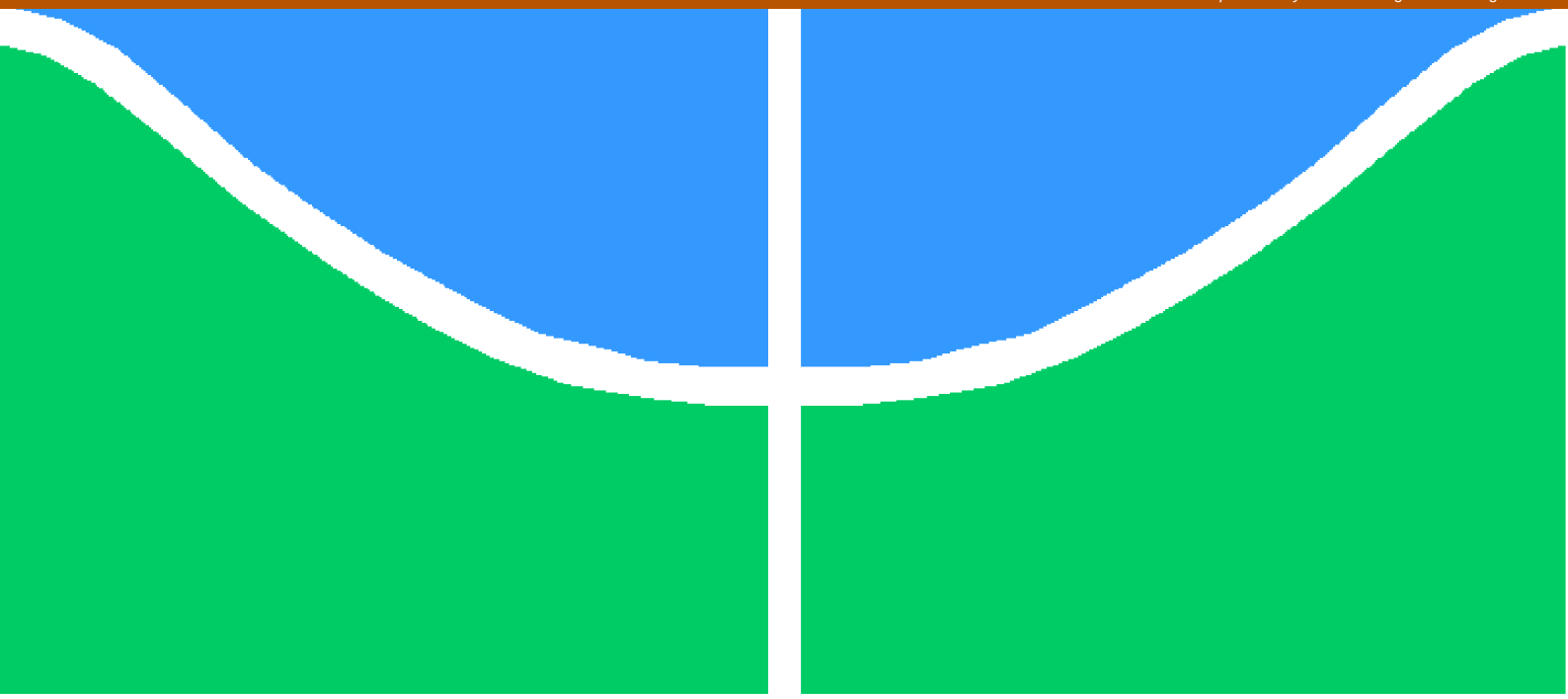


Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Comparação da Acurácia de Estimativas de Esforço Produzidas a Partir de Story Points, Ideal Day e de Ponto de Função COSMIC

Autor: Murilo Duarte Gonçalves

Orientador: Professora Dra. Edna Dias Canedo

Brasília, DF

2018

Murilo Duarte Gonçalves

Comparação da Acurácia de Estimativas de Esforço Produzidas a Partir de Story Points, Ideal Day e de Ponto de Função COSMIC

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

2018

Murilo Duarte Gonçalves
Comparação da Acurácia de Estimativas de Esforço Produzidas a Partir de Story Points, Ideal Day e de Ponto de Função COSMIC/ Murilo Duarte Gonçalves. – Brasília, DF, 2018-
86 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 , 2018.

Comparação da Acurácia de Estimativas de Esforço Produzidas a Partir de Story Points, Ideal Day e de Ponto de Função COSMIC

CDU 02:141:005.6

Murilo Duarte Gonçalves

Comparação da Acurácia de Estimativas de Esforço Produzidas a Partir de Story Points, Ideal Day e de Ponto de Função COSMIC

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, 13 de Julho de 2018:

Professora Dra. Edna Dias Canedo
Orientadora

Professora Msc. Cristiane Soares Ramos
Convidado 1

Professor Msc. Ricardo Ajax
Convidado 2

Brasília, DF
2018

Dedico este trabalho à minha família. Sem vocês nada teria sentido. Sempre estiveram ao meu lado incentivando e ajudando nos momentos mais difíceis desta jornada.

Agradecimentos

Agradeço a minha mãe Valéria Duarte Gonçalves, que me apoiou e incentivou nas horas difíceis. Sou grato também a minha namorada Bernadete Cristina Gouvêa Quirino da Costa, que não me deixou ser vencido pelo cansaço, me estimulando durante todo o projeto e compreendeu minha ausência pelo tempo dedicado aos estudos. Meus agradecimentos aos irmãos, sobrinhos, tios e avós, que de alguma forma também contribuíram para que o sonho da faculdade se tornasse realidade.

Agradeço a todos os professores, especialmente a Dra. Edna Dias Canedo, que me deu todo o suporte com suas correções e incentivos.

Agradeço imensamente à Deus, por ter me concedido saúde, força e disposição para fazer a faculdade e o trabalho de final de curso. Sem ele, nada disso seria possível. Também sou grato ao Senhor por ter dado saúde aos meus familiares e tranquilizado o meu espírito nos momentos mais difíceis da minha trajetória acadêmica até então.

Agradeço à Universidade de Brasília (UnB), por me proporcionar um ambiente criativo e amigável para os estudos. Sou grato à cada membro do corpo docente, à direção e a administração dessa instituição de ensino.

Murilo Duarte Gonçalves

"Um sonho sonhado sozinho é um sonho. Um sonho sonhado junto é realidade."

(Yoko Ono)

Resumo

No intuito de produzir software de qualidade, aspectos como planejamento do processo produtivo são primordiais, afinal é necessário obter indicadores do que irá acontecer, para definir a alocação dos recursos de um projeto de desenvolvimento. Este trabalho, propõe melhorar o entendimento dos aspectos relacionados à prática de estimativas de esforço, isto é, característica da alocação recursos para desenvolver um software sob a perspectiva do tempo. Para isso, aspectos como a acurácia de três técnicas de estimativa de esforço, Story Points, Ideal Day e Ponto de Função COSMIC, são investigados. Perspectivas de qual dessas técnicas é a mais utilizada em projetos de desenvolvimento, e os principais fatores influenciadores de baixas acurácias de estimativas de esforço são analisadas. Este estudo possui natureza aplicada, onde as questões levantadas são respondidas com base em uma revisão bibliográfica e devem ser confrontadas em um trabalho prático futuro, portanto, traz uma base para que um estudo de caso seja realizado, onde a acurácia das três técnicas são comparadas, confirmando assim a natureza empírica deste estudo.

Palavras-chaves: Desenvolvimento Ágil de Software; Estimativas de software; Métricas de software; Estimativa de Esforço; Acurácia de Estimativas; Story Points; Ideal Day; Ponto de Função COSMIC;

Abstract

In order to produce quality software, aspects such as production process planning are paramount, after all it is necessary to obtain indicators of what will happen, to define the allocation of the resources of a development project. This work proposes to improve the understanding of the aspects related to the practice of effort estimates, that is, characteristic of the allocation resources to develop software from the perspective of time. For this, aspects such as the accuracy of three effort estimation techniques, Story Points, Ideal Day and COSMIC Function Point, are investigated. Perspectives of which of these techniques is the most used in development projects, and the main influencing factors of low estimates of effort estimates are analyzed. This study has an applied nature, where the questions raised are answered based on a bibliographical review and should be confronted in a future practical work, therefore, it provides a basis for a case study to be carried out, where the accuracy of the three techniques are compared , confirming the empirical nature of this study.

Key-words: Agile Software Development; Software estimates; Software metrics; Estimating Effort; Accuracy of Estimates; Story Points; Ideal Day; Function Point COSMIC;

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

TCC	Trabalho de Conclusão de Curso
US	User Story
COSMIC	Common Software Measurement
ISO	International Organization for Standardization
PFC	Ponto de Função COSMIC
SP	Story Point
AEB	Agência Espacial Brasileira
DINF	Divisão de Informática
CPM	Coordenação de Planejamento e Modernização
DPOA	Diretoria de Planejamento, Orçamento e Administração
MCTI	Ministério da Ciência, Tecnologia e Inovação
XP	Extreme Programming
CMMI	Capability Maturity Model Integration
TI	Técnologia da Informação
PSW-AEB	Processo de Desenvolvimento de Software AEB
TCU	Tribunal de Contas da União
QP	Questão de Pesquisa
NBR	Norma Brasileira
ABNT	Associação Brasileira de Normas Técnicas
SISP	Sistema de Administração dos Recursos de Tecnologia da Informação

Sumário

1 Introdução

1.1 Contexto

O desenvolvimento de um produto de software é uma tarefa complexa, que envolve desde os desafios de definição de tecnologia até a escolha de equipes e o estabelecimento de cronogramas. Com isso, cria-se um processo único e que gera um produto exclusivo em sua concepção (BHARDWAJ; RANA, 2016).

Para obter um acompanhamento de qualidade para esse processo complexo é necessário obter indicadores do que está ocorrendo e para isso utiliza-se da aplicação e coleta de métricas. Segundo (PRESSMAN, 2006), o ato de mensurar é praticado no processo de desenvolvimento de software ou em partes de um produto com a perspectiva de melhorá-lo continuamente.

As métricas de software podem ser usadas para auferir uma grande variedade de informações, como: A qualidade do produto entregue ao cliente; O progresso de um projeto de software; As estimativa de custos; As estimativas de tamanho e complexidade (PADMINI; BANDARA; PERERA, 2015).

As estimativas de custo são um importante aspecto em um projeto de desenvolvimento de software, e na maioria dos casos o custo é fruto do trabalho empregado para obter o produto final (CAO, 2008). Então, estimar o esforço de desenvolvimento é essencial para o planejamento, gerenciamento e controle de um projeto de software. Subestimar essa métrica por um lado aumentará a pressão da equipe elevando as taxas de defeitos e custos do desenvolvimento, por outro lado, superestimar resultará em desperdício de recursos e prejuízos na competitividade dos contratos firmados (CAO, 2008).

A busca por respostas mais rápidas a mudanças de projeto, adaptabilidade e redução do tempo de desenvolvimento, levam o olhar diretamente para as metodologias ágeis de desenvolvimento de software. Essas contemplam abordagens iterativas e incrementais, isto é, reagem mais facilmente a mudanças de requisitos do produto. Assim, é possível propor pequenos ciclos de entrega de valor ao cliente, aproximando-o do desenvolvimento do produto e possibilitando detecção de alterações menores (TORRECILLA-SALINAS et al., 2015). A mensuração de custo e tempo em metodologias ágeis torna-se incerta e difícil, levando em conta as flexibilizações do projeto, além disso, parte dos métodos mais usados de estimação em ágeis atualmente não são eficientes, já que desconsideram fórmulas matemáticas para cálculos precisos de esforço e custo (POPLI; CHAUHAN, 2014).

No contexto de desenvolvimentos ágeis de software Story Points, Ideal Day e Ponto de Função COSMIC são técnicas de estimativas de esforço importantes.

A técnica story points é uma medida relativa, usada para estimar o esforço funcional dos requisitos (RAITH et al., 2013). Portanto, aspectos como qualificação da equipe de desenvolvimento são considerados e influenciam no uso da técnica, já que a estimativa é realizada com a atribuição de pontuações para as funcionalidades, pela própria equipe (OWAIS; RAMAKISHORE, 2016).

A técnica Ideal Day, corresponde a estimar levando em conta a quantidade de trabalho que um profissional da área consegue entregar em um dia dedicado a este. Esta, é uma medida empírica utilizada em grande parte por especialistas (ALVES; FONSECA, 2008).

Por fim, a técnica de estimativa por Ponto de Função COSMIC fornece uma estimativa do tamanho funcional, a partir dos requisitos funcionais do usuário, onde cada funcionalidade é discriminada com base em seus movimentos e manipulações de dados (ABUALKISHIK et al., 2017). Foi desenvolvido para ser adequado para uma ampla gama de domínios e aplicações. É considerado um método da 2ª geração dos métodos de medição do tamanho funcional, e foi elaborado para cumprir a norma ISO / IEC14143, que define os conceitos da Medição de Tamanho Funcional (Di Martino et al. 2016).

1.2 Justificativa

Tendo como premissa obter um melhor acompanhamento e controle, bem como, aumentar as chances de sucesso de um projeto, é preciso aplicar métricas para monitorar o desenvolvimento, ter estatísticas e comparações, já que não há forma de controlar aquilo que não se pode medir (FERREIRA; HAZAN, 2008).

Sabendo que os projetos de desenvolvimento ágeis de software contam com os fatores de serem iterativos e incrementais, tendo assim uma evolução e movimentação constante nos requisitos, a aplicação e obtenção de indicadores de métricas de estimativa são cruciais para o gerenciamento e apresentam uma maior dificuldade de aproximar a o estimado do real (TANVEER; GUZMáN; ENGEL, 2016).

A escolha da métrica de estimativa tem uma grande influência na geração de indicadores precisos e que revelem um valor significativo para o planejamento e tomadas de decisão.

Portanto, neste trabalho serão comparadas três técnicas de estimativa de esforço: **story points, ideal day e Ponto de Função COSMIC, com relação a suas acurácias**. Gerando resultados que auxiliem na escolha da métrica que apresente uma maior confiança, reduzindo assim a probabilidade de superestimação ou subestimação no planejamento do uso da força de trabalho dos projetos.

1.3 Questões de Pesquisa

Com o foco de comparar as técnicas de estimativas de esforço, as seguintes questões de pesquisa (QP) foram levantadas:

(QP.1) No contexto de desenvolvimento ágil quais são os aspectos que diminuem a acurácia das técnicas de estimativas de esforço?

(QP.2) Quais das técnicas de estimativas no contexto de desenvolvimento ágil são mais utilizadas: Story Point, Ideal Day ou Ponto de função COSMIC?

(QP.3) No contexto de desenvolvimento ágil entre as estimativas Story Point, Ideal Day e Ponto de função COSMIC qual possui uma maior acurácia?

1.4 Objetivos

1.4.1 Objetivo Geral

O objetivo geral deste trabalho é apresentar informações que auxiliem na escolha de uma ferramenta ou técnica de estimativa de esforço ágil, e com base nessas informações entregar uma proposta de desenvolvimento de um software que apoie, facilite e melhore o modo e a acurácia das estimativas de esforço em processos ágeis de desenvolvimento de software.

1.4.2 Objetivo Específicos

Para alcançar o objetivo geral, foram definidos alguns objetivos específicos:

- Estudo da relevância do uso das métricas no desenvolvimento de software.
- Estudo das técnicas de estimativa em métodos de desenvolvimento ágil de software.
- Identificar a partir da bibliografia levantada quais são os maiores influenciadores de baixas em acurácia de métricas de estimativa de esforço.
- Mapear quais são os maiores influenciadores de baixas em acurácia de métricas de estimativa de esforço.
- Mapear as principais técnicas de estimativas de esforço em processos ágeis.
- Identificar a técnica de estimativa de esforço mais em utilizada em processos ágeis de desenvolvimento.
- Propor o desenvolvimento de um software que apoie o método de estimativa.
- Apresentar um relatório com os resultados obtidos.

1.5 Metodologia de Pesquisa

O propósito básico de uma pesquisa científica é empreender pressupostos válidos, estabelecendo conclusões de um todo ou parcela, sendo essas válidas e aptas a explicar um universo maior e mais complexo (EV; GOMES, 2014).

A metodologia de pesquisa deste trabalho foi classificada quanto à abordagem, natureza, objetivos e procedimentos técnicos.

Essa pesquisa foi definida para ser realizada com uma abordagem quantitativa, onde existe uma preocupação com a mensuração do resultado obtido. Logo, utiliza-se da linguagem matemática para descrever as justificativas de um fenômeno, enfatizando o raciocínio dedutivo, as regras da lógica e os aspectos medidos da experiência humana, considerando que a realidade só pode ser entendida com base em análise de dados bruto (FONSECA, 2002).

Do ponto de vista da natureza, foi definida pesquisa aplicada, onde objetiva-se gerar conhecimentos para aplicação prática, guiada pela solução de problemas específicos, envolvendo sobretudo verdades e ambições do ambiente aplicado (GERHARDT; SILVEIRA, 2009).

Quanto ao objetivo é de pesquisa descritiva, onde o investigador fornece e busca várias informações da área de pesquisa, descrevendo fatos e fenômenos encontrados. Nesse tipo de pesquisa os dados são avaliados de forma indutiva pelos pesquisadores, tendo como foco o significado e o processo (GIL, 2008).

Sobre os procedimentos, com o propósito de ter uma aproximação do entendimento e da atuação desses fenômenos, definiu-se o procedimento primário de pesquisa bibliográfica como método de investigação literária, feita com o levantamento de referências teóricas já analisadas, e publicadas por meios eletrônicos e escritos (FONSECA, 2002). Além disso, para concretizar a natureza aplicada foi definido o procedimento secundário de estudo de caso, onde o foco é conhecer com uma maior perícia, o "Como?" e o "Porque?" da relação entre o tema de pesquisa e o ambiente aplicado, buscando a essência do que é mais significativo (GERHARDT; SILVEIRA, 2009).

Então, os procedimentos definidos foram:

- **Pesquisa bibliográfica:** É definida como um estudo elaborado e fundamentado em materiais publicados em bases eletrônicas, revistas, livros, ou seja qualquer material de relevância e que tem acesso público. Neste trabalho algumas bases como Scopus, Capes e IEEE foram utilizadas para a construção da base literária (MORESI et al., 2003).
- **Estudo de caso:** Pode ser caracterizado como o estudo de uma entidade bem

definida, como uma instituição, um sistema ou até uma pessoa. É um método de pesquisa de campo, que busca investigar fenômenos à medida que ocorrem, sem que a entidade interfira na percepção do investigador (GERHARDT; SILVEIRA, 2009).

Assim, com base nos conceitos metodológicos científicos explorados em (GERHARDT; SILVEIRA, 2009), foi desenhado um diagrama do processo metodológico de pesquisa definido para este trabalho, de forma que fique mais claro e especificado. O diagrama é apresentado na Figura 1.

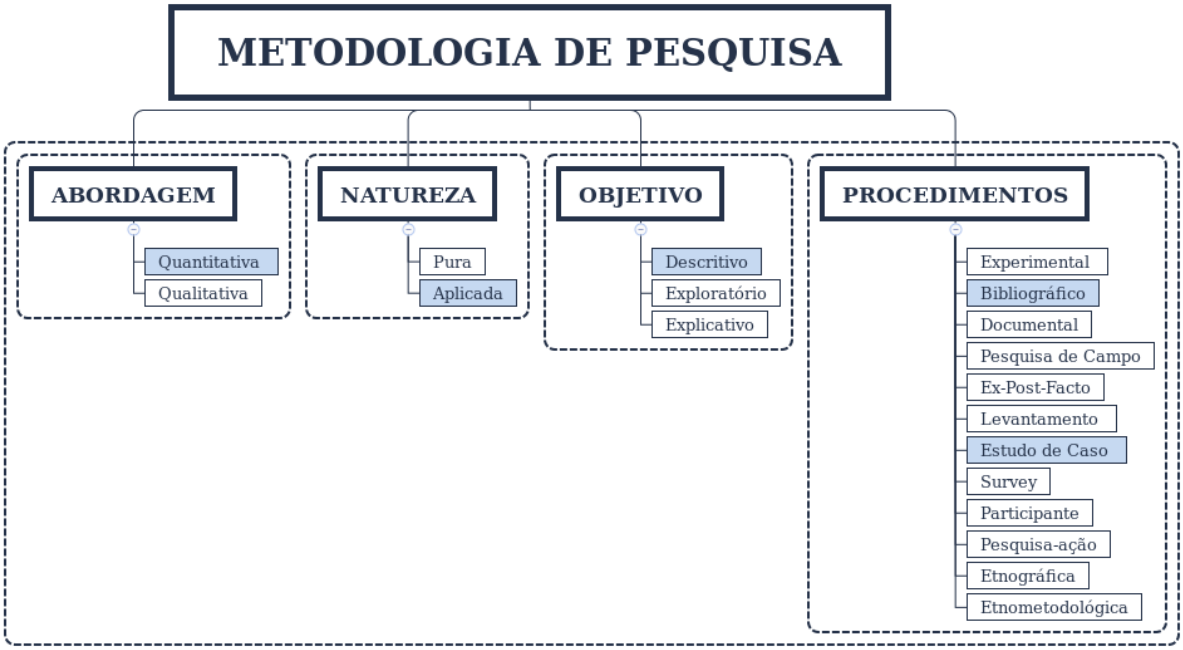


Figura 1 – Visão Geral da Metodologia de Pesquisa. Fonte: Autor

As etapas gerais de realização dessa pesquisa são respectivamente: Na primeira etapa, o **Planejamento da pesquisa**, que busca identificar o escopo de atuação, questões de pesquisa e objetivos da realização desse trabalho; Na segunda etapa a **Coleta e análise de dados** é o cerne, onde são coletados e analisados os dados para busca do entendimento e resultados pré-estabelecidos no passo anterior; Por fim, na terceira e última fase o **Relato dos resultados** é realizado, de forma que se formalizam os resultados, obtidos na análise dos dados, por meio de redação dos resultados da pesquisa. A Figura 2 apresenta o protocolo de pesquisa proposta.

Figura 2 – Visão Geral do Protocolo da Pesquisa. Fonte: Autor

1.6 Organização do Trabalho

Este trabalho está organizado em seis Capítulos, além deste.

No *Capítulo 2 - Referencial Teórico*, são apresentados conceitos essenciais para o entendimento deste trabalho, como: Qualidade de software; Estimativas de Software; Estimativas de Esforço de Desenvolvimento de Software; Story Points; Ideal Day; Ponto de Função.

No *capítulo 3 - Materias e Métodos*, é apresentado um detalhamento das fases de desenvolvimento da pesquisa realizada através do procedimento de estudo de caso.

No *Capítulo 4 - Análise dos Dados*, são apresentados os resultados de comparação dos dados.

No *Capítulo 5 - Desenvolvimento*, são apresentados os documentos e a proposta de desenvolvimento da ferramenta deste estudo.

