos Ágeis de Software: Estudo Comparativo entre Pontos de Função e Story Points

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Trabalho aprovado. Brasília, DF, 11 de Dezembro de 2017 – Data da aprovação do trabalho:

Professora Dra. Edna Dias Canedo
Orientadora

Professor Msc. Cristiane Soares Ramos
Convidado 1

Professor Msc. Ricardo Ajax Dias Kosloski
Convidado 2

Brasília, DF
2017

Dedicamos este trabalho às pessoas que sempre estiveram ao nosso lado incentivando e nos ajudando nos momentos mais difíceis desta jornada.

Agradecimentos

Meus sinceros agradecimentos a minha família, e amigos que nos últimos anos me apoiaram e compreenderam minha ausência em diversos momentos. Um agradecimento especial a minha mãe Marilda, que sempre foi a maior incentivadora de todos os meus sonhos,e sem a qual jamais conseguiria chegar ao fim deste curso. Ao meu pai Idalício, pelo amor incondicional e por acordar cedo todos os dias para me levar a parada de ônibus sem nunca reclamar. Ao meu parceiro de curso Maxwell, que dividiu comigo a obrigação de realizar este trabalho, sempre com muita responsabilidade e persistência. Agradeço também ao meu namorado Danilo, por toda paciência durante esses últimos dois anos, me acalmando nos momentos de tensão e me incentivando a ir em frente sempre. E, por último, e não menos importante, a minha orientadora Edna Dias Canedo que com sua disposição e eficiência, guiou o nosso trabalho da melhor maneira possível.

Dandara Pereira Aranha

Devo agradecer primeiramente à Deus por me dar a oportunidade de estar desenvolvendo um Trabalho de Conclusão de Curso em uma Universidade Federal conceituada como a UnB. Devo também, sinceros agradecimentos à minha família, em especial à minha mãe Eliana e ao meu pai Carlos Alberto pelo apoio nos momentos difíceis nesta jornada, sempre me incentivando e mostrando o valor dos estudos. Á minha parceira de curso Dandara por dividir as responsabilidades deste trabalho com muita competência. E não poderia deixar de agradecer à nossa orientadora Edna Dias Canedo por sempre estar disposta à nos ajudar e nos guiar para que o trabalho seja concluído da melhor maneira possível.

Maxwell de Oliveira Cardoso

"Sonhos determinam o que você quer.

Ações determinam o que você conquista."

(Aldo Novak)

Resumo

Nos últimos anos, os métodos ágeis de desenvolvimento de software ganharam muita atenção no campo da engenharia de software. A natureza dos projetos ágeis de software é diferente dos projetos de software tradicionais. Portanto, usar técnicas tradicionais de estimativas de esforço, tempo e custo pode produzir estimativas imprecisas. Várias técnicas foram propostas por vários autores e desenvolvedores. Um dos objetivo deste trabalho então é pesquisar na literatura através de uma Revisão Sistemática (RS) as práticas atuais de estimativas no desenvolvimento de software ágil e as métricas de tamanho mais utilizadas como insumos para essas estimativas, coletando-as em um estudo para uma futura comparação de precisão das mesmas. Com a realização da RS, foi concluído que o Story Point e o Ponto de Função são as métricas mais utilizadas em projetos ágeis como base para relizar estimativas de tamanho, tempo, esforço, produtividade e custo. Com base nessas duas métricas de tamanho, um estudo de caso foi executado com a realização de estimativas de esforço, tempo e custo para um projeto ágil de uma fábrica de software, onde os valores reais de desenvolvimento foram comparados com as estimativas para analisar qual delas forneceu as estimativas que mais se aproximaram dos valores reais.

Palavras-chaves: Desenvolvimento Ágil de Software; Estimativas de software; Métricas de software; Scrum;

Abstract

In recent years, agile methods of software development have gained a lot of attention in the field of software engineering. The nature of agile software projects is different from traditional software projects. Therefore, using traditional techniques of estimates of effort, time, and cost may produce imprecise estimates. Several techniques have been proposed by various authors and developers. One of the objectives of this work is to search the literature through a Systematic Review (SR) the current practices of estimates in the development of agile software and the most used size metrics as inputs for these estimates, collecting them in a study for a future comparison of their accuracy. With the realization of SR, it was concluded that Story Point and Point of Function are the metrics most used in agile projects as the basis for estimating size, time, effort, productivity and cost. Based on these two size metrics, a case study was performed with estimates of effort, time, and cost for an agile project of a software factory, where actual development values were compared with estimates to analyze which of them provided the estimates that were closest to the real values.

Key-words: Agile Software Development; Software Estimation; Software Metrics; Scrum;

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

TCC	Trabalho de Conclusão de Curso
APF	Análise de Pontos de Função
RS	Revisão Sistemática
US	User Story
SP	Story Points
PF	Pontos de Função

Sumário

1 Introdução

O desenvolvimento de software ágil é um conjunto de métodos iterativos e incrementais de engenharia de software que são defendidos com base em uma "filosofia ágil"descrita no Manifesto Ágil (FOWLER; HIGHSMITH, 2017) e que surgiu como alternativa aos chamados métodos tradicionais de desenvolvimento de software. Os métodos tradicionais se concentram no planejamento inicial e no gerenciamento rigoroso da mudança, já os métodos ágeis foram projetados para aceitar e gerenciar eficientemente as mudanças.

O desenvolvimento ágil enfatiza ciclos de desenvolvimento curtos, entregas freqüentes, comunicação contínua "frente a frente"e o aprendizado. Valorizam indivíduos e interações, software de trabalho, colaboração com clientes e respondem a mudanças durante todo o processo. Os métodos de desenvolvimento ágil mais populares incluem Scrum (SCHWABER; BEEDLE, 2002) e Extreme Programming (BECK; ANDRES, 2004).

De acordo com (PULFORD; KUNTZMANN-COMBELLES, 1995), usamos métricas todos os dias para entender, controlar e melhorar o que fazemos e como o fazemos, e o autor nos dá as seguintes motivações para o uso de métricas: planejamento e estimativa do projeto, gerenciamento e acompanhamento de projetos, compreensão a qualidade e os objetivos do negócio, comunicação, processos e ferramentas aprimorados para o desenvolvimento de software.

Sendo assim, a mensuração é aplicada no processo de desenvolvimento de software ou atributos de um produto com o objetivo de melhorá-lo de forma contínua, que utilizando ao longo do projeto auxilia na estimativa, no controle de qualidade, na avaliação da produtividade e no controle do projeto (PRESSMAN, 2006).

Ao estimar esforço, duração e custo de projetos de desenvolvimento de software, o tamanho do software é um pré-requisito. Semelhante ao uso de um plano de piso (tamanho em pés quadrados) como base para estimar um projeto de construção de edifícios, o tamanho funcional de software a ser entregue é uma base sólida para estimar um projeto de desenvolvimento de software. Uma das maneiras mais comuns de se obter o tamanho funcional de um software é através da Análise de Pontos de Função (APF).

Combinando o tamanho de um sistema em pontos de função por exemplo com muitas outras métricas, pode-se realizar estimativas para o projeto de desenvolvimento, e definir um plano de ações voltado para o atendimento das várias metas (DIAS, 2003).

1.1 Justificativa

O sucesso do desenvolvimento ágil de software e as estimativas ainda continuam desafiando projetos e organizações nos dias atuais. Apesar da importância das métricas e estimativas para projetos de desenvolvimento de software, pesquisas relacionadas ao tema no contexto de projetos ágeis ainda permanecem escassas, o que torna as estimativas e o planejamento ineficiente e/ou impreciso (KITCHENHAM, 2009).

Para um gerente de projetos de desenvolvimento de software, o desconhecimento de quantitativos como o prazo de duração do projeto, alocação de recursos e o esforço a ser empregado é, no mínimo, preocupante. Um erro nesses quantitativos pode levar ao completo caos (WEBER KIVAL CHAVES;ROCHA, 2001).

A análise dos resultados obtidos com este estudo pode indicar quais os modelos e métricas mais adequadas para o desenvolvimento ágil de software. Deve ser considerada, ainda, a importância teórica deste estudo, em virtude da escassez de literatura. Os livros e artigos encontrados sobre o assunto são reduzidos.

Esse trabalho de conclusão de curso contribuirá para a comunidade ágil, mais especificamente do Scrum, no levantamento das melhores métricas utilizadas como insumos para realizar estimativas que ajudem no planejamento de projetos ágeis de desenvolvimento de software.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral deste trabalho é realizar um estudo de caso em uma fábrica de software para comparar a precisão dos dois principais métodos de estimativas de projetos ágeis de software encontrados através da execução de uma revisão sistemática (RS).

1.2.2 Objetivos Específicos

Para atingir o objetivo geral, foram definidos os seguintes objetivos específicos:

- Identificar através da revisão sistemática quais métodos e métricas de tamanho são utilizadas quando se desenvolve sistemas utilizando a metodologia Ágil;
- Mapear as métricas de tamanho e os métodos que podem servir de insumo para estimar e planejar o desenvolvimento dos sistemas utilizando métodos ágeis;
- Analisar a precisão dos dois principais métodos de estimativas encontrados na RS através de um estudo de caso em um projeto de desenvolvimento de uma fábrica

de software para verificar qual dos dois métodos produz estimativas que mais se aproximam dos valores reais.

1.3 Metodologia de Pesquisa

Para reunir evidências e responder as questões de pesquisa deste trabalho optou-se pela realização de uma Revisão Sistemática (RS). Ela que tem como objetivo responder as seguintes questões de pesquisa:

(QP1) Quais os métodos e métricas utilizadas para fazer estimativas de esforço, prazos e custos para o planejamento do projeto de software ágil?

(QP2) Métricas de ponto de função podem ser utilizadas para realizar estimativas de esforço, prazos e custos para o planejamento de software ágil? Caso afirmativo, é a estimativa mais apropriada?

(QP3) Quais as métricas adotadas para verificar a produtividade da equipe em desenvolvimento ágil de software?

(QP4) Que tipo de estudos na área de métricas estão sendo feitos no ambiente ágil de desenvolvimento de software pela academia, e quais resultados estão sendo obtidos?

Definimos o processo de estudo da RS com base em diretrizes já existentes de (KITCHENHAM; CHARTERS, 2007) e (PETERSEN; FELDT; MUJTABA; MATTSSON, 2008). Este processo compreende três fases principais: planejamento, execução e documentação da revisão.

Os trabalhos que forem encontrados nas fontes de busca passarão por um processo de seleção sistêmico feito com base em critérios específicos previamente definidos, assim, os artigos que atenderem à todos os critérios farão parte da base de estudos da revisão sistemática.

Depois do mapeamento das métricas e métodos de estimativas utilizados em projetos ágeis de software feito na RS, será feita uma avaliação dos dois principais métodos através de um estudo de caso em uma fábrica de software para determinar qual deles mais se aproxima das estimativas reais de um projeto de desenvolvimento de software.

No Capítulo 3 será descrito o passo-a-passo para a condução deste trabalho. Além disso, será apresentado o detalhadamente da revisão sistemática e as questões de pesquisas abordadas, bem como o detalhamento da execução da revisão e do estudo de caso.

1.4 Organização do Trabalho

Este trabalho está organizado como segue. No Capítulo 2 é apresentado o Referencial Teórico, contendo os conceitos necessários para o entendimento deste trabalho.

O Capítulo 3 apresenta a metodologia de desenvolvimento aplicada na elaboração deste trabalho, bem como o detalhamento da RS e sua execução. Também apresenta o planejamento e a execução do Estudo de Caso. O Capítulo 4 apresenta os resultados obtidos com a realização deste trabalho.

O Capítulo 5 apresenta a conclusão deste trabalho e mostra quais os objetivos alcançados com a elaboração do mesmo, bem como os trabalhos futuros.

2 Referencial Teórico

Este capítulo apresenta conceitos importantes a respeito de metodologias ágeis de desenvolvimento de software, especialmente o Scrum, métricas de software, Análise de Pontos de Função, e planejamento e estimativas de projetos de software. As informações apresentadas foram adquiridas através de pesquisas bibliográficas e tem como objetivo fundamentar os resultados encontrados bem como prover um melhor entendimento a cerca do objeto de estudo deste trabalho.

2.1 Metodologias Ágeis de Desenvolvimento de Software

2.1.1 Definição

O conceito de desenvolvimento ágil foi proposto em 2001 pelo *Agile Team*, e então vários times de desenvolvimento de software e empresas reconheceram e aceitaram o conceito, sendo assim, seu uso aumentou gradualmente em projetos de desenvolvimento de software (LI, 2008).

Os métodos para o desenvolvimento ágil de software constituem um conjunto de práticas para desenvolvimento de software que foram criadas por profissionais experientes. Esses métodos podem ser vistos como uma reação ao método tradicional de desenvolvimento, o qual enfatiza uma abortagem racionalizada e baseada em engenharia em que se afirma que os problemas são totalmente especificáveis e que soluções ótimas e previsíveis existem para qualquer tipo de problema (DYBA; DINGSOYR, 2008).

2.1.2 Manifesto Ágil

Em 2001, um grupo de profissionais publicou o Manifesto Ágil (FOWLER; HIGHSMITH, 2017) o qual trouxe uma nova era ao desenvolvimento de software. Nesse manifesto estão os princípios do desenvolvimento ágil de software, que são:

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

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

Esses princípios geraram um número de práticas que acredita-se que agregam um maior valor ao cliente. No centro destas práticas está a idéia de um time auto-organizável onde os membros não só estão colocados, mas também trabalham à um ritmo que sustenta sua criatividade e produtividade. Os princípios encorajam práticas que acomodam mudanças nos requisitos em qualquer estágio do processo de desenvolvimento. Além disso, o(s) cliente(s) está(ão) envolvido(s) ativamente no processo de desenvolvimento, facilitando o feedback que levam à resultados mais satisfatórios. Os princípios não são uma definição formal de ágil, mas são diretrizes para prover software de alta qualidade de uma maneira ágil(DINGSOYR; NERUR; BALIJEPALLY; MOE, 2012).

Segundo (LI, 2008), existem 12 princípios por trás do Manifesto Ágil, são eles:

1. **Nossa maior prioridade é satisfazer o cliente através de entregas rápidas e contínuas de software:** Desenvolvimento ágil é um método iterativo que utiliza entregas de software de forma incremental. Entregas contínuas refletem o processo iterativo do devenvolvimento ágil.

2. **Requisitos mutáveis, mesmo em estágio avançado no processo de desenvolvimento:** Mesmo estando em um estágio mais avançado no processo de desenvolvimento, os requisitos podem sofrer mudanças adequadas o que faz com que o software tenha um maior valor para o cliente.

3. **Entregas frequentes de software funcional, em um intervalo de semanas ou de meses, dando prferência à menor escala de tempo possível:** Iterações com um período de tempo mais curto podem garantir uma relação mais frequente com o cliente. Em cada iteração, o time do projeto deverá entregar novas funcionalidades ou evoluções com base na iteração anterior.

4. **Os profissionais da área de negócio e de desenvolvimento devem trabalhar juntos diariamente durante o projeto:** As informações podem inevitávelmente sofrer distorções durante sua transmissão. Quando o profissional de negócio descreve um requisito do cliente, os desenvolvedores podem cometer enganos. Sendo assim, durante o processo de desenvolvimento, pessoas de negócio e desenvolvedores necessitam de uma interação mais frequente para detectar problemas o mais cedo possível.

5. **Desenvolva projetos com profissionais motivados. Dê à eles um ambiente e o suporte que eles necessitam, e confie neles para concluir o trabalho:** Somente com confiança, motivação e suporte os profissionais desempenharão seu trabalho com todo o seu potencial.

