UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

KALIANE LARISSA VIESSELI

FRAMEWORK PARA ESTIMAR ESFORÇO DE MANUTENÇÃO EM UM AMBIENTE MULTI-EQUIPE

DOIS VIZINHOS

2020

KALIANE LARISSA VIESSELI

FRAMEWORK PARA ESTIMAR ESFORÇO DE MANUTENÇÃO EM UM AMBIENTE MULTI-EQUIPE

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de Bacharel em Engenharia de Software da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientador: Prof. Dr. Gustavo Jansen de Souza Santos

DOIS VIZINHOS

2020


4.0 Internacional

Esta licença permite compartilhamento, remixe, adaptação e criação a partir do trabalho, mesmo para fins comerciais, desde que sejam atribuídos créditos ao(s) autor(es). Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.

Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COORD. DO CURSO DE ENG. DE SOFTWARE-DV

TERMO DE APROVAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO - TCC

FRAMEWORK PARA ESTIMAR ESFORÇO DE MANUTENÇÃO EM UM AMBIENTE MULTI-EQUIPE

Por

Kaliane Larissa Viesseli

Monografia apresentada às 14 horas 00 min. do dia 13 de Maio de 2021 como requisito parcial, para conclusão do Curso de Bacharelado em Engenharia de Software da Universidade Tecnológica Federal do Paraná, Câmpus Dois Vizinhos. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação e conferidas, bem como achadas conforme, as alterações indicadas pela Banca Examinadora, o trabalho de conclusão de curso foi considerado APROVADO.

Banca examinadora:

Profa. Alinne Cristinne Correa Souza	Membro
Prof. Rodolfo Adamshuk Silva	Membro
Prof. Gustavo Jansen de Souza Santos	Orientador
Prof. Francisco Carlos Monteiro Souza	Professor responsável TCCII

Documento assinado eletronicamente por (Document electronically signed by) **GUSTAVO JANSEN DE SOUZA SANTOS**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em (at) 20/05/2021, às 09:41, conforme horário oficial de Brasília (according to official Brasilia-Brazil time), com fundamento no (with legal based on) art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por (Document electronically signed by) **RODOLFO ADAMSHUK SILVA**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em (at) 20/05/2021, às 09:50, conforme horário oficial de Brasília (according to official Brasilia-Brazil time), com fundamento no (with legal based on) art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por (Document electronically signed by) **ALINNE CRISTINNE CORREA SOUZA**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em (at) 20/05/2021, às 15:30, conforme horário oficial de Brasília (according to official Brasilia-Brazil time), com fundamento no (with legal based on) art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por (Document electronically signed by) **Kaliane Larissa Viesseli**, **Usuário Externo**, em (at) 20/05/2021, às 15:35, conforme horário oficial de Brasília (according to official Brasilia-Brazil time), com fundamento no (with legal based on) art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site (The authenticity of this document can be checked on the website) https://sei.utfpr.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador (informing the verification code) **2044509** e o código CRC (and the CRC code) **56230335**.

AGRADECIMENTOS

Agradeço primeiramente aos meus pais e minha irmã, por estarem sempre ao meu lado, comemorando nas horas alegres e me amparando nos momentos difíceis, sempre me apoiando e incentivando para que eu chegasse até aqui.

Agradeço a Deus, por estar comigo ao longo de todo esse percurso, me orientando e protegendo.

Agradeço a todos que fizeram parte da minha jornada acadêmica, aos meus amigos por acreditarem em meu potencial e sempre estarem me apoiando, ao corpo docente por todo conhecimento repassado e em especial ao meu orientador, professor Gustavo Jansen de Souza Santos por todas as orientações, suporte e incentivos nos momentos mais difíceis.

A todos que fizeram parte da minha formação, o meu muito obrigado.

"Nosso destino vive dentro de nós, você só tem que ser corajoso o suficiente para vê-lo"

(Chapman, Brenda; Valente, 2012)

RESUMO

VIESSELI, K.. FRAMEWORK PARA ESTIMAR ESFORÇO DE MANUTENÇÃO EM UM AMBIENTE MULTI-EQUIPE. 74 f. Trabalho de Conclusão de Curso – Coordenadoria do Curso de Engenharia de Software, Universidade Tecnológica Federal do Paraná. Dois Vizinhos, 2021.

As estimativas de software são de grande importância, pois a partir dos esforços estimados, é possível tomar decisões sobre como será o andamento das atividades, além de resultarem no sucesso ou fracasso do projeto. Consequentemente, existe um número considerável de pesquisas que utilizam métodos diferentes de estimativa com a intenção de torná-las precisas, ou seja, deixar as estimativas o mais próximo do real esforço. Porém, apenas uma pequena parcela dessas pesquisas são direcionadas à manutenção de software, que é considerada uma das áreas que gera mais custos para as empresas. Este trabalho propõe o desenvolvimento do *framework Web* EstimAi para auxiliar as estimativas de esforço de manutenção, em um ambiente composto por múltiplas equipes trabalhando em um mesmo projeto. Para atingir esse objetivo, o framework utiliza dados históricos. O real esforço de atividades já finalizadas serve como base para o cálculo de estimativa de esforço de novas atividades. Para validar o *framework*, foi realizado um estudo de caso em duas equipes de manutenção, e como resultado, foi obtido que o método de estimativa de esforço segundo a literatura não é precisa. Porém, na opinião dos usuários, a estimativa foi consideravelmente aceitável, o que se conclui que o *framework* necessita de ajustes para melhorar ainda mais a usabilidade e a estimativa gerada.

Palavras-chave: Estimativa de esforço, Manutenção de software

ABSTRACT

VIESSELI, K.. FRAMEWORK FOR ESTIMATING MAINTENANCE EFFORT IN A MULTI-TEAM ENVIRONMENT. 74 f. Trabalho de Conclusão de Curso – Coordenadoria do Curso de Engenharia de Software, Universidade Tecnológica Federal do Paraná. Dois Vizinhos, 2021.

During the software lifecycle, good effort estimation allows teams to make decisions about how activities will progress. Inaccurate or incorrect estimates directly impact software delivery, customer dissatisfaction, and ultimately, project failure. Several approaches proposed methods to calculate effort estimation that is approximate to the real effort. However, there is a gap in the state of practice concerning effort estimation in software maintenance activities. This work proposes a framework to calculate effort estimates during software maintenance, in a company environment composed of multiple teams working on the same project. This framework uses the real effort from previous activities to estimate the effort for new ones. To validate the EstimAi framework, an experimental evaluation was conducted in two maintenance teams during two Sprint iterations. As a result, the approach is not accurate; however, according to developers' feedback, the estimate was acceptable, which leads us to conclude that the framework needs adjustments to further improve usability and the generated estimate.

Keywords: Effort estimation, Software maintenance

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

API	*Application Programming Interface*
ERES	Escola Regional de Engenharia de Software
ERP	*Enterprise Resource Planning*
GQM	*Goal/Question/Metric*
MMRE	*Mean Magnitude of Relative Error*
MRE	*Magnitude of Relative Error*
MRQ	Média das Respostas no Questionário de Avaliação
MS	Mapeamento Sistemático
PNL	Processamento de Linguagem Natural
PO	Product Owner
PRED	*Percentage Relative Error Deviation*
QP	Questão de Pesquisa
RF	Requisito Funcional

SUMÁRIO

1 INTRODUÇÃO

O termo "Engenharia de Software" surgiu na conferência da OTAN em 1969, na qual pesquisadores discutiam sobre vários problemas relacionados ao desenvolvimento de software, como atrasos nas entregas, custos maiores do que o esperado e planejado, falta de confiabilidade e funcionalidades que não contemplavam as reais necessidades dos clientes (SOMMERVILLE, 2019).

Desde então, muito se evoluiu a respeito da Engenharia de Software como tema de pesquisa; porém, ainda se observam softwares sendo entregues atualmente com os mesmos problemas discutidos em 1969. Uma demonstração dessa problemática foi levantada pelo Standish Group (2015), na qual foi estimado que 60% dos projetos não são entregues dentro do tempo planejado, 56% ultrapassam o orçamento inicial estipulado e 44% não atingem seu objetivo dentro do cronograma e orçamento estipulado.

No contexto da Engenharia de Software, pode citar a manutenção de software como a área que possui um grande impacto nos custos de uma empresa. Como apresentado por Koskinen (2015), o custo para manter e gerenciar a evolução de um software é mais de 90% de seu custo total e cerca de 75% dos custos são causados por melhorias. A manutenção de software é inevitável, pois o software certamente sofrerá mudanças após a entrega para o cliente, podendo ser por motivos de defeitos encontrados, adaptação ou porque o cliente precisou de novas funcionalidades (PRESSMAN, 2016).

As empresas demandam de esforço e de tempo bem maior para realizar a manutenção se comparado a um sistema novo e isso implica diretamente no custo de manutenção. Por esse motivo, persiste a necessidade por técnicas que auxiliem nas estimativas de esforço para projetos de manutenção, para que as empresas possam gerenciar melhor seus projetos (FREIRE; BELCHIOR, 2006).

Para auxiliar na gestão e no planejamento de projetos de software, é muito utilizada a metodologia ágil *Scrum*, que possui como principais argumentos a constante inspeção e adaptação do desenvolvimento de um projeto, além de ser focada na gestão de ciclos

iterativos (AUDY, 2015). Tradicionalmente, o *Scrum Team* ou time de desenvolvimento é composto por até nove pessoas que colaboram entre si. Porém, em grandes projetos, necessita-se criar múltiplas equipes que trabalharão paralelamente no mesmo projeto, tornando a gestão mais complexa. Caso as equipes não sejam auto organizadas, estas podem ter suas entregas atrasadas, além do aumento de defeitos encontrados.

A proposta deste trabalho consiste em escolher e implantar métodos para auxiliar na estimativa de esforço no setor de manutenção de software de uma empresa localizada na região sudoeste do Paraná e que atua na área de gestão para supermercados e lojas de materiais de construção. A empresa utiliza a metodologia ágil *Scrum* e possui múltiplas equipes trabalhando paralelamente em um mesmo projeto. Como produto deste trabalho, será desenvolvida uma ferramenta *Web*, na qual seja possível centralizar todas as estimativas de esforço propostas e implementar métodos para a geração automática de estimativas.

1.1 MOTIVAÇÃO

Ao reconhecer a necessidade de manutenção de um software para suprir as novas necessidades do cliente, sempre surge o questionamento de quanto tempo será necessário para concluir o projeto e, para chegar a uma conclusão, são utilizadas estimativas. Porém, muitos fatores dificultam a obtenção de uma resposta confiável. Para estimar o tempo que será necessário, inicialmente necessita-se estimar o esforço empregado na execução do projeto (VAZQUEZ; SIMÕES; ALBERT, 2013).

No sudoeste do Paraná está localizada uma empresa de desenvolvimento de software; esta possui setores específicos para o desenvolvimento dos sistemas, comumente denominados de fábricas. Na empresa em questão, existem duas fábricas: a fábrica de inovação voltada à criação de novos produtos, e a fábrica de manutenção com o princípio de melhorar e manter os softwares mais antigos.

Para o presente trabalho, será utilizado como base o processo da fábrica de manutenção da empresa citada, que utiliza a metodologia ágil *Scrum*. Na fábrica em questão são aplicadas manutenções corretivas e perfectivas, e ambas possuem uma estimativa de esforço gerada pela própria equipe responsável pela atividade no momento de Planejamento de *Sprint*.

O processo de Planejamento de *Sprint*, apresentado na Figura 1, é um dos eventos da metodologia ágil *Scrum*, na qual a equipe responsável pelas atividades de manutenção irá estimar o esforço necessário para as atividades de sua *Sprint*. Neste processo, o *Scrum*

Master tem o papel de levantar o total esforço do time durante a *Sprint*. Este valor é denominado de *Sprint Velocity* (SILVA, 2011) e é controlado por meio de uma planilha, que possui campos referentes à quantidade de dias úteis da *Sprint* e à quantidade de pessoas na equipe, além de campos para armazenar todas as estimativas das atividades. Seguindo o processo, o *Product Owner* (PO) apresenta o item de maior prioridade do *Backlog* e o time realiza a estimativa de esforço para cada tarefa utilizando uma técnica denominada *Planning Poker* (POKER, 2021).

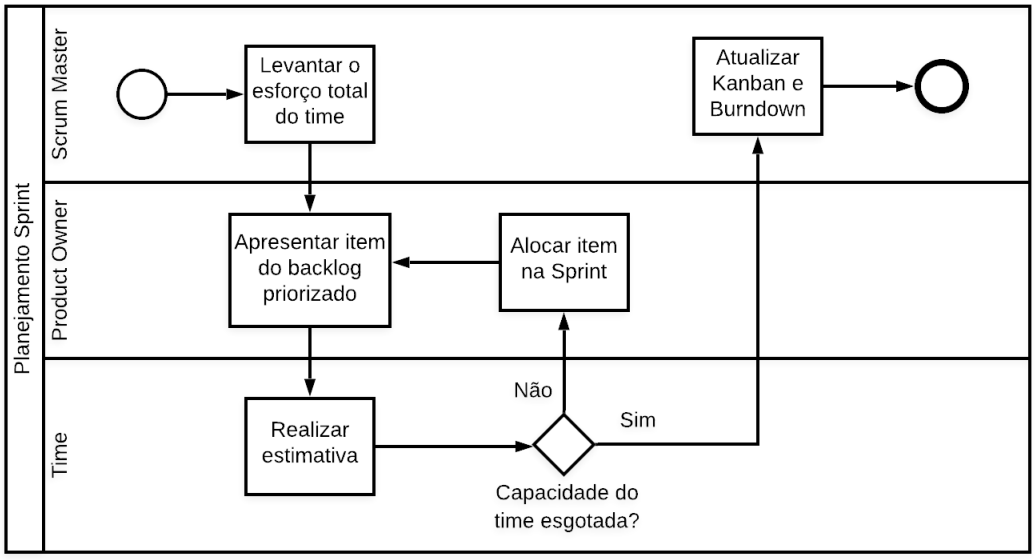

Figura 1: Diagrama de Planejamento da *Sprint*
Fonte: Autoria própria

Após todos da equipe entrarem em consenso sobre a estimativa, a mesma é descontada do esforço total levantado pelo *Scrum Master* na planilha de controle e registrada na IBM Jazz.net, uma ferramenta de gerenciamento que engloba todos os aspectos do desenvolvimento de um software (JAZZ, 2019). Após a etapa de estimativa, é verificado se a capacidade do time está esgotada, ou seja, se a soma de todas as atividades estimadas é igual ao *Sprint Velocity* (SILVA, 2011). Caso não se tenha atingido o valor total do *Sprint Velocity*, continuam-se estimando as tarefas. Em caso contrário, encerra-se o evento de planejamento de *Sprint* e inicia-se a etapa de desenvolvimento, onde é necessário manter atualizado o *Kanban*[1] e o *Burndown*[2], que auxiliam na visualização de como está o andamento da *Sprint*.

[1]Kanban é uma forma de controlar as tarefas e fluxos através de gestão visual, utilizando de colunas e cartões. `www.artia.com/kanban/`

[2]Burndown é um gráfico que apresenta a performance da equipe, comparando o que foi planejado com o que está sendo entregue. `https://blog.acelerato.com/projetos/graficos-burndown-x-burnup/`

No contexto da empresa analisada, uma problemática levantada pelas equipes nas reuniões de encerramento de *Sprint* é o equívoco das estimativas das atividades. Assim, é possível ter estimativas superestimadas ou subestimadas, que podem gerar como consequência atrasos na *Sprint* e na entrega das atividades, além de prejuízos à empresa.

A empresa trabalha com um sistema *Enterprise Resource Planning* (sistema de gestão integrado - ERP), no qual cada equipe é responsável por uma área diferente do sistema. Porém, pode ocorrer casos em que alterações que são de conhecimento específico da equipe X, tenham que ser alterados pela equipe Y que não possui conhecimento sobre o determinado módulo, assim gerando uma grande imprecisão nas estimativas. Nesse contexto, a base de dados que será gerada pela ferramenta é um ponto muito importante, pois mesmo que a equipe Y não conheça o módulo que deve ser alterado, a equipe X já o alterou várias vezes e suas análises poderão ser utilizadas pelo framework para ajudar a outra equipe a gerar estimativas de esforço mais precisas.

Vazquez, Simões e Albert (2013) argumentam que uma base de dados históricos estimados e realizados podem ser utilizados como fonte de informação para projetos futuros, por meio do qual as novas estimativas podem ser realizadas com mais confiança e segurança, gerando decisões mais rápidas e com menor custo para a organização. Tendo em vista tais problemáticas encontradas, surge a necessidade de estudar novas alternativas para ajudar no processo de estimativa de esforço da empresa sob estudo.

1.2 OBJETIVO GERAL

A partir da problemática encontrada na fábrica de manutenção, o objetivo deste trabalho é auxiliar e melhorar as estimativas da empresa estudada, assim desenvolvendo um *framework*, no qual serão aplicados métodos já existentes de estimativa de esforço da área de manutenção de software.