No *Capítulo 6 - Conclusão e trabalhos futuros*, é apresentado a conclusão deste trabalho, bem como, eleição e disposição de potenciais trabalhos futuros.

2 Referencial Teórico

Este Capítulo aborda conceitos relacionados ao processo de desenvolvimento de software e a abordagem ágil de desenvolvimento, medição de software, o processo de medição de software e as métricas de estimativa em diversos aspectos.

2.1 Processo de Desenvolvimento de Software

Um processo de produção é resultado de ações que se inter-relacionam de forma direcionada e dinâmica com objetivo de transformar determinados elementos. São conhecidos como fatores os bens utilizados com fins produtivos, os elementos de entrada que passam a ser elementos de saída, oriundos de um processo que incrementa o seu valor. (MARTINS; LAUGENI, 2013)

Segundo (CRUZ, 2005) um processo é uma transformação de entradas em saídas, atendendo sobretudo as expectativas e desejos do cliente, a partir da conformidade com várias características de qualidade. Para isso, o processo deve ser estável. Controlar as entradas de um processo é uma tarefa difícil, podendo interferir diretamente na estabilidade, bem como afetar o ambiente de produção de um software.

Um processo contém mais outros dois elementos básicos para o seu andamento: o controle e os recursos. Desse modo, um processo é formado basicamente por quatro elementos, como destaca (CRUZ, 2005):

- **Entradas:** são as matérias-primas, materiais ou informações que oferecem o subsídio para o processo ser realizado;
- **Saídas:** são os serviços ou produtos alcançados pelo processo;
- **Controle:** é o responsável pela verificação da conformidade com diretrizes, métodos, objetivos, procedimentos e padrões na realização do processo;
- **Recursos:** são a provisão de tudo que se faz indispensável para a realização do processo como recursos administrativos, financeiros, humanos, materiais, informações, entre outros.

Conforme Watts Humphrey, citado por (CONTI; TREIN, 2012), que instituiu o programa de melhoria de processos de software do Carnegie Mellon University, o CMMI, grande parte do que compõe o processo de desenvolvimento de um software são atividades

de engenharia, empenhos técnicos e gerenciais, fundamentais para transformar uma coleção de necessidades, anseios e ressalvas do cliente em uma solução, conforme apresentado na Figura 3.

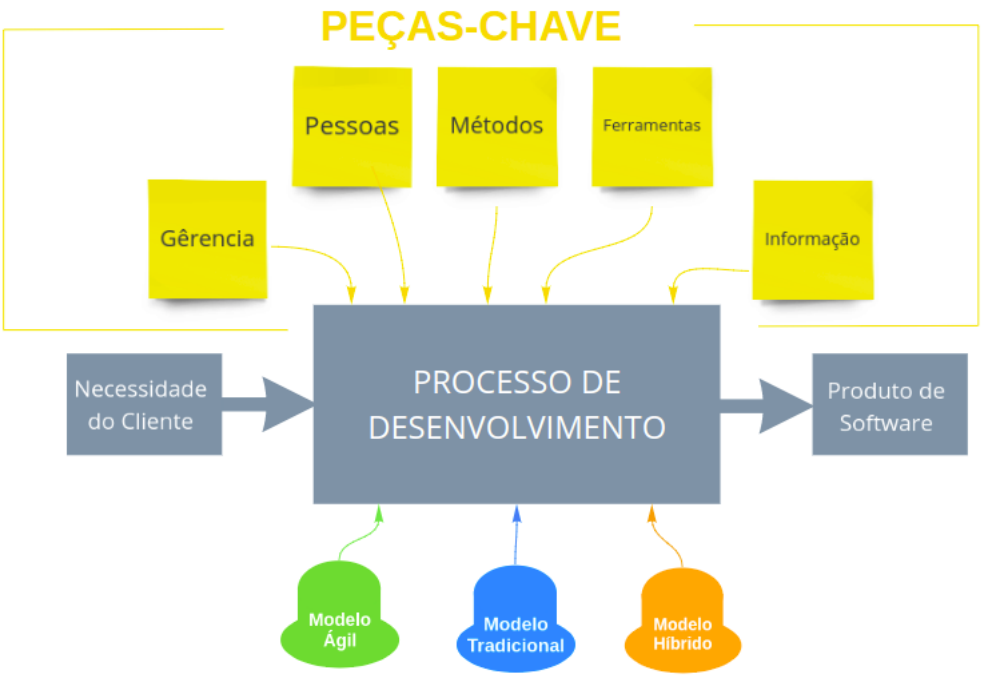


Figura 3 – Processo Básico de Desenvolvimento de Software (ROCHA; MALDONADO; WEBER, 2001), Adaptado

Portanto, é indispensável investir tempo na escolha e montagem do processo de desenvolvimento para se alcançar software de qualidade. Os fatores culturais e organizacionais devem ser responsavelmente ponderados, influenciando diretamente na escolha e definição, uma vez que esses são tão ou mais importantes do que os aspectos técnicos da execução, tornando-se fundamentais para o sucesso dos projetos.(ROCHA; MALDONADO; WEBER, 2001).

2.1.1 Metodologias Ágeis do Desenvolvimento de Software

Na história do desenvolvimento de software foram introduzidas inúmeras abordagens diferentes, tradicionais e ágeis, das quais apenas algumas sobreviveram e são utilizadas (ABRAHAMSSON; SALO; RONKAINEN, 2002).

Esses processos apresentavam uma teoria bem desenvolvida onde transmitiam boas perspectivas, mas na prática seus ideais nem sempre funcionaram bem. As mudanças de ideias do produto por parte dos clientes eram um fator determinante para o insucesso, já que depois de meses, ou mesmo anos, das coletas de requisitos e construção de documentos, os clientes ainda não tinham certeza do que eles queriam. Um segundo ponto são os requisitos, que tendiam a mudar no meio do processo de desenvolvimento, e quando eram

alterados fica difícil parar a cadeia de produção para incluir as mudanças (COHEN; LINDVALL; COSTA, 2003). Entre os processos manifestados o processo cascata foi um grande percursor e base para criação de outros métodos.

O modelo cascata deveria resolver o problema de mudanças nos requisitos, congelando e não permitindo qualquer alteração por parte dos cliente e da equipe de desenvolvimento, mas os profissionais perceberam que os requisitos não poderiam ser simplesmente fixados, como a metodologia havia antecipado (COHEN; LINDVALL; COSTA, 2003).

A busca por entender e melhorar os processos e projetos de desenvolvimento tiveram grandes protagonistas. No início dos anos 90, a IBM Consulting Group contratou o cientista da computação Alistair Cockburn para desenvolver um método de desenvolvimento aprimorado que seguisse o princípio de projetos orientado a objetos. Cockburn construiu um processo com base nas melhores práticas e lições aprendidas no desenvolvimento, e entrevistou as equipes de desenvolvimento da IBM submetidas ao processo. Ele percebeu que as equipes bem sucedidas se desculparam por não terem seguido o processo formal, e usado ferramentas de alta tecnologia, ao invés disso, eles apenas sentaram um perto do outro e discutiam enquanto produziam. Já as equipes que falharam seguiram processos formais e ficaram confundidos com o motivo de não ter funcionado, afirmando em alguns momentos que talvez eles não tinham seguido o processo tão a risca. Cockburn usou o que aprendeu com está experiência na IBM para desenvolver sua metodologia ágil, o Crystal. (COHEN; LINDVALL; COSTA, 2003).

O mundo do desenvolvimento estava estimulado às mudanças, apesar dos métodos tradicionais de desenvolvimento ainda serem moda, era visível que eles nem sempre entregavam o pretendido. Então, alguns desenvolvedores reconheciam que cada vez mais eram necessárias novas práticas para lidar melhor com as mudanças de requisitos. E que estas novas práticas deveriam ser orientadas às pessoas, com princípios de flexibilidade tanto em regras quanto contratos, gerando um ambiente dinâmico. (COHEN; LINDVALL; COSTA, 2003).

No início de 2001, profissionais do desenvolvimento de software se reuniram na estação de ski Snowbird, em Utah, nos Estados Unidos para discutir alternativas para desenvolver software de forma mais leve, rápida e centrada em pessoas (PRIKLADNICKI; WILLI; MILANI, 2014). Estes especialistas não eram contrários aos métodos, processos ou metodologias. Defendiam a modelagem e a documentação do software, mas de forma moderada. Percebiam o valor do planejamento, mas reconheciam o limite entre o engessamento do processo e a liberdade de construção, entendendo que o princípio de sucesso era o equilíbrio entre o planejamento a previsibilidade e a flexibilidade (DIAS, 2010). Disso tudo surge o nome e o conceito “Desenvolvimento ágil de software” e “Metodologias Ágeis”, que foi concretizado em um manifesto denominado “Manifesto Ágil” (PRIKLADNICKI; WILLI; MILANI, 2014).

Segundo (COHEN; LINDVALL; COSTA, 2003), grande parte das práticas ágeis não eram novidades. O que era relevante e gerava bastante diferença em relação aos métodos tradicionais eram o foco e os valores por trás dos métodos ágeis. De acordo com (DIAS, 2010), o cerne desse movimento foi a remodelagem do enfoque de desenvolvimento de software baseando os principios em: agilidade; flexibilidade; habilidade de comunicação; capacidade de entregar produtos novos que agreguem valor em curto período de tempo. A agilidade e flexibilidade deixavam de ser motivos de falta de estrutura, e assumiam posições de melhora na habilidade de criar e reagir a mudanças.

2.1.1.1 Manifesto Ágil

Em 2001, um manifesto denominado “Manifesto Ágil”, foi publicado por um grupo de profissionais, o qual abriu a possibilidade para uma nova era no desenvolvimento de software (PRIKLADNICKI; WILLI; MILANI, 2014). Esse manifesto é guiado por quatro valores maiores (ABRAHAMSSON; SALO; RONKAINEN, 2002), que são:

1. **Indivíduos e interações** acima de processos e ferramentas;
2. **Software em execução** acima de documentação abrangente;
3. **Colaboração com o cliente** acima do contrato negociado;
4. **Responder a mudança** acima de seguir um plano.

Os valores ágeis apresentados à esquerda não excluem os valores da direita, apenas devem ser mais observados e considerados nas tomadas de decisão dos projetos.

De acordo com Highsmith e Cockburn (2001), citado em (ABRAHAMSSON; SALO; RONKAINEN, 2002), "O que é novo no conceito sobre métodos ágeis não são as diferentes práticas utilizadas, mas o reconhecimento das pessoas como os principais impulsionadoras do sucesso do projeto, juntamente com um foco intenso em eficácia e manobrabilidade. Isso produz uma nova combinação de valores e princípios que definem uma visão de mundo ágil ".

Esses quatro valores, levam a observar doze princípios que ajudam o processo de desenvolvimento a se tornar menos complexo e a aumentar a possibilidade de resposta às mudanças que ocorrem ao longo do mesmo, levando sempre em conta o ponto de vista do cliente. Estes princípios fornecem uma boa base para identificar os pressupostos subjacentes aos processos ágeis. Na tabela 1, segundo (TURK; FRANCE; RUMPE, 2014), identificamos os princípios e respectivamente os seus pressupostos.

Tabela 1 – Princípios dos métodos ágeis de desenvolvimento de software. Fonte: (TURK; FRANCE; RUMPE, 2014)

Doze Princípios Ágeis
1. "Nossa maior prioridade é **satisfazer o cliente através da entrega rápida e contínua** de software significativo".
2. "Pessoas da **área de negócio e programadores devem trabalhar juntos**, diariamente, ao longo de todo o projeto".
3. "As **mudanças de requisitos**, mesmo que numa etapa avançada do desenvolvimento, **devem ser consideradas**".
4. " **Entregue continuamente** novas versões do software".
5. "O **software em funcionamento determina** o percepção de **progresso** do projeto".
6. "As pessoas do projeto devem estar motivadas. O **ambiente propício e todo apoio necessário deve ser fornecido as pessoas**, acreditando em sua capacidade de realização do trabalho".
7. " **Equipes auto-organizadas emergem com mais eficiência** as melhores arquiteturas, requisitos e projetos".
8. "A **comunicação face a face é o método mais eficiente** de distribuir a informação e o conhecimento entre uma equipe de desenvolvimento".
9. "Processos ágeis geram **desenvolvimento sustentável**".
10. "A atenção contínua na **excelência técnica** e ao bom design de projeto **aumentam a agilidade**".
11. "É **essencial simplicidade**".
12. "**Equipes de projeto avaliam seu desempenho** em intervalos regulares, **ajustando seu comportamento de acordo com os resultados**".

2.1.2 Scrum

Na literatura as primeiras referências do termo “Scrum” apontam para o artigo de Takeuchi e Nonaka (1986), apresentando o processo adaptativo de desenvolvimento de produtos, rápido e auto-organizável, originário no Japão. Observa-se também o uso do termo Scrum em estratégias de jogo de rugby, onde traz o significado de “Trazer uma bola de fora do jogo de volta” (ABRAHAMSSON; SALO; RONKAINEN, 2002).

A termo Scrum na área de tecnologia é uma abordagem desenvolvida para gerenciar o processo de desenvolvimento de softwares. É de natureza empírica, aplicando os conceitos da teoria de controle de processos industriais ao desenvolvimento de sistemas, resultando em características como flexibilidade, adaptabilidade e produtividade nos projetos (SCHWABER; BEEDLE, 2002).

Para (SCHWABER; SUTHERLAND, 2013) o Scrum é um framework de desen-

volvimento e manutenção de projetos de produtos de software, que entrega produtos complexos com alto valor para o projeto.

Conforme Udo e Koppensteiner (2003) em (DIAS, 2010), o Scrum possui um maior destaque em relação aos outros métodos ágeis por dar maior ênfase ao gerenciamento de projeto. Atividades específicas de feedback e monitoramento são definidas, onde em grande parte, reuniões rápidas e diárias com a equipe, objetivando à identificação e à correção de deficiências e/ou impedimentos no processo de desenvolvimento, são base para a gestão.

O Scrum é uma forma de melhorar as práticas de engenharia existentes (por exemplo práticas de teste) em uma organização, envolvendo atividades de gerenciamento contínuas, com o objetivo de identificar sempre as deficiências e/ou impedimentos no processo de desenvolvimento, bem como nas práticas que são usadas (ABRAHAMSSON; SALO; RONKAINEN, 2002).

Para a aplicação do Scrum deve-se empregar uma estrutura iterativa e incremental, de forma que no início de cada iteração a equipe decida o que deve ser feito levando em conta a necessidade de entregar um incremento de valor ao produto, no final de cada iteração (LIBARDI; BARBOSA, 2010).

A cada iteração a equipe faz um apanhado de análises, entre elas, análise dos requisitos, análise tecnologia, habilidades da equipe, e com base nisso um planejamento é gerado de forma que possam construir e entregar uma nova iteração com o melhor produto de software possível adaptando-se sempre, conforme surgem as complexidades, dificuldades e surpresas. Assim, a iteração é o coração do Scrum (COHN, 2005).

As principais características para nortear a aplicação do Scrum se encontram na tabela 2.

Tabela 2 – Características e valores básicos da aplicação do Scrum. Fonte: (DIAS, 2010).

Característica	**Valores**
Tamanho da Equipe	Recomenda-se equipes de trabalho de até sete pessoas, de forma dividida por área de atuação.
Duração das iterações	Recomenda-se o uso de iterações com duração em média de quatro semanas.
Equipes distribuídas	O projeto pode ser gerado por várias equipes pequenas, há possibilidade de que estas estejam distribuídas (descentralizadas).

2.1.3 Extreme Programming

A metodologia de desenvolvimento Extreme programming foi introduzida por Beck, Jeffries, et. em 1998, tendo maior popularização em 1999 com “Extreme Programming Explained: Embracing Change” (BECK, 1999b), além de outros artigos produzidos por desenvolvedores desencantados com métodos tradicionais, procurando algo novo.

O Extreme Programming (XP) surgiu da evolução de problemas causados pelos longos ciclos de desenvolvimento dos modelos de desenvolvimento tradicionais (BECK, 1999a). Embora boa parte das práticas do XP não eram novidade, o XP trouxe uma nova visão, onde essas técnicas foram coletadas e alinhadas para funcionarem entre si de uma forma inovadora, formando assim uma nova metodologia para desenvolvimento de software. O termo "Extreme"vem de levar esses princípios e práticas de senso comum a níveis extremos (BECK, 1999b).

Beck (BECK, 1999b) estabelece o XP como uma metodologia eficiente e de baixo risco, flexível, previsível, científica e adequada para pequenas e médias equipes de desenvolvimento de software, onde os requisitos do produto são voláteis e/ou vagos.

Segundo Cohen (COHEN; LINDVALL; COSTA, 2003) e Sommerville (SOMMERVILLE, 2007), o Extreme Programming (XP) é sem dúvida um dos Método Ágeis mais conhecido e utilizado no desenvolvimento de software.

O custo de mudança de um software durante o desenvolvimento, já foi motivo para repensar métodos e técnicas. Mas, ao contrário do que se pensava, esse custo não aumenta exponencialmente com o avanço do projeto (DIAS, 2010). Novas técnicas desenvolvidas, como: bancos de dados relacionais, programação modular, dentre outras, possibilitaram apoiar mudanças no projeto com um custo reduzido, como visto na Figura 4.

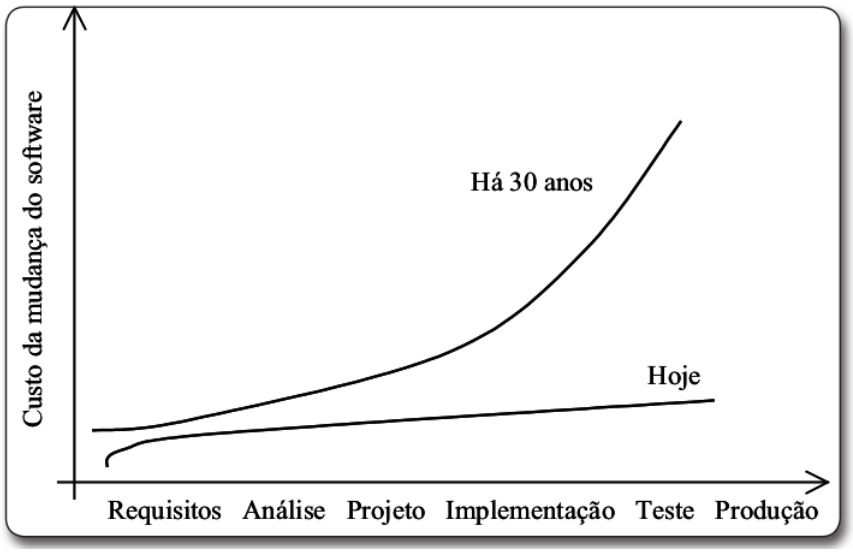


Figura 4 – Custo da Mudança do Software. Fonte: Beck (2004), em (DIAS, 2010).

Portanto dedicar esforço na tentativa de evitar mudanças e/ou adequações no produto em desenvolvimento já não se faz necessário, abrindo espaço para o aspecto de flexibilidade dos métodos ágeis (DIAS, 2010).

2.1.3.1 Valores e Regras

Beck (BECK, 1999b) menciona que para encontrar o sucesso no uso da metodologia XP é necessário seguir seu processo com disciplina, atendendo às suas recomendações e práticas. Então, **doze regras** concisas e diretas, são definidas para o desenvolvimento com Extreme Programming (BECK, 1999b). Para Cohen (COHEN; LINDVALL; COSTA, 2003), o método é tão compreensível que poderia ser aplicado sem ler uma página do livro de Beck.

1. **Jogo de planejamento:** Clientes, gerentes e desenvolvedores se reúnem para elaborar, estimar e priorizar os requisitos (são chamados de "histórias de usuários") para a próxima versão.

2. **Pequenas versões:** Após as primeiras iterações uma primeira versão do sistema é posta em produção. Posteriormente, as versões funcionais são colocadas em produção pouco a pouco, com o avanço das iterações.

3. **Metáfora:** Clientes, gerentes e desenvolvedores geram uma metáfora da identidade visual ou conjunto de metáforas após a modelagem do sistema.

4. **Design simples:** Os desenvolvedores são estimulados a manter o design tão simples quanto possível.

5. **Testes:** Os desenvolvedores criam testes primeiro, antes mesmo de escreverem o próprio código. Todos os testes devem ser executados ao final de cada iteração.

6. **Refatoração:** Com o passar das iterações os trabalhos devem garantir que tanto o design quanto o código sejam desenvolvidos mantendo-os o mais simples possível.

7. **Programação de par:** Os desenvolvimentos devem ser em par, onde um desenvolvedor reviso o código enquanto o outro escreve, e vice-versa.

8. **Integração contínua:** O código deve estar integrado com um ambiente de teste, onde o código só é aprovado se todos os testes passam, ou o código deve ser descartado.

9. **Propriedade coletiva:** O código é de propriedade de todos os desenvolvedores, e eles podem fazer alterações em qualquer lugar no código, quando quiserem.

10. **Cliente no local:** O cliente trabalha com a equipe de desenvolvimento em todos os momentos, respondendo a perguntas, realizando testes de aceitação e garantindo que o desenvolvimento avance conforme o esperado.

11. **Semanas de 40 horas:** As histórias devem ser selecionadas para cada iteração, de modo que os desenvolvedores não precisam de horas extra para finalizá-las.

12. **Área de trabalho aberta:** Os desenvolvedores trabalham em um espaço de trabalho comum.

O sucesso da aplicação do XP não surge apenas da aplicação das doze regras sozinhas, mas das propriedades alcançadas com as junções delas. Segundo (COHEN; LINDVALL; COSTA, 2003), Highsmith elenca **cinco princípios** fundamentais do XP, que são aprimorados à medida que são praticados: **comunicação**, **simplicidade**, **feedback**, **coragem** e **trabalho de qualidade**.

Somente a aplicação do modelo XP não basta para obtenção de sucesso, **quatro princípios** devem ser observados com maior foco para que o método funcione da melhor forma:

- **Tamanho da equipe:** A equipe deve trabalhar na mesma localidade, portanto o tamanho da equipe é limitado ao número de pessoas que cabem no espaço de produção, recomenda-se algo perto de 2 a 10 pessoas;

- **Comprimento de iteração:** O XP possui a recomendação de tempo de iteração mais curto dos métodos ágeis, recomenda-se 2 semanas;

- **Suporte para equipes distribuídas:** Equipes distribuídas não são suportadas, devido ao foco do XP na comunidade e na co-localização;

- **Aplicabilidade:** o XP não é necessariamente orientado para um tipo específico de software a ser desenvolvido. Não deve haver nada no próprio XP que limite a sua aplicabilidade.

2.1.3.2 Etapas

Os projetos executados com a metodologia Extreme Programming contam com a aplicação de **seis fases**, sendo: **fase de exploração**, **fase de planejamento**, **fase de iteração**, **fase de Produção**, **fase de Manutenção** e **fase de morte**.