6. **O método mais eficiente e efetivo de transmitir uma informação dentro de um time de desenvolvimento é conversando cara a cara:** Em um time

muito grande, ao invés de transmitir conhecimento através de documentos, é mais eficiente e efetivo a comunicação entre os membros do time cara a cara. Geralmente os times de desenvolvimento ágil, giram em torno de 7 à 10 pessoas, o que faria com que o esforço e tempo gastos na escrita de um documento serem um desperdício.

7. **Software funcional é a principal maneira de medir o progresso:** Em desenvolvimento ágil, o objetivo de cada iteração é entregar um incremento de software funcional, então a medição de progresso não é feita através do número de linhas de código escritas, ou do número de testes escritos, mas da quantidade de funcionalidades testadas e aprovadas.

8. **O processo ágil promove um desenvolvimento sustentável. Os patrocinadores, desenvolvedores, e usuários devem ser capazes de manter um rítmo constante de trabalho:** Em empresas de desenvolvimento de software, horas extras de trabalho são comuns. Entretanto, desenvolvimento ágil se opõe à essa prática, pois horas extras de trabalho farão com que os profissionais fiquem fadigados e entediados, o que diminuirá a eficiência do trabalho.

9. **Atenção contínua para a excelência técnica e um bom design melhora a agilidade:** Em desenvolvimento ágil é necessário responder positivamente às mudanças, a comunicação entre os membros do time também é importante, as pessoas prestam mais atenção para um bom design e tecnologia.

10. **Simplicidade - a arte de maximizar a quantidade de trabalho não concluído - é essencial:** Times ágeis defendem que todo mundo deveria se atentar em qual é a maneira mais fácil de resolver um determinado problema. Times ágeis nao defendem a utilização de tecnologias complexas na implementação de software.

11. **As melhores arquiteturas, os melhores requisitos, e designs emergem de times auto - organizáveis:** Os times auto - organizáveis são capazes de se comunicar positivamente uns com os outros a fim de formar uma cultura e uma ética de trabalho comuns para todo o time. Eles não necessitam de muitas instruções, isso permite aos membros do time ter mais confiança em seu trabalho.

12. **Em intervalos regulares, o time reflete sobre como se tornatem mais efetivos, e então ajustam seu comportamento de acordo com o que foi refletido:** Durante o processo de desenvolvimento, não só os requisitos poderão sofrer mudanças, mas também podem ocorrer mudanças no time de desenvolvimento. É difícil trabalhar de forma ágil seguindo um plano de trabalho pré - definido. É necessário que em determinados intervalos de tempo, o time reflita sobre seu trabalho e faça os ajustes apropriados.

2.2 Scrum

De acordo com (SCHWABER; SUTHERLAND, 2013), o Scrum é um framework no qual as pessoas podem resolver problemas adaptativos complexos, de forma produtiva e criativa, desenvolvendo produtos com o mais alto valor possível.

De acordo com o Guia (SCHWABER; SUTHERLAND, 2013), o Scrum é:

- Leve.
- Simples de entender.
- Difícil de ser gerenciado.

O Scrum é um framework de processo utilizado para gerenciar desenvolvimento de produtos complexos desde a década de 1990. O Scrum não é um processo ou uma técnica para desenvolver produtos, mas é um framework em que se podem empregar vários processos e técnicas. O Scrum deixa claro a eficácia relativa de suas práticas de desenvolvimento e gerenciamento de produtos para que você possa melhorá-los.

O Framework Scrum consiste em um *Time Scrum* e seus respectivos *papéis, eventos, artefatos e regras*. Cada componente dentro do framework têm um objetivo específico e é essencial para o sucesso do Scrum e seu uso.

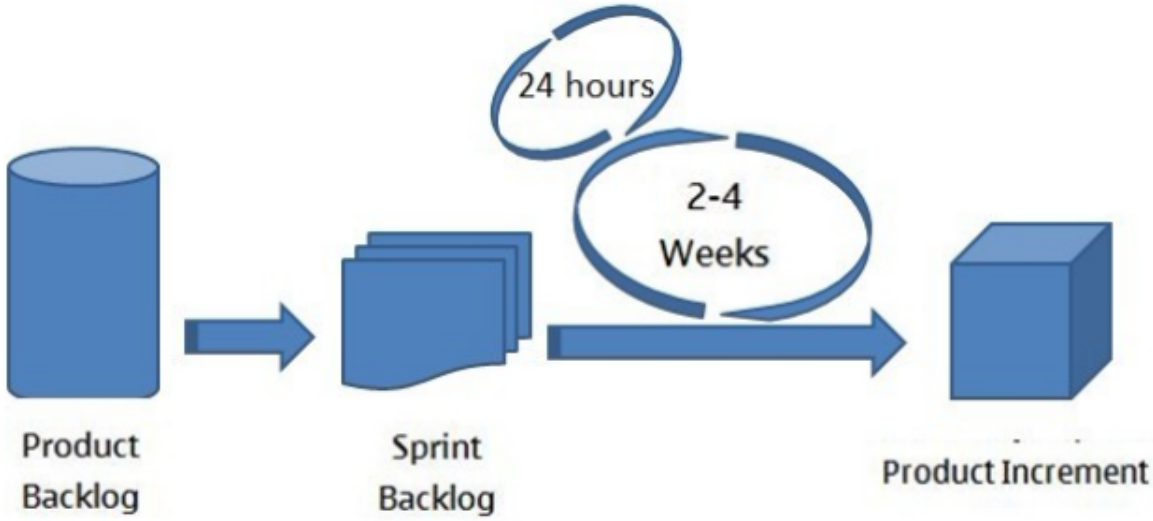

Figura 1 – Visão Geral do Processo do Scrum (LI, 2008)

.

Segundo (LI, 2008), a idéia básica do Scrum é desenvolver software como um desenvolvimento de um novo produto, não se pode definir no começo do processo todas as especificações finais do produto. O processo requer pesquisas, criatividade, tentativa e erro, pois não existe um processo para o sucesso de um projeto. O Scrum tem um objetivo

final claro, o qual é necessário para o processo de desenvolvimento. Com alta autonomia e flexibilidade, comunicação direta e cooperação para resolver os problemas. Garantir que todos os dias, cada estágio tenha um incremento claro. Sendo assim, o Scrum é ideal para projetos de desenvolvimento de produtos.

O processo de desenvolvimento do Scrum é um ciclo iterativo de 2 à 4 semanas, como apresentado na Figura 1. Cada iteração é chamada de Sprint. Um novo produto começa a ser desenvolvido a partir de uma requisição do cliente. O time de desenvolvimento deve se esforçar para gerar resultados à cada Sprint. O time tem uma reunião diária de normalmente 15 minutos para relatar o progresso de cada membro, a fim de entender as dificuldades e tentar resolvê-las, então decisões são tomadas para organizar as tarefas do dia seguinte (LI, 2008).

De acordo com (SCHWABER; SUTHERLAND, 2013), a equipe Scrum é composta por:

- Proprietário do Produto (PO), única pessoa responsável por gerenciar o Product Backlog.
- Desenvolvedores, que consiste em um grupo de profissionais que constroem um incremento funcional do produto a cada sprint
- Scrum Master, que é o responsável por garantir que o Scrum foi entendido, avaliando se o time está aderindo à teoria, às práticas e às regras do Scrum (SCHWABER; SUTHERLAND, 2013)

As equipes Scrum são auto-organizadas e multifuncionais, ou seja escolhem a melhor maneira de realizar o trabalho e possuem todas as competências necessárias para realizar o trabalho sem depender de outras pessoas fora do time.

2.3 Extreme Programming

O Extreme Programmin (XP) é uma metodologia de desenvolvimento ágil. XP define o desenvolvimento de software através de valores e práticas pensadas para trabalhar juntos na prática. No XP, um bom gerenciamento de projeto e um envolvimento constante do cliente são cruciais para o sucesso do projeto. No entanto, o XP provê pouco suporte para o gerenciamento de projeto e o cliente está constantemente sob pressão durante o processo de desenvolvimento (VALKENHOEF; TERVONEN; BROCK; POSTMUS, 2011).

O XP é motivado por 2 elementos cruciais: comunicação efetiva entre os entre as pessoas envolvidas no projeto e a divisão de responsabilidades entre as pessoas da área

de *business* e as pessoas da área técnica. Os processos e práticas seguidas pelo XP são baseados em um conjunto de princípios que enfatizam a colaboração, receptividade ao feedback, respeito e honestidade. Os processos e práticas do XP foram projetados para apoiar o compartilhamento de informações (WOIT; BELL, 2014).

O XP pertence à família ágil de metodologias de desenvolvimento, e portanto, possui um processo iterativo e incremental. O projeto é composto por um número de *releases*(versões), e cada release é implementada dentro de um número de iterações. As releases ocorrem com frequência, normalmente a cada um à três meses. Cada release provê um incremento na funcionalidade do sistema, e constitui um sistema de trabalho completo. Uma nova funcionalidade é incorporada frequentemente, normalmente várias vezes ao dia(WOIT; BELL, 2014).

Cada release é regida por um Plano de Release (ou Release Plan), o qual é estabelecido pelos desenvolvedores, pelo cliente e pelos gerentes. O(s) cliente(s), com o auxílio dos desenvolvedores, seleciona um grupo de histórias para serem inclusas na próxima release. Esta seleção é feita com base no escopo (funcionalidade desejada) ou tempo (histórias que são possíveis de serem implementadas em um determinado espaço de tempo). As histórias de usuário para uma determinada iteração são selecionadas pelo cliente dentre as histórias da release corrente (WOIT; BELL, 2014).

Uma prática obrigatória para o XP é o Daily Meeting (reunião diária), onde todos os desenvolvedores e pessoas do negócio devem participar. Como parte da reunião, os desenvolvedores fornecem atualização sobre o andamento das tarefas e registra essas atualizações em um quadro (WOIT; BELL, 2014).

2.4 Métricas e Medição de Software

A base para as métricas de software foi estabelecida nos anos setenta. O primeiro artigo sobre o assunto foi publicado em 1968 (RUBEY; HARTWICK, 1968). A partir desses trabalhos, surgiram mais trabalhos a respeito e resultados interessantes.

As medições foram criadas principalmente para garantir que indicativos pudessem ser obtidos e assim houvesse uma otimização dos custos de produção já que na década de 90, bilhões de dólares eram gastos em softwares que não atendiam as empresas da época (S., 2011).

(DEMARCO, 1982) afirma "Você não pode controlar o que você não pode medir."Métricas e medições são essenciais para obter um controle significativo sobre o processo de software ao longo do ciclo de vida. Consequentemente, precisa-se de um conjunto de diferentes métricas para se obter informações sobre o produto e sobre como o projeto está se desenvolvendo.

Neste trabalho, a definição escolhida para métricas de software é a dada por (DEMARCO, 1982): "Uma métrica é uma indicação mensurável de algum aspecto quantitativo de um sistema."

Neste contexto, a palavra "sistema"refere-se a ambos: o produto desenvolvido e o processo de desenvolvimento em todo o ciclo de vida. Isso nos leva às seguintes definições utilizadas por (DCONTE; DUNSMORE; SHEN, 1986):

- As métricas de processo quantificam os atributos do processo de desenvolvimento e do ambiente de desenvolvimento (por exemplo, o custo do desenvolvimento);
- As métricas do produto são medidas do produto de software (por exemplo, o número de páginas de documentos do projeto).

Ainda segundo o (IEEE., 1990), podemos separar as métricas em alguns aspectos:

- **Atributo** - Propriedade física ou abstrata mensurável de uma entidade.
- **Medição** - Ato ou processo de atribuir um número ou uma categoria a uma entidade para descrever aquela entidade.
- **Medida** - Um número, extensão ou quantidade que resulta de uma medição.
- **Medir** - Aplicar uma métrica, ou atribuir valor por comparação com uma norma.

A lista de áreas e ocasiões em que são necessárias diferentes métricas é muito longa. Já mencionamos planejamento e controle de projetos de software. Outros exemplos típicos são contratos ou responsabilidades, especificações de requisitos, garantias, estimativa de custos, escolha de métodos, escolha de técnicas básicas, garantia de qualidade, testes, etc. As métricas também podem ser usadas para avaliar projetos e produtos, respondendo a perguntas como:

1. Por que nosso projeto custou três vezes mais do que o previsto?
2. Por que nosso produto contém dez vezes mais falhas do que o previsto?

As Métricas de Software devem ser a rigor práticas, com baixo custo e confiáveis (COSTA; HENRIQUE; SÉRGIO; LISBOA, 2011). Elas se dividem basicamente em diversos tipos, dentre eles:

- Métricas Diretas;
- Métricas Indiretas;

- Métricas Orientadas a Tamanho;
- Métricas Orientadas a Função;
- Métricas de Produtividade;
- Métricas de Qualidade e Métricas Técnicas.

Os focos deste trabalho são as métricas orientadas a tamanho e métricas orientadas a função, principalmente Pontos de Função, onde o objetivo é saber como são utilizadas dentro de um contexto ágil de desenvolvimento de software. Na Seção 2.4.1, será explicado com mais detalhes o que são essas métricas e aprofundaremos os estudos em um dos métodos de medição mais utilizados para dimensionar o tamanho de um software, a Análise de Pontos de Função (APF).

2.4.1 Métricas orientadas a função

2.4.1.1 Análise de Pontos de Função

A Análise de Pontos de Função (APF) é um método padronizado que determina o tamanho do software através de seus requisitos funcionais, considerando as funcionalidades a serem implementadas. Foi desenvolvido para ser aplicado independente da linguagem de programação e tecnologias utilizadas (JUNIOR MARCOS ; FANTINATO, 2015).

2.4.1.1.1 International Function Point User Group - IFPUG

Organização criada em 1986 como uma organização sem fins lucrativos para promover e disseminar a gestão eficaz do desenvolvimento de software e manutenção por meio da Análise de Pontos de Função. O IFPUG é atualmente a agência reguladora da APF, responsável pela melhoria e desenvolvimento das regras estabelecidas no Counting Practices Manual(CPM) (JUNIOR MARCOS ; FANTINATO, 2015).

A APF é bastante utilizada como uma referência para derivar outras medições como a de esforço de desenvolvimento, produtividade ou custo. A principal relevância da APF é vista em casos no Brasil, onde o Governo Federal determinou que todos os desenvolvimentos de software devem cumprir uma métrica específica. Além disso, o governo recomenda que os membros do SISP - Sistema de Administração dos Recursos de Tecnologia da Informação apliquem a APF (JUNIOR MARCOS ; FANTINATO, 2015).

2.4.1.1.2 Processo de Contagem de Pontos de Função

De acordo com (JUNIOR MARCOS ; FANTINATO, 2015), o procedimento para aplicar a APF inclui os seguintes passos:

1. ***Definir limite de contagem, escopo e propósito:*** O limite de software é definido estabelecendo uma borda lógica entre o software a ser medido, seus usuários e outros sistemas de software. O limite pode ser subjetivo, consequentemente, a dificuldade de se limitar onde um software termina e onde o outro começa pode ser alta.

2. ***Medir funções de dados:*** Uma função de dados atende aos requisitos funcionais relacionados ao armazenamento ou referência de dados.

3. ***Medir funções transacionais:*** Uma função transacional é a funcionalidade provida pelo software ao usuário para processamento de dados.

4. ***Calcular o tamanho funcional:*** Fórmulas diferentes podem ser utilizadas para calcular o tamanho funcional considerando o tipo do projeto, o escopo e o propósito da contagem: desenvolvimento, melhoria, ou aplicação.

5. ***Documentar a contagem e divulgar os resultados:*** Salvar as premissas e interpretações durante a contagem assegura o rastreamento do tamanho funcional resultante da contabilização usando a APF.