De maneira geral, o processo da ferramenta pode ser observado na Figura 2, que inicia com a extração das informações da nova atividade (ação 1). A partir da nova atividade, o *framework* irá extrair todas as informações pertinentes, como por exemplo os requisitos propostos. A ação 2 tem como objetivo identificar as atividades já finalizadas que são similares à nova e, para a realização desta busca, será utilizada a ação 3 que consiste na base com todas as informações das atividades já finalizadas. Para realizar a identificação de atividades similares, é feito um cálculo de similaridade textual, comparando as palavras utilizadas na descrição das atividades. A partir das atividades similares encontradas,

inicia-se a ação 4, na qual são extraídas as informações referentes ao real esforço utilizado. Parte-se do argumento que, se descrição é similar, então o tempo utilizado também será parecido. Para a realização desta ação, foi utilizado novamente a ação 3 que é a base que armazena estas informações.

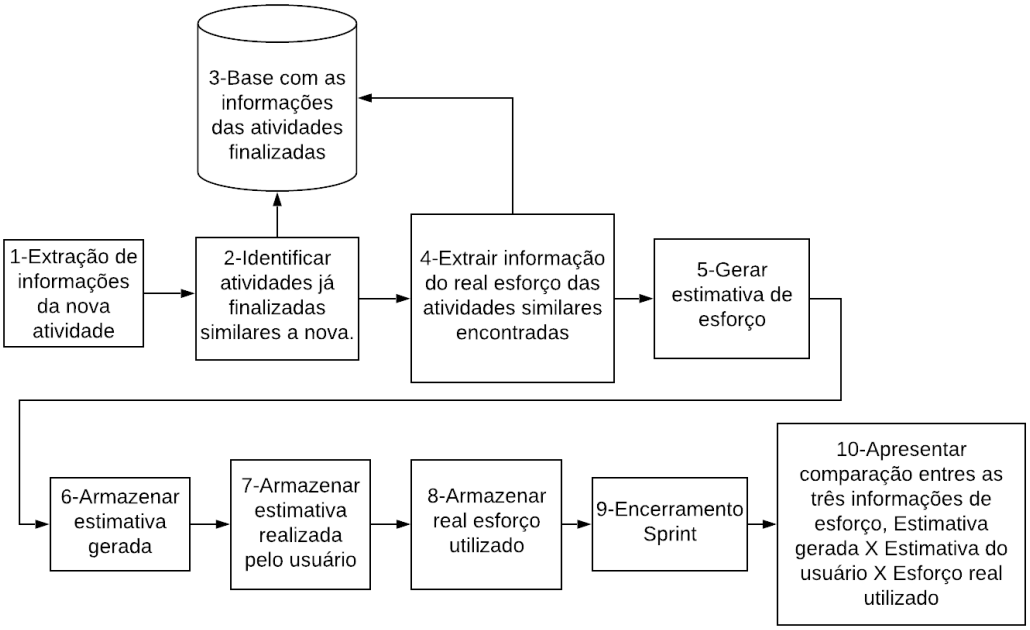

Figura 2: Processo do *framework* para estimativa de esforço
Fonte: Autoria própria

Por fim, a ação 5 irá gerar a estimativa de esforço a partir dos dados extraídos até o momento, sendo as atividades similares e seus respectivos esforços reais utilizados. A partir desse cálculo, é gerada a estimativa de esforço para a nova atividade, e seu armazenamento em banco é representado na ação 6. Referente à nova atividade, na ação 7 é realizado o armazenamento das estimativas realizadas pelo usuário e na ação 8 o armazenamento do real esforço utilizado. Ao final da *Sprint*, tem-se o processo de encerramento da *Sprint* descrito como ação 9 e, por último, a ação 10 irá apresentar ao usuário uma comparação entre as três informações armazenadas pelo *framework*, que são: a estimativa gerada, a estimativa realizada pelo usuário e o esforço real utilizado.

Como objetivo final para validar a real efetividade e desempenho dos métodos utilizados, será realizada um estudo de caso. A ferramenta irá ser implantada no período de duas *Sprints*, no setor da fábrica de manutenção de software da empresa.

1.3 OBJETIVOS ESPECÍFICOS

Com base no objetivo geral, foram definidos os seguintes objetivos específicos:

- Investigar métodos para estimativa de esforço na área de manutenção de software;
- Adaptar os métodos de estimativa de esforço para o contexto da empresa;
- Identificar tecnologias para o desenvolvimento da ferramenta *Web*;
- Desenvolver a ferramenta *Web* no qual serão implementados os métodos adaptados de estimativa de esforço;
- Aplicar estudo de caso.

1.4 CONTRIBUIÇÕES

As contribuições deste trabalho são sumarizadas a seguir.

- Um método de cálculo de estimativa de esforço, realizado a partir de dois métodos utilizados no estado da arte: similaridade textual para recuperar atividades similares, e análise de regressão linear para cálculo da estimativa;
- Um *framework Web* que implementa este método, resolvendo uma lacuna do estado da prática. O *framework* foi projetado para adaptar outros cálculos de estimativa e permitir a integração com diversas ferramentas de gestão. Dessa forma, contribui-se para o reuso do *framework* para outras pesquisas similares;
- Um procedimento de avaliação do método por meio de um estudo de caso. O estudo proposto analisou a precisão do método em relação ao real esforço utilizado, assim como a percepção dos desenvolvedores em relação ao *framework* desenvolvido. Além disso, o estudo lista uma série de obstáculos e lições aprendidas na condução do estudo. Portanto, contribui-se para a replicação do estudo em outras empresas que possuam as mesmas necessidades.

2 ASPECTOS CONCEITUAIS

Neste capítulo serão apresentados os conceitos fundamentais relacionados ao tema proposto. Na Seção 2.1 serão apresentados os conceitos relacionados a engenharia de software. Depois, na Seção 2.2 será explicado sobre os processos de produção de software e a Seção 2.3 apresentará sobre a manutenção de software e seus tipos. A conceitualização sobre estimativa de esforço e seus métodos serão explicados na Seção 2.4. Por fim, na Seção 2.5 serão apresentadas as considerações finais do capítulo.

2.1 ENGENHARIA DE SOFTWARE

Atualmente, o software está inserido em todos os aspectos de nossas vidas, gerando assim grandes interesses por mais funcionalidades e aplicações e, consequentemente, produzindo softwares inflados e complexos. Não apenas os softwares do nosso dia-a-dia, mas também os softwares empresariais estão sujeitos a falhas. Para esses últimos, as consequências das falhas se mostram de formas diferentes, pois elas podem acarretar em grandes prejuízos para as empresas. Este cenário apenas reforça a importância de métodos e técnicas que auxiliem os profissionais a desenvolver softwares de qualidade (PRESSMAN, 2016).

A “Engenharia de Software” é considerado um termo bastante amplo e que possui como objetivo apoiar o desenvolvimento de um sistema, desde as primeiras fases de levantamento de requisitos até a manutenção de software. Portanto, é importante destacar que a engenharia de software engloba todos os aspectos da produção de um software, desde gerenciamento de projeto, gerenciamento da qualidade, os processos de produção e os artefatos gerados (SOMMERVILLE, 2019).

2.2 PROCESSO DE SOFTWARE

Segundo Sommerville (2019), o processo de software consiste em várias atividades relacionadas que tem como objetivo a produção de um produto de software. Apesar de existirem vários modelos de processos diferentes, todos devem possuir quatro atividades principais: (I) especificação de software; (II) projeto e implementação de software; (III) validação de software; e (IV) evolução de software.

Ainda que todos os processos devam utilizar as atividades descritas, isso não impediu dos processos de software de evoluir, assim aproveitando melhor a capacidade das pessoas da organização, além das características do sistema em desenvolvimento. Como exemplo dessa evolução, pode citar as metodologias ágeis, na qual o processo é realizado de forma gradativa, tornando a realização de alterações mais fácil conforme as mudanças dos clientes (SOMMERVILLE, 2019). A partir desse conceito de desenvolvimento incremetal, foram criadas diversas metodologias ágeis para facilitar o processo de desenvolvimento, entre elas tem a metodologia *Scrum*.

2.2.1 *SCRUM*

O *Scrum*, segundo Schwaber e Sutherland (2017), é um modelo de processo no qual os integrantes podem gerenciar o trabalho em produtos complexos e resolver problemas adaptativos, enquanto entregam produtos com o maior valor possível. Para o uso e sucesso do *Scrum*, é necessário que o time faça o uso dos papéis, artefatos e eventos.

O time *Scrum* é descrito como auto-organizável e multifuncional, o que significa que eles escolhem a melhor forma para completar seus trabalhos e não dependem de ajuda de pessoas externas ao time (SCHWABER; SUTHERLAND, 2017). Cada time *Scrum* é composto por três papéis fundamentais:

- ***Product Owner* (PO):** também conhecido como dono do produto, é responsável por gerenciar o *Backlog* do produto, o qual contêm todas as atividades a serem desenvolvidas, além de priorizar os itens a serem produzidos pelo time.
- ***Scrum Master*:** é responsável por garantir que todo o time siga as regras e valores propostos pelo *Scrum*, remover impedimentos que estão atrapalhando o progresso do time e trabalhar juntamente com outros *Scrum Masters* com o objetivo de alinhar os projetos e aumentar a eficácia da aplicação do *Scrum*.

- **Time de desenvolvimento:** é formado por vários profissionais que trabalham com o objetivo de entregar um incremento potencialmente liberável do produto ao final de cada *Sprint*. Os times de desenvolvimento devem possuir um tamanho ideal, sendo entre três a nove integrantes, assim facilitando o gerenciamento interno.

Além dos papéis, o *Scrum* também define os eventos, que são utilizados para criar uma rotina e diminiur reuniões não definidas. Segundo Schwaber e Sutherland (2017) existem cinco eventos chave para o *Scrum*, todos possuindo uma duração máxima planejada. Os eventos podem ser observados a seguir:

- **Planejamento da *Sprint*:** é o evento inicial da *Sprint* e tem como objetivo planejar as funcionalidades que serão desenvolvidas durante a *Sprint*. O time de desenvolvimento avalia quais atividades poderão ser completadas ao longo da *Sprint*, por meio das estimativas de esforço das atividades e da capacidade atual do time.
- ***Sprint*:** é o momento no *Scrum*, tendo a duração de um mês ou menos, no qual é desenvolvida a parte incremental do produto e após o termino do mesmo, outra *Sprint* tem inicio de imediato.
- **Reunião diária:** é um evento com duração máxima de 15 minutos realizada diariamente, no qual o time de desenvolvimento inspeciona o progresso realizado para atingir os objetivos da *Sprint*.
- **Revisão da *Sprint*:** é realizada ao final da *Sprint* e tem como objetivo apresentar, para o time e aos demais interessados, o que foi realizado durante a *Sprint*, inclusive os incrementos prontos do produto.
- **Retrospectiva da *Sprint*:** ocorre após o evento de revisão da *Sprint*, e tem como objetivo avaliar como foi a última *Sprint*, em relação às pessoas, relacionamentos, processos e ferramentas, e por fim criar um plano para melhorias que devem ser aplicadas na próxima *Sprint*.

Outro componente importante do *Scrum* são seus artefatos, os quais são projetados para gerar transparência das informações. Schwaber e Sutherland (2017) citam três artefatos principais, sendo eles:

- ***Backlog* do Produto:** é uma lista ordenada de todas as mudanças que são necessárias no produto e é de responsabilidade do PO mantê-lo atualizado, disponível e ordenado conforme os requisitos mais prioritários.

- ***Backlog* da *Sprint*:** é um conjunto de itens do *Backlog* do Produto selecionados para serem desenvolvidos durante a *Sprint*. Com ele é possível tornar visível para o time todo o trabalho que será realizado para atingir o objetivo da *Sprint*.
- **Incremento:** é a soma de todos os itens do *Backlog* do Produto que foram terminados durante a *Sprint* com todos os incrementos das *Sprints* anteriores.

O *Scrum* é considerado leve e simples de entender (SCHWABER; SUTHERLAND, 2017), por esse motivo, o mesmo é amplamente utilizado para o controle de entregas, podendo ser utilizado tanto para o desenvolvimento de um novo produto quanto para a manutenção de um software.

2.3 MANUTENÇÃO DE SOFTWARE

Muitas empresas tem a maior parte do seu tempo dedicado a manter softwares e é por isso que a manutenção é uma etapa fundamental para preservar a vida útil dos mesmos, pois é ela que evita a deterioração do sistema, por meio de alterações e melhorias contínuas (NGUYEN; BOEHM; DANPHITSANUPHAN, 2011).

A partir do momento que um software é entregue ao cliente, inicia-se o processo de manutenção de software, que é caracterizada por modificações no sistema enquanto o mesmo está sendo utilizado pelos usuários. Assim, é comum as empresas terem mais de uma versão do sistema, onde uma é a versão que está em produção e que o cliente está utilizando, e a outra está sendo evoluída pela equipe de desenvolvimento para que seja lançada futuramente (RAJLICH, 2014), (SPINOLA, 2011).

Segundo Lientz e Swanson (1980) existem quatro principais tipos de manutenção de software, as quais podemos definir como:

- Manutenções corretivas que se referem à correção de erros;
- Manutenções adaptativas que são alterações no software para adequá-lo conforme novas mudanças externas, como regras de negócio, leis e constituições;
- Manutenções perfectivas ou evolutivas que irão acrescentar novas funcionalidades; e
- Manutenções preventivas ou reengenharia, a qual altera o software para melhorar a estrutura para futuras manutenções ou para tratar uma falha antes que aconteça.

2.4 ESTIMATIVA DE ESFORÇO

Todo projeto inicia com um planejamento das atividades, no qual deve-se levantar questionamentos importantes sobre como será o andamento do mesmo. Pressman (2016) cita que o planejamento do software é composto por cinco atividades: análise de risco, estimativa, cronograma, planejamento da gestão da qualidade e por último, planejamento do gerenciamento de alteração.

Em um projeto, existe a necessidade de determinar como será o andamento do mesmo, se ele será desenvolvido e quais os prazos para entrega. Para isso, as estimativas são responsáveis por determinar quanto de esforço será empregado, qual o custo, qual o cronograma e o escopo do projeto (VAZQUEZ, 2009). Porém, existem muitos fatores que podem dificultar na geração de uma estimativa mais precisa, como por exemplo requisitos imprecisos e mal detalhados, falta de conhecimento por parte dos estimadores e nova atividade muito diferente de tudo que já foi realizado.

O presente trabalho abordará a estimativa de esforço, a qual tem como principal objetivo identificar qual o tempo necessário para a conclusão de uma atividade. O esforço pode ser medido em horas, dias ou até em semanas e manifesta-se em estimativa de esforço ou estimativa de prazo. A estimativa de esforço resultará em uma quantidade de horas corridas e a estimativa de prazo leva em consideração número de pessoas envolvidas e horas trabalhadas por dia para determinar em quantos dias a atividade será entregue.

Existem diversas técnicas no âmbito das estimativas de esforço e várias maneiras de classificá-las. Por exemplo, Khatibi e Jawawi (2011) classificam todas as técnicas em apenas dois grupos, os métodos não-algorítmicos e os métodos algorítmicos.

2.4.1 MÉTODOS NÃO-ALGORÍTMICOS

Os métodos não-algorítmicos não utilizam equações matemáticas para obter um resultado e baseiam-se em comparações, inferências e analogias. Para se utilizar os métodos não-algorítmicos, normalmente se faz necessário o uso de um conjunto de dados anteriores do mesmo projeto ou de um projeto semelhante (KHATIBI; JAWAWI, 2011).

Como exemplo de método não-algorítmico, podemos citar a opinião especializada, que gera as estimativas dos projetos por meio do parecer de pessoas que possuem experiência em projetos similares (KHATIBI; JAWAWI, 2011). Baseada no método de opinião especializada, existe uma técnica denominada *Planning Poker*, na qual cada membro

da equipe utiliza de sua própria experiência em projetos para estimar as atividades por meio de um baralho de cartas com uma determinada sequência númerica. O objetivo do *Planning Poker* é auxiliar a equipe a chegar em um consenso entre os valores das cartas jogadas e assim definir uma estimativa para a atividade (POKER, 2021).

2.4.2 MÉTODOS ALGORÍTMICOS

Os métodos algorítmicos geram a estimativa por meio de equações matemáticas e, para isso, necessitam de dados iniciais e métricas do projeto (KHATIBI; JAWAWI, 2011). As métricas são uma forma de medir quantitativamente e objetivamente atributos de um software, com o objetivo de melhorar a visualização e construção de um projeto. As medidas são os resultados obtidos por meio das métricas, e podem indicar a quantidade, dimensão, tamanho e capacidade de um determinado software que está passando pelo processo de medição (SOMMERVILLE, 2019) (PRESSMAN, 2016).

Segundo Sommerville (2019), existem dois tipos de métricas: (i) as métricas de controle que são voltadas ao processo de gerenciamento e ajudam gerentes a decidir se alterações no processo devem ou não ser feitas, como por exemplo métricas de esforço médio e tempo para correção de defeitos; e (ii) as métricas de previsão, também conhecidas como métricas de produto, que auxiliam a realizar estimativas para o sistema e são relacionadas diretamente ao software, como por exemplo métricas de complexidade, tamanho, coesão e acoplamento, entre outros.

Entre os métodos classificados como algorítmicos, podemos citar como exemplo o método de analogia, o qual define uma função que compara projetos reais já finalizados de um banco de dados, e apresenta como resultado o nível de similaridade entre esses projetos finalizados e o novo projeto a ser estimado (KHATIBI; JAWAWI, 2011). Outro exemplo é a similaridade textual, que é uma maneira de mensurar a relação entre dois textos. Podendo se tornar um grande desafio, pois mesmo que o texto utilize as mesmas palavras, uma mudança na ordem das mesmas pode influenciar no significado geral do texto e no resultado da similaridade (CAVALCANTI, 2018).