1. Na **fase de exploração**, anterior a construção do software, os clientes descrevem em cartões de histórias o que eles desejam incluir no primeiro lançamento. Cada cartão de história descreve um recurso a ser adicionado ao programa. Ao mesmo tempo, o

time do projeto se familiariza com as ferramentas, tecnologia e práticas que estarão usando no projeto. A tecnologia a ser utilizada será testada e as possibilidades de arquitetura para o sistema são exploradas através da construção de um protótipo do sistema (ABRAHAMSSON; SALO; RONKAINEN, 2002).

2. A **fase de planejamento inicial**, define a ordem de prioridade para as histórias e firmam um acordo sobre o conteúdo entregue na primeira versão. Os programadores primeiro estimam quanto esforço cada história exige, e então o cronograma é montado (ABRAHAMSSON; SALO; RONKAINEN, 2002).

3. A **fase de iterações da release**, inclui a definição das várias iterações dos sistemas antes da primeira versão. Na primeira iteração deve-se criar toda a arquitetura do software. O cliente é quem decide quais histórias são selecionadas para cada iteração (ABRAHAMSSON; SALO; RONKAINEN, 2002).

4. A **fase de modo de produção**, exige testes extras e verificação do desempenho do sistema antes que o sistema possa ser liberado para o cliente. Nesta fase, novas mudanças ainda podem ser encontradas e decisões de inclusão ou não das mudanças devem ser tomadas (ABRAHAMSSON; SALO; RONKAINEN, 2002).

5. Na **fase de manutenção**, depois que o primeiro lançamento, de valor para o cliente, é realizado os projetos em XP devem manter o sistema em modo de produção em execução, além de produzir e agregar novas funcionalidades. Para fazer isso, a fase de manutenção requer um esforço também para tarefas de suporte ao cliente (ABRAHAMSSON; SALO; RONKAINEN, 2002).

6. A **fase de morte**, corresponde ao término de um projeto de XP. Isso é, quando o cliente já não possui histórias a serem implementadas. Este é o momento no processo XP que a documentação necessária do software escrita, pois não são feitas mais alterações na arquitetura, design ou código. A morte do projeto também pode ocorrer se o sistema não estiver entregando os resultados desejados, ou se ele for muito caro para o desenvolvimento (ABRAHAMSSON; SALO; RONKAINEN, 2002).

A Figura 5, apresenta um compilado das seis fases, bem como a integração das doze regras da metodologia XP (BECK, 1999b).

Figura 5 – Projeto Extreme Programming. Fonte: (WELLS, 2000).

2.2 Medição de Software

Desenvolver software de qualidade é uma tarefa complexa, envolvendo desde desafios tecnológicos até os desafios mais comuns de todo projeto, como montar a equipe certa e manter todo o processo alinhado e controlado (BHARDWAJ; RANA, 2016).

O processo de produção é indispensável para geração de produtos de qualidade, e auxilia na busca por uma visão maior e mais equilibrada dos aspectos essenciais e acidentais da engenharia de software e do processo. De acordo com (ROCHA; MALDONADO; WEBER, 2001), para a construção de um software com qualidade é necessário um processo com normas, procedimentos e gerência bem definidos, diferentemente da visão de processo de produção centrado em fases/produtos.

Com essa visão o papel da gerência passa a ser observado como um aspecto técnico e não como um aspecto exterior ao conhecimento da engenharia de software, onde deve procurar focar na aquisição de dados sobre o processo, bem como, a transformação destes em conhecimento. (ROCHA; MALDONADO; WEBER, 2001)

Segundo (FENTON; PFLEEGER, 1997), avaliar quantitativamente processos e produtos com o objetivo de facilitar análises, estimativas e controle do processo de desenvolvimento, é propriamente o conceito de medição aplicada à área da engenharia de software.

Em (SOMMERVILLE, 2007), a medição de software representa a obtenção de um valor numérico para um atributo de um sistema, processo ou componente de software. Com a medição, conclusões de qualidade do software, eficiência e eficácia de métodos, e a relevância de ferramentas devem ser percebidas.

2.2.1 Estimativa de Software

Segundo (PRESSMAN, 2006), a definição de quanto tempo e os recursos que devem ser empregados na construção de um software, ou produtos derivados, deve ser realizada com base em estimativas.

A estimativa deve ser considerada a atividade básica do planejamento de projetos de software. Estimativas eficientes permitem a percepção de viabilidade de projetos, bem como, são base para o controle e escolha de ferramentas a serem empregadas. A definição de cronogramas, orçamentos, plano de riscos, não possui credibilidade sem o uso de estimativas confiáveis.

A subestimação e superestimação dessas medidas, por um lado causará uma pressão maior aos projetos, resultando em altas taxas de defeitos e custo de desenvolvimento. Por outro, causarão desperdícios de recursos, prejudicando a competitividade do contrato (CAO, 2008).

Quase 60% de todos os projetos de software falham ou são desafiados com o excesso de custo ou tempo, conforme relatório do Grupo Standish, Group CHAOS 2006. Logo, a estimativa é um desafio presente em cada projeto de software (CAO, 2008).

Em (VAZQUEZ; SIMÕES; ALBERT, 2003), o processo de estimativa de projetos de software envolve **quatro atividades**:

1. **Estimar o tamanho do produto** a ser desenvolvido;
2. **Estimar o esforço** aplicado na execução do projeto;
3. **Estimar o prazo (duração)** do projeto;
4. **Estimar o custo** do projeto.

Para (HAZAN, 2008a), o processo básico de estimativas de projetos de software deve ser aderente a área de processo e planejamento. A primeira entrada desse processo é o documento de requisitos. Para garantir a qualidade da estimativa, o responsável pelas estimativas deve analisar antecipadamente o documento de requisitos prevendo quaisquer intempéries. Com a realização da estimativa de tamanho e com base em dados históricos de projetos é possível obter outras medidas, como: estimativas de esforço; prazo (cronograma); custo (orçamento). A Figura 6 demonstra a relação entre as estimativas e a possibilidade e necessidade de iniciar um ciclo constante de melhoria nelas.

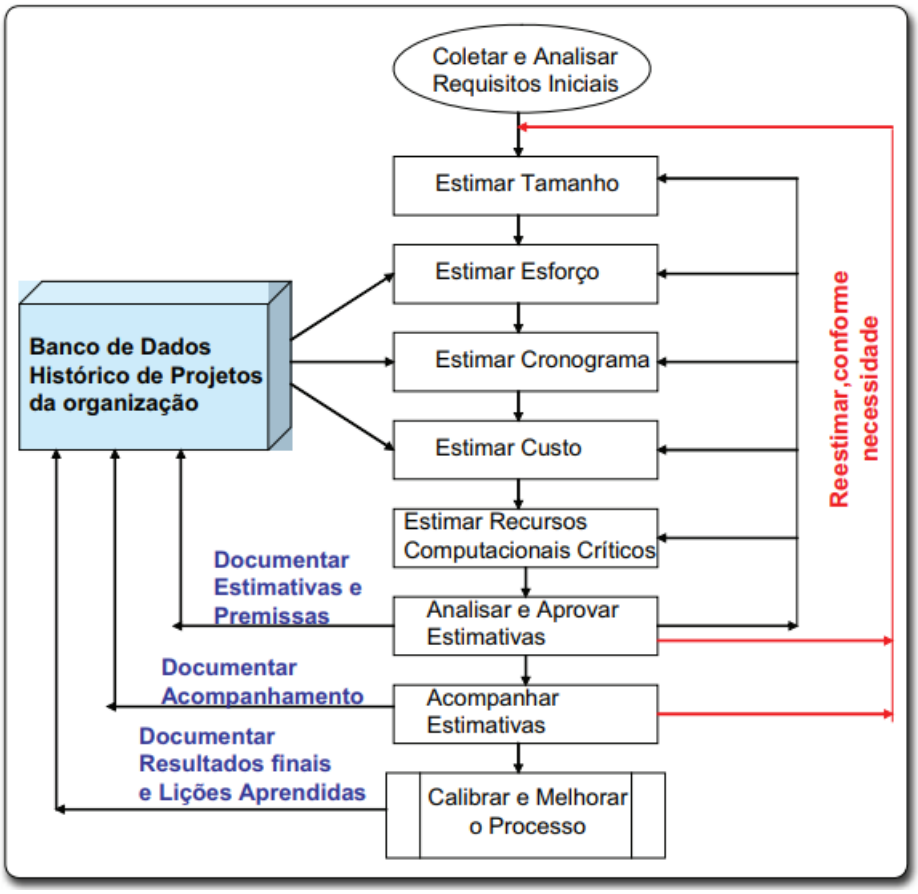


Figura 6 – Processo de Estimativas de Projetos de Software. Fonte: (HAZAN, 2008a)

O modelo de estimativa utilizado pelas equipes de projetos de software pode variar de acordo com o modelo do processo utilizado por elas, mas o uso de conhecimentos prévios nas diversas técnicas, bem como, estabelecer uma base histórica de estimativas já realizadas em outros projetos, agregam um valor e qualidade altos às técnicas aplicadas.

As incertezas diminuem significativamente à medida que novos conhecimentos são obtidos durante o projeto. De qualquer forma, no início, quando não se conhece muitas informações, a estimativa do esforço é muito mais difícil. À medida que o projeto avança, as estimativas tornam-se cada vez mais precisas como apresentado na Figura 7. Na fase inicial do projeto, o cone de incerteza mostra que o valor real varia de 40% a 250%, segundo McConnell, 2006 por (DRAGICEVIC; CELAR; TURIC, 2017) ou de 60% a 160% segundo Kan, 2002 por (DRAGICEVIC; CELAR; TURIC, 2017), comparando com a estimativa.

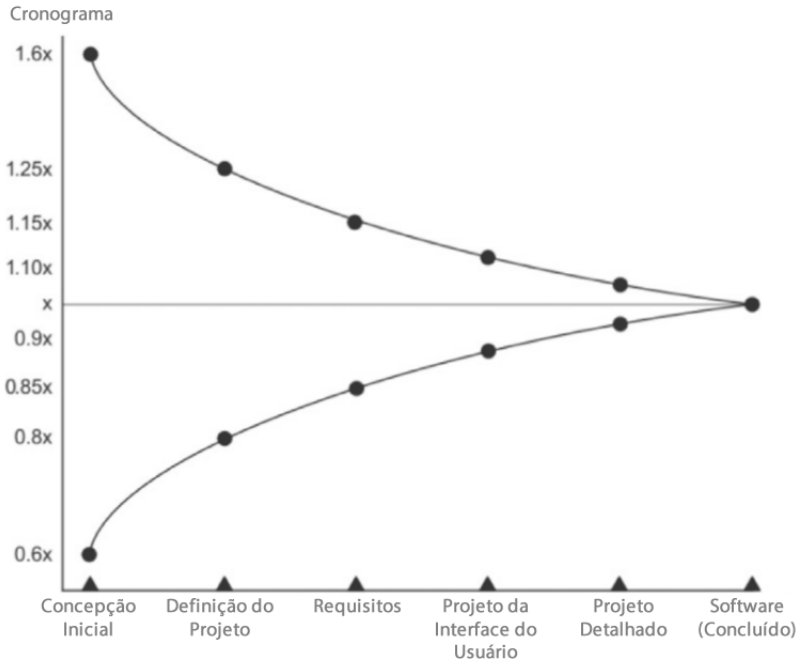


Figura 7 – Cone de incertezas. Fonte: (DRAGICEVIC; CELAR; TURIC, 2017) obtido de (KAN, 2002)

Segundo (SOMMERVILLE, 2007), as técnicas utilizadas para estimar tem como base esses princípios:

- **Técnicas baseadas em experiências**:

  A utilização de conhecimento previamente adquirido de projetos anteriores de mesmo domínio, onde o gerente faz um juízo fundamentado, de quais são os requisitos de esforço.

- **Técnicas baseadas em modelagem algorítmica de custos**

  É definida uma fórmula matemática para antecipar os custos do projeto, de forma que esta considere diversos fatores, como: Tamanho; Tipo do projeto; Tamanho da equipe; Características do processo e produto.

Portanto, estimar medidas de um projeto não é uma tarefa fácil, já que com base em previsões qualitativas deve-se alcançar, entender e estabelecer uma visão quantitativa. A realização dessas previsões não partem do uso apenas da experiência, mas da utilização de técnicas disponíveis na literatura.

2.2.1.1 Estimativa de Tamanho

O ponto de partida para realização de estimativas de software é o tamanho do produto, já que este indicador é o insumo principal para as métricas de estimativa de

custo, esforço e tempo de desenvolvimento. Assim, a precisão desse indicador influencia diretamente na precisão de outros indicadores, tornando-se fundamental (HAZAN, 2008a).

O tamanho do software significa diretamente a quantidade de trabalho, em uma unidade de medida especificada, a ser empreendido no desenvolvimento de um projeto.

A elaboração da estimativa de tamanho é dependente direta do conhecimento e detalhamento do projeto, portanto, o controle dos projetos é indispensável. Tendo em vista a aplicação da estimativa de tamanho na primeira fase do processo de desenvolvimento, faz-se necessário uma reciclagem dessas estimativas durante todo o projeto, levando sempre em conta as mudanças (MONTEIRO, 2005).

Duas abordagens são mais utilizadas para medir tamanho funcional, a primeira, "Abordagem Posterior", utiliza como parâmetro as linhas de código (LOC), e a segunda, "Abordagem Anterior", que são técnicas que estimam o tamanho de um software com base nos requisitos definidos (TRAN-CAO; LEVESQUE; ABRAN, 2002).

2.2.1.2 Estimativa de Esforço

A estimativa de esforço é tida como o empenho aplicado para desenvolver um software sob a perspectiva do tempo, onde defini-se a força de trabalho necessária para completar um atividade do projeto. Geralmente é expressa como um total de horas, dias, meses ou semanas gastos por uma força de trabalho humana para finalizar as atividades (FERREIRA; HAZAN, 2008; FREIRE, 2008).

(FREIRE, 2008) destaca que o esforço, é definido como o tempo exigido para completar uma atividade ou outro elemento de um projeto, não sendo confundido com duração.

Para obter o esforço estimado do projeto geralmente utiliza-se um cálculo da estimativa de tamanho do produto. O processo de desenvolvimento, a elaboração de documentos, implementação de protótipos, plano de projeto do produto, revisão e teste do código, devem ser considerados nesse cálculo (MONTEIRO, 2005).

Portanto, estimativas são decisivas na gestão de projetos, onde o uso inadequado levará a falhas, com subestimação ou superestimação de prazos e alocação de equipe no projeto (CAO, 2008).

O método de estimativa de esforço que prevalece como mais usado em desenvolvimentos de software é com base no julgamento, embora o uso de modelos de estimativas formais seja bastante explorado na engenharia de software. Mas, existem evidências claras que o uso de estimativas baseadas no julgamento tendem ao superotimismo (JøRGENSEN, 2004a).

Segundo (VAZQUEZ; SIMÕES; ALBERT, 2003), a utilização de dados internos

da própria organização é a melhor forma de se chegar à estimativa de esforço de um projeto de desenvolvimento. Alguns dados como o número de horas destinados em projetos passados, tamanho dos projetos, dimensionado em suas diversas fases, devem ser relevados na estimativa.

Cerca de 60% a 80% dos projetos de desenvolvimento de software ultrapassam a quantidade de esforço e/ou duração estimados, de acordo com a pesquisa de (MOLOKKEN; JORGENSEN, 2003).

A pesquisa do (STANDISH GROUP, 2009) em (JøRGENSEN, 2004b) revela dados semelhantes, onde, apenas 32% dos projetos de desenvolvimento de software são concluídos com sucesso, e 44% dos projetos atrasam e/ou ultrapassam o orçamento, sendo entregues com menos funcionalidades do que propuseram.

Para Jorgensen e Ostvold (2004) em (KOSLOSKI; DE, 2005), existe a necessidade de distinguir os fatores que provocam erros de estimativa, para que as precisões sejam melhoradas. Essa melhora pode ser alcançada buscando comparar e encontrar as semelhanças entre características como os registros de dados históricos de estimativas, e o plano de estimativa vigente, levando sempre à reflexão de pontos de aperfeiçoamento.

Os professores S.D. Conte, H.E. Dunsmore e V.Y. Shen conforme (MCCONNELL, 2006), propõem que uma boa abordagem de estimativa deve fornecer estimativas dentro de 25% a 75% do tempo dos resultados reais. Sendo este o padrão de avaliação de estimativas mais comumente usado para avaliar a precisão das estimativas.

2.2.2 Acurácia

Qualquer medida, seja de estimativa ou não, está sujeita a influência de diversificados tipos de erros, sendo de natureza grosseira, sistemática ou aleatória (randômicos) (MONICO et al., 2009).

De acordo com a norma ISO 5725-1 em (REVISTABW, 2014), acurácia pode ser entendida como a combinação entre a exatidão e a precisão. Exatidão por sua vez, é a aproximação de uma medição com o valor considerado real (ou de referência).

$$\textbf{Acurácia} = \textbf{Exatidão} + \textbf{Precisão}$$

Para (MIKHAIL; ACKERMAN, 1976), a acurácia é o nível de semelhança de uma estimativa com seu parâmetro, ao passo que precisão expressa o nível de consistência da medida com sua média. Acrescentam ainda que acurácia representa a proximidade do valor do parâmetro a uma grandeza estatística que ela foi estimada.

Como apresentado na Figura 8, o afastamento entre o valor de referência e o valor estimado é a acurácia, e a precisão o fator de dispersão dos dados do valor estimado.

Figura 8 – Relação da Acurácia e Precisão. Fonte: (REVISTABW, 2014)

2.2.2.1 Acurácia x Precisão

O uso de exemplos de tiro para explicar os conceitos de acurácia e precisão é bastante comum. Portanto, as Figuras 9 e 10, apresentam a atuação de 4 atiradores, A,B,C e D, onde o objetivo deles é acertar o alvo central.

Na Figura 9, podemos perceber que tanto o atirador A quanto o B possuem baixa tendência ao acerto, com uma diferença, o atirador B apresenta menor dispersão (melhor precisão). Assim, o atirador B é mais preciso e acurado do que o atirador A, embora os dois tenham tendência nula (MONICO et al., 2009).

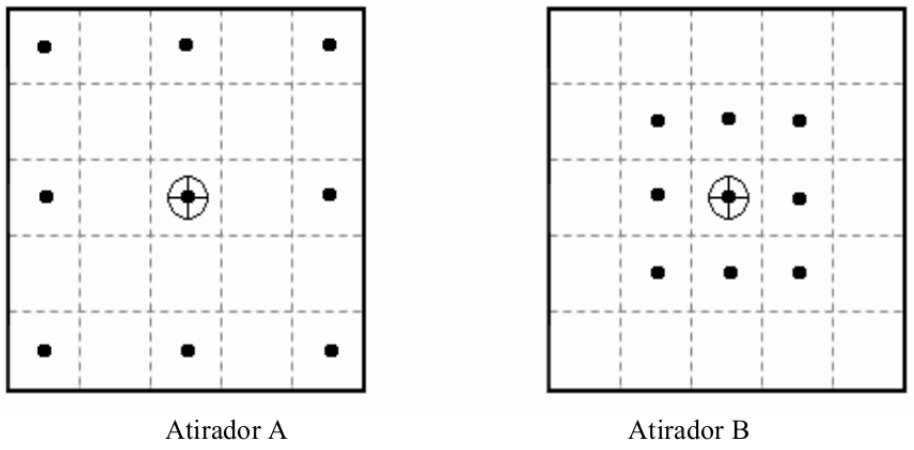


Figura 9 – Acurácia e precisão com tiro ao alvo, atiradores sem tendência. Fonte: (MONICO et al., 2009)

Na Figura 10, outros dois atiradores são apresentados, mas com a diferença de que o atirador D possui uma tendência (atuação de um fator externo). Tanto o atirador C quanto o D apresentam alta precisão nos tiros, porém, somente o atirador C apresenta alta acurácia. Alguns fatores como efeitos sistemáticos e aleatórios foram considerados para produzir essa análise de acurácias, por isso, a tendência do atirador D afeta a qualidade de seus resultados (MONICO et al., 2009).

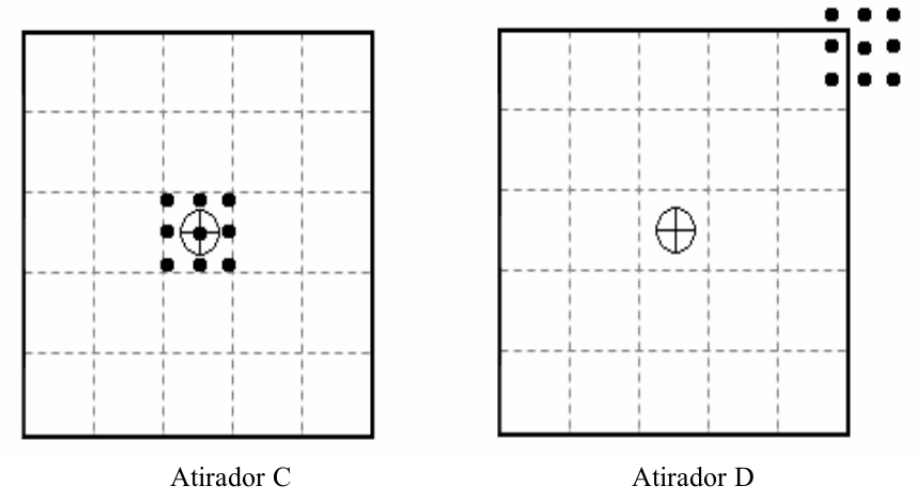


Figura 10 – Acurácia e precisão com tiro ao alvo, atiradores com tendência. Fonte: (MONICO et al., 2009)

Observando todo o cenário dos tiros realizados pelos 4 atiradores, é visível que o atirador C é o mais preciso e acurado, contudo se a tendência, que pode ser de um problema com o equipamento utilizado, do atirador D fosse removida provavelmente ele se igualaria ao atirador C. Em termos de precisão, com base em dispersão temos a seguinte ordem de qualificação dos atiradores, C e D, B, A, variando de menor dispersão à maior (MONICO et al., 2009).

2.3 Métricas de Software

Atribuir números, símbolos a uma entidade, descrevendo o contexto com base em padrões bem definidos, é realizar um processo de medição. Isto é, o processo de medição tem por objetivo capturar dados sobre atributos de uma entidade ou processo para poder descrevê-la com teor científico e sistematizado (FENTON; BIEMAN, 2014)

A ISO/IEC 9126-1 (ISO/IEC. . . , 2001) define medição como "uso de métrica para se obter valor (o qual pode ser numérico ou categoria), obtido por meio de uma escala de um atributo de uma entidade".