2.5 Planejamento e Estimativas de Software

Ao se calcular métricas, pode-se aperfeiçoar uma das tarefas mais importantes da Gerência de Projetos que é o planejamento. Segundo (S., 2011) a medição de software torna possível que gerentes e profissionais entendam melhor o processo de engenharia de software e o produto (software) que ele produz. Usando medidas diretas e indiretas, as métricas de produtividade e qualidade podem ser definidas. Pose-se identificar também o esforço, custo e tempo estimados para um empreendimento do projeto.

Sendo assim, a estimativa é uma das principais atividades do planejamento de software. Elas fornecem dados que permitem prever o tempo necessário e os custos do projeto. Não é possível elaborar cronograma e orçamento sem o uso de estimativas. Geralmente, estima-se (ABRAN; MOORE, 2004):

- O número de horas-pessoa necessárias para completar o desenvolvimento ou manutenção (esforço).

- A duração das tarefas com horários de início projetados, duração individual por tarefa e horários finais.

- O custo do projeto com base nas necessidades de recursos, como pessoas ou ferramentas.

Estimativas são realizadas com base em métricas. Com a aplicação de estimativas é possível coletar métricas que permitam prever a quantidade de pessoas necessárias, o tempo necessário e os custos para o desenvolvimento do projeto. "Assim, torna-se importante o investimento na implantação de um processo de estimativas"(HAZAN, 2008). Métricas de tamanho, duração, produtividade e esforço estão entre as mais utilizadas (ABRAN; MOORE, 2004).

A estimativa pode ser realizada também por um engenheiro de software ou especialista que analisa os requisitos garantindo a qualidade, e então estima o tamanho do projeto de software. Após a definição do tamanho, é realizada a derivação para determinação do esforço, o prazo (cronograma) e custo (orçamento). Caso ocorram mudanças nos requisitos é necessário re-estimar (HAZAN, 2008).

Entretanto fatores como: custo e tempo de desenvolvimento que são estimados usando-se apenas uma experiência do profissional envolvido, são normalmente imprecisos e isso, muitas vezes, acarreta desperdício de recursos, descumprimento de prazos, o que gera prejuízo para o desenvolvedor e atraso na entrega do produto ao cliente (ZADRA; CARVALHO; CARDOSO, 2014).

A Figura 2 apresenta as principais etapas para a realização de estimativas de projetos:

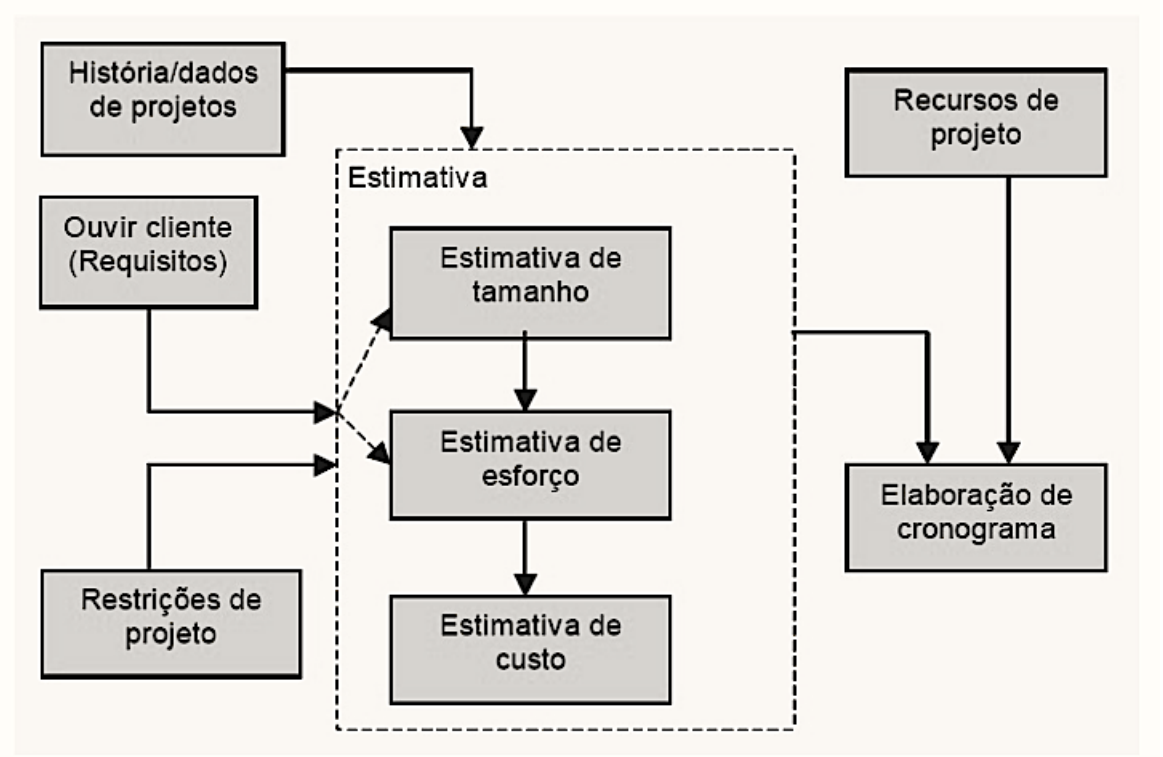

Figura 2 – Principais Etapas para Realização de Estimativas (FILHO, 2014)

.

As próximas Seções discutem de maneira mais detallhada os principais conceitos

relacionados a cada uma das estimativas, as quais são extremamente dependentes uma das outras. Também será apresentado um processo geral de como essas estimativas são feitas.

2.5.1 Estimativas de Esforço

A gestão bem sucedida de projetos de software começa com uma estimativa precisa do esforço de desenvolvimento (S., 2011) (SOMMERVILLE, 2007). Imprecisão nas estimativas continua a ser um dos fatores-chave que contribuem para falhas de projeto de software (S., 2011). Projetos de software gastam 30 a 40 por cento mais esforço do que é estimado pelos modelos existentes (MOLOKKEN; JORGENSEN, 2003). Subestimar o esforço a ser gasto gera pressões que comprometem a qualidade do desenvolvimento do projeto, enquanto que, superestimar pode elevar o custo e diminuir a competitividade.

O esforço é entendido como a quantidade de horas de trabalho a ser executada nas atividades de desenvolvimento do projeto que entregará, ao seu final, o produto/serviço de software, podendo ser apurado utilizando-se diversos métodos que estão disponíveis no mercado (ABRAN; MOORE, 2004).

A popularidade do uso de Pontos de Função mostrou que ele poderia ser colocado em modelos de predição de esforço em vez do tradicional LOC (Linhas de Código) porque o processo de computação de Pontos de Função é mais confiável e preciso do que LOC. As vantagens da Análise por pontos de função motivaram outros pesquisadores a inventarem novos modelos de estimativas baseados em Pontos de Função como (KEMERER, 1987), (MATSON, 1994). A introdução de uma nova versão do COCOMO (COnstructive COst MOdel), nomeado de COCOMO II em 2000 (BOEHM, 2000), é um acontecimento significativo neste domínio. COCOMO II é considerado o modelo mais detalhado para realizar estimativas de esforço.

A vantagem destes métodos reside no facto de serem independentes da tecnologia ou da linguagem de programação utilizadas e podem ser utilizadas durante todo o ciclo de vida do desenvolvimento (TRENDOWICZ; JEFFERY, 2014). Com os métodos da Análise de Pontos de Função podemos estimar o tamanho de uma aplicação de software e, portanto, o esforço de desenvolvimento da mesma no início do processo de desenvolvimento, o que pode não ser o caso com outros métodos.

2.5.2 Estimativas de Custo

A literatura apresenta várias técnicas para a estimativa de custos de software e não há um consenso sobre a totalidade das técnicas. A maioria dos autores entretanto, concorda com a utilização destas três: Julgamento Especialista(ou Parecer Técnico), Analogia e Modelos Algorítmicos.

Dentre os modelos Algorítmicos se destacam : Linhas de código (ALBRECHT; GAFFNEY, 1983), Modelo de Estimativa de Putnam (PUTNAM, 1978), Modelo de Custo Construtivo (COCOMO) (**??**), Pontos de Função (ALBRECHT, 1979), Pontos de Particularidade (JONES, 1998), Ciência do Software de Halstead(KAN, 1995) e Número Ciclomático de McCabe (KAN, 1995).

Cada uma dessas técnicas apresenta vantagens e desvantagens e, tanto (BOEHM, 1981), quanto (SHEPPERD; SCHOFIELD; KITCHENHAM, 1996), afirmam que se deve usar mais do que uma técnica, por exmplo, parecer técnico e um dos modelos algorítmicos. Não fica claro, porém qual deve ser utilizado.

Geralmente o primeiro passo na estimativa do custo de desenvolvimento de um dado projeto é estimar o volume da aplicação a ser desenvolvida. Em outras palavras, quão grande é esse esforço. Há uma variedade de abordagens para estimar o volume.

O próximo passo é ajustar esta estimativa para fatores ambientais específicos, como capacidade da equipe, complexidade do problema, ambiente de desenvolvimento, etc. O resultado é uma estimativa do esforço total e do tempo assumindo que o projeto é desenvolvido nesse ritmo ótimo. No mundo real, não é incomum para as pressões do negócio exigir que o projeto seja desenvolvido em um ritmo não-ótimo, tipicamente acelerado. Isso é possível, mas a eficiência cai e os custos aumentam. Na Seção 2.5.3, discute-se abordagens para estimar o tempo.

2.5.3 Estimativas de Tempo

Para determinar o tempo de desenvolvimento, é necessário estimar a duração das atividades do software. Estimativas da duração do software dependem do tamanho do software que é necessário produzir e da produtividade dos profissionais alocados.

No campo de estimativas de software, é prática comum usar o tamanho do software como a variável independente para prever o esforço do projeto e o esforço previsto como variável independente para prever a duração do projeto (AHMED; AHMAD; ALGHAMDI, 2013) (BERLIN; RAZ; GLEZER; ZVIRAN, 2009) (BOURQUE; OLIGNY; ABRAN; FOURNIER, 2007). O tamanho de um produto de software é medido principalmente em pontos de função ou linhas de código fonte (SHEETZ; HENDERSON; WALLACE, 2009), enquanto a duração de um projeto de software é geralmente medida em meses.

2.6 Estimativas de Projetos Ágeis de Software

As metodologias ágeis propõem um vasto conjunto de técnicas para estimar e planejar projetos, principalmente em termos de modelos não baseados em algoritmos (COHN,

2004) (HIGHSMITH, 2009). A maioria das técnicas de estimativa ágil concentram-se no uso de "histórias de usuários"também conhecidas como *User Stories* (US). As Histórias de Usuários foram introduzidas pela primeira vez por eXtreme Programming (BECK; ANDRES, 2004) e, em seguida, popularizado por Mike Cohn (COHN, 2004).

Existem diversas técnicas para estimativa de projetos ágeis de software, entretanto serão discutidas aqui: *Planning Poker* (COHN, 2005), *The planning game* (SHORE; WADEN, 2008), e *Ideal Day* (ALVES; FONSECA, 2008).

2.6.0.1 Planning Poker

Foi inicialmente usado para planejar o trabalho em uma iteração. (COHN, 2005) e (HIGHSMITH, 2009) propõem essa técnica para realizar estimativas no nível do projeto, comparando histórias de usuários em vez de tarefas técnicas. Toda a equipe de desenvolvimento estima um conjunto de recursos nesta técnica. Cada membro da equipe tem um baralho de cartas com um subconjunto discreto de valores (por exemplo 1, 2, 3, 5 e 8), representando os pontos a serem atribuídos a cada recurso. Um a um, o representante do cliente explica os recursos. Uma vez que um recurso é descrito, os membros da equipe podem fazer algumas perguntas para esclarecer seu escopo. Depois disso, cada membro, ao mesmo tempo, mostra um card do seu deck com a estimativa. Se elas são todas iguais, o recurso recebe a estimativa; Se não, os membros que propõem a estimativa mais alta e a mais baixa explicam seus pontos de vista e novas rodadas são jogadas até chegar a um consenso. Então, a equipe prevê a velocidade (ou seja, o número de pontos que eles podem entregar em uma iteração, por meio de dados históricos, uma iteração de teste ou uma suposição educada) e estabelece o comprimento da iteração. O número de iterações é obtido pela divisão do número total de pontos pelo *velocity*. Da mesma forma, a duração do projeto é calculada multiplicando o número de iterações pelo tamanho.

2.6.0.2 The planning game

Essa técnica, proposta pela eXtreme Programming, pressupõe que os clientes têm a maior parte das informações sobre o que deve ser desenvolvido e os desenvolvedores têm a maior parte das informações sobre como implementar esses recursos. Os desenvolvedores estimam o custo de cada recurso e os clientes priorizam a importância relativa de cada recurso entre o restante dos recursos. Estas duas etapas são repetidas até que todas as características sejam estimadas e organizadas. Durante o processo, desenvolvedores e clientes interagem para resolver dúvidas sobre prioridades e estimativas. Inicialmente, esta técnica foi utilizada para avaliar o trabalho em uma iteração, mas também pode ser usada para realizar um planejamento de liberação completa (LOGUE; MCDAID, 2008).

2.6.0.3 *Ideal Day*

Muito utilizado em projetos Ágeis de Software,*Ideal Day* corresponde à quantidade de trabalho que um profissional da área consegue concluir em um dia de trabalho (ALVES; FONSECA, 2008).

A velocidade é calculada a partir do número de horas que a equipe gasta para implementar um trabalho equivalente a um *Ideal Day* (MARTINS, 2001). Caso o item passe um dia de trabalho, é sugerido decompor esse item em itens menores que se consiga implementar em apenas um dia.

Segundo (MARTINS, 2001) para efetuar o cálculo dos dias estimados utiliza a seguinte fórmula:

$$DE = \frac{IED}{1 - IEDREAL}$$

Onde:

DE: representa a quantidade de dias estimados para concluir a tarefa;

IED: representa o prazo necessário para implementar o item. Esse prazo é definido pela equipe; IEDREAL: representa o percentual que indica a estimativa de quanto tempo do dia o desenvolvedor ficará dedicado a implantação do item.

2.7 Estudo de Caso

Estudo de Caso é uma modalidade de pesquisa bastante utilizada nas ciências biomédicas e sociais. Atualmente é encarada como a mais adequada para a investigação de um fenômeno contemporâneo dentro de seu contexto real, onde os limites entre o fenômeno e o contexto não são claramente percebidos (YIN, 2013).

De acordo com (RUNESON, 2008), estudo de caso em engenharia de software é um estudo empírico para investigar uma instância (ou um pequeno número de instâncias) de um fenômeno de engenharia de software contemporâneo dentro de seu contexto de vida real, principalmente quando a fronteira entre o fenômeno e o contexto não pode ser claramente especificada.

Ainda segundo (RUNESON, 2008) um estudo de caso compreende as seguintes fases: Planejar e Desenhar o Estudo de Caso, contendo tanto o planejamento do estudo quanto a preparação, em que são realizadas a caracterização do objeto e a definição do protocolo; Coletar Dados, onde as técnicas são empregadas na execução do estudo e o resultados são obtidos; Analisar Dados, transformando-os em informações essenciais para a tomada de decisões; e Relatar Estudo, onde ocorre a redação de todo o processo do estudo do caso selecionado.

3 Metodologia

Este trabalho de conclusão de curso é composto de duas etapas. Na primeira etapa definimos um processo de estudo de revisão sistemática adaptado de (KITCHENHAM; CHARTERS, 2007) (GALSTER et al., 2014).