Para medir a similaridade textual, existem diversos métodos já criados, entre eles pode-se citar o coeficiente de Sørensen-Dice, também conhecido como índice de Sørensen ou coeficiente de Dice. O mesmo foi proposto por Dice (1945) e Sørensen (1948), e é um algoritmo que utiliza operações de conjuntos para calcular a similaridades entre dois conjuntos de amostras discretas. Porém, o modelo foi abstraído e também é utilizado para identificar a proximidade entre duas *strings* contando as intersecções entre elas.

Para realizar o cálculo de intersecção entre duas *strings*, o modelo propõe o uso de *n-grams*[1], portanto, a ordem em que as *strings* aparecem não influencia no resultado final. Assim, dadas duas *strings* A e B, é possível calcular a similaridade por meio da fórmula:

$$s = \frac{2|A \cap B|}{|A| + |B|} \tag{1}$$

onde, $|A|$ e $|B|$ correspondem ao número de *n-grams* de cada *string*, e $|A \cap B|$ representa a cardinalidade do conjunto de intersecção (OLIVEIRA, 2015). O resultado do coeficiente de Sørensen-Dice pode variar entre 0 e 1, no qual 0 significa que os textos não são similares e 1 significa que são exatamentes iguais.

Outro exemplo de método algorítmico é o método de análise de regressão linear, o qual é uma equação matemática que avalia projetos em andamento utilizando dados de projetos antigos já finalizados (FEDOTOVA; TEIXEIRA; ALVELOS, 2013). A análise de regressão tem como objetivo realizar uma análise estatística, com o intuito de encontrar uma relação funcional entre duas variáveis (x) e (y), dispostas em um plano cartesiano de duas dimensões, e por fim estabelecer uma equação para representar a regressão estudada (MONTGOMERY; PECK; VINING, 2012 apud MIGUEL, 2016).

Com base na análise de regressão e da função linear gerada a partir da relação dos parâmetros informados, é linear para estimar um valor de uma variável observada (no caso, y), conforme valores da variável de entrada (x) (FEDOTOVA; TEIXEIRA; ALVELOS, 2013). Para analisar o comportamento das variáveis, pode-se utilizar o gráfico denonimado de diagrama de dispersão, representado na Figura 3. O diagrama de dispersão apresentado é uma representação de uma situação ideal da utilização da regressão linear, no qual os dados tem um comportamento linear. Porém, podem existir situações que encontrem valores extremos, que são denominados de *outlier*. Os *outlier* podem afetar os algoritmos, e como resultado, refletir negativamente nos dados finais.

2.5 CONSIDERAÇÕES FINAIS

Este capítulo apresentou o referencial teórico sobre a engenharia de software e duas de suas áreas, sendo elas, o processo de software e a manutenção de software, além de explicar sobre as estimativas de esforço.

[1] *N-grams* é um conceito encontrado em Processamento de Linguagem Natural (PNL), e pode significar uma sequência de N palavras. `https://blog.xrds.acm.org/2017/10/introduction-n-grams-need/`

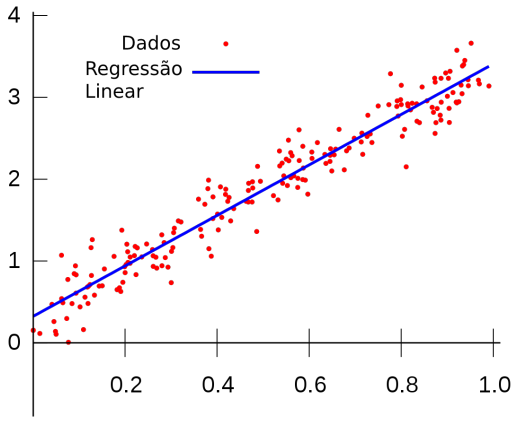

Figura 3: Diagrama de dispersão
Fonte: Montgomery, Peck e Vining (apud MIGUEL, 2016)

A partir deste estudo, foi possível entender os principais métodos de estimativa apresentados neste trabalho. O método de *Planning Poker*, o qual gera as estimativas por meio de um jogo de cartas, é utilizado pela equipe de desenvolvimento de software na empresa estudada. Já o método de similaridade textual implementado com o coeficiente de Sørensen-Dice e o método de análise de regressão linear, serão utilizados em conjunto pelo *framework* proposto nesse trabalho, para a geração de estimativa de esforço de forma automática. Por fim, este estudo foi fundamental para a compreensão dos diversos tópicos apresentados, além de auxiliar na criação do *framework* a ser apresentado no Capítulo 5.

3 MAPEAMENTO SISTEMÁTICO

Com o objetivo de identificar quais pesquisas estavam sendo realizadas na área de estimativa de esforço para manutenção de software, foi conduzido um mapeamento sistemático (MS), o qual visa identificar, interpretar e avaliar todas as pesquisas pertinentes à questão de pesquisa definida (KITCHENHAM, 2004). O MS foi realizado seguindo quatro etapas, sendo o primeiro passo a definição da questão de pesquisa (QP), apresentada a seguir:

- **QP:** Quais métodos, métricas ou ferramentas de estimativa de esforço têm sido aplicadas à manutenção de software?

A QP foi definida com o intuito de encontrar diferentes abordagens para a estimativa de esforço que fossem voltadas especificamente à manutenção de software. A partir da QP, deu-se início a segunda etapa, que seria a pesquisa dos estudos primários relacionados ao tema.

Para a realização da pesquisa, foi definida a *string* de busca, composta por termos em inglês das palavras chaves "estimativa de esforço" e "manutenção de software", resultando na seguinte *string* de busca: ***("effort estimation" AND "software maintenance")***. Posteriormente, quatro bases de dados foram selecionadas para a realização da busca dos estudos, sendo elas: *ACM Digital Library*, *IEEE Xplore*, *Scopus* e *Springer*.[1].

Após a execução da *string* de busca nas bases selecionadas, foram retornados 521 resultados. A partir desses resultados, teve início a terceira etapa do MS, que representa a triagem dos dados. Nesta etapa foram aplicados os critérios de inclusão e exclusão e, após a aplicação dos mesmos, sobraram 17 estudos, os quais foram selecionados para a realização da análise.

Dentre os 17 estudos selecionados, 13 apresentaram algum método e 11 estudos utilizaram métricas. Pôde-se observar que a métrica mais utilizada foi linhas de código

[1] `http://dl.acm.org/` | `http://ieeexplore.ieee.org/` | `http://scopus.com/` | `http://link.springer.com/`

(6 de 11 estudos), sendo usado tanto para medir o tamanho total do sistema, quanto identificar linhas alteradas, incluídas ou excluídas. Dentre os métodos, o mais usado é a análise de regressão linear (5 de 13 estudos), o que indica que os estudos preferem utilizar dados históricos para gerar novas estimativas. Quanto às ferramentas, observa-se uma grande escassez no âmbito de ajudar o usuário a estimar as atividades, pois entre os resultados, apenas um *framework* foi proposto (MIGUEL et al., 2016).

O mapeamento sistemático foi publicado em forma de artigo na Escola Regional de Engenharia de Software (ERES)[2] (VIESSELI; SILVA; SANTOS, 2020). O artigo completo e com maiores detalhes sobre o MS pode ser analisado no Apêndice A.

[2] `https://eres2020.github.io/`

4 FRAMEWORK

Um *framework* pode ser descrito como uma estrutura de base utilizada para implementar funções e componentes pré-definidos. Para o desenvolvimento da solução para este trabalho, optou-se pelo desenvolvimento de um *framework*. Nele, é possível realizar diversos cadastros (usuários, equipes, atividades), além do cálculo de estimativas de esforço para novas atividades. Como pontos de variabilidade e extensão do *framework* proposto, tem-se a escolha dos métodos para cálculo de estimativas, assim como a integração com bancos de atividades em diferentes ferramentas de gestão.

Portanto, no presente capítulo será descrito o processo de desenvolvimento do *framework* EstimAi, desde a seleção dos métodos de estimativa de esforço até a construção do *framework*. Na Seção 4.1 é descrito o processo de seleção e desenvolvimento do método para estimativa de esforço e na Seção 4.2 é discutida a seleção da tecnologia e o processo de desenvolvimento do *framework*. Por fim, na Seção 4.3 é apresentado o *framework* EstimAi pronto, com suas respectivas telas desenvolvidas e seus respectivos funcionamentos.

4.1 MÉTODOS PARA ESTIMATIVA DE ESFORÇO

Nesta seção será descrito o processo de seleção dos métodos que irão gerar a estimativa de esforço, além do desenvolvimento e funcionamento do método proposto para este trabalho.

4.1.1 SELEÇÃO DE MÉTODOS PARA ESTIMATIVA DE ESFORÇO

Perante o mapeamento sistemático, foi possível observar diferentes métodos e métricas utilizados para a geração de estimativa de esforço, como por exemplo, rede neural, análise de regressão linear, COCOMO, pontos de função, entre outros. Porém foi possível identificar que a maioria dos métodos reportados optou por fazer uma análise de dados históricos. Com isso um dos métodos escolhidos para gerar a estimativa neste trabalho foi a análise de regressão linear.

A partir do contexto da empresa, foram definidos alguns critérios de como o método deveria comportar-se. O método (i) deveria utilizar dados da base histórica para gerar a estimativa; (ii) teria que permitir a geração separada das estimativas de desenvolvimento e de teste das atividades; e (iii) mediria o esforço em horas. Esses critérios foram criados a partir do processo da empresa para facilitar a aplicação do estudo de caso, mas nada impede que outras empresas também utilizem o método proposto. Os critérios de estimar desenvolvimento e teste separados e gerar a estimativa em horas podem ser facilmentes alterados, de acordo com as características do processo de cada empresa.

Com os critérios definidos, foram selecionados dois métodos que vão gerar a estimativa de esforço. Inicialmente, será utilizada a base de dados histórica para encontrar atividades semelhantes à nova atividade a ser estimada. Para encontrar esse nível de semelhança entre as atividades, será utilizado o texto descritivo dos requisitos para aplicar o método de similaridade textual, que também foi usado pelo artigo Miguel et al. (2016). A partir desses dados, será utilizado o método de análise de regressão linear, que consiste em uma equação matemática capaz de gerar uma estimativa para a nova atividade, por meio das informações de nível de semelhança das atividades e o tempo real utilizado para a conclusão das mesmas.

Os dois métodos escolhidos, a similaridade textual e análise de regressão linear, complementam-se para que possam gerar a estimativa de esfoço. Apesar de trabalharem juntos, cada método possui suas próprias definições e características que podem ser analisadas no Capítulo 2.

Por fim, para auxiliar o desenvolvimento do método de estimativa de esforço, foram utilizadas *Application Programming Interfaces* (Interfaces de Programação de Aplicações - API), que possuem rotinas prontas para realizar determinadas ações, como a similaridade textual e análise de regressão linear.

Portanto, segundo o método proposto para geração das estimativas, após a busca por atividades antigas, é realizada uma comparação a partir da similaridade textual, para encontrar quais atividades destas antigas são mais similares à nova atividade informada. Para isso foi utilizada uma API denominada *string-similarity*, a qual é capaz de encontrar a semelhança entre duas *strings* e utiliza como base o coeficiente de Sørensen-Dice (STRING-SIMILARITY, 2021). O resultado do coeficiente de similaridade calculado pela API, pode variar entre 0 e 1, no qual 0 significa que os textos não são similares e 1 significa que são extamentes iguais.

Logo após a identificação das atividades semelhantes, é extraída a informação

referente ao real tempo utilizado para a conclusão das mesmas e, a partir desses dados, é realizada a análise de regressão linear para gerar a estimativa de esforço. Para a execução desta parte, foi utilizada a API *simple-statistics*, a qual possui vários cálculos estatísticos, incluindo o de regressão linear utilizado neste trabalho. A função de regressão linear da API é um algoritmo capaz de encontrar uma linha entre um conjunto de coordenadas, sendo essas coordenadas o nível de similaridade das atividades no eixo X e suas respectivas horas reais utilizadas no eixo Y. A partir dessa linha e do valor de similaridade máxima informado no eixo X, é possível calcular o valor da estimativa de esforço da nova atividade no eixo Y (SIMPLE-STATISTICS, 2021).

4.1.2 DESENVOLVIMENTO DO MÉTODO

O processo do método está representado na Figura 4, que inicia na atividade 1 com o usuário fornecendo informações referentes à nova atividade, tais como o título, a descrição detalhada sobre o defeito e o número correspondente da atividade na ferramenta utilizada para gestão, por exemplo, o Jazz. Com os dados informados e salvos, o usuário irá pressionar o botão para que se dê início a geração de estimativa. A partir dessa ação, a ferramenta irá extrair as informações da atividade (atividade 2) e buscar no banco de dados as atividades já finalizadas e similares à nova (atividade 3); nesse momento, será utilizado o método de similaridade textual.

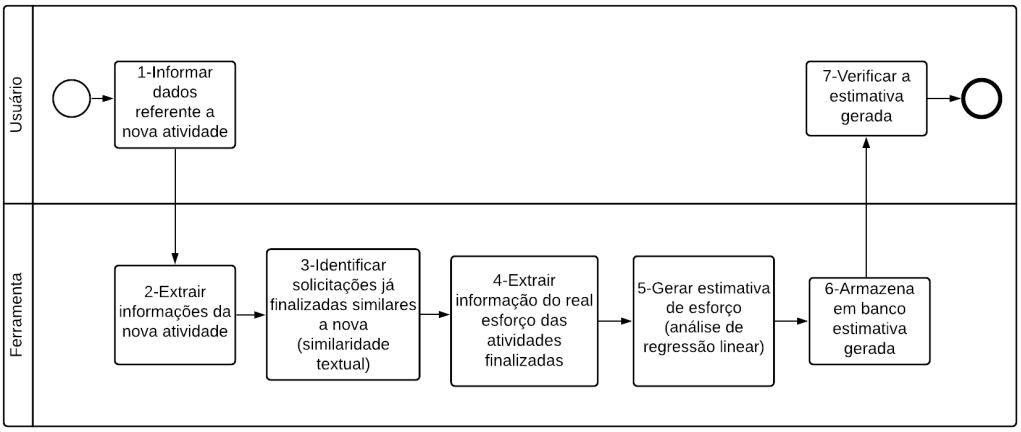

Figura 4: Fluxo de atividades para o método de estimativa
Fonte: Autoria própria

Após encontrar as atividades similares, a ferramenta irá extrair o real esforço delas (atividade 4) e irá utilizar esse valor extraído para gerar as estimativas a partir da análise de regressão linear (atividade 5). O método de regressão linear irá considerar as

informações referentes ao nível de similaridade das atividades e as horas reais utilizadas, e a partir desses dados cruzados será possível gerar a estimativa de esforço para a nova atividade. Com a estimativa gerada, a mesma é gravada em banco (atividade 6) e por fim na atividade 7, o usuário final verifica a estimativa de esforço gerada.

4.2 DESENVOLVIMENTO DO *FRAMEWORK* ESTIMAI

Na presente seção serão descritos com mais detalhes a seleção das tecnologias que serão utilizadas para o desenvolvimento do framework, além do processo de desenvolvimento do mesmo.

4.2.1 SELEÇÃO DA TECNOLOGIA PARA O *FRAMEWORK*

Considerando os resultados do mapeamento sistemático, observou-se também uma grande lacuna no estado da prática de ferramentas, *framework*, ou sistemas, que pudessem ajudar o usuário na geração das estimativas. Com isso, o presente trabalho optou por desenvolver um *framework Web* que permitiria ao usuário a geração facilitada da estimativa de esforço de suas atividades, sem a necessidade de outras ferramentas de análise complementares. Além disso, o *framework* foi desenvolvido de maneira a englobar não apenas as características da metodologia ágil *Scrum*, como planejamento e encerramento das *Sprints*, como também foi pensado no ambiente de múltiplas equipes, assim permitindo que cada equipe seja cadastrada e tenha suas próprias atividades cadastradas e estimadas.

Para o desenvolvimento do *framework Web* denominado EstimAi, foi selecionada a linguagem JavaScript[1], uma linguagem de programação interpretada e baseada em scripts. A mesma foi escolhida por ser uma das linguagens mais populares[2] e com isso possuir uma grande comunidade na *internet*, com vários tutoriais e vídeos explicando seu funcionamento, assim facilitando o aprendizado. Outro motivo para a seleção do JavaScript são as várias aplicações e tecnologias baseadas na linguagem que facilitam o desenvolvimento do *framework*, incluindo os métodos de similaridade textual e regressão linear.

Portanto, para apoiar o desenvolvimento do *framework* na linguagem JavaScript, foram utilizadas algumas tecnologias, as quais podemos citar:

- **React:** é uma biblioteca JavaScript que facilita a criação de interfaces para o usuário

[1] `https://www.javascript.com/`
[2] `https://insights.stackoverflow.com/survey/2020#most-popular-technologies`

final, sendo assim utilizada para o desenvolvimento do *front-end* do *framework* EstimAi (REACT, 2021).