As métricas são operações matemáticas que buscam obter informações mais específicas com teor científico, onde tamanho e complexidade são dois aspectos dentre o diversos que podem ser obtidos. Além disso, as métricas de software objetivam prestar um suporte aos desenvolvedores e gestores, oferecendo possibilidades de consolidar a administração de projetos e controlar todo o ciclo de vida da engenharia de software. "Nenhuma investigação humana pode realmente ser chamada de ciência se não pode ser demonstrada matematicamente" (Leonardo da Vinci) (KOSCIANSKI; SOARES, 2006).

As métricas, no processo de desenvolvimento de software, em grande parte são

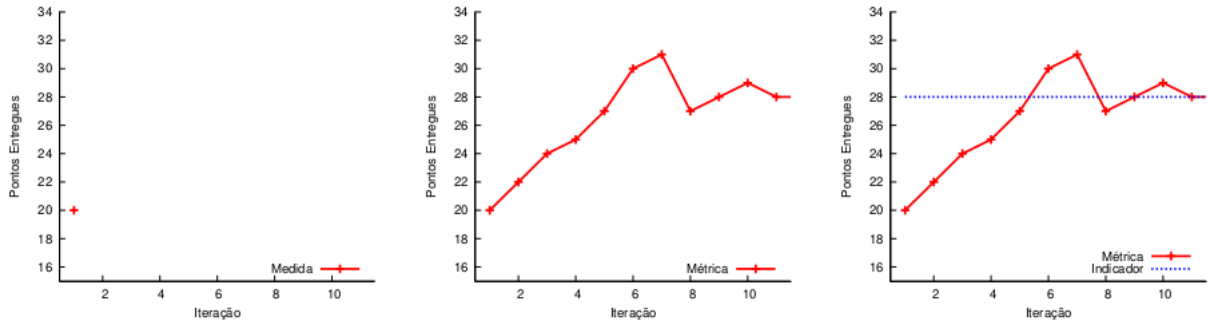
direcionadas e definidas nos primeiros estágios do processo, ampliando assim a visão dos engenheiros (PRESSMAN, 2006). Em (SOMMERVILLE, 2007) as métricas apoiam um ciclo de melhoria constante do processo de desenvolvimento, fornecendo dados quantitativos para que decisões e aperfeiçoamentos sejam tomados.

Segundo (COHN, 2006), alguns critérios devem ser observados no julgamento de qualificação de uma métrica ágil, ela deve: fortalecer princípios ágeis; medir resultados e não saídas; seguir tendências e não números; responder uma pergunta peculiar para uma pessoa real; pertencer a um conjunto pequeno de métricas e diagnósticos; ter facilidade em ser coletada; indicar ao invés de esconder seu contexto e suas variáveis; fornecer feedback contínuo; impulsionar para o alto o nível de qualidade.

Explorar a área de conhecimento das métricas requer uma atenção especial com alguns conceitos, por isso é necessário entender as diferenças entre **medidas, métricas e indicadores**.

- **Medida:** Parte do princípio de avaliar algo diante de um padrão (IEEE, 1983). No desenvolvimento de software, podemos avaliar a adesão de um projeto aos princípios de de maturidade do CMMI, sendo assim, a medida é a qualidade o padrão é o CMMI (Capability Maturity Model Integration).
- **Métrica:** Parte do princípio de determinar se um componente, processo ou sistema, apresenta uma certa característica (IEEE, 1990). As métricas para se diferenciarem das medidas, que são baseadas em uma única informação, cruzam mais duas ou mais medidas para que resultem em uma informação. No desenvolvimento de software, após rodar uma bateria de testes e avaliar essa medida, poderíamos relacionar com a fase encontrada, assim duas medidas estariam cruzadas.
- **Indicador:** Parte do princípio de analisar com base em resultados de um processo ou incidente de alguma condição específica, a determinação de um estado (IEEE, 1990). No desenvolvimento de software, após rodar uma bateria de testes poderíamos determinar que baterias com resultados abaixo de 80% representam algum característica do estado atual do desenvolvimento.

A Figura 11 explica todo esse conceitos de maneira mais didática. A imagem (a) demonstra o conceito de medida, onde os padrões são colocados. A imagem (b) demonstra a métrica, onde com um padrão de medida se realiza uma medição e obtêm-se dados. Na última imagem é possível ver a combinação dos três cenários, onde a partir de um padrão encontra-se uma métrica e com base nos resultados da métrica determina-se um indicador de estado.



(a) Medida (b) Métrica (c) Indicador

Figura 11 – Medida, métrica e indicadores de maneira mais didática. Fonte: (SATO, 2007)

Alguns critérios são utilizados para classificar as métricas, assim uma pessoa que assume o papel de Tracker em um projeto pode identificar qual a categorização de métricas alcança a sua necessidade, bem como, qual métrica deve utilizar para atingir seus objetivos.

- **Métricas Objetiva / Subjetiva**
  - As **métricas objetivas** buscam relacionar medidas com base apenas no objeto, descartando o ponto de vista de quem está interpretando. Um exemplo é obter informações de ferramentas, o número de commits que uma pessoa deu em um repositório é uma métrica objetiva, não sendo influenciada por terceiros (SATO, 2007).
  - Já as **métricas subjetivas** levam em conta tanto o objeto questionado quanto o ponto de vista do interpretador. Um exemplo é a relação de qualidade de código, onde a escala de qualidade pode variar de 0% a 100% e depender diretamente do ponto de vista de quem está envolvido (SATO, 2007).
- **Métricas Quantitativa / Qualitativa**
  - As **métricas quantitativas** são representadas em grande parte por um valor numérico, permitindo assim um padrão de comparação. Os exemplos apresentados acima representam o uso de perspectiva quantitativa, onde números são apresentados (SATO, 2007).
  - Já as **métricas qualitativas**, são representados em grande parte por palavras, símbolos ou figuras. Um exemplo de métrica qualitativa é a percepção do humor da equipe (SATO, 2007).

A utilização de **métricas qualitativas e quantitativas** em um mesmo estudo é considerada uma boa combinação, e os ganhos são enormes tendo em vista a

abrangência da cobertura dessas técnicas. A classificação dessas técnicas agrega um valor implícito da presença dos conceitos das técnicas objetivas/subjetivas, isto é, as métricas quantitativas geralmente estão associadas à objetivas e respectivamente métricas qualitativas à subjetivas (SATO, 2007).

- **Métricas Organizacionais / Acompanhamento**
  - As **métricas organizacionais**, medem a quantidade de valor negocial entregue ao cliente (SATO, 2007). Alguns exemplos de métricas que se enquadram nesse contexto são, "Funcionalidades Testadas e Entregues" métrica proposta por Ron Jeffries, "Tempo Médio de Ciclo" proposta pela abordagem Lean obtido e "Net Promoter Score" proposta por Reichheld, obtidas em (SATO, 2007).
  - Prover informações para auxiliar o time de desenvolvimento, o controle e melhoria de processos, é em parte responsabilidade das **métricas de acompanhamento**. Os dados fornecidos por essas métricas geralmente não são associados a indivíduos, além disso a utilização e observação deve ser entorno de um contexto particular, uma vez que o ecossistema influencia diretamente nos resultados. A velocidade de uma equipe de desenvolvimento é um bom exemplo de métrica de acompanhamento (SATO, 2007).

2.3.1 Planejamento e Estimativas de Projetos Ágeis de Software

Diante das inúmeras opções de técnicas de estimativas para o planejamento de projetos ágeis, três são bastante conhecidas quando o foco é estimativas de tamanho e esforço, Story Points, Ideal Day e Ponto de Função, sendo estas objeto de pesquisa deste trabalho.

2.3.1.1 Story Points

As Story Points (Pontos de História), expressam uma medida relativa do tamanho geral de uma história de usuário por meio de unidade de medida padrão (COHN, 2006).

As histórias de usuário são obtidas por membros da equipe de desenvolvimento, transformando assim a visão de negócio em uma visão um pouco mais tecnológica.

Para isso, o dono do produto deve transmitir os seus requisitos do software para a equipe de desenvolvimento. Por sua vez, a equipe discute o trabalho envolvido para alcançar os requisitos pretendidos e expostos pelo cliente. Para atingir um entendimento comum, os membros da equipe estimam em particular o esforço necessário, e expõe para todos da equipe. Quando as estimativas estiverem muito divergente, os membros discutem até chegarem em um denominador comum (MAHNIč; HOVELJA, 2012; HAUGEN, 2006).

Logo, o Story Points é uma medida relativa, onde o time decide o valor real de um ponto, e com base nisso, estima e define quantos pontos cada requisito a ser desenvolvido possui (HAMOUDA, 2014).

Duas formas principais são utilizadas para estimar Story Points, sendo a primeira identificando o menor requisito do backlog, atribuindo a ele o valor de um ponto, e respectivamente presumindo o resto dos requisitos com base nessa medida. E a segunda, com a utilização da técnica de Planning Poker.

Os Planejamentos com estimativas mais eficientes contam com uso da relatividade, mantendo sempre a integridade de que uma história estimada com dois pontos demande o dobro de esforço em relação a uma história estimada com somente um ponto (COHN, 2006).

2.3.1.1.1 Planning Poker

O Planning Poker é uma técnica de estimativa de pontos para histórias de usuário, que se baseia no princípio de uma cabeça pensando sozinha gera resultados viciados, portanto, tem como objetivo aumentar a confiabilidade das estimativas, com base na opinião da equipe de desenvolvimento do projeto acerca do que será produzido (COHN, 2006).

A técnica Planning Poker não foi criada a partir de algo totalmente novo, ela é uma variação da técnica de Delphi, planejada pela corporação RAND na década de 1950, objetivando estimular e refinar a perspectiva de uma equipe acerca de algo. Para isso a técnica Delphi definiu **três princípios** básicos:

1. **Respostas anônimas:** Onde os participantes teriam o direito de opinar de forma anônima.
2. **Controle da Iteração e feedback:** Controle de rodadas de feedback por meio de ciclos e de um moderador.
3. **Estatísticas:** Estatística das respostas do grupo.

Portanto, todos os participantes das rodadas de Planning Poker devem expor suas opiniões, sendo eles de cargo ou especialidade mais relevante ou não.

Para (HAUGEN, 2006), rodadas de estimativas que não tenham essa estruturação acabam estimulando os participantes a desistirem, por exemplo, se percebem que não possuem as habilidades suficientes para se pronunciar acerca de um tema. Um estudo sobre estimativas especializadas em desenvolvimento web demonstrou, que pessoas com maior conhecimento nem sempre têm as melhores respostas, já que as pessoas com menos conhecimento empregam diversas formas diferentes na resolução daquele requisito.

O Planning Poker, foi definida por James Grenning em 2002, e difundida por Mike Cohn. Fundamenta-se no alcance de medidas de estimativa por meio de um jogo de cartas. O princípio básico é permitir que todos os membros da equipe de desenvolvimento participem agregando sua visão de complexidade sobre o software desenvolvido, permitindo a aproximação de um denominador comum para o conceito de esforço (HAUGEN, 2006).

Para isso, cada integrante possui um baralho de doze cartas semelhante ao da Figura 12, de forma que as cartas estejam numeradas de acordo com a sequência números de Fibonacci.

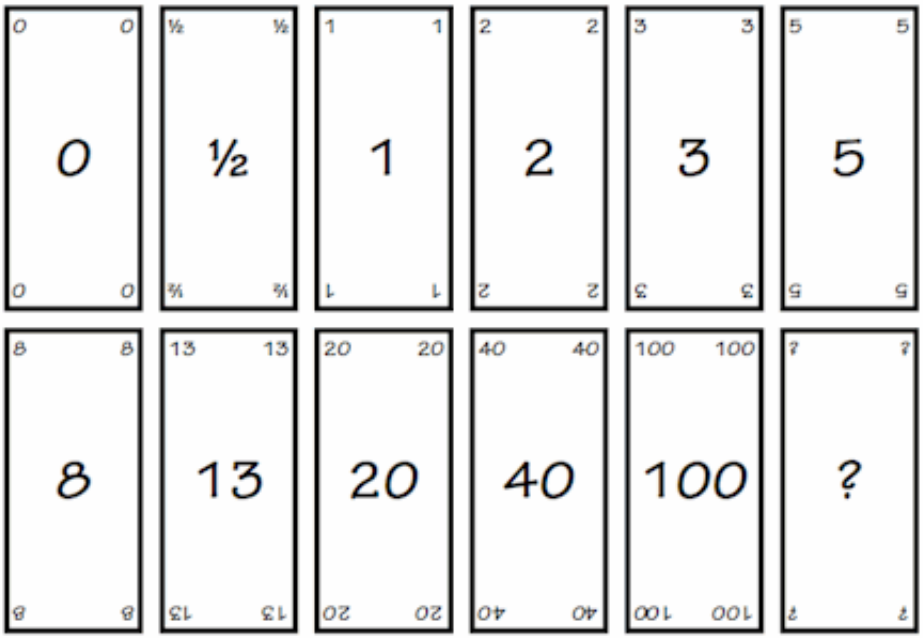


Figura 12 – Baralho Planning Poker com base na sequência de Fibonacci. Fonte: (RITTER, 2016).

A utilização da sequência de fibonacci não vem do acaso. O motivo de utilizar esses números é que o intervalo entre os valores permite visualizar melhor as desigualdades de complexidade entre as funcionalidades. Então, é possível diminuir a granularidade das histórias de usuário, evitando uma falsa ideia de precisão nas estimativas (COHN, 2006).

(COHN, 2006), define os **sete passos** dos procedimentos para o uso da técnica:

1. **Deve-se pontuar a história de usuário** de menor valor com uma pontuação pequena, isto é, a história que apresenta um menor custo de esforço para a execução pela equipe.

2. Todas as **histórias devem apresentadas** uma a uma pelo dono do produto, oferecendo informações mais detalhadas que agreguem o valor de negócio da história de usuário.

3. Os membros da equipe deve **pensar de forma individual no tempo e esforço necessário para implementar as histórias apresentadas**. Cada um deve transmitir seu pensamento definindo o esforço necessário para finalizar a tarefa com base nas cartas do baralho. Aspectos como desenvolver, testar e criar design devem ser considerados na decisão. Por fim, guarde a sua estimativa para você mesmo.

4. Se todos os membros já realizaram as estimativas individuais **todos da equipe devem revelar as estimativas** de forma simultânea.

5. Com todos as estimativas expostas, **a equipe deve avaliar se houveram convergências** entre todas ou parte delas.

6. Caso contrário, **a menor e maior estimativa divergentes devem ser discutidas** entre os nomeadores, de forma que toda a equipe reflita os motivos. Assim, uma nova rodada é realizada, seguindo nesse ciclo até que as estimativas de esforço sejam convergidas.

7. Por fim, a **estimativa escolhida para a história de usuário deve ser a que tiver maior ocorrência** ou a média entre as estimativas informadas. E assim seguem para a próxima história repetindo o processo até terminar as estimativas de todas as histórias de usuário do Backlog do produto.

Para (HAUGEN, 2006), a aplicação da técnica de Planning Poker para estimar Story Points, demonstra uma melhora significativa na eficiência das estimativas com pontos.

2.3.1.2 Ideal Day

O Ideal Day é uma técnica de estimativas ágeis, empírica utilizada por especialistas para desenvolvimentos que apoiam em exploração adaptativa, utilizada no planejamento de projetos e iterações (GAMBA; BARBOSA, 2010). Segundo (NGUYEN-CONG; TRAN-CAO, 2013), o Ideal Day é uma técnicas muito utilizadas por especialistas, de acordo com seus estudos de técnicas de estimativas.

Um Ideal Day (Dia Ideal) representa a quantidade de trabalho que um profissional de nível sênior, munido de ferramentas e tecnologias, desenvolve em oito horas de trabalho dedicado (sem interrupções) (COHN, 2006; DRAGICEVIC; CELAR; TURIC, 2017).

É importante a compreensão que um Ideal Day (Dia Ideal) é apenas uma forma de consolidar uma padrão de medida, que provavelmente não será alcançado pela equipe. Logo, essa medida deve servir apenas como um padrão, moeda estável, para quantificar o tamanho de referência e balizar a produtividade. Assim, uma tarefa estimada com o uso de Ideal Day, que requer oito horas ideais, pode levar dois ou três dias com as interrupções do cotidiano (DRAGICEVIC; CELAR; TURIC, 2017).

Para utilizar essa técnica é necessário manter as tarefas com uma granularidade menor, já que as tarefas devem caber dentro do espaço de execução de oito horas ideais. Se uma tarefa é muito grande para ser executada em oito horas ideais, é importante decompô-la em tarefas menores, se adequando a regra de desenvolvimento em oito horas ideais (COHN, 2006).

Conforme (MARTINS; LAUGENI, 2013), a seguinte fórmula deve ser utilizada para realizar os cálculos de estimativas em Ideal Days:

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

No qual:

- **DE:** Corresponde a quantidade de dias estimados para finalizar a tarefa;
- **IED:** Corresponde ao prazo, definido pela equipe, necessário para finalizar a tarefa;
- **IEDREAL%:** Corresponde ao percentual de dedicação de um desenvolvedor empreendido na finalização de uma tarefa designada.

Para manter a integridade e unidade da equipe, (ALVES; FONSECA, 2008) afirma, que não é necessário expor a equipe a quantidade de tempo que um integrante levou para executar suas atividades. Como é um dinâmica de grupo isso pode influenciar na produtividade. Se for necessário o time deve decidir pela utilização de técnicas que potencializam a equipe, como: pair-programming.

2.3.1.3 Ponto de Função COSMIC

O técnica de estimativa COSMIC, COSMIC Full Function Points (COSMIC – FFP), foi concebida em 1997 com o nome de Full Function Points v.1, a partir de adaptações da métrica FPA/IFPUG, para softwares real time. Em 1999, o grupo Common Software Measurement International Consortium – COSMIC propôs algumas mudanças na abordagem, tornando a métrica independente de outras métricas e alterando a sua abrangência operacional, passando a ser tanto para aplicações de negócio quanto para softwares em tempo real, assim tornou-se COSMIC – FFP – V.2.1 (Ponto de função Cosmic cheio – V.2.1) (CALAZANS, 2003; MONTEIRO, 2005).

O COSMIC foi criado com o objetivo de estimar ou medir o tamanho de um software com base em suas funcionalidades e processos funcionais. Assim, sua abrangência atinge diversos aspectos, tanto de requisitos de software embarcados como de softwares de tempo real, mas com independência de domínio e indicando medidas para diferentes pressupostos, sempre considerando a visão do usuário e do desenvolvedor (CALAZANS, 2003; MONTEIRO, 2005).

A (ISO/IEC..., 2011) segundo (COMMEYNE; ABRAN; DJOUAB, 2016), determina que o COSMIC é um padrão internacional para medir requisitos funcionais de software. O produto obtido com a sua aplicação é um valor numérico que representa o tamanho funcional do próprio software, podendo ser utilizado para comparação ou estimativas. Estando em conformidade com a ISO o COSMIC foi projetado para extrair dados de requisitos funcionais já desenvolvidos mas também dos requisitos do software antes mesmo de ser desenvolvido.

Um processo funcional é um componente básico de um conjunto de requisitos de funcionalidades do usuário, incluindo movimentos únicos dos dados, coesos e executados de forma independente (BERARDI; SANTILLO, 2010).

Conforme (COMMEYNE; ABRAN; DJOUAB, 2016), os fluxos de dados podem ser caracterizados por **quatro tipos distintos de movimento de um grupo de dados**:

- Os movimentos de **entrada (E - Entry)** e **saída (X - Exit)** de um grupo de dados, onde os dados são movimentados entre processo funcional através da fronteira para o processo funcional de requisição;
- Os movimentos de **leitura (R - Ready)** e de **escrita (W - Write)** de um grupo de dados, onde um processo funcional e um hardware de armazenamento persistente trocam dados.

Os movimentos de dados mencionados acima, são atribuídos a um tamanho padrão expresso em Pontos de Função COSMIC, de forma que um movimento de dados represente um PFC.

Para realizar o processo de medição com o COSMIC, (COMMEYNE; ABRAN; DJOUAB, 2016) indica **três fases**:

1. **A fase de estratégia de medição:** Consiste em especificar a finalidade e alcance da medição.
2. **A fase de mapeamento:** Consiste em identificar os processos funcionais do software, de acordo com os artefatos disponibilizados, atribuindo a pontuação aos requisitos de acordo com o PFC.
3. **A fase de medição:** Consiste em agregar as medidas individuais em uma medida de tamanho total PFC consolidado.

A utilização do COSMIC para as estimativas, gera algumas características peculiares ao produto final (CALAZANS, 2003):

- Aplicação da visão do usuário e do desenvolvedor em uma mesma medida;
- A detecção de camadas (Boa parte das tecnologias existentes trabalha com este paradigma);
- A flexibilidade (O COSMIC ajusta a pontuação dos movimentos de acordo com quantos movimentos são efetuados);
- Redução da ambiguidade, medidas mais simples reduzindo ambiguidade.

3 Metodologia de Desenvolvimento

Este Capítulo detalha o plano de coleta de estimativas. As Seções deste Capítulo caracterizam o objeto de estudo, explorando as fases de execução e os aspectos relacionados ao planejamento e configuração do estudo de caso.

3.1 Estudo de Caso

Para definir uma linha de processo para o estudo de caso este trabalho utilizou o trabalho de Runeson (2008) como referência, onde o processo de execução do estudo de caso conta com a realização de quatro etapa, sendo: Planejar e Desenhar, no qual o estudo é planejado e preparado, realizando a caracterização do objeto de pesquisa e do protocolo a ser seguido; Coletar Dados, que consiste em aplicar as técnicas do estudo de caso na coleta para obtenção de dados para análise; Analisar Dados, consiste em transformar os dados coletados em informações significativas. Relatar Estudo, onde se consolida todo o processo de estudo de caso, reflexão do valor dos dados coletados e sua influência no meio (RUNESON, 2008).

Para um melhor entendimento e visualização das etapas do processo, a Figura 13 retrata os quatro passos do processo de estudo de caso e suas atividades.

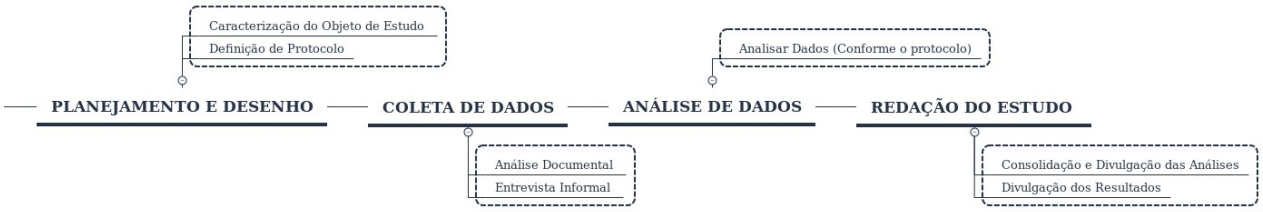


Figura 13 – Etapas de execução do estudo de caso. Fonte: Autor

3.1.0.1 Planejamento e Desenho

Essa fase contém diversos elementos cruciais, desde o planejamento até o desenho de como será executado. Aspectos como: o objeto e o contexto de estudo de caso, os objetivos a se alcançar com estudo, os métodos e protocolos estabelecidos, são explorados e exemplificados (RUNESON, 2008).