A segunda etapa do trabalho é a realização de um estudo de caso em um projeto de desenvolvimento ágil de uma fábrica de software. O objetivo é realizar estimativas de tamanho, esforço, praz e custo da primeira release do projeto utilizando duas diferentes técnicas para assim comparar os resultados com os valores reais e identificar quais dos dois mais se aproxima desses valores.

3.1 Revisão Sistemática

Tem-se como objetivo identificar trabalhos científicos que apresentam soluções de métodos e métricas para metodologias de desenvolvimento de software ágil. Em particular, identificar práticas comuns de medição e controle dos projetos, métricas de tamanho utilizadas nos projetos, tendências de pesquisa na área de desenvolvimento ágil, questões abertas e tópicos de pesquisa para melhoria das estimativas dos projetos de desenvolvimento ágil . Além disso, categorizar a evidência dos estudos primários de acordo com um esquema bem definido e estruturado.

Este processo consiste em três fases principais: planejar, executar e documentar a revisão, conforme apresentado na Figura 3.1:

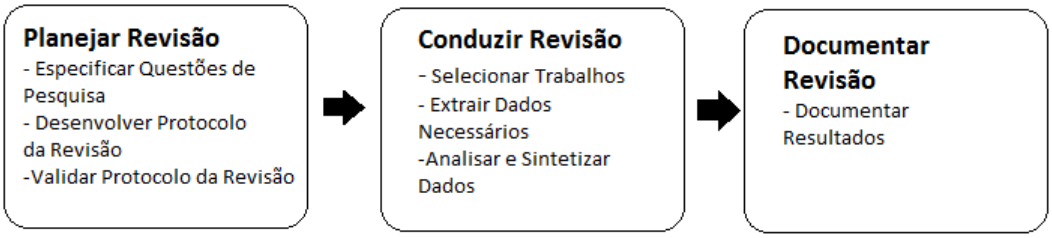

Figura 3 – Visão Geral do Processo da Revisão Sistemática. (Autoria própria, 2017)

A fase de planejamento da Revisão Sistemática (RS) está focada na especificação do protocolo, que descreve atividades sistemáticas para reunir evidências disponíveis. O protocolo é bem detalhado e serve para apoiar os pesquisadores envolvidos durante todo o processo de revisão. Ele fornece uma definição clara de questões de pesquisa, estratégia de busca para reunir estudos relevantes, critérios para inclusão e exclusão de estudos

primários, rastreio de processos de documentos, bem como análise de extração de dados e síntese.

A segunda fase, a fase de Execução da RS, envolve a aplicação do protocolo de pesquisa para buscar estudos, extrair dados e sintetizar o conhecimento relevante relacionado ao tema do trabalho. O resultado desta fase é a evidência gerada a partir de todas as atividades do protocolo. A Figura 4 apresenta o processo de seleção dos estudos.

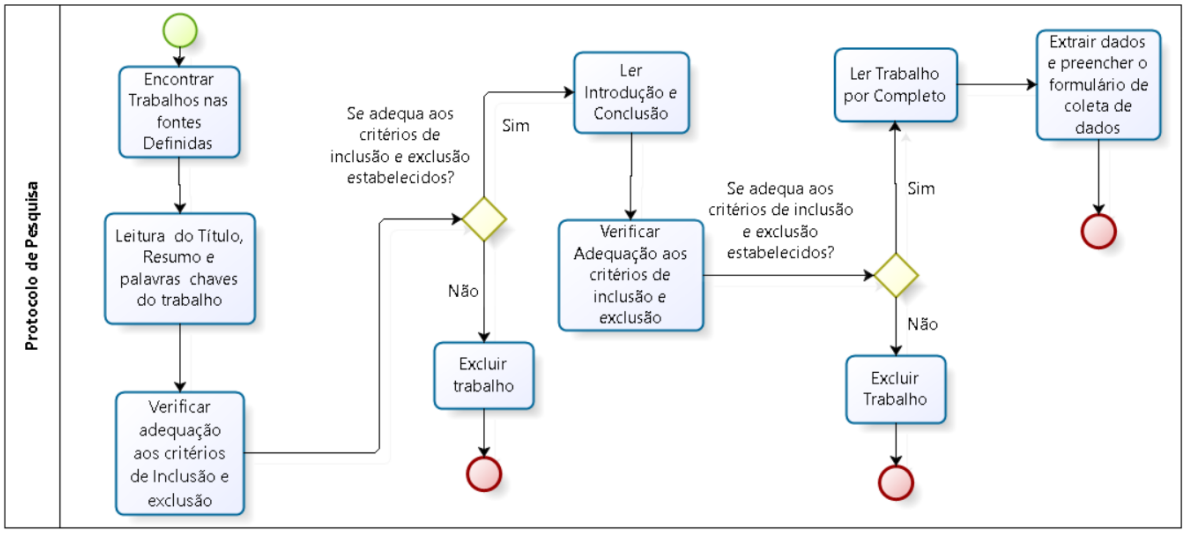

Figura 4 – Visão Geral do Processo de Pesquisa. (Autoria Própria, 2017)

A fase de documentação da revisão relata os resultados do estudo de mapeamento. Nesta fase, os alunos envolvidos consolidaram todas as informações, escreveram o relatório, revisaram e publicaram os resultados.

3.1.1 Questões de Estudo

Esta revisão sistemática de literatura visa responder às seguintes questões:

(QP1) Quais os métodos e métricas utilizados para fazer estimativas de esforço, prazos e custos para o planejamento do projeto de software ágil?

(QP2) Métricas de ponto de função podem ser utilizadas para realizar estimativas de esforço, prazos e custos para o planejamento de software ágil? Caso afirmativo, é a estimativa mais apropriada?

(QP3) Quais as métricas adotadas para verificar a produtividade da equipe em desenvolvimento ágil de software?

(QP4) Que tipo de estudos na área de métricas estão sendo feitos no ambiente ágil de desenvolvimento de software pela academia, e quais resultados estão sendo obtidos?

3.1.2 Estratégia de Busca

Para realizar a busca de trabalhos nas bases de dados definidas é necessário estabelecer uma estratégia de busca.

3.1.2.1 Definição da String de Busca

A definição da *String* de busca é baseada na população, na intervenção, na comparação e no resultado.

População: A população é o desenvolvimento ágil de software. Para procurar a população, usamos as palavras-chave "Agile Software Development".

Intervenção: A intervenção é estimar esforço, custo e prazo, e também identificar como a Análise de Pontos de Função é realizada nos projetos Ágesi de software. Sendo assim os termos: prediction and estimation.

Comparação: O foco do estudo não se limitou a estudos comparativos. Portanto, a comparação não foi considerada na estratégia de pesquisa.

Resultado: Procuramos métricas e métodos para estimativas que sejam utilizados em estudos científicos e ou pela indústria, conforme relatos de pesquisadores. Assim, a cadeia de pesquisa continha palavras como *empirical, validation, evaluation,size, effort etc.*.

A string de busca selecionada para o trabalho é a seguinte:

("AGILE SOFTWARE DEVELOPMENT") **AND** (ESTIMATION OR PREDICTION) **AND** ((EMPIRICAL) OR (STUDY) OR (VALIDATION) OR (EVALUATION) OR (SIZE) OR (EFFORT) OR (SCHEDULE) OR (METRICS))

3.1.2.2 Fontes de Busca

As buscas foram feitas em bases de dados digitais. Foram selecionadas 2 bases, sendo elas:

Periódicos CAPES (http://www.periodicos.capes.gov.br/)

Scopus (https://www.scopus.com/)

Muitas vezes os estudos se repetem por isso a escolha de apenas 2 bases de dados. Scopus por exemplo indexa diversas outras fontes como por exemplo a IEEXplore Digital Library e a Springer, assim como a base de periódicos da Capes, que contém trabalhos científicos de diversas fontes (conferências e Periódicos).

3.1.2.3 Idioma

Para o presente trabalho, serão selecionados preferencialmente artigos na língua inglesa, entretanto trabalhos relevantes encontrados na língua portuguesa também serão considerados.

3.1.3 Seleção dos Estudos

Foram definidos critérios para inclusão e exclusão dos estudos encontrados.

3.1.3.1 Critérios de Inclusão e Exclusão

A inclusão de estudos adotou os seguintes critérios:

1. Os trabalhos devem estar disponíveis em bases de dados digitais previamente definidas, de forma gratuita.
2. Serão considerados trabalhos publicados a partir do ano de 2007. Entretanto pode-se encontrar fontes clássicas com definições (livros com conceitos clássicos ou artigos pioneiros) que também serão considerados.
3. O trabalho deve possuir menções a métricas de software em projetos de desenvolvimento ágil.
4. O trabalho possuir menção a pontos de função em projetos de desenvolvimento Ágil.
5. Trabalhos que propõem métodos ou métricas para realizar estimativas e planejamento de projeto ágil de software.

A exclusão dos estudos considerou os seguintes critérios:

1. Trabalhos publicados antes do ano de 2007 que não sejam de fontes clássicas e renomadas em relação ao tema do trabalho.
2. Trabalhos não disponíveis em bases de dados digitais e/ou que não estejam disponibilizados de forma gratuita.
3. Trabalhos que, forem identificados como fora do escopo e tema do trabalho, que não sejam voltados a métricas de projetos ágeis de software.
4. Trabalhos que não propõem métodos ou métricas para relizar estimativas de projetos ágeis de software
5. Trabalhos duplicados, que foram publicados em mais de uma base de dados.

6. Trabalhos incompletos - publicados como Short Paper, Key notes, resumo de conteúdos de conferências, tutoriais, cursos, workshops, apresentações, biografias, ou publicações sem referências..

3.1.4 Processo de Seleção dos Estudos

Os artigos encontrados nas fontes de busca passarão por um processo de seleção sistêmico feito com base em critérios específicos previamente definidos, assim, os artigos que atenderam à todos os critérios farão parte da base de estudos da revisão sistemática. O objetivo principal é selecionar os artigos que abordam corretamente as questões de pesquisa. Sendo assim, a estratégia de pesquisa definida é:

1. Pesquisa de trabalhos nas fontes definidas utilizando as strings de busca. Para apoio e documentação, a ferramenta denominada StArt será utilizada para extração e estruturação dos dados.

2. Leitura do título, resumo e palavras chaves dos trabalhos aplicando os critérios de inclusão e exclusão definidos neste protocolo. Com base nesses critérios os trabalhos serão pré-selecionados.

3. Leitura da introdução e conclusão dos trabalhos pré-selecionados, aplicando novamente os critérios de inclusão e exclusão.

4. Os trabalhos que forem selecionados no passo anterior deverão ser lidos por completo, e por fim as informações presentes no formulário de coleta de dados deverão ser extraídas destes trabalhos.

3.1.5 Coleta de Dados

A coleta de dados foi dividida em duas etapas, sendo a primeira etapa a realização dos três primeiros passos definidos na Seção 3.1.4, catalogando os artigos selecionados. Na segunda etapa, aplica - se o último passo definido na Seção 3.1.4 para os artigos selecionados para o trabalho.

3.1.5.1 Primeira Etapa da Coleta de Dados

A pesquisa utilizando a *String* definida no protocolo foi realizada nas bases e obteve-se um total de 291 trabalhos, sendo 91% dos artigos da biblioteca digital *Scopus* e 9% dos artigos foram provenientes da biblioteca digital da *CAPES* conforme apresentado na Figura 5.

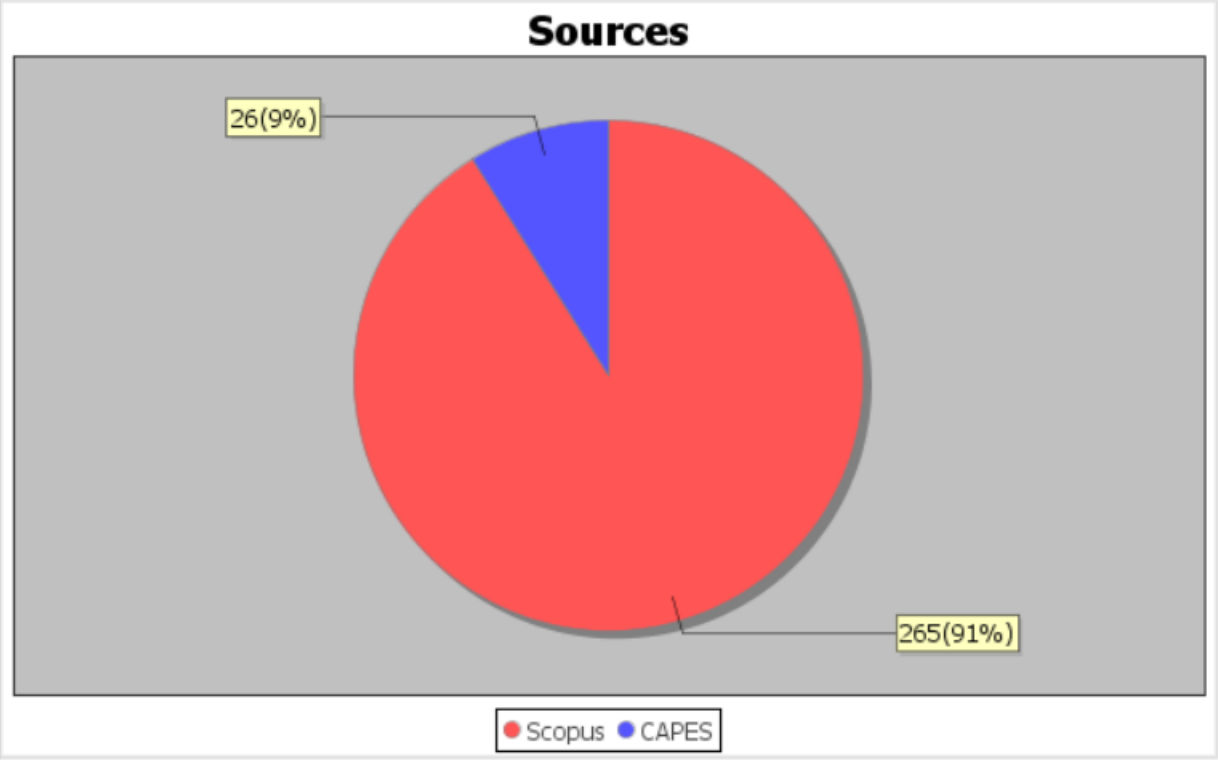

Figura 5 – Quantidade de artigos relevantes por Biblioteca Digital. Autor.

Após a leitura dos títulos, resumos e palavras chaves dos trabalhos, foram classifcados 45 estudos de acordo com os critérios definidos no protocolo. Com o apoio da ferramenta START, também foi classificado a prioridade de leitura do trabalho selecionado de acordo com as partes lidas. A Figura 6 apresenta que inicialmente a maioria dos trabalhos tinha prioridade baixa de leitura:

Figura 6 – Prioridade de Leitura dos Artigos na etapa 1

3.1.5.2 Segunda Etapa da Coleta de Dados

Após a a finalização da primeira etapa com vários artigos selecionados, foi realizada a leitura mais detalhada dos artigos aceitos, e dentre os artigos, somente 27 foram aceitos, pois realmente trouxeram um conteúdo relevante e respostas para as questões de pesquisa. Sendo assim, 58% foram aceitos e 42% foram rejeitados pois tinham conteúdo irrelevante, ou seja fora do escopo do trabalho, conforme apresentado na Figura 7.

A Figura 8 apresenta a prioridade de leitura dos artigos depois da leitura inicial da introdução e conclusão terem sido efetuadas e as Figuras 9 e 10 apresentam quais critérios os artigos aceitos atenderam e quais os critérios de exclusão os artigos rejeitados atenderam. E por fim, a Tabela 1 apresenta todos os artigos selecionados para a extração, com seus respectivos identificadores (IDs) de acordo com a ferramenta *StArt*.