- **Node.js:** pode ser descrito como um ambiente de execução javaScript e foi utilizado para o desenvolvimento do *back-end* do *framework*, sendo o responsável pelos tratamentos dos dados, validações, além de gerar as estimativas (NODE.JS, 2021).
- **PostgreSQL:** é um banco de dados relacional que usa a linguagem SQL como base e foi o responsável por armazenar todos os dados gerados e utilizados pelo EstimAi (POSTGRESQL, 2021).
- **Knex.js:** é uma API para realizar conexão com bancos relacionais, além de facilitar a criação de consultas através da linguagem JavaScript. O Knex.js foi utilizado no *back-end* para conexão e consultas com o banco PostgreSQL (KNEX.JS, 2021).

Para a geração de estimativas de esforço, o método proposto executa uma busca por atividades antigas, a qual é realizada em uma base de dados da empresa: um banco de dados DB2. Desenvolvido pela IBM, o banco de dados DB2 é relacional e além de possuir alta escalabilidade e velocidade, ele também pode ser executado em várias plataformas e aceita diversas linguagens de programação (DATABASE, 2021). A fim de facilitar a conexão do *framework* com o banco de dados DB2, foi usada uma API denominada *ibm_ db*, a qual foi desenvolvida para ser aplicada no Node.js e utilizar conexões assíncronas e síncronas (SMITH; VERWEIRE; IBM, 2010).

4.2.2 PROCESSO DE DESENVOLVIMENTO

Com o propósito de iniciar a arquitetura do sistema, foram identificados os requisitos funcionais (RFs) que seriam utilizados no desenvolvimento do *framework* EstimAi, os quais podem ser observados na Tabela 1.

Tabela 1: Requisitos funcionais do *framework* EstimAi

RFs	Descrição
RF01	O usuário realiza o login para ter acesso ao sistema, e o mesmo ocorre por meio de um e-mail e senha.
RF02	O usuário pode alterar seu perfil, alterando as informações como nome, e-mail, celular e senha.

Continuação na próxima página

Tabela 1 – continuação da página anterior.

RFs	Descrição
RF03	O usuário pode cadastrar equipes, sendo solicitado apenas o nome da equipe. A empresa é composta por múltiplas equipes, o cadastro de equipes serve para gerar uma melhor organização e separação das atividades.
RF04	O usuário pode gerenciar as equipes já cadastradas, podendo alterá-las ou inativá-las.
RF05	O usuário pode cadastrar uma *Sprint*, sendo as informações necessárias o nome dado a *Sprint* e a qual equipe ela pertence. A *Sprint* pode significar um ciclo de desenvolvimento, ou o tempo necessário para entrega de uma release, para modelos de processo diferentes do *Scrum*.
RF06	O usuário pode gerenciar as *Sprints*, alterando suas informações já cadastradas ou visualizando todas as atividades relacionadas à *Sprint*.
RF07	O usuário pode cadastrar atividades, sendo solicitadas informações como título, descrição, estimativas da atividade (tanto de desenvolvimento como de teste), o número da atividade correspondente no outro sistema de gestão das atividades, e também a *Sprint* na qual a atividade será realizada.
RF08	O usuário pode gerenciar as atividades, podendo alterar as informações já cadastradas ou inserindo novas, como o tempo real utilizado de desenvolvimento e teste, para a conclusão da atividade.
RF09	O usuário pode encerrar a *Sprint*, assim podendo visualizar as diferenças entre as estimativas e o tempo real utilizado nas atividades.
RF10	O administrador é o único que pode realizar cadastro de novos usuários, assim não tendo a possibilidade de pessoas fora da empresa tentarem se cadastrar.
RF11	O administrador pode alterar outros usuários para administrador para que também tenham controle sobre os demais usuários ou inativá-los para que não possam mais acessar o sistema.
RF12	O *framework* pode gerar a estimativa, a partir das informações extraídas da nova atividade cadastrada e do método de estimativa proposto.
RF13	O *framework* pode gerar gráficos de estimativa *versus* real utilizado, a partir das informações extraídas das atividades da *Sprint* selecionada, assim facilitando a visualização das diferenças entre os mesmos.

Fonte: Autoria própria

Com o objetivo de melhorar a compreensão sobre as funcionalidades do *framework*, foi elaborado o diagrama de caso de uso, o qual pode ser observado na Figura 5. No diagrama de caso de uso é possível analisar três atores, o Usuário, o Administrador que herda as mesmas funções do Usuário e possui mais dois casos de uso específicos para ele, e por último a própria Ferramenta.

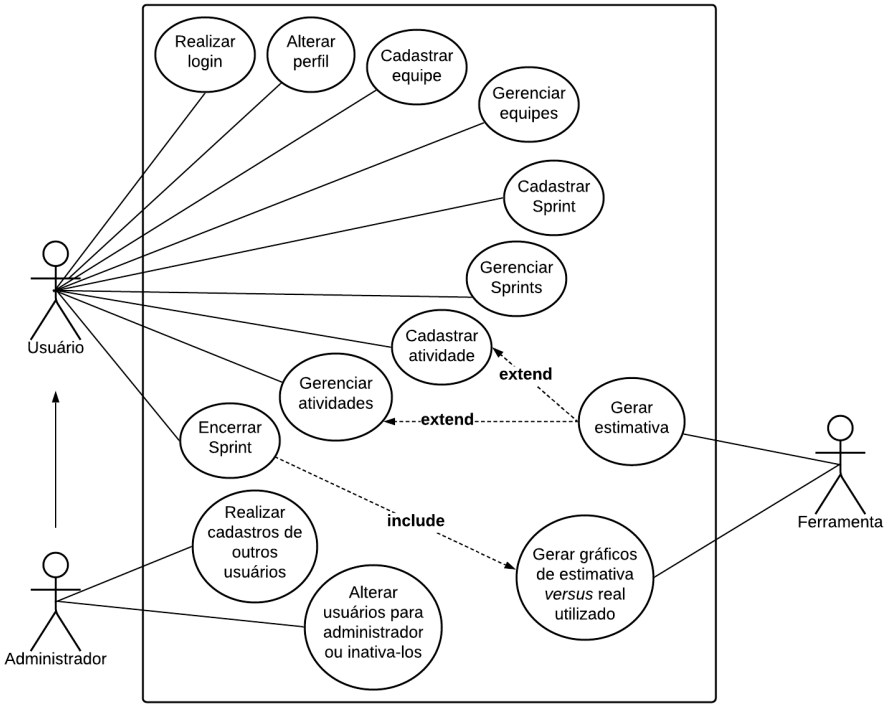

Figura 5: Diagrama de caso de uso do *framework* EstimAi
Fonte: Autoria própria

Além disso, podemos observar que o diagrama possui ao todo 13 casos de uso que fazem referência direta aos requisitos funcionais e que cada caso de uso está ligado ao seu ator correspondente, além de alguns casos de uso estarem relacionados a outro caso de uso. Desse relacionamento entre casos de uso, temos o primeiro tipo que é opcional (*extend*), como por exemplo, ao "cadastrar atividade" é opcional, podendo ou não "gerar estimativa". E temos o relacionamento obrigatório dos casos de uso (*include*), que temos como exemplo o caso de uso "encerrar sprint" onde é obrigatório o ator Ferramenta executar o caso de uso "gerar gráficos de estimativa *versus* real utilizado".

Após a elaboração do diagrama de caso de uso, foi criado um protótipo de baixa fidelidade da tela de cadastro de atividades do *framework* EstimAi, com o próposito de reduzir incertezas quanto a campos necessários e a usabilidade do mesmo. O protótipo pode ser visualizado na Figura 6.

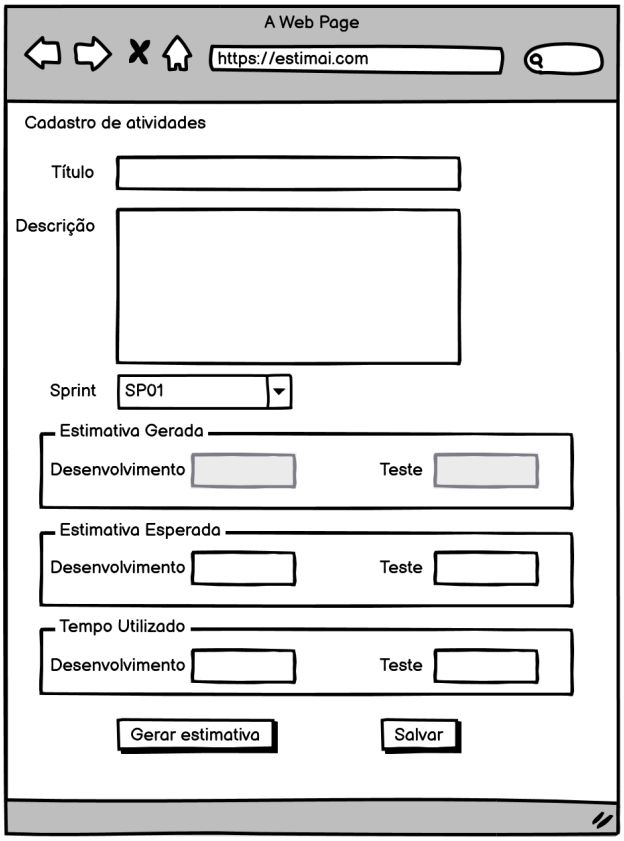

Figura 6: Protótipo da tela de cadastro de atividades do EstimAi
Fonte: Autoria própria

No protótipo em questão, é possível visualizar os campos de título e descrição, que serão responsáveis por armazenar uma descrição resumida da atividade e uma descrição completa, respectivamente. O campo de *Sprint*, é colocado como uma lista, onde o usuário teria a opção de escolher a qual *Sprint* a atividade pertence. Por fim, existem os campos de estimativas e tempo utilizado, sendo os mesmos divididos em três grupos, (i) estimativa gerada, que seria referente à estimativa gerada pelo *framework*; (ii) estimativa esperada, seria a estimativa que o usuário definiu para a atividade; e (iii) tempo utilizado, que referencia o real tempo utilizado para a conclusão da atividade. Os três grupos possuem campos para informar as horas de desenvolvimento e as horas de teste da atividade. Ao final da página é possível visualizar os dois botões, o primeiro de geração da estimativa, o qual irá acionar o processo de geração da estimativa para a atividade descrita, e o segundo é o botão salvar, que irá armazenar em banco os dados informados.

Com o objetivo de facilitar a visualização de como serão armazenadas as informações referente ao *framework* EstimAi, foi produzida uma representação do modelo relacional do banco de dados, apresentado na Figura 7.

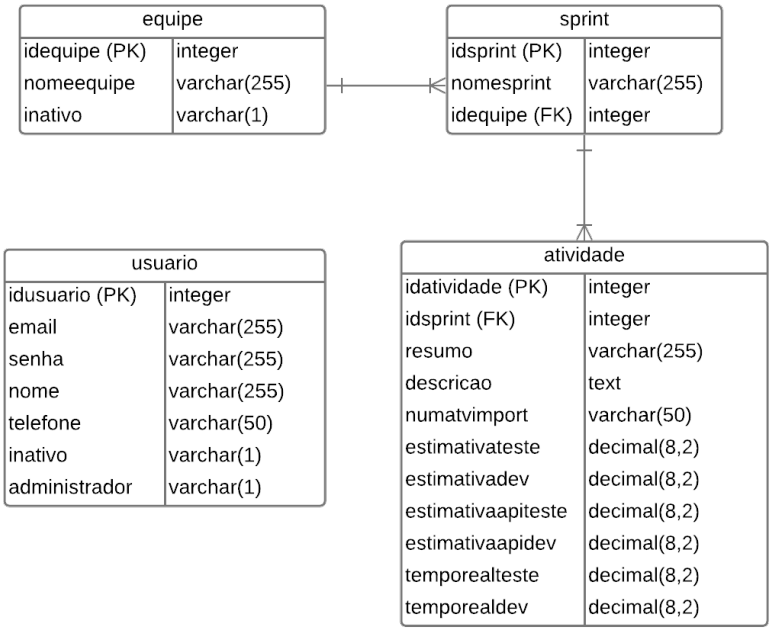

Figura 7: Estrutura do banco de dados do EstimAi
Fonte: Autoria própria

Na estrutura apresentada do banco de dados, é possível observar que o sistema será composto por quatro tabelas. A primeira tabela "usuario" é referente às informações dos usuários cadastrados, como dados pessoais, e o email e senha que serão usados para realizar o login no sistema. A segunda tabela "equipe" armazena o nome da equipe correspondente a uma equipe real dentro da empresa. Já a terceira tabela "sprint" possui o nome da *Sprint* e a qual equipe pertence essa *Sprint* criada. A última tabela "atividade" mantém todas as informações referentes à atividade criada, como resumo, descrição, o número da atividade correspondente na ferramenta de gestão (*e.g.*, Jazz) que seria o campo "numatvimport". Além de possuir outros campos como, as estimativas, o tempo real utilizado, e realizar a ligação da atividade a uma *Sprint* já existente.

Já no método de estimativa de esforço, para a identificação das atividades similares é utilizado o banco de dados da empresa que contém toda a base histórica das atividades já concluídas. O banco de dados é relacional e utiliza a arquitetura DB2. Na Figura 8 é representado o modelo relacional do banco de dados, no qual é possível analisar apenas as tabelas e colunas utilizadas para a identificação das atividades. As demais informações foram ocultadas por não serem relevantes para o presente trabalho.

O diagrama apresenta três tabelas principais, a primeira "work_item" armazena as informações pertinentes às atividades, como "resumo" que representa o título da atividade,

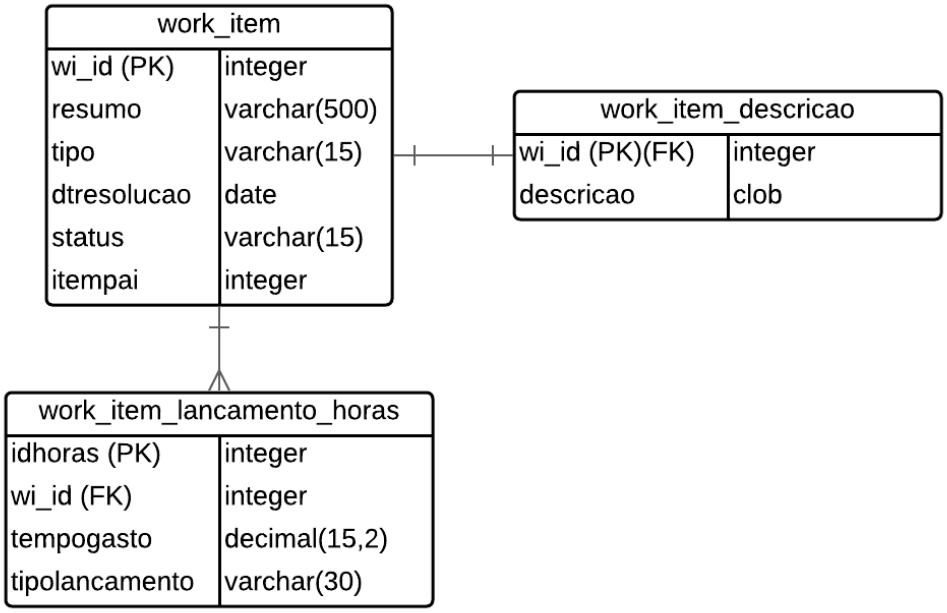

Figura 8: Estrutura do banco de dados da base de atividades da empresa
Fonte: Autoria própria

o "tipo" identifica se é defeito ou estória do usuário, a "dtresolucao" a data que foi terminada a atividade, o "status" armazena se a atividade está pronta, cancelada ou em andamento, e por último o "itempai" que é uma referência à mesma tabela e identifica o item ao qual a atividade atual está ligada. Um exemplo de como as atividades se relacionam, um defeito é considerado como uma atividade pai e é armazenado como uma linha na tabela "work_item". A partir desse defeito, existem várias atividades filhas, sendo que cada atividade é uma linha diferente na tabela referenciando o item pai. São nessas atividades filhas que estão relacionadas as horas gastas em desenvolvimento e em teste para este defeito.

A tabela "work_item_descricao" armazena as descrições detalhadas da atividade e na tabela "work_item_lancamento_horas" temos os dados como o "tipolancamento" que identifica se é horas de desenvolvimento ou teste, e a coluna "tempogasto" que identifica as horas gastas para a conclusão da atividade.

Para executar a busca por atividades encerradas, é realizada a conexão com o próprio banco da empresa. Nessa busca, são selecionadas apenas as atividades que são defeitos e extraídas suas descrições para comparação posterior e depois são buscados os itens filhos correspondentes a essa atividade, para identificar as horas gastas de desenvolvimento e teste da atividade. Vale ressaltar que este processo descrito e a estrutura do banco

apresentado são baseados em uma empresa, porém outras empresas podem utilizar o método proposto. Nesses casos, o líder interessado tem a opção de criar um banco DB2 com estrutura semelhante ao apresentado na Figura 8, ou atualizar a busca de informações conforme a estrutura de seu próprio banco, sendo essas informações: a descrição, status, data do término da atividade e horas gastas.