3.1.0.1.1 Caracterização do Objeto de Estudo

Este trabalho está inserido no contexto de uma organização da administração Pública Federal, denominada de Agência Espacial Brasileira (AEB). A AEB é uma au-

tarquia vinculada ao Ministério da Ciência, Tecnologia e Inovação (MCTI), e tem como responsabilidade formular, coordenar e executar a Política Espacial Brasileira. Assim, é responsável por estabelecer a Política Nacional de Desenvolvimento das Atividades Espaciais, definindo os objetivos e regulamentos para os programas e projetos nacionais da área espacial.

Para isso, conta uma grande força de colaboradores, sendo, quatro diretorias (DTEL, DSAD, DPEI e DPOA) e dez coordenadorias (CTE, CIN, CSA, CPDI, CAA, CPP, CPM, COF, CRH e CRL), a Figura 14 exemplifica toda a estrutura organizacional e as subordinações.

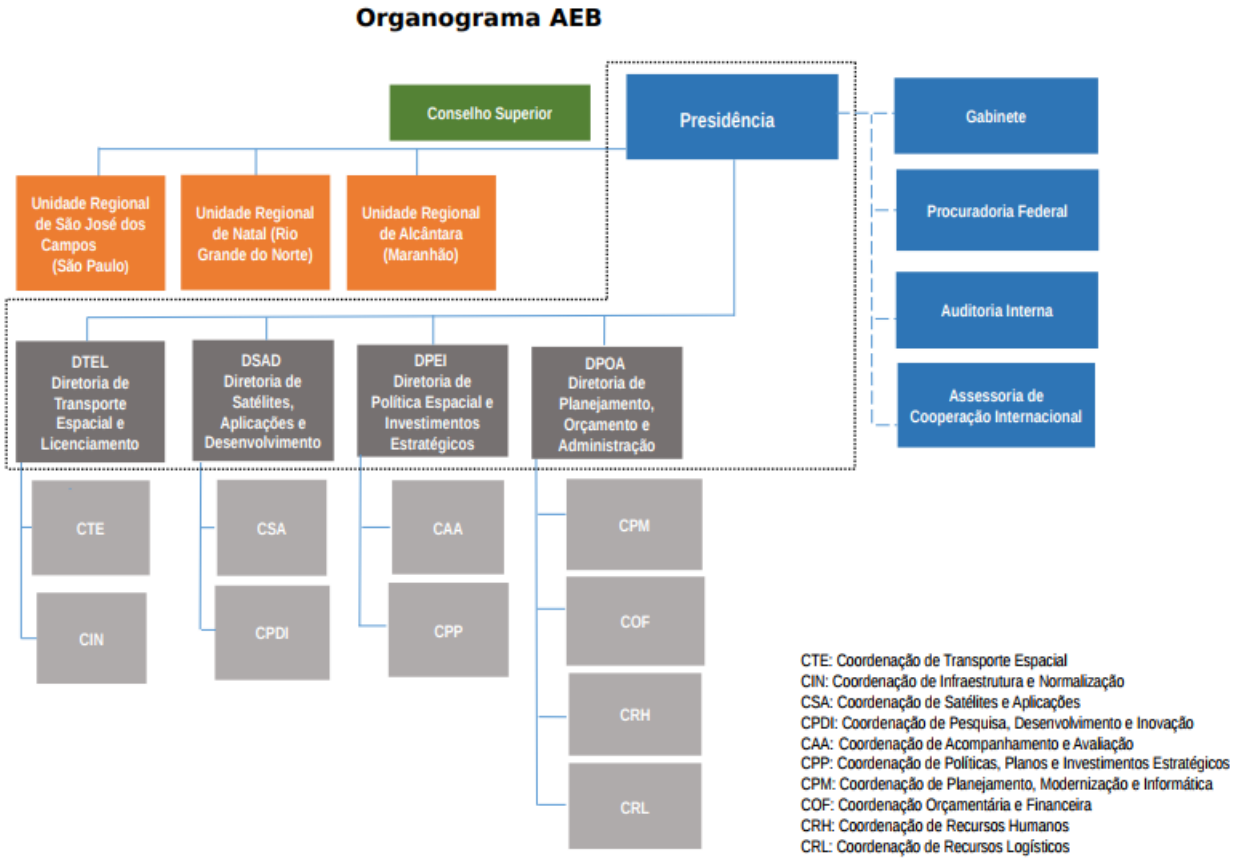


Figura 14 – Estrutura organizacional AEB. Fonte: (PRE-AEB, 2017).

Buscando estabelecer uma Administração Pública consolidada e com excelência na prestação de serviços aos cidadãos, a AEB possui uma divisão de tecnologia e informática (DINF), que tem por missão "administrar, desenvolver, propor e prover soluções e recursos de TI na organização, para que todos os setores consigam realizar suas atribuições de maneira eficiente e eficaz" (CGTI-AEB, 2016), respondendo de forma estratégica e inovadora às competências de desenvolvimento e manutenção de infraestrutura tecnológica necessárias.

A DINF realiza suas atribuições com base no processo de desenvolvimento de

Software da AEB que é parte da Política de Desenvolvimento, Manutenção e Aquisição de Software, objetivando gerenciar o ciclo de vida dos produtos de software da casa.

Estão no escopo deste processo a caracterização da demanda, que deve nascer na área de negócio, passar pela gestão estratégica da área e é encaminhada à DINF. A partir daí a sua viabilidade é avaliada, utilizando os critérios normativos existentes e aplicando critérios de priorização e balanceamento, tendo como premissa o atendimento aos objetivos estratégicos da instituição.

Uma vez aprovado, o projeto entra na etapa de análise técnica por parte da DINF, onde se analisam critérios tais como disponibilidade de soluções livres e públicas, recursos disponíveis para desenvolvimento e/ou aquisição de soluções de software. É nesse momento que ocorre a caracterização do tipo de projeto a ser executado: desenvolvimento/customização de solução de software ou aquisição de software proprietário.

As Figuras 15 e 16 demonstram a estrutura organizacional da DINF, que é vinculada à Coordenação de Planejamento e Modernização (CPM) e à Diretoria de Planejamento, Orçamento e Administração (DPOA).

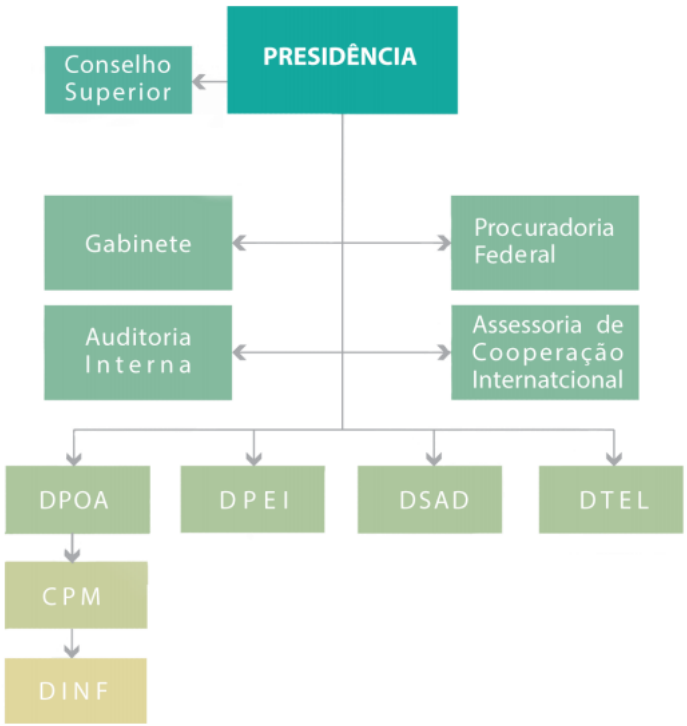


Figura 15 – Posicionamento da DINF na estrutura organizacional da AEB. Fonte: (CGTI-AEB, 2016).

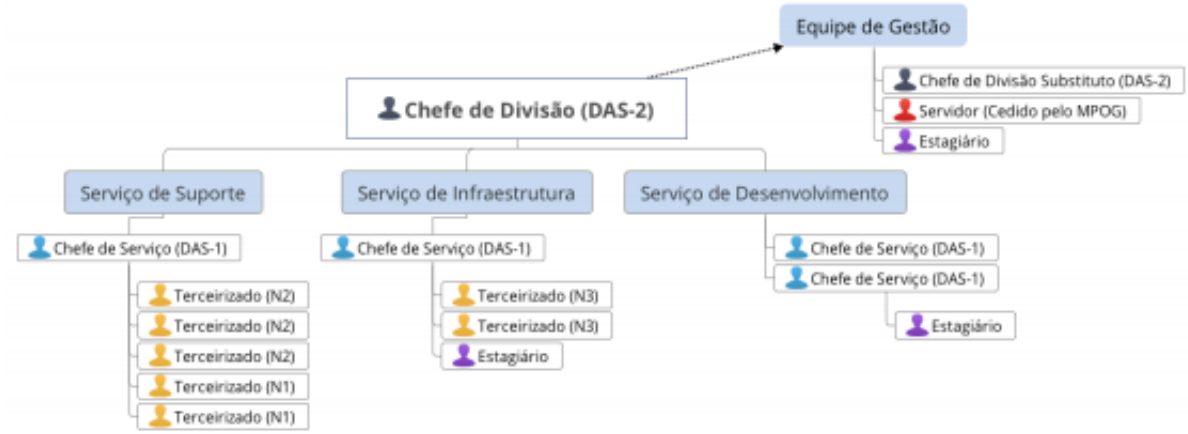


Figura 16 – Divisão da força de trabalho da DINF da AEB. Fonte: (CGTI-AEB, 2016).

A busca pela melhoria na qualidade e produtividade do desenvolvimento de software dentro das entidades provedoras, envolve basicamente três condições: processo, tecnologia e pessoas. A Tríade popularmente conhecida como triângulo da qualidade (PAULK et al., 1999).

Uma parte importante da engenharia de software é adaptar o processo. De acordo com (KRUCHTEN, 2000), "um processo não deveria ser seguido cegamente, gerando trabalho inútil e produzindo artefatos de pequeno valor adicionado". Portanto, um processo não deve ser um decreto rígido de como desenvolver um software. Ao contrário, é uma abordagem ajustável que deve estar em constante evolução e manutenção, se adequando às necessidades do operante, impactando na escolha do coleção apropriada de ações e tarefas.

Segundo a síntese dos relatórios e recomendações da Controladoria-Geral da União (CGU, 2012) à AEB , foram obtidas recomendações de definir e implantar uma metodologia sistemática e documentada de desenvolvimento de sistemas, bem como a inclusão de métodos e métricas para o monitoramento do progresso do desenvolvimento de softwares.

Segundo (PRESSMAN, 2006), para aumentar as chances de um projeto ser bem sucedido é preciso aplicar métricas para monitorar o desenvolvimento, ter estatísticas e comparações, já que não há forma de controlar aquilo que não se pode medir.

É possível obter indicadores com a utilização de estimativas, permitindo prever a quantidade de pessoas necessárias, o tempo necessário e o os custos para o desenvolvimento do projeto. Portanto, é relevante a busca de estabelecer e implantar um processo de estimativas (HAZAN, 2008a).

3.1.0.1.2 Fundamentação do Estudo de Caso

Como orientação do estudo de caso uma questão de pesquisa foi definida. Para isso, levou-se em conta a (QP.1) e os aspectos abordados no **Tópico 1.4.1** do **Capítulo 1.4**. Com isso, para que guie o objeto de estudo de caso uma questão de pesquisa foi levantada:

(QP.EC) No contexto de desenvolvimento ágil a ferramenta de apoio ao processo de estimativa de esforço em Story Points, melhorou e otimizou os resultados?

Para tal, foram definidas três hipóteses:

- **H1 -** Sim, a ferramenta apoiou o processo e aumentou consideravelmente a acurácia das estimativas.
- **H2 -** Sim, a ferramenta apoiou o processo porém não aumentou a acurácia das estimativas.
- **H3 -** Não, a ferramenta não apoiou o processo e não afetou positivamente a acurácia das estimativas.

A questão de pesquisa levantada, bem como, as hipóteses, devem ser verificadas por um protocolo básico apresentado na Figura 17.

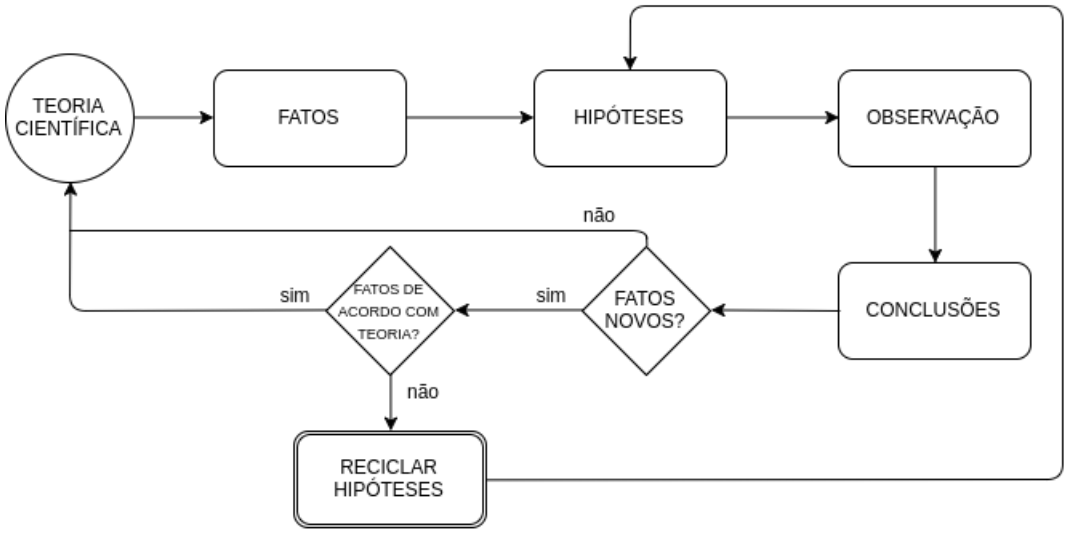


Figura 17 – Protocolo de validação das hipóteses. Fonte: Autor

3.1.0.2 Coleta de Dados

Utilizar várias fontes de dados em um estudo de caso é uma premissa importante para se evitar vícios de interpretação. Em um estudo de caso, também é importante

levar em consideração os pontos de vista de diferentes papéis e investigar as diferenças (RUNESON, 2008).

A realização da etapa de coleta de dados é feita a partir de **três níveis**:

1. **Método Diretos:** Este nível consiste no aplicação de contato direto entre o pesquisador e a pesquisa, possibilitando a coleta de dados em tempo real, a com a aplicação de entrevistas e observações.
2. **Métodos Indiretos:** Consiste na coleta de dados sem interferir nem influenciar a coleta interagindo com os sujeitos. Geralmente é executada com o auxílio de ferramentas da engenharia de software.
3. **Análise de Artefatos e de Trabalho:** Por fim, este nível é a execução da análise documental, on artefatos já concluídos são os insumos.

Neste trabalho serão utilizados apenas dois níveis, Métodos Diretos e Análise de Artefatos de Trabalho.

3.1.0.3 Análise de Dados

Está fase compreende realizar ações voltadas validar os dados coletados e extrair o maior número de informações deles. Por isso, está é uma fase de comparação dos dado extraídos com os dados da teoria apresentado neste trabalho. Essa comparação confirma a coerência dos dados, verificando a convergência entre a teoria e a prática.

Os dados obtidos neste trabalho terão natureza quantitativa, pois quantificam características das estimativas do processo de desenvolvimento de software da AEB, bem como, apontam com base nos dados quantificados critérios e definições que devem ser estabelecidos ou para a execução do planejamento de estimativa no processo.

3.1.0.4 Redação do Estudo

Na fase de redação do estudo, um compilado do estudo é gerado, isto é, os resultados são comunicados. Para consolidar os resultados um relatórios deve ser redigido, garantindo assim a sistematização e qualidade do estudo (RUNESON, 2008).

4 Análise de Dados

Nesta Seção são explorados os resultados das questões de pesquisa deste trabalho, obtidos com base na revisão literária. As questões de pesquisa encorajam indagações sobre o processo ágil de desenvolvimento de software e suas estimativas, onde são exploradas as diversas influências das técnicas de estimativa nas medições dos processos de desenvolvimento.

4.0.1 Análise Questão de Pesquisa 1 (QP1)

(QP.1) No contexto de desenvolvimento ágil quais são os aspectos que diminuem a acurácia das técnicas de estimativas de esforço?

Para (MOLOKKEN; JORGENSEN, 2003), um dos maiores desafios dos pesquisadores em responder essa questão está vinculado a necessidade de evitar que os entrevistados sejam tendenciosos. Isto é, as avaliações de precisão de estimativas devem ser desenvolvidas por revisores, e não por membros da equipe, evitando que tomem a atitude de culpar ou punir alguém da equipe por este meio.

(MOLOKKEN; JORGENSEN, 2003) durante as revisões de artigos publicados, segundo o estudo de Phan, encontraram que cerca de 51% dos entrevistados acreditam que realizar estimativas muito otimistas afetam diretamente a acurácia das estimativas, e cerca de 50% dos entrevistados veem mudanças de projeto ou de implementação como um fator determinante para o sucesso e qualidade da estimativa.

No estudo de Lederer e Prasad segundo (MOLOKKEN; JORGENSEN, 2003), foram avaliadas vinte e quatro razões para uma possível imprecisão das estimativas, sendo três as principais:

- Frequentes solicitações de mudanças por parte dos usuários;
- Falta de entendimento por parte dos usuários dos requisitos necessários;
- Tarefas negligenciadas pela equipe de desenvolvimento.

(JøRGENSEN, 2004a) descreve o processo de estimativa com duas visões, uma como um modelo "racional"e outra "político".

O modelo racional possui processo de estimativas semelhantes aos livros de texto sobre estimativas, tendo a precisão de estimativas como o principal foco. Já o político, descreve o processo de estimativas como um “Exaltador”, gerando objetivos individuais e conflitos e poder.

Portanto (JøRGENSEN, 2004a), define seis metas para reduzir os problemas com viés humano, equilibrando os modelos de estimativas e reduzindo a imprecisão dos resultados:

- Avalie a precisão da estimativa, mas evite avaliações sob pressão.
- Evite metas de indicadores de estimativas conflitantes.
- Peça aos estimadores para justificar e criticar suas estimativas.
- Evite informações de estimativas irrelevantes e pouco confiáveis.
- Use dados documentados de tarefas de desenvolvimento anteriores.
- Encontre especialistas em estimativas com antecedentes de domínio altamente relevantes e bons registros de estimativa.

Conforme os estudos de (SCHWEIGHOFER; KLINE; PAVLIC; HERICKO, 2016), os fatores de pessoal em relação aos fatores de processo dos projetos antecipam e premeditam o sucesso dos resultados. Tornando assim, o fator pessoal como o aspecto mais importante e demandante de observação nos projetos.

(SCHWEIGHOFER; KLINE; PAVLIC; HERICKO, 2016), relacionou e listou quatro aspectos mais importantes para conquistar uma boa acurácia das estimativas:

- Os fatores de pessoal devem antecipar-se aos fatores de processo, necessitando de uma dedicação e observação superior, em processos de estimativa de esforço ágil. Isso significa, que o conhecimento e a experiência da equipe, bem como, a composição e formação da mesma são cruciais para as estimativas;
- A comunicação deve ser mantida, reduzindo assim os buracos de falta de informação, e antecipando mudanças e tarefas durante todo o projeto;
- Manter o controle de tarefas do projeto, além de avaliar essas tarefas, é fundamental para o sucesso.
- Apresentar um feedback das estimativas para toda a equipe, agrega uma visão processual a todos da equipe, e são parte da obtenção de bons resultados de estimativas.

(TANVEER; GUZMáN; ENGEL, 2016), conduziram um estudo com questionários sobre estimativas, que foi divido em três partes:

1. Na primeira parte, os entrevistados foram questionados a partir de uma questão aberta, que os indagava sobre as considerações mais importantes de estimativas. Os

entrevistados, elegeram a experiência de implementação do desenvolvedor como o fator mais relevante. Seguido do conhecimento e capacidade de desenvolvimento.

2. Na segunda parte, os entrevistados avaliaram sete fatores com pontuações variando de um a cinco, sendo cinco muito relevante e um não relevante.

   A Tabela 3 traz os fatores definidos para avaliação pelos entrevistados.

Tabela 3 – Definição dos fatores para análise de impactos nas estimativas. Fonte: (TANVEER; GUZMáN; ENGEL, 2016).

Nome dos Fatores	**Definição dos Fatores**
Experiência do desenvolvedor com estimativas	A experiência do desenvolvedor com estimativas de tarefas de desenvolvimento.
Conhecimento dos itens do backlog pelo desenvolvedor	Refere-se ao conhecimento de um desenvolvedor sobre o sistema e/ou componente em que está trabalhando, assim como um determinado item do backlog a ser implementado.
Experiência do desenvolvedor com implementação	Refere-se à experiência de implementação passada e a familiaridade com o sistema/componente estimado de um desenvolvedor, bem como com o sistema/componente atual em que ele está trabalhando.
Dependências entre itens do backlog	Refere-se às dependências existentes entre os itens do backlog a serem implementados.
Complexidade dos itens do backlog	Refere-se à complexidade prática de implementalçao dos itens do backlog.
Analisar o impacto dos Itens do backlog no que já foi desenvolvido	Refere-se ao impacto da implementação de um novo item do backlog no sistema existente.
Considerar estimativas feitas para um item do backlog semelhante em sprints já realizadas	Obter o esforço real de implementação de um item do backlog com base em estimativas e desenvolvimentos de itens passados.

Esses itens foram, analisados com relação à relevância para os entrevistados, com pontuações variando de um a cinco, resultando na Tabela 4.

Tabela 4 – Relevância dos fatores na acurácia de estimativas. Fonte: (TANVEER; GUZMáN; ENGEL, 2016).

Nome dos Fatores	**Pontuação Mínima**	**Pontuação Máxima**	**Pontuação Média**
Experiência do desenvolvedor com estimativas	1	5	4
Conhecimento dos itens do backlog pelo desenvolvedor	1	5	5
Experiência do desenvolvedor com implementação	1	5	5
Dependências entre itens do backlog	2	5	4
Complexidade dos itens do backlog	3	5	5
Analisar o impacto dos Itens do backlog no que já foi desenvolvido	3	5	4
Considerar estimativas feitas para um item do backlog semelhante em sprints já realizadas	2	4	2

A Tabela 4, mostra a percepção dos entrevistados da relevância de alguns fatores.