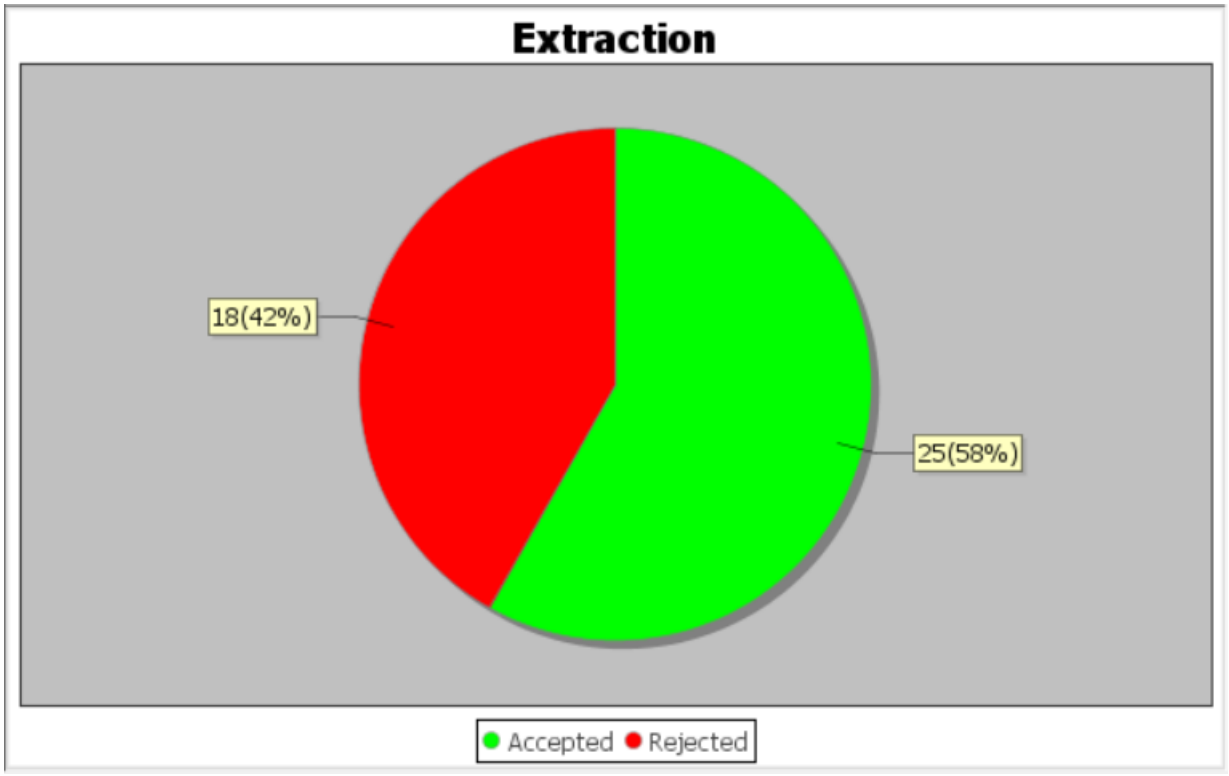

Figura 7 – Porcentagem de artigos aceitos na segunda etapa. Autor.

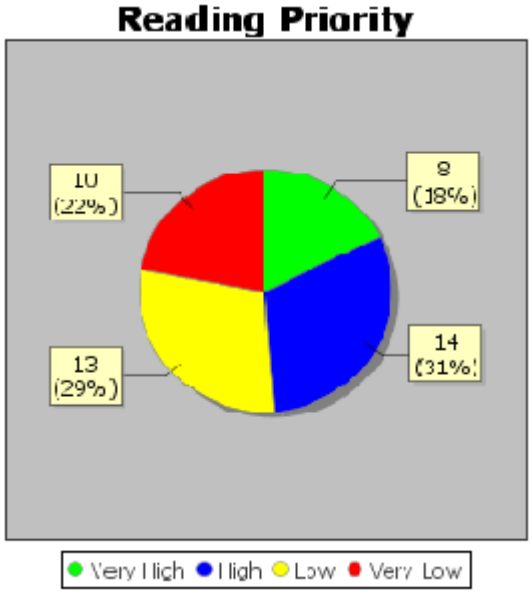

Figura 8 – Prioridade de Leitura dos Artigos na Segunda Etapa

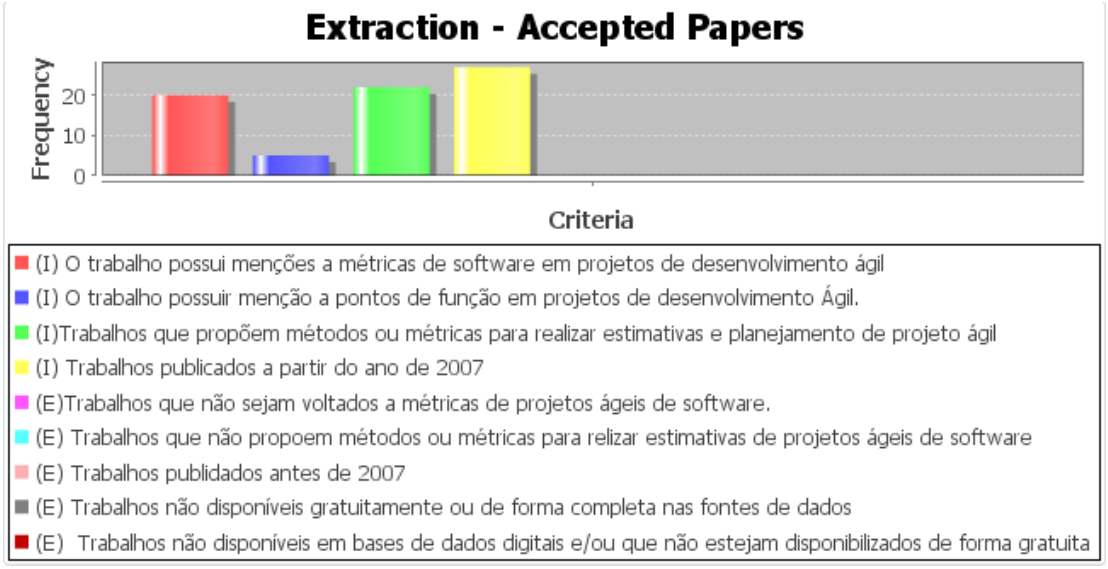

Figura 9 – Critérios de Inclusão dos Artigos Aceitos

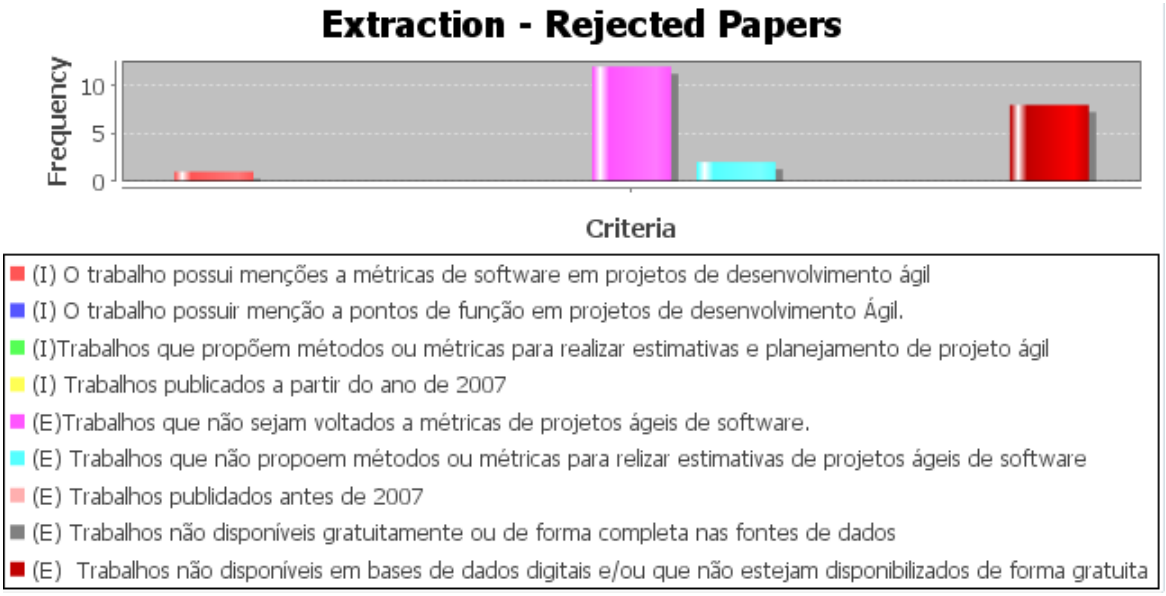

Figura 10 – Critérios de Exclusão dos Artigos Rejeitados

3.2 Estudo de Caso

Nesta seção serão documentadas as etapas do desenvolvimento do estudo de caso adotadas neste trabalho. Na figura 11 é possível visualizar a ordem e as fases do Estudo baseado em (RUNESON, 2008):

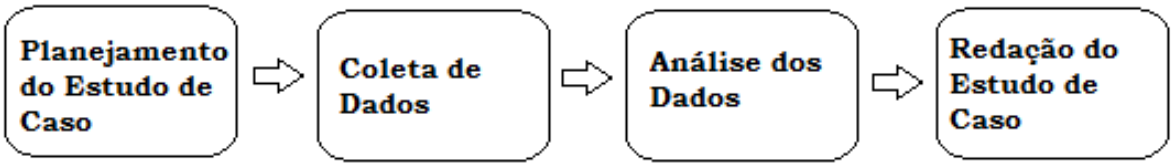

Figura 11 – Principais Etapas para Realização de um Estudo de Caso.

3.2.1 Planejamento do Estudo de Caso

3.2.1.1 Visão Geral do Projeto X

A Basis Tecnologia da Informação S.A. é uma sociedade anônima, brasileira, de capital fechado, com presença nas cidades de Brasília/DF, Belo Horizonte/MG e São Paulo/SP. Atua no mercado de TI com foco em desenvolvimento de soluções personalizadas. A empresa é especializada na prestação de serviços de fábrica de software, tecnologias de ECM – Enterprise Content Management, consultoria, BPM – Business Processes Management, soluções analíticas e outsourcing.

A empresa definiu que Projeto X seria estudado neste trabalho. Por questões contratuais o nome do projeto e informações como especificações e código não serão revelados neste trabalho.

O foco do estudo será a relase 1 do projeto. Uma release de software é definida como um ciclo de desenvolvimento que perpassa sequencialmente as fases de iniciação, de construção e de transição, nessa ordem, com objetivo de entregar, ao final do ciclo, um produto pronto para implantação em produção, que agregue valor ao negócio em questão e que seja passível de ser utilizado pelo usuário final.

O processo do Projeto X é iterativo e incremental e baseado no Scrum. Toda sprint inicia com a Reunião de Planejamento, em que se definem as User Stories (US) e os itens de trabalho a serem desenvolvidos ao longo da iteração. Itens de trabalho planejados e US não entregues ao final da sprint voltam para o Release Backlog e serão utilizados, a

critério do cliente, em planejamento de sprints futuras. Na Reunião de Planejamento são definidos:

- As *User Stories* da sprint;
- Os critérios de aceite;
- Os critérios de teste;
- O valor a ser pago pela sprint
- A autorização do cliente para iniciação das atividades da sprint.

Como padrão e quando não especificado em contrato, considerar-se como dia útil o período de 6 horas úteis. Considerar-se como hora útil o intervalo de uma hora dentro de um dia útil. Como padrão e quando não especificado em contrato, todos os prazos serão contados em horas úteis ou dias úteis.

Os produtos entregues ao final da Sprint serão validados conforme critérios de aceite definidos na Reunião de Planejamento da sprint.

O dimensionamento atual do projeto é feito por meio da técnica de pontos de função não ajustados. As contagens de Ponto de Função seguem as regras estabelecidas no Manual de Práticas de Contagem (CPM) (IFPUG, 2010) versão 4.3.1 (ou superior) do IFPUG, acrescidas das definições do Guia Interno de Contagem do Cliente (versão mais atual) e do Roteiro de Métricas de Software do SISP na versão 2.0 (ou superior) (SISP, 2016).

3.2.1.2 Objetivos do Estudo de Caso

Este estudo tem como objetivos:

1. Estimar o tamanho, o tempo, custo e esforço da release 1 do Projeto X utilizando como base nas duas métricas de tamanho que segundo a RS deste trabalho são as mais utilizadas em Projetos ágeis de software : Ponto de Função e Story Point.
2. Comparar as estimativas realizadas no início do projeto com valores reais do desenvolvimento do projeto para definição de qual das métricas fornece estimativas que mais se aproximam dos valores reais.

3.2.1.3 Questões do Estudo de Caso

O estudo de caso busca responder a seguinte questão de pesquisa:

Q1 - Quais das métricas utilizadas como insumo para estimar tempo, custo e esforço fornece estimativas que mais se aproximam dos valores reais em um projeto ágil de desenvolvimento: Story Points ou Pontos de Função?

3.2.1.4 Fontes de Dados

Todos os dados referentes ao Projeto X são documentados no Redmine, um software livre gerenciador de projetos baseados na web e ferramenta de gerenciamento de bugs. Todas as informações referentes ao backolog, sprints, histórias de usuário, estimativas e outras informações são armazenados nesta plataforma. Logo esta será a principal fonte de dados do projeto. Os autores também participarão de algumas reuniões, de onde poderão extrair informações adicionais através da observação direta do time e perguntas.

3.2.1.5 Procedimentos

O seguintes procedimentos foram adotados para a execução do estudo:

- Análise Documental;
- Entrevistas;
- Observação;

3.2.2 Coleta de Dados

Foram coletados dados referentes ao backlog, valores estimados de SP e PF e da execução das Sprints do Projeto X, necessários para a realização das estimativas do estudo. Nos próximos itens é possível a visualização desses dados.

3.2.2.1 Backlog da Release

Após diversas reuniões do time junto ao cliente, o Backlog para a release 1 do Projeto X foi definido conforme mostra a tabela 2. As User Stories estão em ordem de prioridade. Os valores estimados em SP e PF também estão presentes.

3.2.2.2 Execução das Sprints

A cada sprint finalizada, os dados referentes ao backog da sprint, pontuação em SP e PF User Stories, itens aceitos e recusados, horas de trabalho eram armazenados para comparação posterior com as estimativas realizadas no início do projeto e das sprints. As tabelas mostram como ocorreram as sprints do projeto:

A tabela 3 ilustra a distribuição das histórias e seus respectivos valores de Story Points e Pontos de Função.