4.3 O *FRAMEWORK* ESTIMAI

A partir das funcionalidades propostas, foi desenvolvido o *framework* denominado EstimAi, o qual tem como objetivo gerar as estimativas para a atividade descrita pelo usuário. Seguindo o processo da ferramenta, inicialmente o usuário irá realizar o login conforme o e-mail e senha passados pela empresa. O usuário não consegue realizar seu próprio cadastro, pois para evitar que pessoas de fora da empresa tenham acesso, o cadastro dos usuários é de responsabilidade dos administradores do sistema. Após o login, será apresentada ao usuário a tela inicial, a qual apresenta as ações disponíveis para serem realizadas. A tela inicial é apresentada na Figura 9.

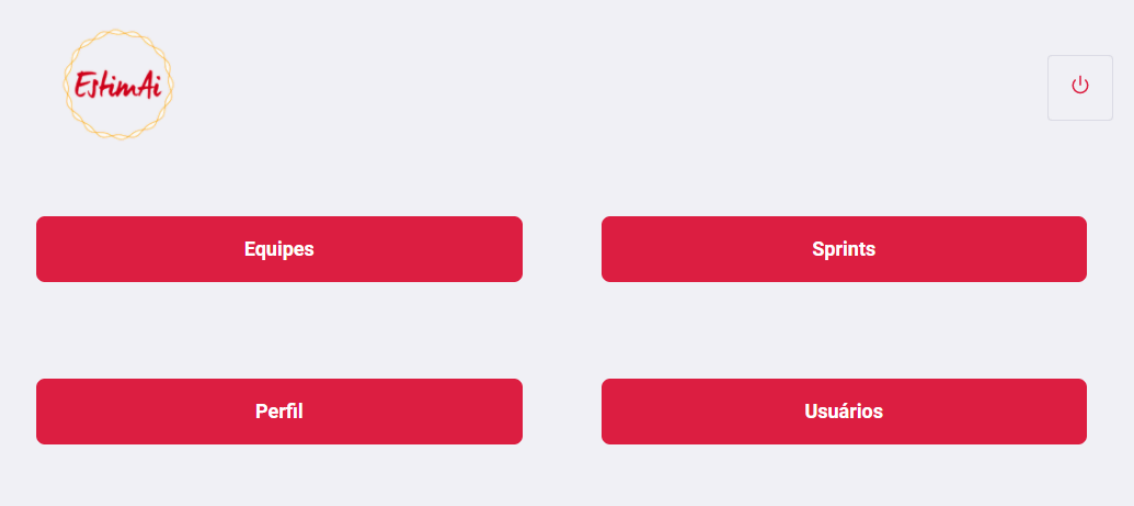

Figura 9: Tela inicial (após login) do *framework* EstimAi
Fonte: Autoria própria

Após o primeiro login, o usuário deve realizar alguns procedimentos, por exemplo a alteração da senha que foi passada pela empresa, a qual pode ser realizada através da opção "Perfil" da tela inicial. Na tela de perfil, é possível alterar algumas informações pessoais, como nome, e-mail, celular e senha. A tela de edição de perfil é apresentada na Figura 10.

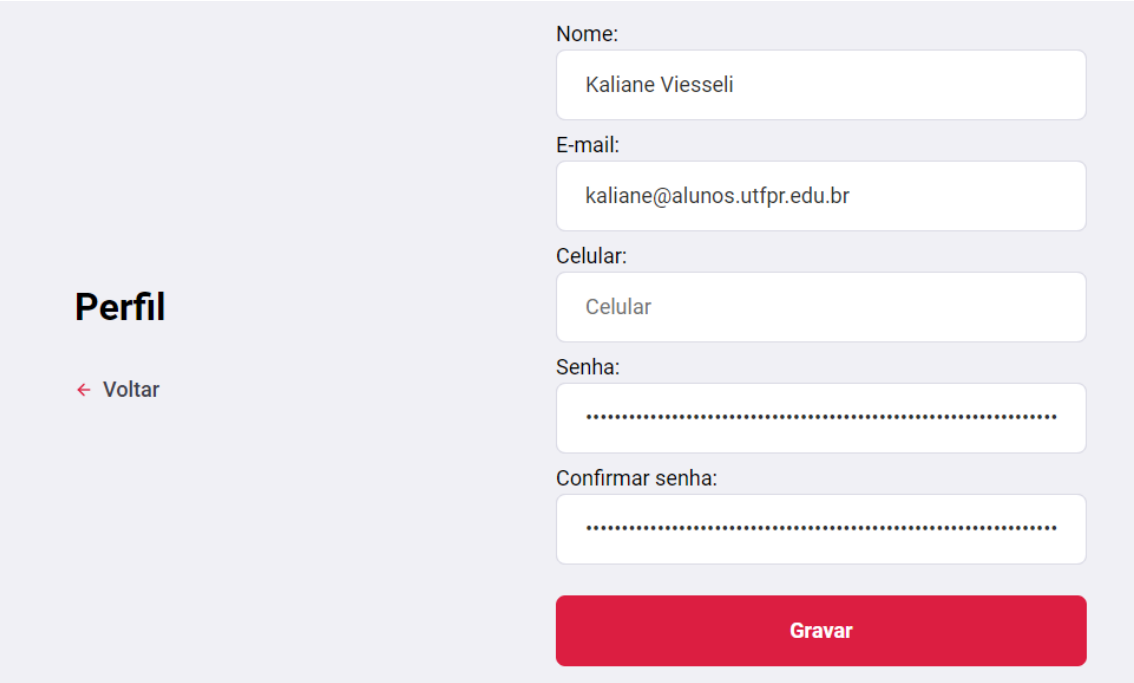

Figura 10: Tela de alteração do perfil
Fonte: Autoria própria

Caso o usuário esteja cadastrado como administrador do sistema, será apresentada na tela inicial a opção "Usuários", na qual é possível visualizar todos os usuários cadastrados no sistema. O usuário com o poder de administrador também pode modificar informações dos demais usuários, podendo inativá-los ou torná-los administradores, além de ter a possibilidade de cadastrar um novo usuário. A tela de usuários é exibida na Figura 11.

Figura 11: Tela de usuários para o perfil de Administrador
Fonte: Autoria própria

Caso o usuário administrador escolha a opção de adicionar um novo usuário, será apresentada a tela de cadastro conforme a Figura 12, na qual deve-se informar um e-mail e senha, que devem ser repassados para o novo usuário, assim o mesmo poderá realizar o login no sistema.

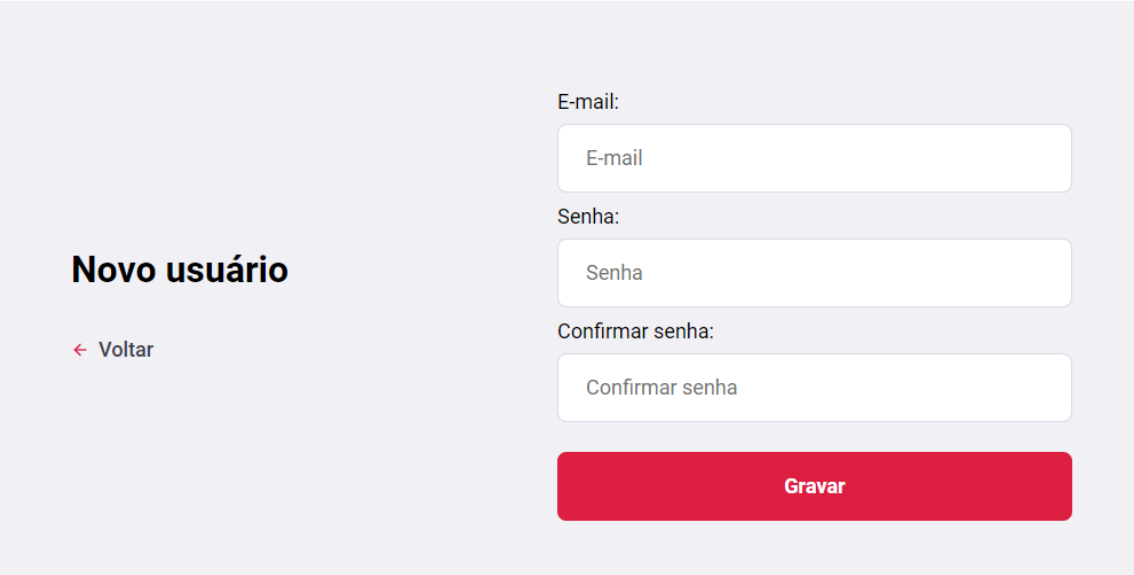

Figura 12: Tela de cadastro de usuário
Fonte: Autoria própria

Antes de iniciar a geração de estimativas, o usuário deve realizar alguns cadastros importantes, como o cadastro de equipe. Por meio da opção "Equipes" da tela inicial, o usuário pode visualizar todas as equipes cadastradas no sistema, além de poder inativá-las, editá-las ou adicionar uma nova equipe. A tela de equipes é apresentada na Figura 13.

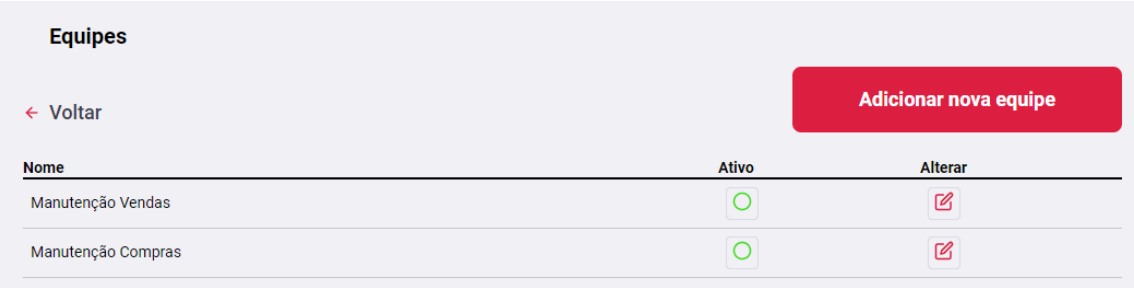

Figura 13: Tela de equipes
Fonte: Autoria própria

Ao selecionar a opção de adicionar nova equipe, o usuário será direcionado para a tela de cadastro, onde deve informar apenas o nome da equipe, conforme apresentado na Figura 14.

Após o cadastro de equipes, é necessário realizar o cadastro de *Sprints* que pode ser acessado por meio da opção "Sprints" da tela inicial. Ao acessar a tela representada na Figura 15, é possível visualizar as *Sprints* cadastradas e a qual equipe elas pertencem. Além de ter as opções de alterar, detalhar, encerrar e adicionar uma nova *Sprint*.

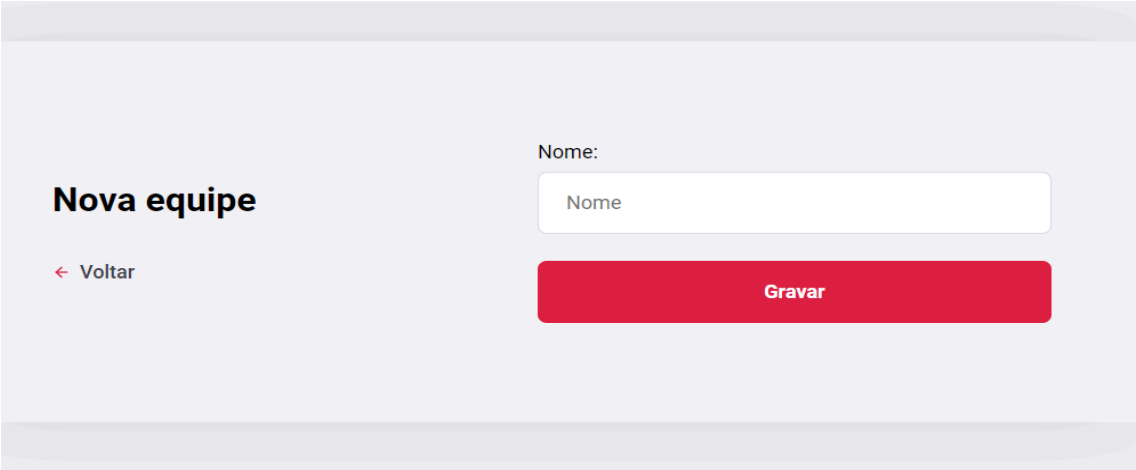

Figura 14: Tela de cadastro de equipe
Fonte: Autoria própria

Figura 15: Tela de Sprints
Fonte: Autoria própria

Caso o usuário escolha a opção de adicionar nova *Sprint*, o mesmo será conduzido para a tela de cadastro apresentada na Figura 16, na qual deve ser informada o nome da *Sprint* e selecionar a qual equipe a mesma pertence, a partir das equipes já cadastradas. Com esses cadastros prontos, facilita a visualização das atividades pelo usuário, pois as atividades estarão separadas por equipe e *Sprint*.

A partir da tela de *Sprints*, o usuário pode escolher a opção de detalhar uma *Sprint* em específico. Será apresentada a tela de atividades, que pode ser observada na Figura 17. Nessa tela é possível visualizar todas as atividades cadastradas para a *Sprint* escolhida com suas respectivas estimativas, além de ter as opções de alterar as atividades ou adicionar uma nova.

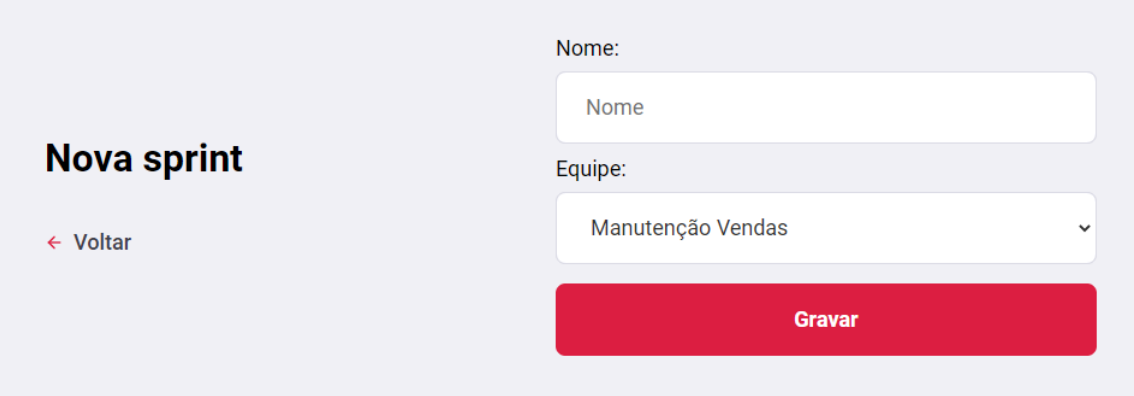

Figura 16: Tela de cadastro de Sprint

Fonte: Autoria própria

Figura 17: Tela de atividades da Sprint

Fonte: Autoria própria

Ao escolher alterar ou cadastrar uma nova atividade, o usuário será direcionado para a tela de cadastro de atividade, apresentada nas Figuras 18 e 19. O cadastro inicia com um campo "ID" que corresponde ao número identificador da atividade no banco de dados do EstimAi. Já o campo "ID importado" é referente ao número da atividade na ferramenta de gestão utilizada pela empresa, exemplo Jazz, assim tendo como objetivo armazenar essa ligação entre as duas ferramentas. Outras informações solicitadas são o título e a descrição da atividade, sendo este último o campo utilizado para a geração das estimativas.

Em seguida, é necessário informar as estimativas e tempo utilizado na atividade, sendo as mesmas divididas em três grupos principais: (I) Estimativa gerada, o qual irá apresentar as estimativas geradas pelo *framework* tanto para desenvolvimento quanto para teste; (II) Estimativa esperada, que irá armazenar as estimativas de desenvolvimento e teste que foram definidas pela equipe; e (III) Tempo real utilizado, que também possui

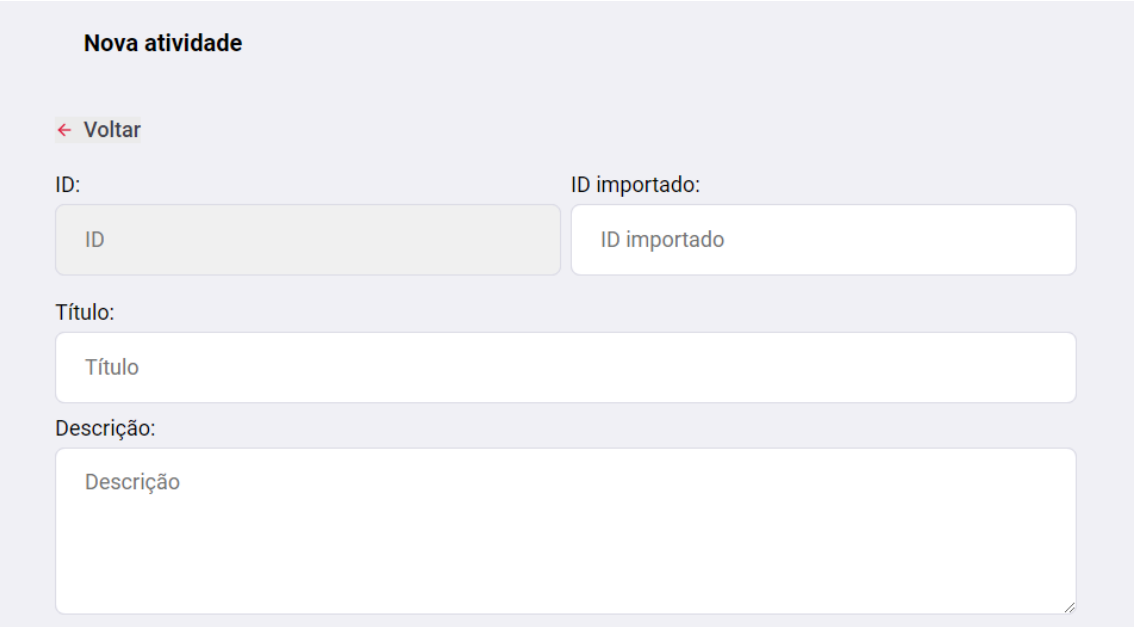

Figura 18: Tela de cadastro de atividade - Parte 1
Fonte: Autoria própria

os campos referentes ao desenvolvimento e teste, no qual o usuário irá informar o tempo utilizado para a conclusão da atividade. Por fim, a tela possui dois botões, sendo que o primeiro é responsável por gerar as estimativas e o segundo por gravar as informações no banco de dados.