Quase todos os fatores foram considerados relevantes, somente “Considerar estimativas feitas para um item do backlog semelhante em sprints já realizadas”, não revelou significativa importância na redução de erros de estimativas.

3. Na terceira parte, os fatores da Tabela 3 foram apresentados novamente na Tabela 5, porém agora sob a perspectiva do potencial que os fatores tem para melhorar a precisão da estimativa, a pontuação varia de um a cinco, onde um significa não relevante e cinco muito relevante.

Tabela 5 – Relevância dos fatores na melhora da precisão das estimativas. Fonte: (TANVEER; GUZMáN; ENGEL, 2016).

Nome dos Fatores	**Pontuação Mínima**	**Pontuação Máxima**	**Pontuação Média**
Experiência do desenvolvedor com estimativas	2	5	4
Conhecimento dos itens do backlog pelo desenvolvedor	4	5	5
Experiência do desenvolvedor com implementação	3	5	4
Dependências entre itens do backlog	2	5	4
Complexidade dos itens do backlog	1	5	4
Analisar o impacto dos Itens do backlog no que já foi desenvolvido	2	5	4
Considerar estimativas feitas para um item do backlog semelhante em sprints já realizadas	2	5	3

Quase todos os fatores foram classificados como tendo um alto potencial para melhorar a precisão da estimativa. No entanto, as equipes aparentemente não tinham certeza de que "Considerar estimativas feitas para um item do backlog semelhante em sprints já realizadas" tem qualquer efeito na melhoria da precisão da estimativa.

(USMAN; BRITTO, 2016) conduziram um estudo, que chegou a definição de **oito categorias de fatores mais relevantes na influência da acurácia dos resultados** das estimativas.

- **Questões relacionadas aos requisitos:** A falta e/ou dificuldade em definir um requisito e as suas mudanças, são citadas como a principal razão de estimativas imprecisas;
- **Questões relacionadas a equipe:** Membros inexperientes e/ou não qualificados, com baixa habilidade, são responsáveis pela imprecisão de estimativas;
- **Questões relacionadas a barreiras globais:** No desenvolvimento descentralizado, fuso horário, distância geográfica e cultura, são influenciadores da acurácia dos resultados de estimativas;

- **Questões relacionadas ao gerenciamento de projetos e controle de mudanças:** Gerenciamento incorreto de processos de desenvolvimento aumentam o tempo, custo e os riscos, assim, aumentam as chances de imprecisão das estimativas;
- **Questões de excesso de otimismo:** Considerar apenas o melhor cenário, ou subestimar propositalmente um projeto para ganhar competitividade nos contratos, são fatores que além de violar o código de ética de engenheiros, aumentam muito a chance de insucesso de projetos além da baixa acurácia das estimativas;
- **Questões relacionadas à falta de um processo formal:** A indecisão sobre um processo formal de projeto, corrobora para a inexperiência da equipe nos projetos, e aumenta a imprecisão das estimativas;
- **Questões relacionadas a testes:** Ignorar esforços nas obtenção de testes levam a subestimação do projeto, aumentando a necessidade de tempo do ciclo de manutenção e evolução de software.
- **Questões relacionadas à falta de envolvimento do cliente:** A falta de presença do cliente no dimensionamento de histórias e visão do produto, bem como, a falta de colaboração do cliente, estão diretamente relacionados a baixa acurácia de estimativas.

Um estudo com um total de 28 entrevistados, respondendo a questões abertas, conduzido por (USMAN; MENDES; BöRSTLER, 2015), para discutir uma série de possíveis razões para as imprecisões das estimativas, resultaram na organizaram dos fatores mais influentes em sete categorias, que foram resumidas na Tabela 6:

Tabela 6 – Razões para imprecisão de estimativas em projetos ágeis. Fonte: (USMAN; MENDES; BöRSTLER, 2015).

Categorias de motivos de imprecisão de estimativas	**Indicações (% de indicação por participantes)**
Problemas relacionados aos requisitos (Falta e/ou alteração nos requisitos)	9 (32%)
Problemas de gerenciamento de projetos (Falta de controle, equipe sem orientação e escopo mal desenhado)	7 (25%)
Problemas relacionados à equipe (Inexperiência, falta de compartilhamento de conhecimento na equipe e membros desqualificados)	5 (18%)
Superotimismo (Planejando sempre o melhor cenário)	3 (10%)
Falta de processo formalizado	3 (10%)
Ignorar esforços em testes	2 (7%)
Envolvimento insuficiente do cliente no dimensionamento das histórias	2 (7%)

4.1 Análise Questão de Pesquisa 2 (QP2)

(QP.2) Quais das técnicas de estimativas no contexto de desenvolvimento ágil são mais utilizadas: Story Point, Ideal Day ou Ponto de função COSMIC?

A técnica Story Points é a técnica mais utilizada segundo os estudos de (NGUYEN-CONG; TRAN-CAO, 2013). Nos estudos, ela aparece com 34.38% de indicações de uso, logo é a mais utilizada. Em seguida a técnica Ideal Day e PF COSMIC com a mesma "%" de indicação com 6,25% de indicações nos estudos.

Nos estudo de (USMAN; MENDES; BöRSTLER, 2015), o Story Points também é a técnica mais frequente e utilizada de acordo com suas análises. O Story Points apresenta-se em 61,67% dos estudos encontrados. Em uma segunda posição o Ponto de Função é bem expressivo, com 28.33% de indicações em seus estudos. Por fim, a técnica Ideal Day, não é mencionada neste estudo, logo, é considerada como a técnica menos expressiva.

Considerando que diversos estudos avaliam as frequências de indicações de uso de técnicas de estimativa nos processos de desenvolvimento, mas nem todos apresentam o mesmo tipo de avaliação e comparação de técnicas, foram mapeadas a partir de nove estudos, sete técnicas de estimativas mais expressivas no uso, e avaliadas de acordo com a frequência de indicação de uso nos estudos, como pode ser visto na Tabela 7.

Tabela 7 – Indicações de técnicas de estimativas ágeis. Fonte: Autor.

Técnica	**Frequência(%)**	**Referência**
Story Point	88%	(NGUYEN-CONG; TRAN-CAO, 2013), (SCHWEIGHOFER; KLINE; PAVLIC; HERICKO, 2016), (USMAN; MENDES; BöRSTLER, 2015), (USMAN; BöRSTLER; PETERSEN, 2017), (USMAN; MENDES; WEIDT; BRITTO, 2014), (TANVEER; GUZMáN; ENGEL, 2016), (POPLI; CHAUHAN, 2014), (GAMBA; BARBOSA, 2010)
Ideal Day	22%	(NGUYEN-CONG; TRAN-CAO, 2013), (GAMBA; BARBOSA, 2010)
Pontos de Função	55%	(IDRI; HOSNI; ABRAN, 2016), (NGUYEN-CONG; TRAN-CAO, 2013), (USMAN; MENDES; WEIDT; BRITTO, 2014), (GAMBA; BARBOSA, 2010), (USMAN; MENDES; BöRSTLER, 2015)
Pontos de Caso de Uso	44%	(NGUYEN-CONG; TRAN-CAO, 2013), (USMAN; BöRSTLER; PETERSEN, 2017), (USMAN; MENDES; BöRSTLER, 2015), (USMAN; MENDES; WEIDT; BRITTO, 2014)
Pontos de Função COSMIC	22%	(NGUYEN-CONG; TRAN-CAO, 2013), (USMAN; MENDES; BöRSTLER, 2015)
Ideal Hour	11%	(NGUYEN-CONG; TRAN-CAO, 2013)
Julgamento de Especialista	55%	(SCHWEIGHOFER; KLINE; PAVLIC; HERICKO, 2016), (USMAN; MENDES; BöRSTLER, 2015), (POPLI; CHAUHAN, 2014), (IDRI; HOSNI; ABRAN, 2016), (USMAN; BöRSTLER; PETERSEN, 2017)

Portanto, a técnica de Story Points novamente foi a técnica mais indicada em frequência de uso em projetos de desenvolvimento. O Story Points obteve 88% de frequência de indicação nos estudos avaliados.

A técnica Ponto de função quando avaliada sem modificações para o modelo COSMIC, assume o segundo lugar com 55% de indicações de uso em projetos de desenvolvimento. Quando assume a variação do modelo tradicional e atua com características do modelo Ponto de Função COSMIC, fica com a mesma "%" de indicação da técnica Ideal Day, de 22% de frequência.

Apesar de não ser foco de discussão dessa questão de pesquisa, a técnica Pontos de Caso de Uso apresentou relevância significativa na indicação de frequências de uso, com 44% de indicações nos estudos.

4.2 Análise Questão de Pesquisa 3 (QP3)

(QP.3) No contexto de desenvolvimento ágil entre as estimativas Story Point, Ideal Day e Ponto de função COSMIC qual possui uma maior acurácia?

No estudo de (GAMBA; BARBOSA, 2010), o Ideal Day, foi o que mais se aproximou a estimativa do realizado, logo, é a técnica mais acurada. Não muito atrás, com uma diferença mínima, a técnica Story Points ficou em segundo lugar no aspecto de acurácia. Por fim, com um resultado muito distante e abaixo das outras técnicas, o Ponto de Função aparece em terceiro lugar, quando comparado com as outras duas técnicas.

Para (GAMBA; BARBOSA, 2010), o Ideal Day se destaca pela facilidade em estimar os prazos em dias empreendidos para finalizar um requisito. O Story Point se aproximou muito do Ideal Day, e isso deve ao fato de que são técnicas empíricas.

O resultado tão divergente da técnica Ponto de Função, surge da falta de análise da complexidade da iteração do usuário com o sistema em desenvolvimento (GAMBA; BARBOSA, 2010).

De acordo com (UNGAN; CIZMELI; DEMIRORS, 2014), quando compara as técnicas de Ponto de Função e Story Points, a técnica mais acurada é o Story Points. Essa conclusão é obtida com base no MMRE, onde o Story Points apresenta 20,15% e o Ponto de Função COSMIC 22,48%.

No estudo de (COMMEYNE; ABRAN; DJOUAB, 2016), o um erro médio das técnicas PFC e Story Points, são respectivamente de 28% e 58%.

Na pesquisa de (USMAN; MENDES; WEIDT; BRITTO, 2014), o Story Point com o uso do Planning Poker, apresenta 48% de acurácia quando analisado sob a perspectiva de uma MMRE.

5 Desenvolvimento

Tendo em vista que, o Story Points e respectivamente o Planning Poker, identificados na (QP.2) e (QP.3), são o conjunto de técnicas e ferramentas mais utilizado e acurado para estimativa de esforço em processo ágil de desenvolvimento de software, e levando em conta o mapeamento de diversos aspectos responsáveis pela baixa acurácia das estimativas por meio da (QP.1), **uma ferramenta de suporte foi planejada para otimizar e apoiar o processo de desenvolvimento de software**, bem como, aumentar a acurácia das estimativas de esforço.

5.1 Visão do Software

O modelo de planejamento para o desenvolvimento do software da ferramenta é o modelo ágil, acrescido de outras duas documentações que se mostraram de grande importância, o “Modelo de Domínio” e o “Diagrama de Casos de Uso”, que vão apoiar o entendimento e o desenvolvimento prático do software.

Assim, o planejamento foi dividido em duas seções, sendo a primeira na "Visão do Software", contendo o "Contexto", "Problema", “modelo de domínio” e o "Diagrama de Casos de Uso", e a segunda seção onde o modelo conceitual é gerado a partir do “protótipo de baixa” e "Backlog"onde toda a rastreabilidade das histórias de usuário é demonstrada e consolidada.

5.1.1 Contexto

A AEB, que é objeto de estudo, possui um processo de desenvolvimento de software oficial baseado no processo ágil XP (Extreme Programming), acrescido de algumas tarefas de cunho burocráticas, inerentes ao processo de negócios específico dos serviços públicos.O PSW-AEB é embasado em diversas referências, entre elas a norma ABNT NBR ISO/IEC 12207:2009, a Instrução Normativa 04/2014 STI/MPOG, o Guia de Processos de Software para o SISP versão 1.0, o Guia de Projetos de Software com Prática de Métodos Ágeis para o SISP versão 1.0 e a Metodologia de Gerenciamento de Projetos para o SISP versão 1.0. Além disso, são levados em consideração o Decreto No. 8.135 de 4 de Novembro de 2013 no que tange à comunicação de dados na administração pública e o acórdão 1233/2012 do TCU referente à Governança de TI na administração pública.

Assim, PSW-AEB como apresentado na Figura 18 possui como seus macroprocessos a divisão de três fases que abrangem todo o ciclo de vida dos produtos de software do órgão: Planejamento, Execução e Manutenção.

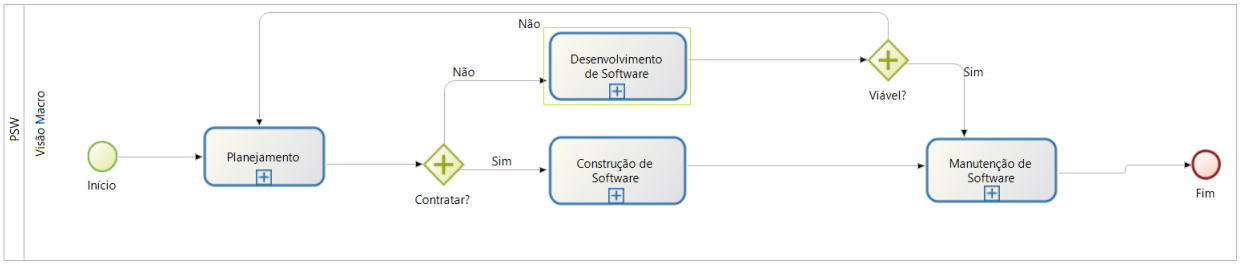


Figura 18 – Macroprocessos do processo de desenvolvimento de sofware da AEB. Fonte: AEB com adaptações do Autor.

1. A etapa de Planejamento, que é a etapa principal de atuação da ferramenta proposta neste estudo, abrange desde a origem da demanda de software, que pode ser iniciada tanto na área-meio quanto na própria DINF.Portanto, é papel da DINF realizar a análise de viabilidade da demanda levando em conta o que foi planejado no PDTI.

2. A etapa de Execução é responsável por colocar em prática as decisões da fase de planejamento, isto é, após uma decisão de direcionar o desenvolvimento para a DINF ou de contratação de uma empresa terceirizada, acompanhar e desenvolver o software proposto dentro de qualificações especificadas anteriormente, como: preço, prazo e qualidade.

3. Por último a fase de manutenção é responsável por iniciar o ciclo PDCA, isto é o ciclo de melhoria de produto, que abrange desde a atuação de levantamento de novas necessidades a correções de bugs e melhoria da qualidade do software.

Por sua vez, esses macroprocessos são compostos por outras diversas tarefas, mas o macroprocesso escolhido para a atuação da ferramenta proposta é o de "Desenvolvimento de Software", que é composto por outros quatro macroprocessos, mas a ferramenta foi direcionada para atuação durante apenas dois deles o "Período Pré-jogo"e "Período Construção", como mostra a Figura 19.

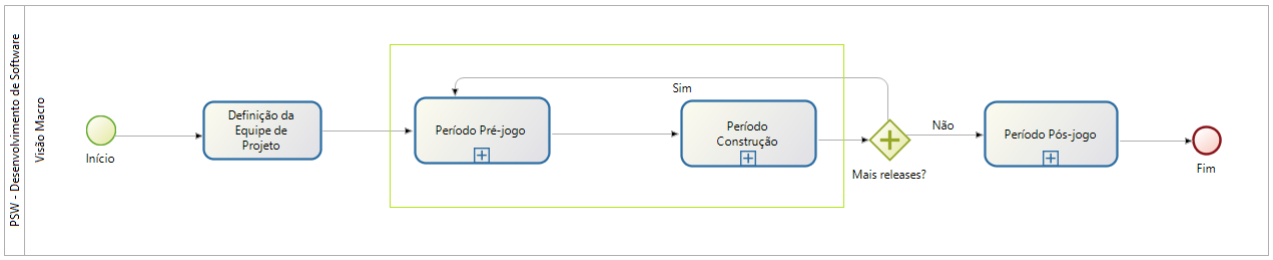


Figura 19 – Macroprocesso "Desenvolvimento de Software"do processo de desenvolvimento de sofware da AEB. Fonte: AEB com adaptações do Autor.

No estágio de "Período Pré-jogo"as atividades relacionadas ao levantamento de requisitos do sistema são realizadas. Também está no escopo desta etapa a elaboração do

cronograma do Projeto e identificação dos atores envolvidos no Projeto.

Já no estágio de "Período Construção" acontecem as atividades relativas ao desenvolvimento prático, isto é, com o ciclo de interações constante entre todos os envolvidos no processo de desenvolvimento de software. A execução do projeto necessita do levantamento das listas de histórias de usuário e do planejamento das releases.

A Figura 20, detalha mais o processo e respectivamente as tarefas executadas para desenvolver os produtos de software da AEB. A ferramenta proposta neste estudo deve atuar de maneira prática durante estas atividades de forma que o processo fique mais otimizado e consequentemente haja uma melhora nas estimativas de esforço de desenvolvimento.

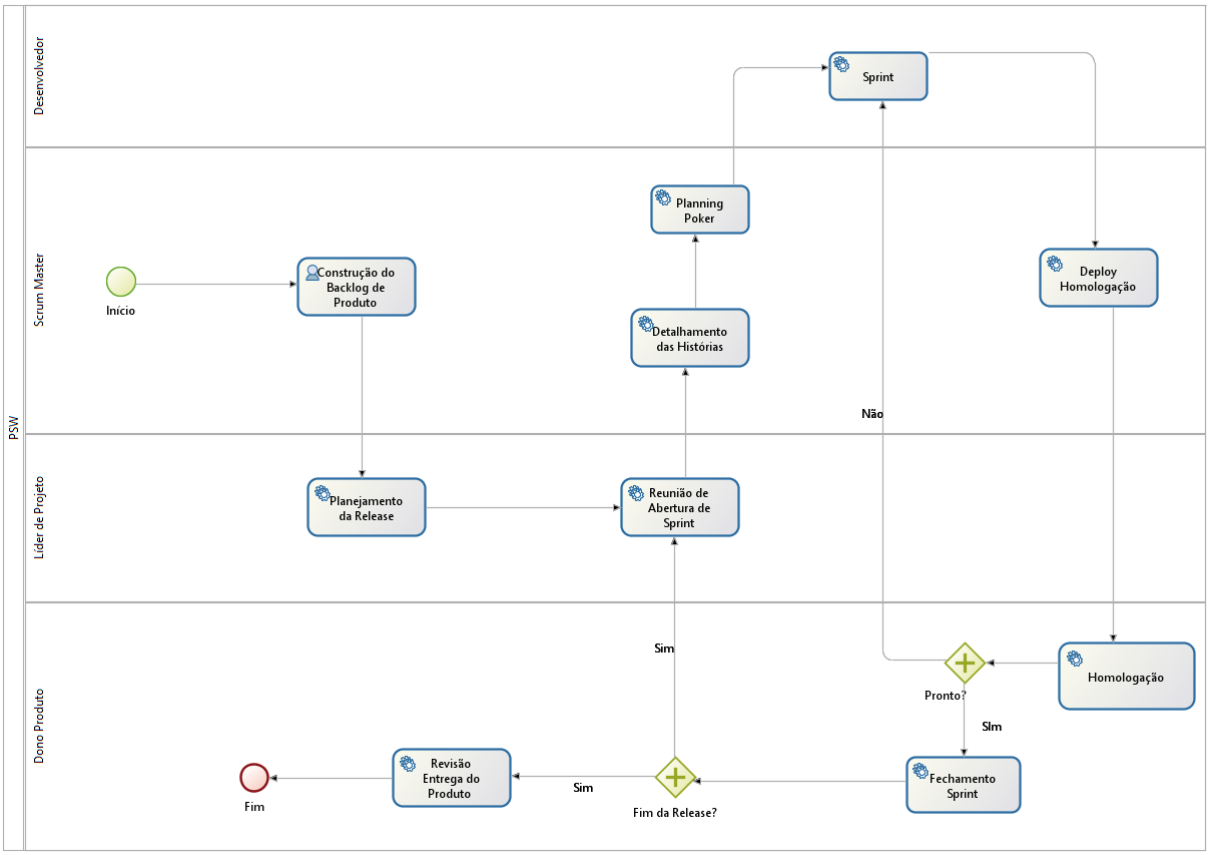


Figura 20 – Macroprocesso "Período Construção"do processo de desenvolvimento de sofware da AEB. Fonte: AEB com adaptações do Autor.

5.1.2 Problema

Então, com o levantamento dos aspectos que afetam a acurácia das estimativas de esforço na (QP.1) e com parte do processo de desenvolvimento de software ágil da AEB, foram definidos quatro aspectos como os mais recorrentes e indicados como precursores do insucesso do planejamento das estimativas de software.

Assim, eles foram formulados e explanados de forma que a ferramenta proposta neste estudo tente sanar ou minimizar esses aspectos que possam vir a ser problemáticos no planejamento do esforço e tempo empregado no desenvolvimento de um software.

As tabelas a seguir foram desenvolvidas visando uma maior compreensão dos aspectos mais recorrentes e que foram definidos para atuação com a ferramenta proposta:

Tabela 8 – Aspecto 1 extraído da (QP.1).

Problema	Falta de comunicação entre cliente e desenvolvedores
Afeta	O resultado de produto final
Cujo impacto é	Insatisfação do cliente e atrasos no projeto
Uma solução bem sucedida seria	Informatizar a comunicação entre a equipe de desenvolvedores e o cliente, aumentando a necessidade da presença do cliente para a definição do que deve ser desenvolvido.

Tabela 9 – Aspecto 2 extraído da (QP.1).

Problema	Cliente não saber ou não entender o que vai ser desenvolvido
Afeta	O resultado de produto final
Cujo impacto é	Insatisfação do cliente e atrasos no projeto
Uma solução bem sucedida seria	Aumentar visibilidade e o acesso do que vai ser desenvolvido melhorando a percepção do cliente sobre o produto pela fase de "metáfora do produto" em que um protótipo é gerado

Tabela 10 – Aspecto 3 extraído da (QP.1).

Problema	Falta de acesso a dados de projetos legado, e de planejamentos passados
Afeta	Qualidade do planejamento
Cujo impacto é	Planejar o desenvolvimento sem se espelhar e provocar superestimação ou subestimação do projeto
Uma solução bem sucedida seria	Guardar planejamentos anteriores e facilitar o acesso a esses planejamentos

Tabela 11 – Aspecto 4 extraído da (QP.1).