Tabela 1 – Artigos Selecionados para Extração de Dados

ID	Artigo Selecionado	Autor	Ano	Base
1543	Measuring and predicting software productivity: A systematic map and review	Petersen, K.	2011	CAPES
1544	iUCP: Estimating Interactive-Software Project Size with Enhanced Use-Case Points	Nunes, N. ; Constantine, L. ; Kazman, R.	2011	CAPES
1547	Enhancing Quality in Scrum Software Projects Points	Mukker, A. ; Mishra, A. K.;	2014	CAPES
1448	On the Current Measurement Practices in Agile Software Development	Javdani, T. ; Zulzalil, H. ; Ghani, A. ; Sultan, A. ; Parizi, R.	2012	CAPES
1552	Survey on agile metrics and their inter-relationship with other traditional development metrics	Misra, S. ; Omorodion, M.	2011	CAPES
1553	Towards a Fuzzy based Framework for Effort Estimation in Agile Software Development	Raslan, A. ; Darwish, N. ; Hefny, H.	2015	CAPES
1562	Function Points, Use Case Points, Story Points: Observations From a Case Study.	Schofield, J.	2013	CAPES
1564	Estimating, planning and managing Agile Web development projects under a value-based perspective	Torrecilla, C.J.; Sedeño, J. ; Escalona, M.J. ; Mejías, M.	2015	CAPES
3800	Effort, duration and cost estimation in agile software development	Owais, M. and Ramakishore, R.	2017	Scopus
3817	Cost and effort estimation in agile software development	Popli, R. Chauhan, N.	2014	Scopus
3826	How is effort estimated in agile software development projects?	Schweighofer, T.; Kline, A.; Pavlic, L.; Hericko, M.	2016	Scopus
3834	Identification of inaccurate effort estimates in agile software development	Raith, F. and Richter, I. and Lindermeier, R. Klinker, G.	2013	Scopus
3841	Effort estimation in Agile software development: A survey on the state of the practice	Usman, M. ; Mendes, E. and Borstler, J.	2015	Scopus
3843	Effort estimation in Agile Software Development: A systematic literature review	Usman, M. ; Mendes, E. Weidt, F. ; Britto, R.	2014	Scopus
3844	Model-based dynamic cost estimation and tracking method for agile software development	Kang, S. Choi, O. Baik, J.	2010	Scopus

3870	NORPLAN: Non-functional requirements planning for agile processes	Farid, W.M. Mitropoulos, F.J.	2013	Scopus
3879	Predicting effort for requirement changes during software development	Basri, S. Kama, N. Haneem, F. Ismail, S.A.	2016	Scopus
3896	Method for personal capability abebment in agile teams using personal points	Celar, S. Turic, M. Vickovic, L.	2015	Scopus
3898	Understanding and improving effort estimation in agile software development- An industrial case study	Tanveer, B. Guzmán, L. Engel, U.M.	2016	Scopus
3900	Agile metrics for a university software engineering course	Matthies, C. Kowark, T. Uflacker, M. Plattner, H.	2016	Scopus
3856	Applying Software Metrics with Scrum	Gamba, M. L. Barbosa, A. C. G	2009	Scopus
3914	On using planning poker for estimating user stories	Mahnic, V. Hovelja, T.	2012	Scopus
3941	Efficiency factor and risk factor based user case point test effort estimation model compatible with agile software development	Parvez, A.W.M.M.	2013	Scopus
3959	Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly Iterative or agile software development processes	Olague, H.M. Etzkorn, L.H. Gholston, S. Quattlebaum, S.	2007	Scopus
3966	Does the use of Fibonacci numbers in planning poker affect effort estimates?	Tamrakar, R. and Jorgensen, M.	2012	Scopus
3973	Improving the user story Agile technique using the INVEST criteria	Buglione, L. Abran, A.	2013	Scopus
3985	Bayesian network model for task effort estimation in agile software development	Dragicevic, S. Celar, S. Turic, M.	2017	Scopus

Tabela 2 – Backlog da Release 1 do Projeto X

User Story	**SP**	**PF**
US014 Manter Eventos de Risco	5	22
US015 Manter Causas de Eventos de Risco	5	22
US016 Manter Conseqüências de Eventos de Risco	5	22
US001 Gerenciar Permissões	8	35
US003 Home Gerenciar Processo	5	32
US002 Autenticação	5	14
US006 Identificação de Eventos de Risco	5	18
US020 Manter Categoria do Risco	3	25
US023 Criar Evento de Risco	13	12
US021 Manter Natureza do Risco	3	25
US017 Manter Controle	3	25
US09 Avaliação de Riscos e Controles	13	18
US019 Manter Operação de Controle	3	25
US018 Manter Desenho de Controle	3	22
US027 Manter Calculadora	5	16
US011 Resposta a Risco	8	12
US010. Validar Etapa Avaliação de Risco - Gestor	5	7
US013. Validar Etapa Resposta e Risco - Gestor	5	7
US022 Manter Glossário	13	28
TOTAL RELEASE	115	387

Tabela 3 – Histórias Desenvolvidas em cada Sprint com seus respectivos valores de Story Points e Pontos de Função

Nome da História	**Pontos de Função**	**Story Points**
[**Sprint 0**]US014: Manter Eventos de Risco administrativa	22	5
[**Sprint 0**]US015: Manter Causas de Eventos de Risco	22	5
[**Sprint 0**]US016: Manter Conseqüências de Eventos de Risco	22	5
[**Sprint 1**]US001: Gerenciar Permissões	35	8
[**Sprint 1**]US003: Home Gerenciar Processo	32	5
[**Sprint 1**]US002: Autenticação	14	5
[**Sprint 1**]MUDANÇA: Alterar as histórias 14, 15 e 16 para exclusão lógica.	9	3
[**Sprint 2**]US006: Identificação de Eventos de Risco	18	5
[**Sprint 2**]US020: Manter Categoria do Risco	25	3
[**Sprint 2**]US023: Criar Evento de Risco	12	13
[**Sprint 2**]US021: Manter Natureza do Risco	25	3
[**Sprint 2**]US017: Manter Controle	25	3
[**Sprint 2**]MUDANÇA: Incluir nas combos de pesquisa por status a opção Todos (US 14, 15 e 16)	9	2
[**Sprint 3**]US09: Avaliação de Riscos e Controles	18	13
[**Sprint 3**]US019: Manter Operação de Controle	25	3
[**Sprint 3**]US018: Manter Desenho de Controle	22	3
[**Sprint 3**]US027: Manter Calculadora	16	5
[**Sprint 3**]MUDANÇA: Não poderá atribuir o mesmo perfil mais de uma vez para o mesmo usuário.	6	2
[**Sprint 3**]US001: Gerenciar Permissões	6	2
[**Sprint 3**][Mudança] caso seja selecionada uma Categoria que não seja Fiscal ou Orçamentária, a natureza deve ser preenchida automaticamente como Não Orçamentária Financeira	6	2
[**Sprint 3**]US006: Identificação de Eventos de Risco	6	2
[**Sprint 3**]US023 Criar Evento de Risco	6	2
[**Sprint 4**]US011: Resposta a Risco	12	8
[**Sprint 4**]US010: Validar Etapa Avaliação de Risco - Gestor	7	5
[**Sprint 4**]US013: Validar Etapa Resposta e Risco - Gestor	7	5
[**Sprint 4**]US022: Manter Glossário	28	13

4 Resultados

4.1 Resultados da Revisão Sistemática

Após a realização da revisão sistemática (RS), obteve-se a respostas para as questões de pesquisa. Este Capítulo apresenta os resultados / respostas encontrados a partir da RS.

4.1.1 Respostas das Questões de Pesquisas – QP

Esta Seção apresenta as respostas encontradas para cada questão de pesquisa definida neste trabalho.

Questão de Pesquisa 1 – QP.1: Quais as métricas e métodos utilizados para fazer estimativas de esforço, prazos e custos para o planejamento de projetos ágeis de software?

A maioria dos modelos de estimativas utilizados em projetos ágeis usam *Story Point* como parâmetro primário (KANG; CHOI; BAIK, 2010), sendo assim *Planning poker* é uma das técnicas mais populares para equipes ágeis no planejamento e estimativa de esforço antes de iniciar cada iteração.

As técnicas de estimativas de tamanho são agrupadas juntamente com as técnicas de estimativa de esforço, pois dentro do contexto de desenvolvimento ágil, estimativa de esforço é frequentemente derivada da estimativa de tamanho e dos cálculos do *Velocity*, como podemos verificar nos trabalhos (3844) e (3843) apresentados na Tabela 2. A técnica *Expert Judgment* (3842), e *Use Case Points* (3842, 1562) também são técnicas de estimativas frequentemente usadas no contexto de desenvolvimento ágil de software (USMAN; MENDES; WEIDT; BRITTO, 2014).

A Tabela 4 apresenta as técnicas de estimativa de esforço no contexto ágil e sua ocorrência nos trabalhos selecionados. Alguns trabalhos se repetem na tabela porque utilizam mais de uma técnica.

Através do número de *Story Points* e do Velocity do time de desenvolvimento pode-se calcular o prazo de desenvolvimento de uma determinada funcionalidade. Como por exemplo, se o número total de Story Points para as funcionalidade desejadas é de 200 e o Velocity do time é de 20, então pode - se concluir que o time necessitará de 20 iterações para completar o desenvolvimento das respectivas funcionalidades (KANG; CHOI; BAIK, 2010). No entanto, o User Story Point (USP) não é objetivo [18] e não pode definir uma prática padrão para estimar o tamanho e a complexidade do software [1548].

Tabela 4 – Técnicas de Estimativa de Esforço Investigadas

Técnica	**ID do Trabalho**	**Frequência**
Opinião de Especialista	1564, 3826, 3842, 3879, 3896, 3898, 3914, 3985	8
Estimativa baseada em modelo (COCOMO, etc.)	1547, 1548, 3879, 3896, 3800	5
Planning Poker	3817, 3841, 3842, 3870, 1562, 3826, 3834, 3914, 3966, 3985, 3856	11
Pontos de Casos de Uso	3842, 1562, 3985	3
Pontos de Função	1562, 1547, 1543 , 3979, 3896, 3856	6
Modelos Customizados	1544, 1564	2
Número de linhas de código	3844, 1543, 3879, 3896	4
Fuzzy based Framework for Estimation	1553	1

Também foi possível identificar trabalhos inovadores na área de estimativas de projetos ágeis. O trabalho (1553) por exemplo, propõe uma estrutura que depende da utilização de lógica fuzzy e tem como objetivo ajudar na produção de estimativas precisas. Já no trabalho (1544), os autores modificam o método de Pontos de Casos de Uso (UCP) para torná-lo apropriado para o desenvolvimento ágil de software e chamam essa nova versão de UCP interativa (iUCP).

O trabalho (3879) propõe um modelo de predição de esforço causado pelas mudanças nos requisitos, o qual integra a análise dos impactos das mudanças com o modelo de estimativa de esforço COCOMO para melhorar a precisão das estimativas de esforço proveniente das mudanças em projetos de desenvolvimento ágil de software.

O trabalho (3985) propõe um modelo chamado Bayesian Network (BN) para ajudar gerentes de projetos ágeis a estimar o esforço do projeto. O BN é um modelo gráfico que descreve as relações probabilísticas entre variáveis relacionadas.

De acordo com o apresentado na Tabela 5, o *Stoty Points* é a métrica de tamanho mais utilizada para realizar as estimativas de projetos ágeis, entretanto podemos ver que os Pontos de Função ainda são bastante utilizados, sendo muitas vezes combinados com outras métricas.

Questão de Pesquisa 2– QP.2: Métricas de ponto de função podem ser utilizadas para realizar estimativas de esforço, prazos e custos para o planejamento de software ágil? Caso afirmativo, é a estimativa mais apropriada?

As métricas de ponto de função podem ser utilizadas para realizar estimativas. (KANG; CHOI; BAIK, 2010) propõe um modelo dinâmico de estimativa de custo para projetos ágeis de software, ou seja, se adapta durante o processo de desenvolvimento e com as mudanças nos requisitos.

Tabela 5 – Métricas de Tamanho utilizadas com insumo para estimativas e planejamento de projetos ágeis de software

Métrica de Tamanho	**Trabalho**	**Frequência**
Story points	3826, 3834, 3842, 3870, 1562, 3800, 3817, 3841, 3973, 3898, 3856	11
Linhas de Código	3844, 1543, 3879, 3896	4
Pontos de Função	1562, 1543, 1547, 3879, 3841,3856	6
Dias Ideais	3966, 3856	2
Horas Ideais	3966	1
Pontos de Casos de Uso	1544, 3842, 1562	3
COSMIC	1548	1

O modelo proposto adota pontos de função como métrica de estimativa base. Neste modelo, o time do projeto estima o tamanho das funcionalidades utilizando pontos de função e calcula o custo para derivar a duração do projeto. Para derivar o plano do projeto, três procedimentos são realizados. Primeiro, o time do projeto calcula os pontos de função das funcionalidades desejadas. Segundo, o modelo de estimativa, o qual é composto pelos pontos de função restantes, é gerado. Terceiro, o time do projeto desenvolvem um plano para a release, iteração, e dia utilizando métricas estimadas de custo tal como pessoas por mês e o número de linhas de código. Este modelo utiliza o *Kalman Filter* como um algorítmo de rastreamento (KANG; CHOI; BAIK, 2010).

Entretanto, apesar de alguns trabalhos, como o que foi citado anteriormente, sugerirem adaptações de modelos tradicionais para estimativas e medições de projetos ágeis de software, (JONES, 1998) afirma que uma das fraquezas das comunidades ágeis ainda é o fracasso em estimar e medir projetos usando métricas padrões, como pontos de função que são amplamente conhecidos e usados dentro da indústria mas que cobrem apenas os requisitos funcionais. Diferente dos *Story Points* por exemplo, que não correspondem a um tamanho de software e nem mesmo ao esforço real, e sim a estimativas (não medição), e que cobrem os requisitos funcionais e não funcionais. Outros estudos identificados também afirmam que Ponto de Função não é adequado para estimativas em projetos ágeis, devido à sua granularidade e ao seu insuficiente apoio ao *feedback* e à mudança. Portanto, apoiam firmemente a idéia de que as empresas que trabalham de forma tradicional ou ágil coletem medidas tradicionais de tamanho (como Pontos de Função) para o gerenciamento de portfólio, gerenciamento de projetos e *benchmarking*; E que as empresas que trabalham de acordo com um método ágil também façam isso, além de coletar medidas de tamanho em uma métrica de tamanho ágil (como *Story Points*) para fins de estimativa.

Questão de Pesquisa 3 – QP.3: Quais as métricas adotadas para verificar a produtividade da equipe em desenvolvimento ágil de software?

O *Velocity* é muitas vezes usado em comunidades ágeis de software no lugar de "produtividade". O *Velocity* é definido como o número de histórias de usuário ou números de pontos de história completos na iteração e é freqüentemente usada para estimar o tempo restante até o final do projeto. Entretanto, esta medida é muito útil para projetos estáveis que são sempre compostos pelos mesmos indivíduos , pois qualquer alteração na composição da equipe invalidará a medida *Velocity*. O Velocity deve ser medido para várias iterações sucessivamente. Esta é única maneira de garantir que ela pode ser útil para realizar estimativas conforme apresentado no trabalho 1553.

No trabalho 1543, também vimos que é possível medir a produtividade por ponto de função desenvolvidos. Entretanto os autores também afirmam que essa abordagem nem sempre é um boa alternativa pois nem todo o esforço de desenvolvimento é contado como pontos de função, a calibração do modelo depende fortemente das habilidades do contador e é influenciada pelo julgamento subjetivo.

Mesmo com um bom método de medição e/ou estimativa, há o grande problema em estimar a produtividade de um membro da equipe em uma determinada tarefa. Alguns testes mostraram que a produtividade pode se diferenciar em mais de 20 vezes de pessoa para pessoa. Os métodos de estimativas têm a tendência de superestimar o tempo necessário para completar tarefas pequenas, porém têm a tendência de subestimar o tempo necessário para completar tarefas grandes conforme apresentado no trabalho 3896.

A Produtividade geralmente indica o sucesso na realização de um determinado trabalho em relação aos recursos utilizados e o trabalho 3896 cita que existem mais de 200 fatores que afetam a produtividade dos desenvolvedores, e lista os principais fatores que mais impactam a produtividade: as habilidades e experiências dos desenvolvedores, a cooperação dos usuários durante a elicitação dos requisitos e design, cronograma ou restrições de recursos, métodos utilizados no projeto, ferramentas disponíveis, escolha de uma linguagem de programação apropriada, complexidade do problema, complexidade do código, complexidade dos dados, estrutura de organização do projeto e o ambiente físico de trabalho.