Com todas as atividades cadastradas, suas respectivas estimativas e tempo utilizado, é possível encerrar uma *Sprint* em específico, a partir da tela de *Sprints*. A tela de encerramento irá mostrar os gráficos, comparando as estimativas da equipe, as estimativas geradas e o tempo utilizado de cada atividade, conforme demonstrado na Figura 20.

A partir do *framework* EstimAi apresentado, é possível observar que o mesmo foi adaptado para trabalhar em um ambiente multi-equipe, com divisões das atividades por *Sprint* e equipe, assim, simplificando a análise nos gráficos gerados. Porém observou-se alguns pontos a serem melhorados para facilitar a análise dos gráficos.

O primeiro ponto observado é que nem todas as atividades são terminadas até o final da *Sprint* e, por esse motivo, os gráficos estão apresentando atividades não encerradas e sem horas lançadas. Uma sugestão de correção é criar um status para a atividade, com o objetivo de identificar se a atividade está pronta, em andamento ou cancelada, assim podendo filtrar apenas as atividades prontas para gerar os gráficos.

Figura 19: Tela de cadastro de atividade - Parte 2

Fonte: Autoria própria

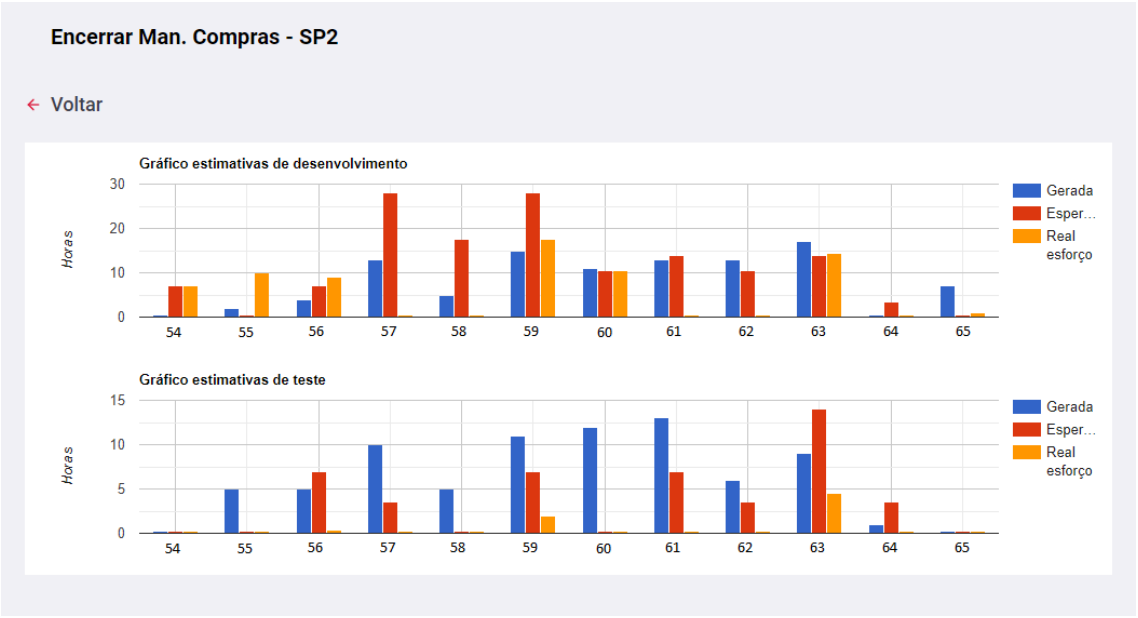

Figura 20: Tela de encerramento da Sprint

Fonte: Autoria própria

O segundo ponto identificado é que na ferramenta de gestão da empresa, muitos defeitos não tinham atividade de teste manual criada. Isso gerou um problema: o *framework*

estava estimando todas as atividades com horas de desenvolvimento e teste; em alguns casos, testes não foram realizados. Portanto, criou-se uma divergência entre a estimativa gerada e o tempo utilizado para a atividade. Como solução, seria necessário um campo que o usuário pudesse informar que não seria realizado teste na atividade, assim não necessitando do *framework* gerar a estimativa de teste.

O *framework* EstimAi foi disponibilizado em código aberto na plataforma GitHub[3] e pode ser acessado a partir do link `https://github.com/kaliane/EstimAi`.

[3]O GitHub é uma plataforma para hospedagem de códigos, assim facilitando o controle de versões e o gerenciamento de projetos. `https://github.com/`

5 ESTUDO DE CASO

O objetivo deste trabalho é auxiliar e melhorar as estimativas de esforço por meio de um *framework* no qual foram aplicados métodos já existentes de estimativa na área da manutenção de software. Para avaliar a solução proposta, foi realizado um estudo de caso (WOHLIN et al., 2012) por seu caráter observacional; ou seja, atividades e real esforço são cadastradas pelos desenvolvedores, enquanto as estimativas do *framework* são geradas automaticamente, sem intervenção nas atividades do desenvolvedor. Neste estudo, foram analisadas a precisão das estimativas geradas pelo *framework* EstimAi e a percepção de utilidade de acordo com os usuários finais do *framework*. Consequentemente, foram utilizados os métodos de observação indireta e entrevista.

A empresa na qual este estudo de caso foi conduzido conta com duas fábricas de desenvolvimento de software. Porém, o estudo foi direcionado apenas à fábrica de manutenção, que possui 10 equipes voltadas à manutenção e melhoria do ERP principal da empresa. A fábrica em questão trabalha com a metodologia ágil *Scrum*, e por esse motivo divide seus trabalhos e entregas em ciclos (*Sprints*) de 15 dias por definição da empresa.

5.1 ESCOLHA DAS ATIVIDADES

O estudo de caso consistiu em coletar as atividades de manutenção da fábrica de manutenção da empresa, incluindo descrição da atividade, estimativa do desenvolvedor e testador, e tempo dispendido na tarefa. Adicionalmente, o *framework* gerou as estimativas para cada atividade. Junto com a gerência da equipe, foi definido que o *framework* iria ser implantado em duas *Sprints*: a primeira teve início no dia 29 de março e encerrou no dia 09 de abril de 2021, e a segunda *Sprint* com início no dia 12 de abril e encerramento no dia 23 de abril de 2021.

Além da *Sprint*, também foi estabelecido que a avaliação seria realizada em duas equipes de manutenção, sendo elas, manutenção de compras composta por 7 integrantes e manutenção de vendas formada por 5 pessoas. Apenas os líderes das equipes escolhidas

estavam cientes e de acordo com a análise que estaria ocorrendo em suas *Sprints*. Os desenvolvedores não estavam cientes de que as atividades cadastradas estavam sendo monitoradas para pesquisa. Essa informação poderia enviesar o preenchimento dos dados na ferramenta, assim como não refletir a situação atual da empresa.

Nesta fase inicial, o estudo enfrentou um primeiro obstáculo, referente ao cadastro das atividades. Ao analisar a base de dados da empresa, foi observado que haviam dois tipos de atividades para serem estimadas: (i) as voltadas a melhorias no sistema que são as manutenções perfectivas, e (ii) as atividades para correção de erros, as manutenções corretivas. Quanto às atividades de melhorias, observou-se que estas não eram separadas em atividades menores. Por exemplo, para uma melhoria em um módulo de açougue, em apenas uma atividade estava descrito todo o processo, desde o desenvolvimento da tela de lançamento, até a gravação final dessas informações.

Dessa forma, a estimativa e o tempo para a conclusão da melhoria se torna muito alto se comparado a uma manutenção corretiva que é mais pontual. Consequentemente, seria observada uma grande distorção nas estimativas geradas pelo método. Por esse motivo, decidiu-se por não utilizar atividades de melhorias para avaliação da abordagem, e a avaliação do método de estimativa foi baseada apenas nas manutenções corretivas.

5.2 PROCEDIMENTO DE COLETA DE DADOS

Com o início de cada *Sprint*, as equipes realizaram a reunião de planejamento e estimaram todas as atividades na ferramenta de gestão. Essas informações são alimentadas ao *framework*, com as descrições e as estimativas realizadas pela equipe. Por último, foi executada a geração das estimativas utilizando o método proposto. No decorrer da *Sprint*, ocorreu o acompanhamento das atividades e repasse das horas utilizadas para o *framework*. Foram inseridas tanto as horas de desenvolvimento quanto as horas utilizadas para testes das atividades já encerradas.

Durante a etapa de análise das atividades antigas, o estudo enfrentou um segundo obstáculo, referente ao sistema de gestão. Após o desenvolvimento do *framework*, a empresa realizou a substituição da aplicação de gerenciamento de projetos, do IBM Jazz[1] para o Jira[2]. Por esse motivo, as atividades ficaram divididas em dois bancos de dados, sendo os dados antigos necessários para a realização das estimativas armazenados na ferramenta antiga Jazz e as atividades novas estão na ferramenta Jira. Com essa mudança,

[1] `https://jazz.net/`
[2] `https://www.atlassian.com/br/software/jira`

o *framework* conectado à ferramenta antiga Jazz consegue gerar as estimativas, porém não consegue coletar as informações referentes às novas atividades que devem ser estimadas.

Com essas mudanças teve-se um problema, no qual o usuário final do *framework* teria o trabalho de informar todos os dados referentes às atividades a serem estimadas, gerando assim um retrabalho, pois esses dados já foram informados no Jira. Contudo, para minimizar o retrabalho dos usuários finais no momento do estudo de caso, as atividades selecionadas no Jira para fazer parte da *Sprint* corrente, serão adicionadas manualmente pela própria autora no EstimAi com todos os seus dados correspondentes.

No encerramento da última *Sprint*, foram apresentados separadamente para as equipes analisadas, tanto o *framework* EstimAi quanto as comparações entres as duas estimativas e o real esforço utilizado. Com essa apresentação, as equipes puderam opinar durante a reunião e por meio de um formulário, apresentado no Apêndice B, sobre o que acharam do método de estimativa proposto e se usariam ou não a ferramenta.

5.2.1 MÉTRICAS

As métricas identificadas para coleta neste estudo estão descritas a seguir:

Precisão da estimativa gerada: Representa o nível de precisão das estimativas geradas pelo método proposto, ou seja, o quanto estão próximas ao real tempo utilizado para a conclusão das atividades. Para realizar o cálculo da precisão das estimativas, foram utilizados dois indicadores de precisão apontados como os mais utilizados de acordo com o Mapeamento Sistemático: *mean magnitude of relative error* (magnitude média do erro relativo em tradução livre, ou MMRE) e *percentage relative error deviation* (desvio de erro relativo percentual em tradução livre, ou PRED) (PORT; KORTE, 2008). As duas medidas são baseadas em um valor denominado *magnitude of relative error* (magnitude do erro relativo em tradução livre, ou MRE), o qual pode ser definido pela seguinte fórmula:

$$MRE_i = \frac{|Y_i - \hat{Y}_i|}{Y_i} \tag{2}$$

onde Y_i é o tempo real utilizado para a conclusão da atividade i e $\hat{Y}_i$ é o tempo estimado para a mesma. Alguns autores representam esse valor em porcentagem. Por exemplo, um MRE de 30% (ou $0,30$) significa que, para uma determinada atividade, o tempo foi sub- ou superestimado em 30% do tempo real utilizado. Já o indicador MMRE

representa a média das N atividades do MRE, assim sua fórmula pode ser definida como:

$$MMRE = \frac{1}{N}\sum_{i=1}^{N} MRE_i \tag{3}$$

Por último temos o indicador PRED(X) que utiliza a fração média de valores MRE que ficaram abaixo de um limiar definido como X, assim podemos definir a fórmula sendo:

$$PRED(X) = \frac{1}{N}\sum_{i=1}^{N} \left\{ \begin{array}{c} 1\, if\ MRE_i \ \leq\ X \\ 0\, otherwise \end{array} \right\} \tag{4}$$

No presente trabalho, foi seguida a definição proposta por Port e Korte (2008), na qual recomenda-se os valores de MMRE $\leq$ 0,25 e PRED(0,30) $\geq$ 0,75 para que a estimativa tenha um nível de precisão aceitável.

Percepção do usuário: Representa a percepção do usuário quanto ao *framework* EstimAi, o qual será medido por meio de um formulário aplicado, onde as respostas são baseadas na escala Likert (LIKERT, 1932) com valores variando de 1 a 6, sendo 1 discordo totalmente e 6 concordo totalmente. Para avaliar a percepção do usuário, será gerada uma média das respostas no questionário de avaliação (MRQ), e a partir desse cálculo será considerado que o valor de MRQ $>$ 3 indica que a percepção do usuário foi positiva quanto ao *framework*.

5.3 REFINAMENTO E ANÁLISE DOS DADOS

No total, foram cadastradas 86 atividades de manutenção durante as duas *Sprints* sob análise. Porém, para realizar os cálculos de precisão das estimativas, foram realizados alguns filtros. Inicialmente, foram excluídas todas as atividades que estavam com status diferente de "pronto", como por exemplo as canceladas ou em andamento, assim restando apenas 50 atividades. A partir dessas atividades, foi possível observar que nem todas realizavam teste manual. Por esse motivo, a comparação da estimativa gerada para essas atividades era imprecisa, pois as mesmas não tinham lançamento de horas utilizadas. Portanto, foram desconsiderados os testes dessas atividades para a realização da avaliação, resultando assim em apenas 24 atividades que serão analisadas as estimativas de testes.

Das 50 atividades de desenvolvimento, quatro atividades não tiveram horas lançadas, assim como duas de teste, apesar de constarem como concluídas. Essas atividades

também foram desconsideradas na avaliação, pois com a fórmula do MRE resultaria em divisão por zero. Com isso, obtivemos para a realização da avaliação, 46 atividades em que serão avaliadas as horas de desenvolvimento e 22 atividades referente às horas de teste, assim resultando em uma avaliação de 68 estimativas.

O procedimento de validação dos dados consiste em verificar quais foram os resultados obtidos pelas ações criadas anteriormente. Para a validação dos dados, foram utilizadas três fontes da seguinte maneira:

- Observações: de maneira indireta, observando o processo de inserção das atividades estimadas e esforço real utilizado pelos membros da equipe de desenvolvimento;
- Entrevistas: após o encerramento da última *Sprint*, foram realizadas entrevistas semi-estruturadas com as duas equipes, a fim de identificar estimativa ou real esforço cadastrados incorretamente;
- Questionário: foi aplicado um questionário para todos os membros das equipes, com o objetivo de avaliar qual a percepção do desenvolvedor quanto à estimativa gerada pelo *framework*.

Utilizando esse processo, foi desenvolvido o estudo de caso, cujos resultados serão apresentados no capítulo a seguir.

6 RESULTADOS

Neste Capítulo são apresentados os resultados obtidos neste estudo de caso. Para realizar a avaliação do método proposto de estimativa de esforço, foram coletadas as informações referentes às atividades realizadas pelas equipes durante duas *Sprints*. Foram analisadas a precisão das estimativas geradas pelo *framework* EstimAi, cujos resultados são apresentados na Seção 6.1, e a percepção dos desenvolvedores quanto ao *framework*, cujos resultados são apresentados na Seção 6.2. Por fim, na Seção 6.3 são discutidas as ameaças à validade do presente trabalho.

6.1 PRECISÃO DAS ESTIMATIVAS

Quanto à precisão das estimativas geradas, foram extraídas das atividades, a estimativa gerada e o real tempo utilizado para a conclusão da mesma. Os dados referentes às 68 estimativas avaliadas, podem ser observados na Tabela 2, a qual apresenta o número da atividade utilizada no EstimAi e suas respectivas estimativas, tanto as estimativas realizas pela equipe quanto as estimativas geradas pelo *framework*, além do tempo real utilizado, para desenvolvimento e teste.

Tabela 2: Estimativas e tempo utilizado para as atividades da empresa

	Desenvolvimento			**Teste**		
ID da atividade	**Estimativa equipe**	**Estimativa gerada**	**Tempo utilizado**	**Estimativa equipe**	**Estimativa gerada**	**Tempo utilizado**
2	7,0	11,0	13,5	3,5	4,0	3,5
3	7,0	2,0	3,5	7,0	3,0	7,0
5	10,5	14,0	10,5			
6	3,5	18,0	2,0			
7	3,5	0,0	3,5	3,5	5,0	2,0
8	3,5	0,0	1,0			
12	5,0	5,0	3,0	3,5	6,0	3,5

Continuação na próxima página

Tabela 2 – continuação da página anterior.