Problema	Falta de métodos de visualização da situação do projeto pela equipe de desenvolvedores e cliente
Afeta	O micro planejamento do projeto
Cujo impacto é	Insatisfação do cliente
Uma solução bem sucedida seria	Centralizar informações de gerenciamento do projeto, de forma que o planejamento e a execução possam ser visualizados e analisados em um mesmo espaço

Portanto, com os aspectos recorrentes definidos como necessidades de melhorias para aumentar a chance de sucesso e a qualidade das estimativas em processos ágeis de desenvolvimentos de software, a ferramenta planejada neste estudo pretende otimizar a forma de atuação do cliente, dos desenvolvedores e do gestor no projeto e no processo de desenvolvimento.

5.1.2.1 Modelo de Domínio

A comunidade de desenvolvedores vê o entendimento do domínio em que o software é inserido como uma ação desafiadora e complexa. Por isso, a modelagem de domínio, que é uma das práticas incentivadas pelo Domain-Driven Design, ajuda a contornar essa complexidade e aproximar os entendimentos (PERILLO, 2010).

Em resumo o modelo de domínio é definido como um sistema de abstrações que descreve aspectos selecionados de um domínio contextual e pode ser empregado para resolver divergências relacionados a esse domínio (PERILLO, 2010).

Portanto, para (PERILLO, 2010), "o modelo de domínio pode ser definido como uma abstração rigorosa e selecionada do conhecimento que os especialistas no domínio possuem".

Na Figura 21, estão os resultados da modelagem de domínio da ferramenta proposta neste trabalho para apoiar o desenvolvimento ágil de software.

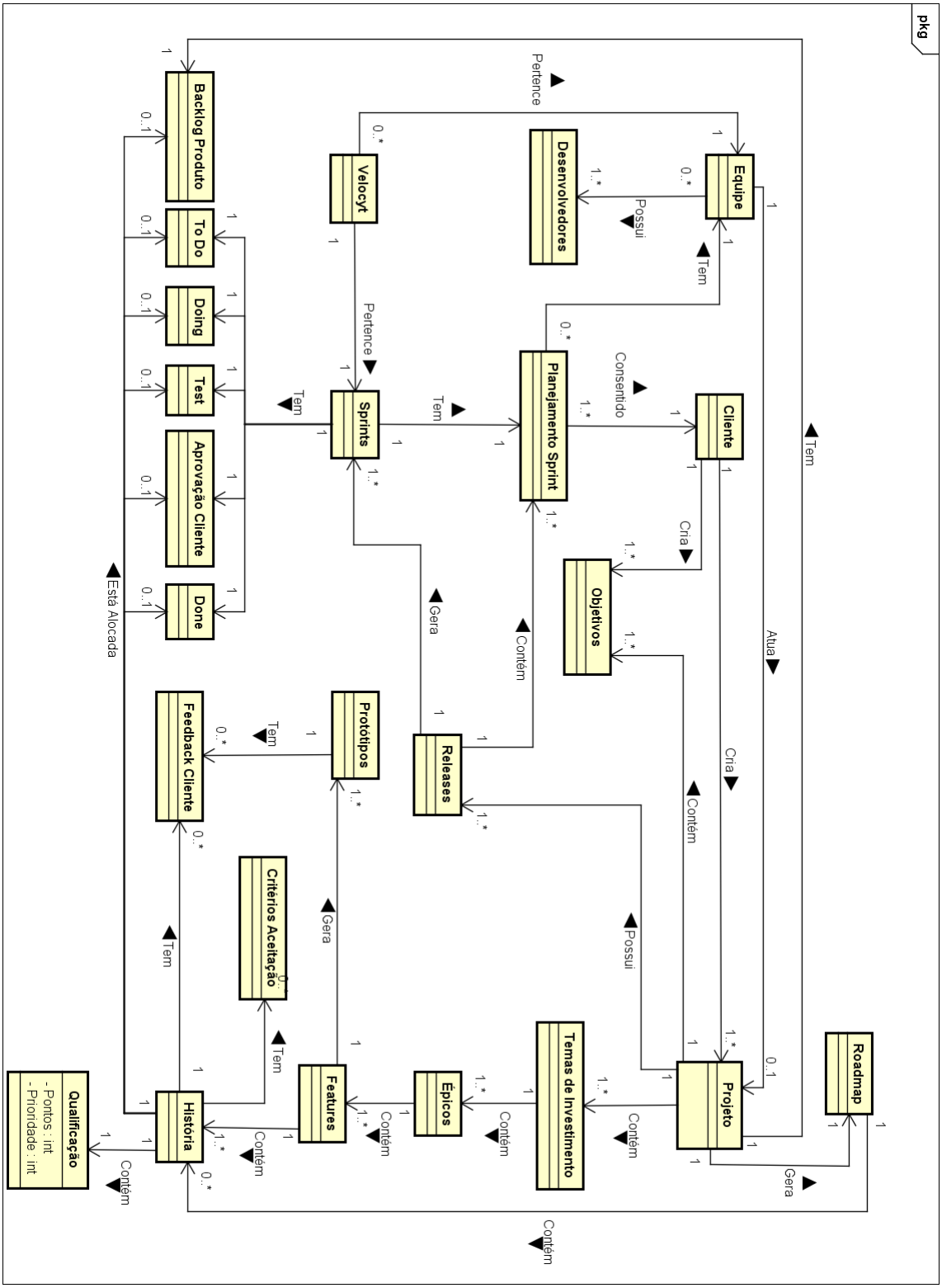


Figura 21 – Modelo de domínio da ferramenta proposta. Fonte: Autor.

5.1.2.2 Diagrama de Casos de Uso

O Diagrama de casos de uso oferece um modo de visão externa do sistema e de suas interações com o mundo exterior. Assim, podem representar uma visão de alto

nível de funcionalidades intencionais mediante requisições feitas pelos usuários do sistema (STADZISZ, 2002).

A Figura 22, demonstra essa interação do usuário na ferramenta proposta neste estudo.

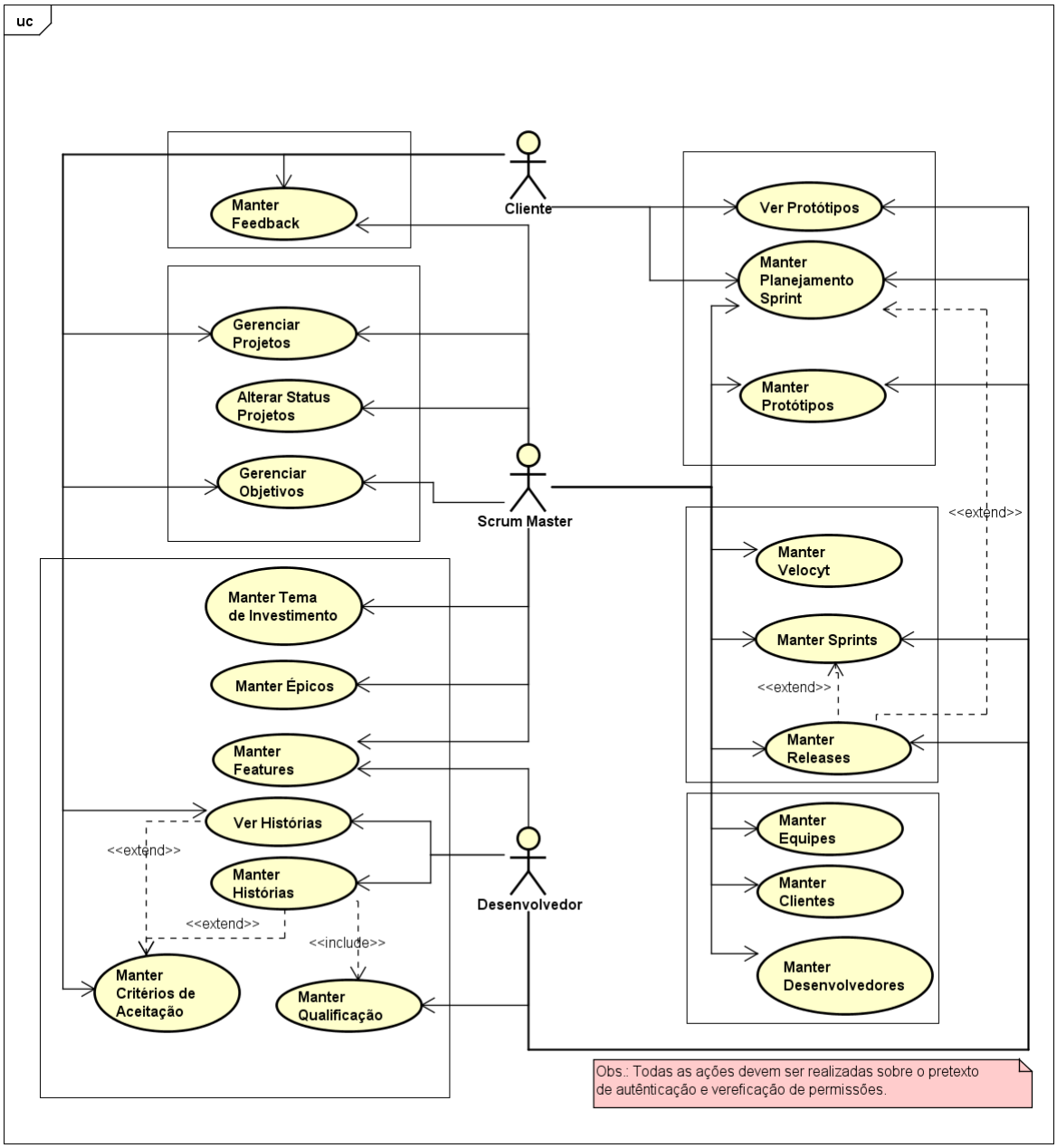


Figura 22 – Diagrama de Casos de Uso da ferramenta proposta. Fonte: Autor.

5.2 Modelo Conceitual

Já com a visão da ferramenta estabelecida na Seção 5.1, foi planejado o desenvolvimento da ferramenta com base na metodologia ágil de desenvolvimento de software, que

é utilizada como processo de desenvolvimento da AEB, objeto de estudo deste trabalho. Logo, resultando no planejamento de quatro Épicos, dezenove Features e setenta e seis User Stories, de forma que os objetivos que surgem dos problemas na Seção 5.1.2 sejam contemplados.

5.2.1 Backlog

O planejamento do desenvolvimento deste software incluiu três seções, a primeira de Épicos, onde uma visão de alto nível é fragmentada e trás uma visão de blocos de soluções que devem ser desenvolvidas. Já a segunda, trás as Features que com base nos Épicos auxiliam e aproximam as soluções que serão desenvolvidas de uma linguagem mais técnica. Por último, na terceira seção, são apresentados as User Stories, que contemplam a consolidação da transformação da visão de negócios em visão técnica, e então, são para o desenvolvimento técnico da solução.

5.2.1.1 Épicos

A Tabela 12 demonstra os Épicos levantados para desenvolvimento separados por um identificador único (ID) utilizado para preservar a rastreabilidade.

Tabela 12 – Épicos da ferramenta. Fonte: Autor.

ID	**Épicos**
1	Gestão de Projetos
2	Gestão dos Stakeholders
3	Gestão do Processo do Desenvolvimento
4	Autenticação de Usuários

5.2.1.2 Features

Para manter a rastreabilidade as Features da ferramenta foram separadas com base nos quatro Épicos (EP) do sistema. As Tabelas foram separadas com identificadores de Épicos (ID EP), identificadores de Features (ID FT) e respectivamente a descrição das Features.

Tabela 13 – Features do Épico 1. Fonte: Autor.

ID EP	**ID FT**	**Features**
1	1	Controle de Projeto
	2	Controle de Objetivos
	3	Controle de Temas de Investimento
	4	Controle de Épicos
	5	Controle de Features
	6	Controle de User Stories
	7	Controle de Critérios de Aceitação
	8	Controle de Protótipos

Tabela 14 – Features do Épico 2. Fonte: Autor.

ID EP	**ID FT**	**Features**
2	9	Controle de Equipes
	10	Controle de Desenvolvedores
	11	Controle de Clientes

Tabela 15 – Features do Épico 3. Fonte: Autor.

ID EP	**ID FT**	**Features**
3	12	Controle de Feedbacks
	13	Controle de Qualificação
	14	Controle de Velocyt
	15	Controle de Sprints
	16	Controle de Planejamento de sprints
	17	Controle de Releases

Tabela 16 – Features do Épico 4. Fonte: Autor.

ID EP	**ID FT**	**Features**
4	18	Controle de autenticidade dos usuários
	19	Controle de permissões dos usuários

5.2.1.3 Users Stories

As User Stories foram apresentadas em uma divisão por Features de Épicos. Portanto, dezenove tabelas foram construídas, onde as tabelas foram organizadas por identificadores de Épicos (ID EP), identificadores de Features (ID FT), identificadores de User Stories (ID US) e a descrição das User Stories.

Tabela 17 – User Stories do Épico 1 e da Feature 1. Fonte: Autor.

ID EP	**ID FT**	**ID US**	**User Stories**
1	1	1	ENQUANTO Scrum Master, DESEJO criar um novo projeto de um cliente, PARA manter a integridade dos dados dos projetos
		2	ENQUANTO Scrum Master, DESEJO editar um projeto, PARA manter a integridade dos dados do projetos
		3	ENQUANTO Scrum Master, DESEJO desativar um projeto, PARA manter a integridade dos projetos descontinuados
		4	ENQUANTO Scrum Master ou Cliente, DESEJO ver informações de um projeto, PARA me manter informado
		5	ENQUANTO Scrum Master, DESEJO alterar o status de um projeto, PARA informar em que posição no processo ele se encontra

Tabela 18 – User Stories do Épico 1 e da Feature 2. Fonte: Autor.

ID EP	**ID FT**	**ID US**	**User Stories**
1	2	6	ENQUANTO Cliente, DESEJO criar objetivos de um projeto, PARA informar metas de negócio que desejo alcança
		7	ENQUANTO Cliente, DESEJO editar objetivos de um projeto, PARAmanter a integridade das informações
		8	ENQUANTO Cliente, DESEJO desativar objetivos de um projeto, PARA manter a integridade dos objetivos do projeto
		9	ENQUANTO Scrum Master ou Cliente, DESEJO ver objetivos de um projeto, PARA me manter informado das metas de negócio do projeto

Tabela 19 – User Stories do Épico 1 e da Feature 3. Fonte: Autor.

ID EP	**ID FT**	**ID US**	**User Stories**
1	3	10	ENQUANTO Scrum Master, DESEJO criar temas de investimento de um projeto, PARA consolidar os objetivos no projeto
		11	ENQUANTO Scrum Master, DESEJO editar temas de investimento de um projeto, PARA manter a integridade dos temas
		12	ENQUANTO Scrum Master, DESEJO cancelar temas de investimento de um projeto, PARA manter a integridade do projeto
		13	ENQUANTO Scrum Master, DESEJO ver temas de investimento de um projeto, PARA me manter informado das diretrizes do projeto

Tabela 20 – User Stories do Épico 1 e da Feature 4. Fonte: Autor.

ID EP	**ID FT**	**ID US**	**User Stories**
1	4	14	ENQUANTO Scrum Master, DESEJO criar épicos dos temas de investimento de um projeto, PARA consolidar os temas do projeto
		15	ENQUANTO Scrum Master, DESEJO editar épicos dos temas de investimento de um projeto, PARA manter a integridade dos épicos
		16	ENQUANTO Scrum Master, DESEJO cancelar épicos dos temas de investimento de um projeto, PARA manter a integridade do projeto
		17	ENQUANTO Scrum Master, DESEJO ver épicos dos temas de investimento de um projeto, PARA me manter informado das diretrizes do projeto

Tabela 21 – User Stories do Épico 1 e da Feature 5. Fonte: Autor.

ID EP	**ID FT**	**ID US**	**User Stories**
1	5	18	ENQUANTO Scrum Master, DESEJO criar features dos épicos de um projeto, PARA consolidar os épicos do projeto
		19	ENQUANTO Scrum Master, DESEJO editar features dos épicos de um projeto, PARA manter a integridade das features
		20	ENQUANTO Scrum Master, DESEJO cancelar features dos épicos de um projeto, PARA manter a integridade do projeto
		21	ENQUANTO Scrum Master, DESEJO ver features dos épicos de um projeto, PARA me manter informado das diretrizes do projeto

Tabela 22 – User Stories do Épico 1 e da Feature 6. Fonte: Autor.

ID EP	**ID FT**	**ID US**	**User Stories**
1	6	22	ENQUANTO Scrum Master, DESEJO criar User Stories das features de um projeto, PARA consolidar as features do projeto
		23	ENQUANTO Scrum Master, DESEJO editar User Stories das features de um projeto, PARA manter a integridade das User Stories
		24	ENQUANTO Scrum Master, DESEJO cancelar User Stories das features de um projeto, PARA manter a integridade do projeto
		25	ENQUANTO Scrum Master ou Cliente ou Desenvolvedor, DESEJO ver User Stories das features de um projeto, PARA me manter informado das diretrizes do projeto
		26	ENQUANTO Scrum Master, DESEJO alterar o status das User Stories de um projeto, PARA manter o controle de estado do projeto
		27	ENQUANTO Cliente, DESEJO alterar o status de uma User Storie, PARA informar se ela foi aceita ou não

Tabela 23 – User Stories do Épico 1 e da Feature 7. Fonte: Autor.

ID EP	**ID FT**	**ID US**	**User Stories**
1	7	28	ENQUANTO Cliente, DESEJO criar critérios de aceitação de uma User Storie, PARA determinar o que é imprescindível para a aceitação da User Storie
		29	ENQUANTO Cliente, DESEJO editar critérios de aceitação de uma User Storie, PARA manter a integridade dos critérios de aceitação
		30	ENQUANTO Cliente ou Desenvolvedor, DESEJO ver critérios de aceitação de uma User Storie, PARA me informar sobre as diretrizes da User Storie
		31	ENQUANTO Cliente, DESEJO cancelar critérios de aceitação de uma User Storie, PARA manter a integridade das User Stories

Tabela 24 – User Stories do Épico 1 e da Feature 8. Fonte: Autor.

ID EP	ID FT	ID US	User Stories
1	8	32	ENQUANTO Desenvolvedor, DESEJO criar protótipos com base em features do projeto, PARA melhorar o entendimento da aplicação prática de uma feature
		33	ENQUANTO Desenvolvedor, DESEJO editar protótipos do projeto, PARA manter a integridade dos protótipos
		34	ENQUANTO Desenvolvedor, DESEJO remover protótipos do projeto, PARA manter a integridade dos protótipos
		35	ENQUANTO Desenvolvedor ou Cliente ou Scrum Master, DESEJO ver protótipos do projeto, PARA melhorar o entendimento da aplicação prática de uma feature
		36	ENQUANTO Scrum Master, DESEJO alterar o status dos protótipos, PARA manter o controle de estado dos protótipos

Tabela 25 – User Stories do Épico 2 e da Feature 9. Fonte: Autor.

ID EP	ID FT	ID US	User Stories
2	9	37	ENQUANTO Scrum Master, DESEJO criar equipes de desenvolvimento de um projeto, PARA direcionar a atuação de grupos específicos nos projetos
		38	ENQUANTO Scrum Master, DESEJO editar equipes de desenvolvimento, PARA manter a integridade das equipes
		39	ENQUANTO Scrum Master, DESEJO ver equipes de desenvolvimento, PARA ver informações dos integrantes e da equipe
		40	ENQUANTO Cliente, DESEJO ver equipes de desenvolvimento dos meus projetos, PARA ver informações dos integrantes e da equipe

Tabela 26 – User Stories do Épico 2 e da Feature 10. Fonte: Autor.

ID EP	ID FT	ID US	User Stories
2	10	41	ENQUANTO Scrum Master, DESEJO cadastrar desenvolvedores, PARA manter os dados dos desenvolvedores seguros
		42	ENQUANTO Scrum Master, DESEJO editar desenvolvedores, PARA manter a integridade dos dados dos desenvolvedores
		43	ENQUANTO Scrum Master, DESEJO desativar desenvolvedores, PARA manter a integridade dos desenvolvedores na ferramenta
		44	ENQUANTO Scrum Master, DESEJO ver desenvolvedores, PARA obter mais informações

Tabela 27 – User Stories do Épico 2 e da Feature 11. Fonte: Autor.

ID EP	**ID FT**	**ID US**	**User Stories**
2	11	45	ENQUANTO Scrum Master, DESEJO cadastrar clientes, PARA manter os dados dos clientes seguros
		46	ENQUANTO Scrum Master, DESEJO editar clientes, PARA manter a integridade dos dados dos clientes
		47	ENQUANTO Scrum Master, DESEJO ver desenvolvedores, PARA obter mais informações

Tabela 28 – User Stories do Épico 3 e da Feature 12. Fonte: Autor.

ID EP	**ID FT**	**ID US**	**User Stories**
2	11	48	ENQUANTO Cliente, DESEJO criar um feedback de uma User Storie ou de um Protótipo, PARA informar necessidade de mudança
		49	ENQUANTO Cliente, DESEJO editar um feedback, PARA manter a integridade do feedback
		50	ENQUANTO Cliente ou Scrum Master ou Desenvolvedor, DESEJO ver um feedback, PARA melhorar a visualização de possíveis mudanças
		51	ENQUANTO Cliente, DESEJO desativar um feedback, PARA manter a integridade dos feedbacks
		52	ENQUANTO Desenvolvedor ou Scrum Master, DESEJO alterar o status de um feedback, PARA informar uma resposta ao feedback

Tabela 29 – User Stories do Épico 3 e da Feature 13. Fonte: Autor.

ID EP	**ID FT**	**ID US**	**User Stories**
3	13	53	ENQUANTO Desenvolvedor, DESEJO criar a qualificação de uma User Storie, PARA priorizar e pontuar as histórias
		54	ENQUANTO Desenvolvedor, DESEJO editar uma qualificação de uma User Storie, PARA manter a integridade das qualificações
		55	ENQUANTO Desenvolvedor, DESEJO ver uma qualificação de uma User Storie, PARA melhorar a informação de prioridades
		56	ENQUANTO Desenvolvedor, DESEJO remover uma qualificação de uma User Storie, PARA manter a integridade das qualificações

Tabela 30 – User Stories do Épico 3 e da Feature 14. Fonte: Autor.

ID EP	ID FT	ID US	User Stories
3	14	57	ENQUANTO Scrum Master, DESEJO criar o velocyt de uma equipe em uma sprint, PARA preservar informações gerenciais
		58	ENQUANTO Scrum Master, DESEJO editar o velocyt de uma equipe, PARA preservar informações gerenciais
		59	ENQUANTO Scrum Master, DESEJO remover o velocyt de uma equipe, PARA manter a integridade dos dados de valocyt
		60	ENQUANTO Scrum Master, DESEJO ver o velocyt de uma equipe, PARA melhorar as informações gerenciais

Tabela 31 – User Stories do Épico 3 e da Feature 15. Fonte: Autor.