Questão de Pesquisa 4 – QP.4: Que tipo de estudos empíricos na área de métricas estão sendo feitos no ambiente ágil de desenvolvimento de software pela academia, e quais resultados estão sendo obtidos?

Através da revisão sistemática foi possível identicar diversos estudos na área de métricas. Alguns destes estudos primários são detalhados a seguir:

- O trabalho 1564 apresenta uma proposta para estimar, planejar e gerenciar projetos Web, combinando algumas técnicas ágeis existentes com princípios de desenvolvimento da Web, apresentando-as como uma estrutura unificada que usa o valor comercial para orientar a entrega de recursos. A proposta é analisada por meio de um

estudo de caso, incluindo um projeto da vida real, para obter conclusões relevantes. Os resultados obtidos após o uso do framework em um projeto de desenvolvimento são apresentados, incluindo resultados interessantes no planejamento e estimativa de projetos, bem como na produtividade da equipe ao longo do projeto. Concluiu-se que o framework pode ser útil para gerenciar melhor os projetos baseados na Web, através de um processo contínuo de gerenciamento e estimativas baseadas em valor.

- No trabalho 3800, tem-se a proposta de um método algorítmico de estimação simples para o desenvolvimento de software ágil onde os autores analisam e comparam os resultados fornecidos pelo método proposto através de um estudo de caso para verificar a viabilidade do método algorítmico em tempo real. O resultado mostra que a duração e o custo do projeto aumentam se o esforço aumentar, mas se recursos forem adicionados aumentando o esforço, a duração e o custo quase permanecem os mesmos.

- O trabalho 3841 cita uma pesquisa realizada em 2005 onde 18 gerentes de projeto de uma empresa de desenvolvimento de software norueguesa foram entrevistados. Dados sobre 52 projetos foram utilizados para analisar as diferenças de cronograma e excessos de esforço entre projetos que utilizam modelos de processos flexíveis (incremental e ágil) e modelos de processos sequenciais. Assim, concluiu-se que os projetos que utilizaram um modelo de processo flexível tiveram menos excesso de esforço se comparado com os projetos que utilizaram modelos de processo sequenciais.

- O trabalho 3898 traz um estudo de caso cujo objetivo é explorar e entender o processo de estimativa em relação à sua precisão no contexto de desenvolvimento ágil de software na perspectiva do time de desenvolvimento. O estudo de caso foi aplicado em uma empresa de desenvolvimento de software alemã chamada SAP SE. Entendeu-se que todos os times fazem estimativas a cada início de sprint. As estimativas são feitas para os requisitos de uma *User Story*. Os times estudados utilizam o MS Excel e a ferramenta para gerenciamento de projetos, JIRA. Fazem as estimativas para priorizar as tarefas, não para avaliação de riscos, por exemplo. Além disso, os times realizam estimativas para desenvolver um entendimento comum entre eles em relação ao que deve ser feito em uma determinada sprint. Os times A e B fazem estimativas individuais, estimando o esforço a partir de *Horas Trabalhadas por Pessoa*. O time C realiza um jogo onde a estimativa de esforço é feita através de um consenso, e é feita utilizando *Story Points*.

- No trabalho 3856, são utilizadas três métricas: *Ideal Day, Story Points* e APF em um estudo de caso em uma organização. Segundo os autores os resultados demonstraram que o *Ideal Day e o Planning Poker* estimaram o tempo com maior precisão que a APF. Os autores aplicaram em um projeto com 34 requisitos.

- O trabalho 1547, apresentaram um guia para otimizar o custo e o esforço em projetos de Scrum usando o ponto de função com o modelo COCOMO. A proposta foi utilizada no estudo de caso real Xsset para realizar estimativas. Além disso, o trabalho mostra as diferentes ferramentas de rastreamento e representação usadas para otimizar o desempenho, como diagrama de execução, JIRA, SVN e SCTM. Esta pesquisa concluiu que a metodologia ágil é muito eficaz no desenvolvimento de software. Além disso, eles concluíram que, ao combinar os dois modelos de estimativa, pode-se melhorar o esforço aplicado e economizar nos custos.

4.2 Resultados do Estudo de Caso

Essa seção apresenta os resultados obtidos com a realização do estudo de caso e a discussão dos mesmos de acordo com os objetivos definidos na fase de planejamento.

4.2.1 Análise dos Dados

Depois de definido o *backlog* da release, todos as histórias de usuário foram estimadas pela equipe utilizando o *Planning Poker* conforme vimos nas tabelas presentes no item de Coleta de Dados. As histórias também foram mensuradas em Pontos de Função conforme Manual de Práticas de Contagem do IFPUG (CPM), versão 4.3.1 (IFPUG, 2010) para realização das estimativas iniciais do Projeto.

4.2.1.1 Estimativas de Tamanho, Esforço, Prazo e Custo utilizando Pontos de Função

Uma contagem estimada em pontos de função das User Stories do backlog do da release 1 foi realizada antes do início da execução das Sprints. A contagem foi realizada pelos autores deste trabalho em conjunto com o time. Alguns membros do time já tinham conhecimento em Análise de Pontos de Função, então não houve dificuldade para a realização da mesma.

Para a estimativa de tamanho das User Stories em pontos de função foram utilizados o Manual de Práticas de Contagem do IFPUG (CPM), versão 4.3.1 (IFPUG, 2010),O Ponto de Função não ajustado foi utilizado. A figura 12 ilustra o procedimento de contagem de pontos de função para as User Stories.

Depois de realizada a contagem estimada da release, o esforço foi calculado. Com base no esforço, o prazo e o custos também foram estimados.

Para a estimativa de esforço utilizando Pontos de Função, nos baseamos no modelo simplificado proposto por (VAZQUEZ, 2012). O Modelo Simplificado de Estimativas consiste em obter um índice de produtividade em horas/PF para o projeto específico em

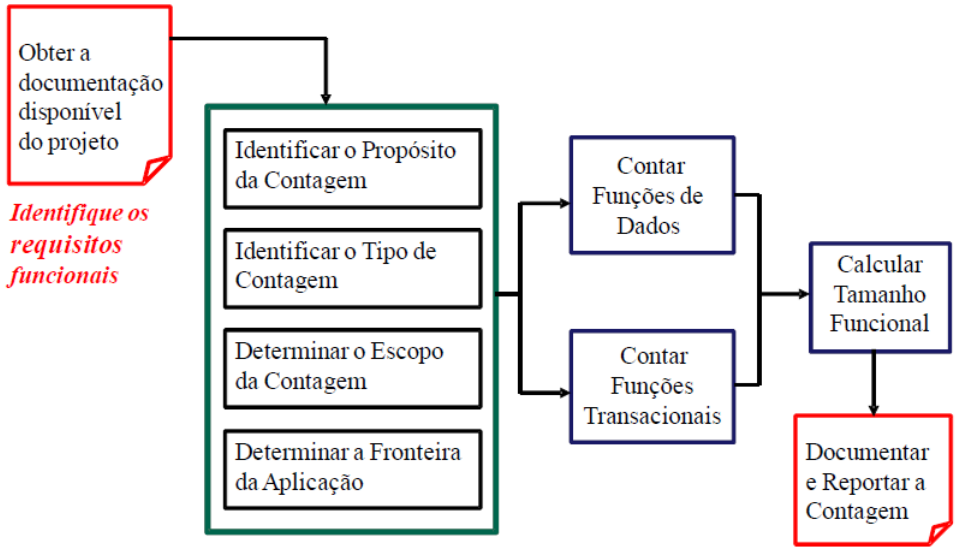

Figura 12 – Procedimento de Contagem de Ponto de Função (SISP, 2016)

.

questão, e então multiplicar o tamanho em PF do projeto pelo índice de produtividade, conforme a fórmula:

Esforço (horas) = Tamanho (PF) x Índice de Produtividade (HH/PF)

O índice de produtividade utilizado foi definido pela organização e pelo time, de acordo com informações de projetos similares desenvolvidos na empresa.

O valor da estimativa de prazo em semanas da release foi definido com base na contagem estimada em pontos de função do escopo da release e na capacidade de produção em pontos de função por sprint da equipe de desenvolvimento do projeto declarada pela Basis.

Quantidade de Sprints = Tamanho (PF) / Produtividade média do time por Sprint (PF).

ou

Quantidade de Sprints = Esforço (horas) / (Produtividade média do time por sprint(PF) * Índice de Produtividade (HH/PF))

Para a estimativa de custos total do projeto utilizando Pontos de Função multiplicou-se o valor do ponto de função da empresa pela tamanho estimado do software:

Custo = Tamanho (PF) * Valor do PF da empresa

O valor total estimado em pontos de função das histórias do backlog da release é de 387 pontos. A equipe de desenvolvimento do projeto é composta por 13 pessoas possui uma produtividade de 78 PF por sprint aproximadamente. A duração desejada de cada sprint é de 2 semanas e o valor em reais por ponto de função da empresa é de

469.68. Com base nessas informações, obteve - se as estimativas iniciais utilizando PF mostradas na tabela 6, com uma incerteza de ±25% definida segundo (COHN, 2005). Os valores reais gastos em cada tarefa foram definidos pelo time e também foram colhidos para comparação e melhoria das estimativas.

A tabela 7 e tabela 8 mostram, respectivamente, as datas de início e término das Sprints e a comparação dos dados estimados com os reais em cada sprint utilizando Pontos de Função. A incerteza também é de ±25%

Tabela 6 – Estimativas Iniciais utilizando Pontos de Função

Estimativa Inicial	**Valor**
Total de Pontos de Função da Release :	$387 \pm 96,75$
Horas de desenvolvimento de 1 PF (HH/P):	$10 \pm 2,5$
Tamanho da Equipe em pessoas:	13
Esforço(HH):	$3870 \pm 967,5$
Produtividade média da Equipe por Sprint (PF/Sprint):	$78 \pm 19,5$
Quantidade Desejada de Dias por Sprint(Dias):	10
Número de Sprints:	5
Duração do Projeto(Semanas)	10
Quantidade de Horas por Iteração(H):	780 ± 195
Quantidade de Horas Totais do Projeto(H):	3900 ± 975
Custo por Iteração(Reais):	$36.635,04 \pm 9.158,76$
Custo Total do Projeto(Reais)	$183.175,2 \pm 45.793,8$

Tabela 7 – Datas de Início e de Término das Sprints

Sprint	**Data de Início**	**Data do Fim**
0	10/07/2017	21/07/2017
1	24/07/2017	04/08/2017
2	07/08/2017	18/08/2017
3	21/08/2017	01/09/2017
4	04/09/2017	18/09/2017

Tabela 8 – Comparação de Dados Estimados com Dados Reais Utilizando Pontos de Função

Sprint	**Dias de Trabalho**	**PFs Estimados**	**PFs Entregues**	**Produtividade Real**	**PFs Restantes**
0	10	$78 \pm 19,5$	66	66	321
1	10	80 ± 20	90	78	240
2	10	80 ± 20	114	90	135
3	10	68 ± 17	93	90,75	54
4	10	$54 \pm 13,5$	54	83,4	0

4.2.1.2 Estimativas de Tamanho, Esforço, Prazo e Custo utilizando Story Points

As estimativas de tamanho, esforço, prazo e custo foram feitas utilizando como insumo os dados referentes à release 1 do Projeto X fornecidas pela empresa. O modelo proposto por (TORRECILLA-SALINAS J. SEDENO, 2015) foi utilizado como base.

O time não possuía muita experiência em realizar estimativas utilizando Story Points. Então um treinamento foi relizado antes da realização das estimativas.

O backlog da release foi estimado com um total de 115 Story Points. E o time definiu a o valor de 13,64 horas por Story Point. A partir daí, foram feitas as estimativas de Velocity, Quantidade Desejada de Dias por Sprint, Número de Sprints, Quantidade de Horas Disponíveis por Sprint, Quantidade de Horas do Projeto, Custo por Sprint e o Custo Total do Projeto.

As fórmulas utilizadas para realizar as estimativas são:

- **Velocity:** A estimativa do Velocity Inicial foi feita baseada em dados de outros projetos desenvolvidos pelo time. Para o restante das sprints, o velocity era estimado dividindo o número de SP restantes pelo número de sprint restantes.
- **Número de Sprints:** Foi definido dividindo o valor total em SP pela Velocity estimado do time.
- **Quantidade Desejada de Dias por Sprint:** O cliente do Projeto X determinou que as Sprints devem ter duração de 2 semanas, ou seja, 10 dias úteis.
- **Quantidade de Horas Estimada por Sprint:** A Quantidade de horas estimadas por Sprint foi calculada multiplicando o valor médio de horas para cada SP (inicial de 13,64 horas) pelo Velocity (inicial de 23 story points), assim, obteve - se o valor de 313,72 horas estimadas por Sprint.

- **Quantidade de Horas Estimada do Projeto:** O Projeto X terá uma duração de 5 sprints com 313,72 horas em cada, então, fazendo a devida multiplicação, temos um total de 1568,6 horas.
- **Custo por Sprint:** O Custo Total por Sprint estimado foi de R$36.660,00, levando em consideração o custo de hora trabalhada de todos os 13 membros da equipe do projeto, considerando 6 horas por dia disponíveis para trabalho (expediente total de 8 horas com 2 horas para almoço).
- **Custo Total do Projeto:** Multiplicando o custo total por Sprint pelo número total de Sprints do projeto, temos um Custo Total Estimado do Projeto em R$183.300,00.
- **Horas Estimadas:** As Horas estimadas para cada sprint foram calculadas multiplicando o valor da hora de 1 SP pelo de SP planejatos para a sprint.
- **Horas do Story Point:** A hora de 1 SP foi calculada dividindo as horas reais pelo número de SP entregues.

As tabelas 7, tabela 9, tabela 10 e tabela 11 apresentam, respectivamente, as datas de início e término das sprints, as estimativas iniciais de cada sprint, a comparação dos valores de story points estimados com os reais e a comparação entre as horas estimadas com as horas reais. Considerando uma incerteza de ±25%.

Tabela 9 – Estimativas Iniciais do Projeto Utilizando Story Points

Total de Story Points :	$115 \pm 28,75$
Horas de 1 SP:	$13,64 \pm 3,41$ horas
Velocity:	$23 \pm 5,75$ Story Points
Quantidade Desejada de Dias por Sprint:	10 dias úteis
Número de Sprints:	5
Quantidade de Horas Estimadas por Sprint:	$313,72 \pm 78,43$ horas
Quantidade de Horas do Projeto:	$1568,6 \pm 392,15$ horas
Custo por Sprint:	$R\$36.660,00 \pm R\$9.165,00$
Custo Total do Projeto:	$R\$183.300,00 \pm R\$45.825,00$

Tabela 10 – Comparação de Dados Estimados com Dados Reais Utilizando Story Points

Sprint	**Dias de Trabalho**	**Velocity Estimado**	**Velocity Real**	**Story Points Entregues**	**Story Points Restantes**
0	10	$23 \pm 5,75$	15	15	100
1	10	$25 \pm 6,8$	18	21	79
2	10	$26,33 \pm 7,3$	21,66	29	50
3	10	$25 \pm 7,3$	23,25	28	22
4	10	$22 \pm 7,75$	24,8	31	0

Tabela 11 – Comparação entre as Horas Estimadas com as Horas Reais

Horas Estimadas	**Horas Reais**	**Hora do SP**
$341 \pm 85,25$	726	48,4
$871,2 \pm 217,8$	801	38,14
$1029,78 \pm 257,44$	798	27,51
$660,41 \pm 165,1$	837	29,89
$927,67 \pm 231,9$	556,2	17,94

4.2.1.3 Comparação entre as estimativas utilizando Pontos de Função e Story Points

O objetivo desse estudo é responder a seguinte questão: Quais das métricas utilizadas como insumo para estimar tempo, custo e esforço fornece estimativas que mais se aproximam dos valores reais em um projeto ágil de desenvolvimento: Story Points ou Pontos de Função?