	Desenvolvimento			Teste		
ID da atividade	**Estimativa equipe**	**Estimativa gerada**	**Tempo utilizado**	**Estimativa equipe**	**Estimativa gerada**	**Tempo utilizado**
13	7,0	11,0	24,0	3,5	9,0	2,5
14	10,5	6,0	4,0	7,0	7,0	6,0
17	7,0	2,0	5,5	3,5	3,0	3,5
18	7,0	13,0	10,5	7,0	5,0	7,0
19	3,5	14,0	1,0	3,5	6,0	1,5
20	3,5	11,0	1,7			
21	0,0	0,0	7,0	0,0	0,0	4,0
23	0,0	4,0	2,5	0,0	0,0	1,0
24	10,5	2,0	3,0			
29	3,5	14,0	7,0			
30	10,5	7,0	10,5			
33	10,5	10,0	11,0			
34	7,0	10,0	4,0			
37	3,5	8,0	17,0	3,5	3,0	3,0
43	3,5	3,0	3,5			
44	3,5	12,0	3,0			
48	3,5	14,0	4,0	3,5	11,0	2,0
51	7,0	8,0	7,0			
54	7,0	0,0	7,0			
55	0,0	2,0	10,0			
56	7,0	4,0	9,0	7,0	5,0	0,3
59	28,0	15,0	17,5	7,0	11,0	2,0
60	10,5	11,0	10,5			
63	14,0	14,0	14,5	14,0	9,0	4,5
65	0,0	7,0	1,0			
66	34,0	17,0	15,0			
68	10,5	10,0	7,0			
69	7,0	11,0	7,0			
71	7,0	0,0	7,0	7,0	3,0	7,0
72	3,5	4,0	3,5			
75	3,5	16,0	7,0	3,5	11,0	3,5
76	7,0	3,0	3,0	0,0	2,0	1,0
78	7,0	9,0	1,0	3,5	3,0	3,5
79	10,5	10,0	2,5	7,0	5,0	7,0
80	3,5	2,0	0,5	0,0	4,0	1,0
81	3,5	8,0	3,5			

Continuação na próxima página

Tabela 2 – continuação da página anterior.

	Desenvolvimento			Teste		
ID da atividade	**Estimativa equipe**	**Estimativa gerada**	**Tempo utilizado**	**Estimativa equipe**	**Estimativa gerada**	**Tempo utilizado**
82	10,5	6,0	2,5			
84	7,0	13,0	2,5			
85	7,0	0,0	11,0			

Fonte: Autoria própria.

Ao analisar a tabela, é nítida a quantidade menor de testes utilizados comparados ao desenvolvimento. O motivo para tal diferença é o processo crescente dentro da fábrica de trocar os testes manuais para os testes automatizados. Portanto, foi gerada estimativa apenas para os testes manuais, pois os testes automatizados eram realizados por outra equipe, e utilizavam de outra atividade sem ligação com o defeito resolvido pela equipe. As informações de execução dos testes automáticos não eram guardadas em histórico e, consequentemente, não puderam ser utilizadas pelo *framework*.

Em relação às estimativas geradas, é possível observar casos de superestimativas (39 estimativas) e subestimativas (29 estimativas). Referente às estimativas acima do valor real utilizado, podemos citar como exemplo a atividade 6, em que foi estimado um valor de 18 horas e foi utilizado apenas 2 horas para o desenvolvimento; outro exemplo é a atividade 48, a qual tanto a estimativa de desenvolvimento quanto de teste ficaram maiores que o tempo utilizado, sendo que foi estimado em 14 horas de desenvolvimento e 11 horas de teste, porém foram utilizadas apenas 4 e 2 horas, respectivamente.

Dentre as várias atividades superestimadas, foram analisadas as atividades com maior divergência das horas estimadas para as horas utilizadas, como nos exemplos citados. A partir da reunião com a equipe após a *Sprint*, foi possível identificar que a maioria das atividades eram referentes a módulos complexos do sistema, os quais normalmente demandam de um tempo superior a outras atividades. Porém, na atividade estimada em questão, foi realizada apenas uma alteração simples e pontual. O método de estimativa não consegue distinguir esse nível de dificuldade, encontra atividades que demandaram muito tempo para serem realizadas, e consequentemente gera uma estimativa muito superior ao tempo utilizado.

Referente às estimativas menores que o tempo real utilizado, é possível citar como exemplos mais críticos as atividades que geraram estimativas zeradas, como nas atividades 7, 8, 21, dentre outras. A partir da reunião com as equipes, foi possível levantar como

possível problemática, o não preenchimento das horas utilizadas. A ação de concluir a atividade e não lançar horas utilizadas, é um fator comum quando as atividades são muito simples e demandam de poucas horas para a conclusão. Porém, é uma conduta errada e que no fim impacta negativamente as estimativas geradas. Outra questão levantada é a dificuldade do método de similaridade textual em encontrar atividades a partir da descrição informada, quando não há atividades com descrições similares. Nesse contexto, um exemplo seria a atividade 85, a qual possui uma descrição bem específica sobre um relatório X, além de ser o primeiro defeito do relatório mencionado. Não haviam atividades passadas relacionadas a esse relatório, e a nova atividade não fornecia mais informações que permitissem o *framework* encontrar atividades similares. Portanto, nesse caso, a estimativa gerada pelo *framework* foi igual a zero.

A partir das 68 estimativas apresentadas, foram aplicadas as fórmulas do MMRE e PRED(0,30) e os resultados podem ser analisados na Tabela 3.

Tabela 3: Resultado cálculo de precisão

Indicadores de precisão	**Valor ideal**	**Resultado**
MMRE	$\leq$ 0,25	1,82
PRED(0,30)	$\geq$ 0,75	0,30

Fonte: Autoria própria

Ao avaliar os resultados, é possível observar que os valores resultantes dos cálculos de MMRE e PRED(0,30) não atingiram o valor esperado. Portanto, conclui-se que as estimativas de esforço calculadas pelo método proposto não são precisas ou aproximadas com relação ao real tempo utilizado.

6.2 ACEITAÇÃO E PERCEPÇÃO DOS DESENVOLVEDORES

Quanto à percepção dos desenvolvedores em relação ao *framework*, foi realizada uma apresentação separadamente para cada equipe selecionada, mostrando o *framework* EstimAi e a comparação entre as estimativas realizadas pela equipe, as estimativas geradas pelo EstimAi e o real tempo utilizado. As comparações foram realizadas a partir de gráficos separados por equipe, *Sprint* e atividade de desenvolvimento ou teste. Um exemplo de gráfico apresentado pode ser observado na Figura 21, a qual apresenta a comparação das horas de desenvolvimento da primeira *Sprint* analisada da equipe de Manutenção Venda.

Para extrair a percepção dos usuários, foi criado um questionário na ferramenta

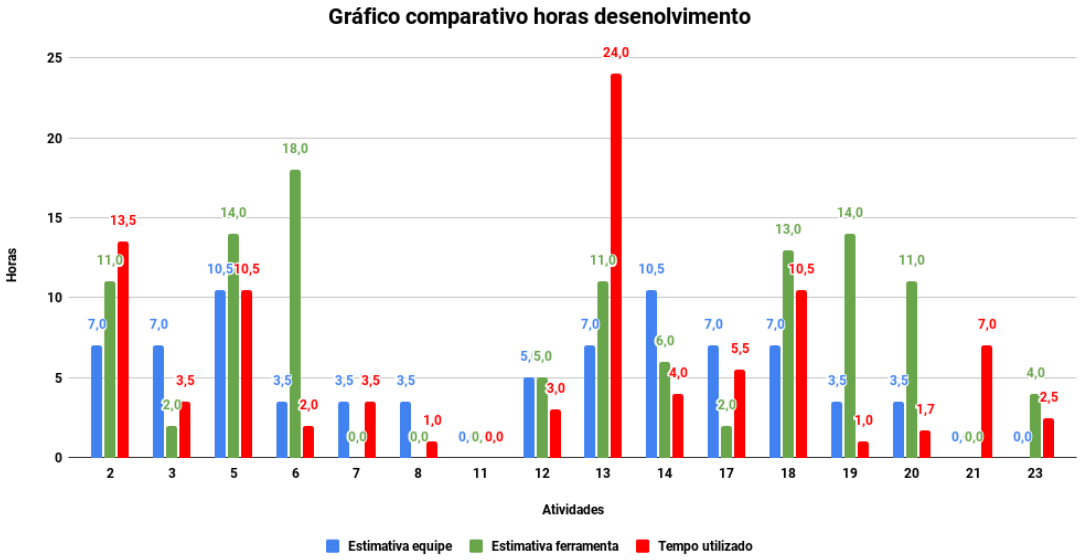

Figura 21: Gráfico comparativo das horas de desenvolvimento

Fonte: Autoria própria

Google Formulários[1], apresentado no Apêndice B. O formulário foi aplicado nas duas equipes avaliadas, assim obtendo um total de doze respostas. A partir das questões objetivas foi calculada a média das respostas e o resultado para cada pergunta pode ser observado na Tabela 4.

Tabela 4: Resultado questionário de avaliação (máximo: 6)

Pergunta	Ideal MRQ	Resultado MRQ
Eu achei interessante a ideia de geração automática das estimativas de esforço	> 3	5,83
Eu achei que a ferramenta gerou estimativas precisas (próximas ao real esforço utilizado)	> 3	4,5
Eu acho que a ferramenta seria útil e que iria agregar na hora de escolher as estimativas para as atividades	> 3	5,5

Fonte: Autoria própria

De modo geral, a avaliação dos usuários foi positiva, porém alguns pontos foram comentados tanto nas questões abertas do formulário, quanto na reunião de apresentação do EstimAi, com o objetivo de tentar melhorar ainda mais as estimativas. Alguns desses pontos, foram referentes à influência dos dados informados pelo usuário na estimativa. Como exemplo, foi discutido a respeito das descrições das atividades, pois descrições incompletas,

[1]O Google Formulários é uma ferramenta para criar questionários de forma simplificada e gratuita. `https://www.google.com/intl/pt-BR/forms/about/`

ou com muitas imagens e pouco texto, podem vir a dificultar o método de geração de estimativa. Outro ponto seria as discrepâncias entre o real tempo utilizado e o informado pelo usuário, pois foi possível identificar uma atividade cujo tempo utilizado foi lançado erroneamente, o que pode representar a existência de outras atividades lançadas erradas, mas que não foram identificadas. Outro ponto de discrepância do tempo utilizado, foi o fato de algumas atividades não terem horas lançadas, o que pode influenciar futuramente no método de estimativa.

Um ponto levantado para melhoria seria desconsiderar valores discrepantes durante o cálculo da estimativa, pois há uma chance de ser um valor digitado incorretamente, ou até mesmo um caso isolado (*outlier*). Esses valores acabam influenciando negativamente na regressão linear e podem causar uma superestimativa ou subestimativa. Outra ideia abordada foi referente à dificuldade do método de gerar estimativas para algumas atividades, podendo ser que a descrição da atividade é muito específica a um requisito e que talvez nunca tenha sido alterado antes, como no exemplo relacionado à manipulação de relatórios discutido na seção anterior. Por esse motivo, foi feita a sugestão de tentar ampliar a busca das atividades; ou seja, utilizar apenas partes da descrição ou palavras chaves, com o intuito de obter mais resultados e assim conseguir realizar a estimativa.

Portanto, a partir dos dados coletados do formulário, é possível afirmar que é positiva a percepção dos usuários quanto ao *framework* EstimAi.

6.3 AMEAÇAS À VALIDADE

A ameaça à validade de construção é referente à ligação entre a teoria e o que é observado (WOHLIN et al., 2012). Como principal ameaça à validade de construção do presente trabalho, podemos citar o fato de não ter sido testada a abordagem com diferentes coeficientes de similaridade, além de não possuir um procedimento caso o método não encontre atividades similares. Outro risco, é a falta de tratamento dos dados na análise de regressão linear, como por exemplo os *outliers*.

A ameaça à validade interna está relacionada a fatores que impactam nos resultados finais e que não podem ser controlados (WOHLIN et al., 2012). No estudo em questão, as estimativas geradas podem ter sido influenciadas pela possibilidade dos usuários terem descrito as atividades de forma muito resumida. Outro fator influenciável, seria o preenchimento das horas utilizadas na atividade de forma incorreta ou até mesmo o não preenchimento da mesma.

A ameaça à validade externa está relacionada à generalização dos resultados para outros cenários (WOHLIN et al., 2012). O estudo de caso foi realizado em uma empresa em específico, a qual pode não representar todas as empresas que utilizam múltiplas equipes. Assim, as questões discutidas na reunião sobre os possíveis fatores que induzem a uma estimativa longe do real esforço utilizado, pode não se aplicar a outras empresas e com isso acabar gerando estimativas mais precisas em outros ambientes.

7 CONCLUSÕES

O presente trabalho apresentou um *framework* denominado EstimAi, com uma estrutura voltada a um ambiente multi-equipe e capaz de gerar estimativas de esforço automáticas para atividades de manutenção de software.

Com o objetivo de validar o método proposto, foi realizada um estudo de caso, em duas equipes de manutenção de software em um período de duas *Sprints*. Para a realização da validação em si, foram utilizados dois indicadores de precisão, além de um questionário avaliativo. Como resultado final, concluiu-se por meio dos indicadores que a estimativa gerada não é precisa, ou seja, não ficou próxima ao real tempo utilizado. Porém, conforme análise e opinião das equipes, fornecido por meio de questionário de satisfação e reunião de encerramento de *Sprint*, a estimativa foi consideravelmente satisfatória.

Portanto, conclui-se que na visão dos usuários o *framework* gerou estimativas de esforço aceitáveis, mas que segundo a literatura, a estimativa não foi gerada de forma precisa. Apesar das duas avaliações se oporem, é nítida a necessidade de melhorar as estimativas geradas.

7.1 LIÇÕES APRENDIDAS

Durante todo o desenvolvimento do trabalho, algumas lições foram aprendidas referente a geração da estimativa de esforço, sendo elas:

- Atividades de diferentes níveis de dificuldade em módulos complexos podem tendenciar a estimativa de esforço, gerando superestimativas;
- Descrição da atividade incompleta e/ou com muitas imagens, a exemplo de capturas de tela, podem dificultar a abordagem do *framework* para encontrar outras atividades similares;
- Descrição da atividade muito específica referente a um requisito ou módulo quase

não alterado, pode refinar muito a busca e não encontrar atividades semelhantes;

- Lançamento de horas utilizadas de forma incorreta ou até mesmo o não preenchimento da mesma, pode influenciar o cálculo das estimativas via regressão linear;
- Atividades semelhantes e com valores de real esforço discrepantes podem influenciar negativamente na regressão linear.

7.2 TRABALHOS FUTUROS

Por fim, ficam abertos para trabalhos futuros, melhorias na geração das estimativas de esforço, podendo aperfeiçoar o uso da regressão linear ou até mesmo fazer uso de novas tecnologias, como inteligência artificial, para assim aprimorar a busca pelas atividades já concluídas. Outras melhorias propostas para o *framework* são: a criação do status da atividade, com o objetivo de identificar se a mesma está finalizada, cancelada ou pendente; e um campo para identificar se vai ser realizada a atividade de teste.

Como possível trabalho futuro, é interessante a aplicação do *framework* em outra empresa de desenvolvimento de software, e assim avaliar as estimativas geradas em um ambiente diferente do realizado no estudo de caso. Outro trabalho possível, é uma análise de como melhorar as descrições das atividades, com o objetivo de facilitar o método de similaridade textual para encontrar atividades semelhantes. E por fim, realizar uma explicação para os usuários para que as próximas atividades tenham descrições cadastradas corretamente e consequentemente melhorando a geração das estimativas de esforço.

REFERÊNCIAS

AUDY, J. **Scrum 360: Um guia completo e prático de agilidade**. [S.l.]: Editora Casa do Código, 2015.

CAVALCANTI, A. P. **Uma medida de similaridade textual para identificação de plágio em fóruns educacionais**. Dissertação (Mestrado em Informática Aplicada) — Universidade Federal Rural de Pernambuco, 2018.

DATABASE, I. D. **IBM Db2 Database**. 2021. Disponível em: <`https://www.ibm.com/br-pt/products/db2-database`>. Acesso em: 16 de abril de 2021.

DICE, L. R. Measures of the amount of ecologic association between species. **Ecology**, Wiley Online Library, v. 26, n. 3, p. 297–302, 1945.

FEDOTOVA, O.; TEIXEIRA, L.; ALVELOS, H. Software effort estimation with multiple linear regression: Review and practical application. **Journal of Information Science and Engineering**, v. 29, n. 5, p. 925–945, 2013.

FREIRE, Y. M. A.; BELCHIOR, A. D. Estimativas de manutenção de software a partir de casos de uso. In: **5th Simpósio Brasileiro de Qualidade de Software–III Workshop de Manutenção de Software Moderna, Vila Velha-ES**. [S.l.: s.n.], 2006.

JAZZ. **IBM Engineering Workflow Management**. 2019. Disponível em: <`https://jazz.net/products/rational-team-concert/`>. Acesso em: 27 de outubro de 2019.

KHATIBI, V.; JAWAWI, D. N. Software cost estimation methods: A review 1. **Journal of Emerging Trends in Computing and Information Sciences**, v. 2, n. 1, p. 21–29, 2011.

KITCHENHAM, B. Procedures for performing systematic reviews. **Keele, UK, Keele University**, v. 33, n. 2004, p. 1–26, 2004.