ID EP	ID FT	ID US	User Stories
3	15	61	ENQUANTO Desenvolvedores e Scrum Master, DESEJO criar sprints de uma release, PARA direcionar os trabalhos das equipes
		62	ENQUANTO Desenvolvedores e Scrum Master, DESEJO editar sprints de uma release, PARA manter a integridade das sprints
		63	ENQUANTO Desenvolvedores e Scrum Master, DESEJO cancelar sprints de uma release, PARA manter a integridade das sprints
		64	ENQUANTO Desenvolvedores e Scrum Master, DESEJO ver sprints de uma release, PARA melhorar as informações das ações do projeto

Tabela 32 – User Stories do Épico 3 e da Feature 16. Fonte: Autor.

ID EP	ID FT	ID US	User Stories
3	16	65	ENQUANTO Desenvolvedores e Cliente, DESEJO criar planejamentos de sprints de uma release, PARA determinar áreas que devam ser atacadas ou que são problemáticas
		66	ENQUANTO Desenvolvedores e Cliente, DESEJO editar planejamentos de sprints, PARA manter a integridade dos planejamentos
		67	ENQUANTO Desenvolvedores ou Cliente, DESEJO ver planejamento de sprints, PARA melhorar as informações sobre o projeto
		68	ENQUANTO Desenvolvedores ou Cliente, DESEJO remover planejamentos de sprints, PARA manter a integridade dos planejamentos de sprints

Tabela 33 – User Stories do Épico 3 e da Feature 17. Fonte: Autor.

ID EP	**ID FT**	**ID US**	**User Stories**
3	17	69	ENQUANTO Scrum Master, DESEJO criar releases, PARA determinar marcos do projeto
		70	ENQUANTO Scrum Master, DESEJO editar releases, PARA manter a integridade das releases
		71	ENQUANTO Scrum Master ou Desenvolvedor ou Cliente , DESEJO ver releases, PARA melhorar as informações do projeto
		72	ENQUANTO Scrum Master, DESEJO cancelar releases, PARA manter a integridade das releases

Tabela 34 – User Stories do Épico 4 e da Feature 18. Fonte: Autor.

ID EP	**ID FT**	**ID US**	**User Stories**
4	18	73	ENQUANTO Cliente ou Scrum Master ou Desenvolvedor, DESEJO me autenticar no sistema, PARA que eu possa realizar operaçãoes de forma segura
		74	ENQUANTO Cliente ou Scrum Master ou Desenvolvedor, DESEJO recuperar dados de autenticidade PARA realizar atenticação mesmo que eu me esqueça de atributos

Tabela 35 – User Stories do Épico 4 e da Feature 19. Fonte: Autor.

ID EP	**ID FT**	**ID US**	**User Stories**
4	19	75	ENQUANTO Scrum Master, DESEJO dar permissões a Desenvolvedores e Clientes no sistema, PARA viabilizar suas movimentações em áreas exclusivas
		76	ENQUANTO Scrum Master, DESEJO retirar permissões a Desenvolvedores e Clientes no sistema, PARA controlar a movimentação no sistema

6 Conclusão

Este trabalho surgiu por meio da necessidade do órgão Agência Espacial Brasileira (AEB) de conhecer e aprofundar aspectos de melhoria para o seu processo interno de desenvolvimento de software. Os estudos dessa área de melhoria envolvem diversas características e categorias de atuação. Dentro do processo ágil de desenvolvimento da AEB, a relação de planejamento e estimativas demonstrou carecer de atenção e aperfeiçoamento, sabendo que esses dados são estratégicos e embasam toda a aplicação de recursos da DINF.

Portanto, a área de estimativas do processo foi escolhida como tema desta pesquisa, e mais precisamente aspectos e técnicas de estimativas de esforço em processos de desenvolvimento ágil.

Desta forma o objetivo deste trabalho é entender e melhorar aspectos de estimativas de esforço em processos de desenvolvimento ágil para auxiliar na escolha de uma ferramenta ou técnica de estimativa, além disso, construir uma proposta de desenvolvimento de um software para apoiar o processo de estimativa e desenvolvimento de software na AEB.

Para que o objetivo geral deste estudo fosse atendido, foram levantadas três questões de pesquisa que foram respondidas na Seção 4 do trabalho. Nesse sentido, como guia da pesquisa foi definida uma metodologia com quatro macro etapas: Planejamento de Projeto, onde foi realizada uma pesquisa bibliográfica para embasar este estudo, e foi planejado um estudo de caso. Análise de dados, onde os dados são coletados para análise desta pesquisa, com o foco alcançar o objetivo geral do estudo. Desenvolvimento, onde com base nos dados coletados uma ferramenta é porposta para apoiar o processo de estimativas da AEB. Relato dos Resultados, onde os dados já analisados fornecem informações significativas para concluir este estudo.

Os Resultados deste estudo permitiram a obtenção e reconhecimento de sete modelos de estimativas de esforço de desenvolvimento, onde o Story Point mostrou-se o modelo mais encontrado e frequentemente indicado pelos estudos analisados.

Por meio da realização deste trabalho, foi possível perceber a importância das estimativas, sobretudo das estimativas de esforço no contexto de desenvolvimento ágil de software. Por isso, questões como de que aspectos mais reduzem a acurácia de estimativas, têm sido tão discutidas na academia. Com este estudo identificamos diversos aspectos que influenciam diretamente os resultados das estimativas de esforço, sendo os fatores pessoais onde o cliente influencia o projeto, otimismo nas estimativas e baixa na qualidade das estimativas como os fatores mais indicados.

Por fim, identificou-se que a técnica mais acurada é o Story Points, seguido do Ideal Day, isso porque são técnicas empíricas. Já o Ponto de Função COSMIC, por ser uma técnica mais complexa e de aplicação padronizada, foi cotada com a técnica menos acurada.

Então, uma ferramenta de apoio ao processo de estimativas da AEB foi proposta, resultando no levantamento de quatro Épicos, dezenove Features e setenta e seis User Stories, de forma que, esse planejamento viabilize o desenvolvimento desta ferramenta proposta.

Como trabalhos futuros, a realização da aplicação prática desse estudo é factivel, onde a acurácia das três técnica, Story Points, Ideal Day e Ponto de Função COSMIC, seriam comparadas com a aplicação prática em um ambiente real de desenvolvimento de software, possibilitando confrontar o experimento com a teoria levantada neste estudo.

Outro ponto interessante seria o desenvolvimento prático da ferramenta proposta, de forma que, reduzisse a possibilidade de ocorrência das circunstâncias levantadas como potenciais colaboradoras para a baixa da acurácia de estimativas de esforço.

Por último, a execução do estudo de caso porposto é necessária e importante, trazendo informações da real importância e consequência do uso da ferramenta proposta na redução de baixa acurácias em processos de desenvolvimento ágil de software.

Referências

ABRAHAMSSON, P.; SALO, O.; RONKAINEN, J. **Agile software development methods Review and analysis**. [S.l.]: VTT Publications 478. 107 p, Juhani Warsta - University of Oulu, 2002. Citado 5 vezes nas páginas 22, 24, 25, 26 e 30.

ABUALKISHIK, A. Z. et al. A study on the statistical convertibility of ifpug function point, cosmic function point and simple function point. **Information and Software Technology**, 2017. Citado na página 16.

ALVES, F.; FONSECA, M. Ideal day e priorização: Métodos Ágeis no planejamento. Engenharia de Software, Rio de Janeiro, 2008. Citado 2 vezes nas páginas 16 e 45.

BECK, K. **Embrace Change with Extreme Programming**. [S.l.]: IEEE Computer, 1999a. Citado na página 27.

_____. **Extreme Programming Explained: Embracing Change**. [S.l.]: Addison-Wesley, 1999b. Citado 3 vezes nas páginas 27, 28 e 31.

BERARDI, E.; SANTILLO, L. Cosmic-based project management in agile software development and mapping onto related cmmi-dev process areas. **International Workshop on Software Measurement-IWSM**, 2010. Citado na página 46.

BHARDWAJ, M.; RANA, A. Key software metrics and its impact on each other for software development projects. **ACM SIGSOFT Software Engineering Notes**, ACM, 2016. Citado 2 vezes nas páginas 15 e 31.

CALAZANS, A. T. S. A utilização do cosmic full function points para estimativa de tamanho de software. 2003. Citado 2 vezes nas páginas 45 e 46.

CAO, L. Estimating agile software project effort: an empirical study. **AMCIS 2008 Proceedings**, 2008. Citado 3 vezes nas páginas 15, 32 e 35.

CGTI-AEB. **Plano Diretor de Tecnologia da Informação da Agência Espacial Brasileira - 2016/2017**. Versão 1.4, 2016. Disponível em: <http://www.aeb.gov.br/wp-content/uploads/2012/08/PDTI-AEB-2016-2017-1.4.pdf>. Citado 4 vezes nas páginas 9, 49, 50 e 51.

CGU. **Relatório de Auditoria Anual de Contas - AEB**. Brasília, DF: MP, 2012. Disponível em: <http://www.aeb.gov.br/wp-content/uploads/2014/02/RA201306049.pdf>. Citado na página 51.

COHEN, D.; LINDVALL, M.; COSTA, P. **A State of the Art Report: Agile Software Development**. [S.l.]: Fraunhofer Center for Experimental Software Engineering Maryland and The University of Maryland, 2003. Citado 5 vezes nas páginas 23, 24, 27, 28 e 29.

COHN, M. Agile estimating and planning. 2005. Citado na página 26.

_____. **Agile Estimating and Planning**. [S.l.]: Massachusetts: Pearson Education, 2006. Citado 6 vezes nas páginas 39, 41, 42, 43, 44 e 45.

COMMEYNE, C.; ABRAN, A.; DJOUAB, R. Effort estimation with story points and cosmic function points - an industry case study. **Journal of the Software Metrics Community**, Software Measurement News, 2016. Citado 2 vezes nas páginas 46 e 62.

CONTI, C. T.; TREIN, F. A. Métodos Ágeis de desenvolvimento de software e o atendimento ao modelo cmmi. **Revista Gestão e Desenvolvimento**, Instituto de Ciências Sociais Aplicadas, 2012. Citado na página 21.

CRUZ, T. **Sistemas, métodos e processos: administrando organizações por meio de processos de negócio**. [S.l.]: Atlas, São Paulo, 2005. Citado na página 21.

DIAS, M. V. B. **Métodos Ágeis de Desenvolvimento de Software**. [S.l.]: Revista Engenharia de Software edição 20, 2010. Citado 7 vezes nas páginas 9, 10, 23, 24, 26, 27 e 28.

DRAGICEVIC, S.; CELAR, S.; TURIC, M. Bayesian network model for task effort estimation in agile software development. **Journal of Systems and Software**, 2017. Citado 4 vezes nas páginas 9, 33, 34 e 44.

EV, L. da S.; GOMES, A. B. P. Entre a especificidade e a teorização: a metodologia do estudo de caso. **Revista Teoria e Sociedade**, UFMG, 2014. Citado na página 18.

FENTON, N.; BIEMAN, J. **Software metrics: a rigorous and practical approach**. [S.l.]: CRC Press, 2014. Citado na página 38.

FENTON, N. E.; PFLEEGER, S. L. **Software metrics: a rigorous and practical approach**. [S.l.]: Intern. Thomson Computer Press [u.a.]. London, 2. ed, 1997. Citado na página 31.

FERREIRA, R. C. da C.; HAZAN, C. Análise de pontos de função: Uma aplicação nas estimativas de tamanho de projetos de software. engenharia de software. **Rio de Janeiro**, 2008. Citado 2 vezes nas páginas 16 e 35.

FONSECA, J. J. S. Metodologia da pesquisa científica. UEC, 2002. Citado na página 18.

FREIRE, Y. M. A. Tucp-m – pontos de casos de uso técnicos para manutenção de software. 2008. Citado na página 35.

GAMBA, M. L.; BARBOSA, A. C. G. Engenharia de software-aplicação de métricas de software com scrum. **Anais SULCOMP**, 2010. Citado 3 vezes nas páginas 44, 61 e 62.

GERHARDT, T. E.; SILVEIRA, D. T. **Métodos de pesquisa**. [S.l.]: Universidade Aberta do Brasil – UAB/UFRGS e Planejamento e Gestão para o Desenvolvimento Rural da SEAD/UFRGS. Porto Alegre: Editora da UFRGS, 2009. Citado 2 vezes nas páginas 18 e 19.

GIL, A. C. Métodos e técnicas de pesquisa social. São Paulo: Atlas, 2008. Citado na página 18.

HAMOUDA, A. E. D. Using agile story points as an estimation technique in cmmi organizations. **Agile Conference (AGILE)**, 2014. Citado na página 42.

HAUGEN, N. C. An empirical study of using planning poker for user story estimation. **Agile Conference**, IEEE, 2006. Citado 4 vezes nas páginas 41, 42, 43 e 44.

HAZAN, C. **Análise de Pontos de Função: Uma Aplicação nas Estimativas de Tamanho de Projetos de Software**. [S.l.]: Engenharia de Software Magazine, Edição 2, pp. 25-31., 2008a. Citado 5 vezes nas páginas 9, 32, 33, 35 e 51.

IDRI, A.; HOSNI, M.; ABRAN, A. Systematic literature review of ensemble effort estimation. **Journal of Systems and Software**, 2016. Citado na página 61.

IEEE. **IEEE standard glossary of software engineering terminology**. [S.l.]: IEEE Std 729, 1983. Citado na página 39.

_____. **IEEE standard glossary of software engineering terminology**. [S.l.]: Std 610.12.40, 1990. Citado na página 39.

ISO/IEC 19761:2011, Software engineering - COSMIC: a functional size measurement method. 2011. Citado na página 46.

ISO/IEC 9126-1:2001, Software engineering - Product quality. 2001. Citado na página 38.

JøRGENSEN, M. A review of studies on expert estimation of software development effort. Journal of Systems and Software, 2004. Citado 3 vezes nas páginas 35, 54 e 55.

_____. A review of studies on expert estimation of software development effort. **Journal of Systems and Software**, 2004. Citado na página 36.

KAN, S. H. **Metrics and Models in Software Quality Engineering**. [S.l.]: Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2002. Citado 2 vezes nas páginas 9 e 34.

KOSCIANSKI, A.; SOARES, M. d. S. **Qualidade de Software: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software**. [S.l.]: São Paulo: Novatec Editora LTDA, 2006. Citado na página 38.

KOSLOSKI, R. A. D.; DE, O. K. M. Melhoria contínua de estimativa de esforço para o desenvolvimento de software. **IV Simpósio Brasileiro de Qualidade de Software**, 2005. Citado na página 36.

KRUCHTEN, P. Introdução ao rup: Rational unified process. 2000. Citado na página 51.

LIBARDI, P. L.; BARBOSA, V. Trabalho da disciplina tópicos em computação no curso de pós-graduação–faculdade de tecnologia, limeira. 2010. Citado na página 26.

MAHNIč, V.; HOVELJA, T. On using planning poker for estimating user stories. **Journal of Systems and Software**, 2012. Citado na página 41.

MARTINS, P. G.; LAUGENI, F. P. **Administração da Produção Fácil**. [S.l.]: Saraiva, 1. ed., 272 p., São Paulo, 2013. Citado 2 vezes nas páginas 21 e 45.

MCCONNELL, S. **Software estimation: demystifying the black art**. [S.l.]: Library of Congress ISBN. Redmond, Wash. Microsoft Press, 2006. Citado na página 36.

MIKHAIL, E.; ACKERMAN, F. **Observations and Least Squares. University Press of America**. [S.l.: s.n.], 1976. Citado na página 36.

MOLOKKEN, K.; JORGENSEN, M. A review of software surveys on software effort estimation. **Empirical Software Engineering, 2003. ISESE 2003. Proceedings.**, IEEE, 2003. Citado 2 vezes nas páginas 36 e 54.

MONICO, J. F. G. et al. Acurácia e precisão: revendo os conceitos de forma acurada. **Boletim de Ciências Geodésicas**, 2009. Citado 4 vezes nas páginas 9, 36, 37 e 38.

MONTEIRO, T. C. Pontos de caso de uso técnicos (tucp): Uma extensão da ucp. 2005. Citado 2 vezes nas páginas 35 e 45.

MORESI, E. et al. Metodologia da pesquisa. universidade católica de brasília. UCB, 2003. Citado na página 18.

NGUYEN-CONG, D.; TRAN-CAO, D. A review of effort estimation studies in agile, iterative and incremental software development. **Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)**, IEEE, 2013. Citado 3 vezes nas páginas 44, 60 e 61.

OWAIS, M.; RAMAKISHORE, R. Effort, duration and cost estimation in agile software development. **Contemporary Computing (IC3), 2016 Ninth International Conference on**, IEEE, 2016. Citado na página 16.

PADMINI, K. J.; BANDARA, H. D.; PERERA, I. Use of software metrics in agile software development process. **Moratuwa Engineering Research Conference (MERCon), 2015**, IEEE, 2015. Citado na página 15.

PAULK, M. C. et al. The capability maturity model guidelines for improving the software process. 1999. Citado na página 51.

PERILLO, R. Domain model: Uma forma mais eficiente de construir aplicações enterprise. 2010. Citado na página 67.

POPLI, R.; CHAUHAN, N. Cost and effort estimation in agile software development. **Optimization, Reliabilty, and Information Technology (ICROIT), 2014 International Conference on**, IEEE, 2014. Citado 2 vezes nas páginas 15 e 61.

PRE-AEB. **Estrutura Organizacional - Organograma Completo**. 2017. Disponível em: <http://www.aeb.gov.br/wp-content/uploads/2017/03/AEB-organograma-completo.pdf>. Citado 2 vezes nas páginas 9 e 49.

PRESSMAN, R. S. **Engenharia de Software**. [S.l.]: McGraw-Hill, 2006. Citado 4 vezes nas páginas 15, 32, 39 e 51.

PRIKLADNICKI, R.; WILLI, R.; MILANI, F. **Métodos ágeis para desenvolvimento de software**. [S.l.]: Bookman Editora, 2014. Citado 2 vezes nas páginas 23 e 24.

RAITH, F. et al. Identification of inaccurate effort estimates in agile software development. **Asia-Pacific Software Engineering Conference**, IEEE, 2013. Citado na página 16.

REVISTABW. **Probabilidade e Estatística: Acurácia, Precisão e Exatidão.Revista Brasileira de Web: Tecnologia**. [S.l.]: Disponível em: <http://www.revistabw.com.br/revistabw/probabilidade-e-estatistica-acuracia-precisao-e-exatidao/> Acesso em: Out, 2017, 2014. Citado 3 vezes nas páginas 9, 36 e 37.

RITTER, R. **Planning Poker e Ideal Day: Técnicas de Abordagem de Estimativa Ágil**. 2016. Disponível em: <https://www.devmedia.com.br/planning-poker-e-ideal-day-tecnicas-de-abordagem-de-estimativa-agil/31220>. Citado 2 vezes nas páginas 9 e 43.

ROCHA, A. R. C. d.; MALDONADO, J. C.; WEBER, K. C. **Qualidade de Software: Teoria e Prática, Orgs.** [S.l.]: Prentice-Hall, São Paulo, 2001. Citado 3 vezes nas páginas 9, 22 e 31.

RUNESON, M. H. Guidelines for conducting and reporting case study research in software engineering. **Lund University**, Department Computer Science, 2008. Citado 2 vezes nas páginas 48 e 53.

SATO, D. T. Uso eficaz de métricas em métodos ágeis de desenvolvimento de software. **Dissertação Apresentada ao Instituto de Matemática e Estatística da Universidade de São Paulo**, 2007. Citado 3 vezes nas páginas 9, 40 e 41.

SCHWABER, K.; BEEDLE, M. **Agile Software Development with Scrum, Vol. 1**. [S.l.]: Prentice Hall Upper Saddle River, 2002. Citado na página 25.

SCHWABER, K.; SUTHERLAND, J. Guia do scrum. um guia definitivo para o scrum: As regras do jogo. **Disponível em: <http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf> Acesso em: out, 2017**, 2013. Citado na página 25.

SCHWEIGHOFER, T.; KLINE, A.; PAVLIC, L.; HERICKO, M. How is effort estimated in agile software development projects? **SQAMIA**, 2016. Citado 2 vezes nas páginas 55 e 61.

SOMMERVILLE, I. **Engenharia de Software**. [S.l.: s.n.], 2007. Citado 4 vezes nas páginas 27, 31, 34 e 39.

STADZISZ, P. C. z. **Projeto de Software usando a UML**. [S.l.: s.n.], 2002. Citado na página 69.

TANVEER, B.; GUZMáN, L.; ENGEL, U. M. Understanding and improving effort estimation in agile software development: an industrial case study. **Lund University**, ACM Press, 2016. Citado 7 vezes nas páginas 10, 16, 55, 56, 57, 58 e 61.

TORRECILLA-SALINAS, C. et al. Estimating, planning and managing agile web development projects under a value-based perspective. **Information and Software Technology**, 2015. Citado na página 15.

TRAN-CAO, D.; LEVESQUE, G.; ABRAN, A. Measuring software functional size: towards an effective measurement of complexity. **IEEE. Software Maintenance. Proceedings. International Conference on. [S.l.]**, 2002. Citado na página 35.

TURK, D.; FRANCE, R.; RUMPE, B. Assumptions underlying agile software development processes. **arXiv preprint arXiv:1409.6610**, Instituto de Ciências Sociais Aplicadas, 2014. Citado 3 vezes nas páginas 10, 24 e 25.

UNGAN, E.; CIZMELI, N.; DEMIRORS, O. Comparison of functional size based estimation and story points, based on effort estimation effectiveness in scrum projects. IEEE, 2014. Citado na página 62.

USMAN, M.; BRITTO, R. Effort estimation in co-located and globally distributed agile software development: A comparative study. **Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA)**, IEEE, 2016. Citado na página 58.

USMAN, M.; BöRSTLER, J.; PETERSEN, K. An effort estimation taxonomy for agile software development. **International Journal of Software Engineering and Knowledge Engineering**, IEEE, 2017. Citado na página 61.

USMAN, M.; MENDES, E.; BöRSTLER, J. Effort estimation in agile software development: a survey on the state of the practice. ACM Press, 2015. Citado 4 vezes nas páginas 10, 59, 60 e 61.

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 2 vezes nas páginas 61 e 62.

VAZQUEZ, C. E.; SIMÕES, G. S.; ALBERT, R. M. **Análise de Pontos de Função – Medição, Estimativas e Gerenciamento de Projetos de Software**. [S.l.]: 3ª. edição. São Paulo: Editora Érica, 2003. Citado 2 vezes nas páginas 32 e 35.

WELLS, D. **Extreme programming project**. [s.n.], 2000. Disponível em: <http://www.extremeprogramming.org/map/project.html>. Citado 2 vezes nas páginas 9 e 31.