Para respondê-la, foram realizadas separadamente estimativas de esforço em horas, tempo e custo em um projeto de desenvolvimento ágil de uma fábrica de software utilizando como base as métricas de tamanho que de acordo com a RS realizada neste trabalho são as mais utilizadas em projetos ágeis de software: Ponto de Função e o Story Point.

O processo de estimativas e o resultado das mesmas foram detalhados nas seções anteriores, entretanto a pergunta ainda foi respondida.

Para determinar qual das métricas é mais precisa, optou-se então por pegar os valores reais do projeto e compará-los com as estimativas realizadas.

As tabela 12 e tabela 13 apresentam os resultados do projeto em relação aos valores do plano inicial. A incerteza associada é de $\pm 25\%$.

Tabela 12 – Resultados do Projeto vs Estimativas Iniciais Utilizando Pontos de Função

	Real	**Estimado**	**Diferença**	**Porc. de desvio em relação ao estimado**
Produtividade Média	81,63	$78 \pm 19,5$	3,63	+4,65
Total de Horas Trabalhadas	3728,5	3900 ± 975	171,5	-4,39
Total de Pontos de Função	417	$387 \pm 96,75$	30	+7,75
Horas por Ponto de Função	8,9	$10 \pm 2,5$	1,1	-11
Dias de Trabalho	50	50	0	0
Custo Total do Projeto	195856,6	$183175,2 \pm 45793,8$	12681,36	+6,92

Tabela 13 – Resultados do Projeto vs Estimativas Iniciais Utilizando Story Points

	Real	**Estimado**	**Diferença**	**Porc. de desvio em relação ao estimado**
Velocity Médio	20,54	$23 \pm 5,75$	2,49	-10,82
Total de Horas Trabalhadas	3728,5	$1568,6 \pm 392,15$	2159,9	+137,69
Total de Story Points	124	$115 \pm 28,75$	9	+7,82
Horas por Story Points	29,98	$13,54 \pm 3,41$	16,44	+126,26
Dias de Trabalho	50	50	0	0

Analisando os valores das duas tabelas, é possível verificar que as estimativas realizadas a partir do Ponto de Função se aproximaram mais dos valores reais do projeto. A diferença entre os valores reais das métricas de Produtividade Média do Time, Total de Horas Trabalhadas, Horas por Ponto de Função, e Custo Total do Projeto, e o estimados tiveram valores dentro da faixa de incerteza definida como aceitável no projeto. O time já utilizava a métrica de tamanho do Ponto de Função para estimar seus projetos de desenvolvimento, logo possuem mais experiência com o método, assim o valor inicial de produtividade e o valor de horas de 1 PF foram mais próximos do real e fez com que as outras estimativas também fossem mais precisas.

Diferentemente do Ponto de função, o time utilizou poucas vezes a métrica de Story Point para realizar estimativas em seus projetos. O que justifica a grande diferença entre as horas estimadas para o desenvolvimento de 1 SP e o valor realmente gasto. O time subestimou o tempo de desenvolvimento, assim tivemos valores de horas bem distantes do estimado. No entanto é possível verificar que a partir da sprint 3 as estimativas começaram a terem valores mais próximos de real. Essa diferença também pode ser explicado pela subjetividade da Métrica, já que a técnica não nos permite utilizar dados de outros times de desenvolvimento ágil, apenas dos projetos desenvolvidos pelo próprio time.

5 Conclusão

Por meio da execução deste trabalho, foi possível perceber a relevância das estimativas de tamanho, esforço, custo e prazo no contexto de desenvolvimento ágil de software e por isso, os métodos e métricas de estimativas têm sido cada dia mais discutidas em trabalhos acadêmicos que procuram as melhores e mais precisas métricas utilizadas em determinado contexto ágil.

Através da Revisão Sistemática, foram investigados os métodos e as principais métricas de tamanho utilizadas em estimativas no contexto ágil. Foi possível concluir que dentre as técnicas mais utilizadas estão: *Planning Poker*, Opinião de Especialista e Análise de Pontos de Função. Sendo as métricas mais utilizadas para as estimativas os *Story Points* e os Pontos de Função.

A Revisão Sistemática também mostrou que os métodos e métricas para estimativas em sua maioria são aplicadas à um determinado contexto de desenvolvimento ágil com adaptações a fim de se adequarem ao projeto em questão. Sendo assim, pode-se concluir também que as métricas de estimativas devem sempre sofrer adaptações para se adequarem ao contexto do projeto, pois cada projeto terá suas próprias características as quais influenciarão no resultado das estimativas.

A partir dos resultados da Revisão Sistemática, um Estudo de Caso foi realizado onde foi feita uma comparação da precisão das estimativas feitas utilizando as métricas de Pontos de Função e Story Points com base nos dados da Release 1 do Projeto X disponibilizados pela empresa Basis.

O estudo de caso mostrou que as estimativas utilizando a métrica de Pontos de Função teve porcentagens de desvio dos valores reais em relação aos valores estimados menor do que as estimativas feitas utilizando a métrica de Story Points. Isso se deve pelo fato de que o time possui uma vasta experiência com a utilização de estimativas com a métrica de Pontos de Função, assim, os valores iniciais das estimativas de produtividade e o valor de horas de 1 PF foram bem mais próximas do real, o que consequentemente fez com que o restante das estimativas fossem mais precisas.

Entretanto, é necessária a realização de diversos experimentos com projetos ágeis de diversas características diferentes para ser possível afirmar com certeza que a métrica de Pontos de Função é mais precisa do que a métrica de Story Points em projetos de desenvolvimento ágil.

Referências

ABRAN, A.; MOORE, J. The guide to the software engineering body of knowledge. SWEBOK, IEEE Computer Society, 2004. Citado 3 vezes nas páginas 26, 27 e 28.

AHMED, M.; AHMAD, I.; ALGHAMDI, J. Probabilistic size proxy for software effort prediction: a framework. Inform. Softw. Technol., 55(2), 2013. Citado na página 29.

ALBRECHT, A. J. Measuring application development productivity,"presented at the ibm applications development. 1979. Citado na página 29.

ALBRECHT, A. J.; GAFFNEY, J. A. Software function, source lines of codes, and development effort prediction: a software science validation. IEEE Trans Software Eng. SE, vol. 9, 1983. Citado na página 29.

ALVES, F.; FONSECA, M. Ideal day e priorização: Métodos Ágeis no planejamento. Engenharia de Software, Rio de Janeiro, 2008. Citado 2 vezes nas páginas 30 e 31.

BECK, K.; ANDRES, C. **Extreme Programming Explained: Embrace Change**. [S.l.]: Addison-Wesley Professional, 2004. Citado 2 vezes nas páginas 14 e 30.

BERLIN, S.; RAZ, T.; GLEZER, C.; ZVIRAN, M. Comparison of estimation methods of cost and duration in it projects. Inform. Softw. Technol., 51, 2009. Citado na página 29.

BOEHM, B. **Software Cost Estimation With COCOMOII**. [S.l.: s.n.], 2000. Citado na página 28.

BOEHM, B. W. Software engineering economics. New Jersey: Prentice-Hall, 1981. Citado na página 29.

BOURQUE, P.; OLIGNY, S.; ABRAN, A.; FOURNIER, B. Developing project duration models in software engineering. J. Comput. Sci. Technol., 22, 2007. Citado na página 29.

COHN, M. **User Stories Applied: For Agile Software Development**. [S.l.: s.n.], 2004. Citado na página 30.

_____. **Agile Estimating and Planning**. [S.l.: s.n.], 2005. Citado 2 vezes nas páginas 30 e 55.

COSTA, A.; HENRIQUE, D.; SÉRGIO, G.; LISBOA, L. Utilizando métricas para dimensionar um software. **Proc. ACM Nat. Conf. pp. 671-677**, 2011. Citado na página 24.

DCONTE, S.; DUNSMORE, H.; SHEN. **Software engineering metrics and models Benjamin/Cure mings**. [S.l.: s.n.], 1986. Citado na página 24.

DEMARCO, T. **Controlling software projects**. [S.l.: s.n.], 1982. Citado 2 vezes nas páginas 23 e 24.

DIAS, R. Análise por pontos de função: Uma técnica para dimensionamento de sistemas de informação. **Computer**, DOAJ, 2003. Citado na página 14.

DINGSOYR, T.; NERUR, S.; BALIJEPALLY, V.; MOE, N. A decade of agile methodologies: Towards explaining agile software development. **Journal of Systems and Software**, 2012. Citado na página 19.

DYBA, T.; DINGSOYR, T. Empirical studies of agile software development: A systematic review. **Information and Software Technology**, 2008. Citado na página 18.

FILHO, A. M. S. Estimativa de custo de software: roteiro e dicas para estimativas de projeto. Revista Espaço Acadêmico, 156, 2014. Citado 2 vezes nas páginas 9 e 27.

FOWLER, M.; HIGHSMITH, J. **Manifesto for Agile Software Development**. 2017. Disponível em: <https://http://agilemanifesto.org/>. Acesso em: 01 abril 2017. Citado 2 vezes nas páginas 14 e 18.

GALSTER, M. et al. Variability in software systems - a systematic literature review. **EEE Trans. Softw. Eng. 40**, 2014. Citado na página 32.

HAZAN, C. Análise de pontos de função: Uma aplicação nas estimativas de projetos de software. Engenharia de Software, Rio de Janeiro, n. , p.25- 30, 2008. Citado na página 27.

HIGHSMITH, J. **Agile Project Management: Creating Innovative Products**. [S.l.: s.n.], 2009. Citado na página 30.

IEEE. **IEEE Standard Glossary of Software Engineering Terminology**. [S.l.: s.n.], 1990. Citado na página 24.

IFPUG. **Manual de Práticas de Contagem de Pontos de Função, Versão 4.3.1**. [S.l.: s.n.], 2010. Citado 2 vezes nas páginas 42 e 53.

JONES, C. **Estimating software costs**. [S.l.: s.n.], 1998. Citado 2 vezes nas páginas 29 e 50.

JUNIOR MARCOS ; FANTINATO, M. . S. V. de F. Improvements to the function point analysis method: A systematic literature review. **IEEE Transactions on Engineering Management**, 2015. Citado na página 25.

KAN, S. H. **Metrics and models in software quality engineering**. [S.l.: s.n.], 1995. Citado na página 29.

KANG, S.; CHOI, O.; BAIK, J. Model-based dynamic cost estimation and tracking method for agile software development. **Proceedings - 9th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2010**, 2010. Citado 3 vezes nas páginas 48, 49 e 50.

KEMERER, C. F. An empirical validation of software cost estimation models. Commun. ACM, vol. 30, 1987. Citado na página 28.

KITCHENHAM, B. What's up with software metrics? – a preliminary mapping study. **Computer**, The Journal of Systems and Software, 2009. Citado na página 15.

KITCHENHAM, B.; CHARTERS, S. Guidelines for performing systematic literature reviews in software engineering. **ech. rep. EBSE 2007-001**, Keele University and Durham University, 2007. Citado 2 vezes nas páginas 16 e 32.

LI, J. Agile software development. 2008. Citado 5 vezes nas páginas 9, 18, 19, 21 e 22.

LOGUE, K.; MCDAID, K. Agile release planning: dealing with uncertainty in development time and business value. Proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, 2008. Citado na página 30.

MARTINS, J. C. C. Técnicas para gerenciamento de projetos de software. Brasport Editora, 2001. Citado na página 31.

MATSON, J. E. Software development cost estimation using function points. IEEE Trans. Softw. Eng., 1994. Citado na página 28.

MOLOKKEN, K.; JORGENSEN, M. A review of surveys on software effort estimation. International Symposium on Empirical Software Engineering, 2003. Citado na página 28.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. Systematic mapping studies in software engineering. ACM, 2008. Citado na página 16.

PRESSMAN, R. S. **Engenharia de Software**. [S.l.]: McGraw-Hill, 2006. Citado na página 14.

PULFORD, K.; KUNTZMANN-COMBELLES, A. **A Quantitative Approach to Software Management: the AMI Handbook**. [S.l.]: Addison-Wesley Longman Publishing Co., 1995. Citado na página 14.

PUTNAM, L. **A general empirical solution to the Macro Sizing and Estimating Problem**. [S.l.: s.n.], 1978. Citado na página 29.

RUBEY, R. J.; HARTWICK, R. D. Quantitative measurement of program quality. **Proc. ACM Nat. Conf. pp. 671-677**, 1968. Citado na página 23.

RUNESON, M. H. **Guidelines for Conducting and Reporting Case Study Research in Software Engineering**. [S.l.]: Sweden, 2008. Citado 2 vezes nas páginas 31 e 41.

S., P. R. **Engenharia de software: uma abordagem profissional**. [S.l.: s.n.], 2011. Citado 3 vezes nas páginas 23, 26 e 28.

SCHWABER, K.; BEEDLE, M. **Agile Software Development with Scrum, Vol. 1**. [S.l.]: Prentice Hall Upper Saddle River, 2002. Citado na página 14.

SCHWABER, K.; SUTHERLAND, J. **The Definitive Guide to Scrum: The Rules of the Game**. [S.l.: s.n.], 2013. Citado 2 vezes nas páginas 21 e 22.

SHEETZ, S.; HENDERSON, D.; WALLACE, L. Understanding developer and manager perceptions of function points and source lines of code. J. Syst. Softw., 82, 2009. Citado na página 29.

SHEPPERD, M.; SCHOFIELD, C.; KITCHENHAM, B. Effort estimation using analogy. INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 1996. Citado na página 29.

SHORE, J.; WADEN, S. **The Art of Agile Development**. [S.l.: s.n.], 2008. Citado na página 30.

SISP. **Roteiro de Métricas de Software do SISP**. [S.l.]: Ministério do Planejamento, Desenvolvimento e Gestão, 2016. Citado 3 vezes nas páginas 9, 42 e 54.

SOMMERVILLE, I. **Engenharia de Software**. [S.l.: s.n.], 2007. Citado na página 28.

TORRECILLA-SALINAS J. SEDENO, M. E. a. M. M. C. Estimating, planning and managing agile web development projects under a value-based perspective. 2015. Citado na página 56.

TRENDOWICZ, A.; JEFFERY, R. Appendix a: measuring software size software project effort estimation. Springer International Publishing, 2014. Citado na página 28.

USMAN, M.; MENDES, E.; WEIDT, F.; BRITTO, R. Effort estimation in agile software development: A systematic literature review. **ACM International Conference Proceeding Series**, 2014. Citado na página 48.

VALKENHOEF, G. van; TERVONEN, T.; BROCK, B. de; POSTMUS, D. Quantitative release planning in extreme programming. **Information and Software Technology**, Elsevier B.V., 2011. Citado na página 22.

VAZQUEZ, C. e. a. **Análise de Pontos de Função: Medição, Estimativas e Gerenciamento de Projetos de Software.** [S.l.: s.n.], 2012. Citado na página 53.

WEBER KIVAL CHAVES;ROCHA, A. R. C. d. N. C. J. d. **Medição da qualidade e produtividade em software**. [S.l.]: Makron Books, 2001. Citado na página 15.

WOIT, D.; BELL, K. Commitment and consistency in the collaborative software development process of extreme programming. **KMIS 2014 - Proceedings of the International Conference on Knowledge Management and Information Sharing**, Elsevier B.V., 2014. Citado na página 23.

YIN, R. K. **Case Study Research: Design and Methods**. [S.l.]: SAGE Publications, 2013. Citado na página 31.

ZADRA, A. N.; CARVALHO, E. O.; CARDOSO, J. F. S. Software metrics: Comparing function points and cocomo. 2014. Citado na página 27.