KNEX.JS. **Knex.js**. 2021. Disponível em: <`http://knexjs.org/`>. Acesso em: 16 de abril de 2021.

KOSKINEN, J. **Software maintenance costs**. 2015. Disponível em: <`https://wiki.uef.fi/display/tktWiki/Jussi+Koskinen`>. Acesso em: 27 de outubro de 2019.

LIENTZ, B.; SWANSON, E. **Software maintenance management: a study of the maintenance of computer application software in 487 data processing organizations**. [S.l.]: Addison-Wesley, 1980.

LIKERT, R. **A Technique for the Measurement of Attitudes**. Archives of Psychology, 1932. (A Technique for the Measurement of Attitudes, Nº 136-165). Disponível em: <https://books.google.com.br/books?id=9rotAAAAYAAJ>.

MIGUEL, M. A. **Giveme effort: um framework para apoiar estimativa de esforço em atividades de manutenção e compreensão de software**. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Juiz de Fora, 2016.

MIGUEL, M. A. et al. A framework to support effort estimation on software maintenance and evolution activities. In: **12th Brazilian Symposium on Information Systems on Brazilian Symposium on Information Systems: Information Systems in the Cloud Computing Era**. [S.l.]: Brazilian Computer Society, 2016. v. 1, p. 31.

NGUYEN, V.; BOEHM, B.; DANPHITSANUPHAN, P. A controlled experiment in assessing and estimating software maintenance tasks. **Information and software technology**, Elsevier, v. 53, n. 6, p. 682–691, 2011.

NODE.JS. **Node.js**. 2021. Disponível em: <`https://nodejs.org/en/`>. Acesso em: 14 de março de 2021.

OLIVEIRA, L. F. G. d. **Comparação de algoritmos para qualificação das saídas de busca em gerenciadores de distribuição Linux**. Monografia (Bacharelado em Engenharia de Software) — Universidade de Brasília, 2015.

POKER, P. **Planning poker**. 2021. Disponível em: <`https://ligaagil.com.br/agile-tools-downloads/planning-poker/`>. Acesso em: 26 de abril de 2021.

PORT, D.; KORTE, M. Comparative studies of the model evaluation criterions mmre and pred in software cost estimation research. In: **2nd ACM-IEEE international symposium on Empirical software engineering and measurement**. [S.l.: s.n.], 2008. p. 51–60.

POSTGRESQL. **PostgreSQL**. 2021. Disponível em: <`https://www.postgresql.org/`>. Acesso em: 14 de março de 2021.

PRESSMAN, R. S. **Engenharia de Software. 8ª edição**. Nova York, USA: Mc Graw Hill, 2016.

RAJLICH, V. Software evolution and maintenance. In: **36th International Conference on Software Engineering**. [S.l.: s.n.], 2014. p. 133–144.

REACT. **React**. 2021. Disponível em: <`https://pt-br.reactjs.org/`>. Acesso em: 14 de março de 2021.

SCHWABER, K.; SUTHERLAND, J. **Guia do Scrum, um guia definitivo para o Scrum: as regras do jogo**. [S.l.]: Share-Alike - Creative Commons, 2017.

SILVA, A. C. **Sistema de Gerenciamento de Tarefas para usuários de Scrum**. Monografia (Engenharia Eletrônica e de Computação) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, Brasil, 2011.

SIMPLE-STATISTICS. **Simple-statistics**. 2021. Disponível em: <`https://www.npmjs.com/package/simple-statistics`>. Acesso em: 14 de março de 2021.

SMITH, L.; VERWEIRE, D.; IBM. **ibm_db**. 2010. Disponível em: <`https://www.npmjs.com/package/ibm_db`>. Acesso em: 16 de abril de 2021.

SOMMERVILLE, I. **Engenharia de Software**. São Paulo: Pearson Education, 2019.

SØRENSEN, T. **A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons**. [S.l.]: Biol. Skr., 1948.

SPINOLA, R. **Manutenção de Software: Definições e Dificuldades - Artigo Revista SQL Magazine 86**. 2011. Disponível em: <https://www.devmedia.com.br/manutencao-de-software-definicoes-e-dificuldades-artigo-revista-sql-magazine-86/20402>. Acesso em: 12 de novembro de 2019.

Standish Group, T. **CHAOS REPORT 2015**. 2015. Disponível em: <https://standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf>. Acesso em: 27 de outubro de 2019.

STRING-SIMILARITY. **String-similarity**. 2021. Disponível em: <https://www.npmjs.com/package/string-similarity>. Acesso em: 14 de março de 2021.

VAZQUEZ, C. E. **Estimativas de Software - Fundamentos, Técnicas e Modelos**. 2009. Disponível em: <https://www.devmedia.com.br/artigo-engenharia-de-software-11-estimativas-de-software-fundamentos-tecnicas-e-modelos/12213>. Acesso em: 12 de novembro de 2019.

VAZQUEZ, C. E.; SIMÕES, G. S.; ALBERT, R. M. **Análise de Pontos de Função: Medição, estimativas e gerenciamento de projetos de software**. São Paulo: Érica, 2013.

VIESSELI, K.; SILVA, A.; SANTOS, G. Estimativa de esforço em atividades de manutenção de software: Um mapeamento sistemático. In: SBC. **Anais da IV Escola Regional de Engenharia de Software**. [S.l.], 2020. p. 97–105.

WOHLIN, C. et al. **Experimentation in software engineering**. [S.l.]: Springer Science & Business Media, 2012.

APÊNDICE A – MAPEAMENTO SISTEMÁTICO

Estimativa de Esforço em Atividades de Manutenção de Software: Um Mapeamento Sistemático

Kaliane L. Viesseli[1], **Arielyn P. Silva**[1], **Gustavo Santos**[1]

[1]COENS – Universidade Tecnológica Federal do Paraná (UTFPR)
Estrada para Boa Esperança, km. 04 – Zona Rural, Dois Vizinhos - PR, 85660-000

{kaliane, arielyn}@alunos.utfpr.edu.br, gustavosantos@utfpr.edu.br

Abstract. *During the software lifecycle, good effort estimation allows teams to make decisions on how development activities will proceed, on the feasibility of new changes, and whether the project will be delivered according to deadlines. In this paper, we describe a systematic mapping to identify evidence with respect to metrics, approaches and frameworks for calculating effort estimation during the software maintenance phase. We analyzed 521 studies and we selected 17 primary studies; most approaches use metrics and historical data do estimate effort for new activities. However, only one study proposed a framework, which emphasizes the importance of proposing tools to calculate effort estimation for maintenance activities.*

Resumo. *Durante o ciclo de vida do software, o correto levantamento de estimativas de esforço permite que a equipe tome decisões sobre como será o andamento das atividades de desenvolvimento e manutenção, qual a viabilidade de novas alterações, e se estas serão entregues de acordo com os prazos. Neste artigo, é apresentado um mapeamento sistemático com o objetivo de identificar evidências na literatura com relação a métricas e abordagens para o cálculo de estimativa de esforço durante a fase de manutenção de software. Foram analisados 521 estudos e selecionados 17 estudos primários; a maioria das abordagens utiliza de métricas ou dados históricos para estimativa de novas atividades, e apenas um framework foi proposto, o que ressalta a importância de gerar novas ferramentas que facilitem a geração das novas estimativas.*

1. Introdução

A manutenção de software abrange os processos de modificação de um sistema após a sua entrega, sejam estes para correções de falhas, melhorias de desempenho ou adaptações do sistema para um ambiente diferente. A manutenção é considerada a fase mais custosa do ciclo de vida de um software, atingindo até 80% dos custos totais de um projeto, e cerca de 75% dos custos são dedicados em melhorias [Erlikh 2000].

As atividades de manutenção em maioria são reativas; isto é, erros são identificados no sistema em uso, melhorias não planejadas no início do projeto são requisitadas pelos clientes, e mudanças na plataforma exigem atualizações no sistema. Independente da atividade, as empresas demandarão esforço e tempo para realizar as alterações necessárias, considerando o estado atual das equipes, a complexidade da atividade, o nível de experiência com o sistema, entre outros fatores. Para isso, as estimativas são responsáveis por determinar quanto de esforço será empregado, qual o custo, qual será o cronograma e o escopo do projeto [Vazquez et al. 2013].

O presente trabalho abordará em específico a estimativa de esforço, a qual almeja identificar qual o tempo necessário para a conclusão de uma atividade. Neste sentido, existem muitos fatores que podem dificultar na geração de uma estimativa mais precisa, como por exemplo requisitos imprecisos e mal detalhados, falta de conhecimento dos estimadores e nova atividade muito diferente do que já foi realizado no projeto, não sendo possível utilizar atividades semelhantes para realizar uma estimativa por analogia.

O objetivo deste trabalho é identificar as abordagens propostas na literatura sobre estimativa de esforço, sumarizar as características principais dos trabalhos encontrados e, consequentemente, situar a atual pesquisa e sugerir novas áreas de investigação para futuras pesquisas. Para realização desse objetivo, foi conduzido um Mapeamento Sistemático (MS) [Kitchenham 2004] em busca de trabalhos relevantes publicados em conferências e periódicos de impacto.

O restante do artigo está organizado da seguinte forma: a Seção 2 descreve o protocolo de pesquisa e condução do MS. A Seção 3 apresenta os resultados e lições aprendidas com o estudo. A Seção 4 discute limitações e ameaças à validade. E, por fim, a Seção 5 apresenta as considerações finais.

2. Mapeamento Sistemático

Com o objetivo de identificar métodos, métricas ou ferramentas de estimativa de esforço utilizados durante a manutenção de software, foi realizado um MS, o qual visa identificar, interpretar e avaliar todas as pesquisas pertinentes à questão de pesquisa definida [Kitchenham 2004]. Este MS foi realizado seguindo quatro passos: (i) definição da questão de pesquisa (QP), (ii) pesquisa dos estudos primários relacionados ao tema, (iii) triagem desses estudos e (iv) extração de dados. Para a condução deste MS, a seguinte QP foi definida:

- **QP:** Quais métodos, métricas ou ferramentas de estimativa de esforço têm sido aplicadas à manutenção de software?

Esta QP visa identificar diferentes abordagens para estimativa de esforço, especificamente aplicadas para apoiar atividades referentes à manutenção de software. Durante esta fase, a complexidade do sistema tende a aumentar, a qualidade do código decai e novas alterações irão demandar mais esforço para compreensão do estado atual do sistema. Consequentemente, estimativas realizadas com base nas primeiras versões do projeto não refletem o real esforço para manutenção.

A partir da QP, foram identificadas as palavras chaves relacionadas ao tema, mais especificamente os termos "estimativa de esforço" e "manutenção de software". A *string* de busca foi obtida pela combinação dos sinônimos em inglês destes termos, obtendo então a seguinte *string*: (***"effort estimation" AND "software maintenance"***).

Posteriormente, quatro bases de dados foram selecionadas para extração dos estudos primários, sendo elas: *ACM Digital Library*, *IEEE Xplore*, *Scopus* e *Springer*.[1]. Por meio da *string* de busca, são buscados automaticamente estudos em conferências e periódicos de impacto na área.

[1] `http://dl.acm.org/` — `http://ieeexplore.ieee.org/` — `http://scopus.com/` — `http://link.springer.com/`

Para a triagem dos estudos obtidos na etapa anterior, foram definidos critérios de inclusão e exclusão. Ambos são importantes para classificar os estudos mais relevantes e que possam contribuir com a QP deste trabalho. Os **Critérios de Inclusão (CI)** agregam: **CI_1**: estudos primários que abordem alguma metodologia ou ferramenta propondo estimativas de esforço voltadas para a área de manutenção de software; e **CI_2**: estudos primários em que a estimativa de esforço seja voltada a tarefas ou a pequenos projetos.

Quanto aos **Critérios de Exclusão (CE)**, são definidos a seguir: **CE_1**: estudos primários que não sejam *full paper* ou *short paper*; **CE_2**: estudos primários que estejam escritos em outro idioma que não seja Inglês ou Português; **CE_3**: estudos primários incompletos e que não possuem avaliação dos métodos; **CE_4**: estudos primários indisponíveis para *download*; **CE_5**: estudos primários que são versões anteriores de um estudo mais completo sobre a mesma investigação.

A condução deste MS é apresentado na Figura 1. As buscas automáticas nas bases de pesquisa foram realizadas no período de outubro a novembro de 2019.

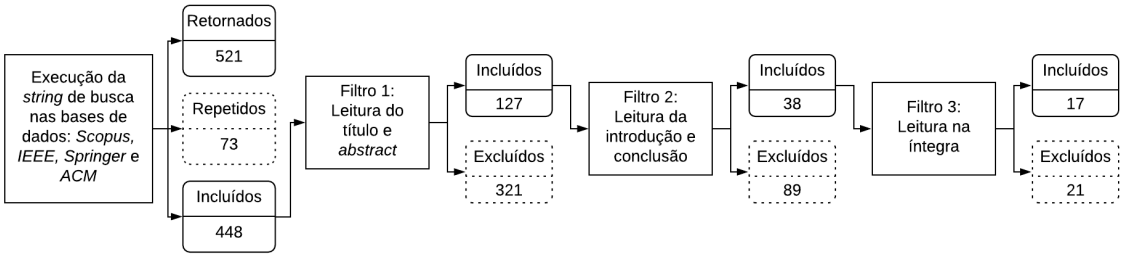

Figura 1. Processo de condução do MS

Após a execução da *string* de busca nas bases selecionadas, foram retornados 521 estudos, sendo 19 estudos na *ACM*, 204 no *IEEE*, 95 no *Scopus*, e 203 no *Springer*. A partir dos resultados obtidos, identificou-se 73 artigos repetidos, os quais foram excluídos do mapeamento, assim restando 448 estudos a serem analisados. Na próxima fase, foram aplicados os critérios de inclusão e exclusão a partir da leitura do título e resumo dos estudos selecionados, sendo incluídos 127 estudos e excluídos 321 estudos. Em seguida, os mesmos critérios foram aplicados, desta vez a partir da leitura da introdução e conclusão dos estudos; foram incluídos 38 estudos e excluídos 89 estudos nesta fase. Por fim, o último filtro foi baseado na leitura integral dos estudos, dos quais 17 estudos foram incluídos e 21 foram excluídos.

É importante destacar que, para melhor compreensão do contexto de cada estudo primário, foram criadas três categorias para classificação desses estudos conforme os objetivos da QP. As categorias são:

- **C_1 – Métricas:** reúne estudos que sugerem o uso de métricas, *e.g.*, linhas de código e pontos de função, ou fórmulas para o cálculo da estimativa de esforço;
- **C_2 – Métodos:** reúne estudos que propõem um método, técnica ou abordagem para o cálculo da estimativa de esforço;
- **C_3 – Ferramentas:** reúne estudos que apresentam uma ferramenta, protótipo, sistema, software, ou *framework* para o cálculo de estimativa de esforço.

Após a leitura na íntegra, foi possível associar alguns estudos a mais de uma categoria. Os conjuntos de categorias observadas foram: C_1 e C_2; e C_2 e C_3.

3. Resultados

A partir da execução do MS, foram selecionados 17 estudos que apresentavam algum método para estimativa de esforço na área de manutenção de software. Considerando o meio de publicação, nota-se que a maioria dos estudos foi publicado em conferências, correspondendo a 58,8% (10/17), já os demais foram publicados em periódicos – 29,4% (5/17), simpósio – 5,9% (1/17) e workshop – 5,9% (1/17).

Quanto ao ano de publicação, pode-se observar que essa área possui estudos ao passar dos anos, tendo artigos desde 1997 até 2019, sendo que os anos que obtiveram mais artigos foram 2008, 2011 e 2018, possuindo dois estudos cada ano. Utilizando como base o país de origem do primeiro autor, os 17 artigos estão distribuídos em nove países diferentes, sendo Índia o país com mais estudos nessa área, totalizando quatro artigos; Estados Unidos e Brasil produziram três estudos cada, e a Coréia do Sul publicou dois artigos. Nas Figuras 2 e 3 são apresentadas as distruibuições de estudos por ano de publicação e país do primeiro autor, respectivamente.

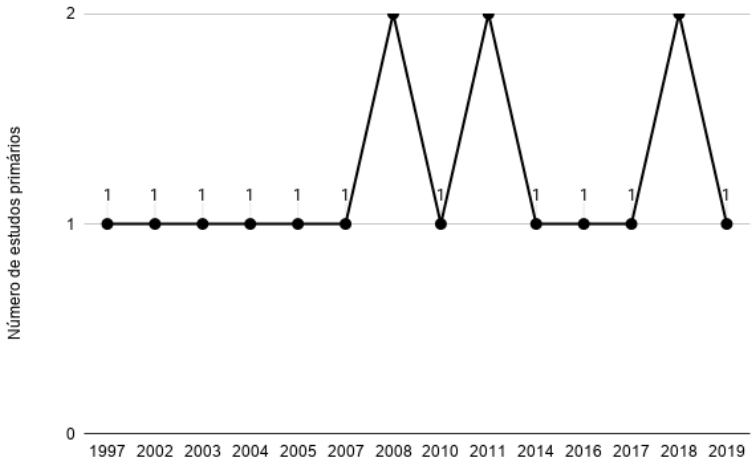

Figura 2. Visão geral dos estudos primários de acordo com o ano de publicação

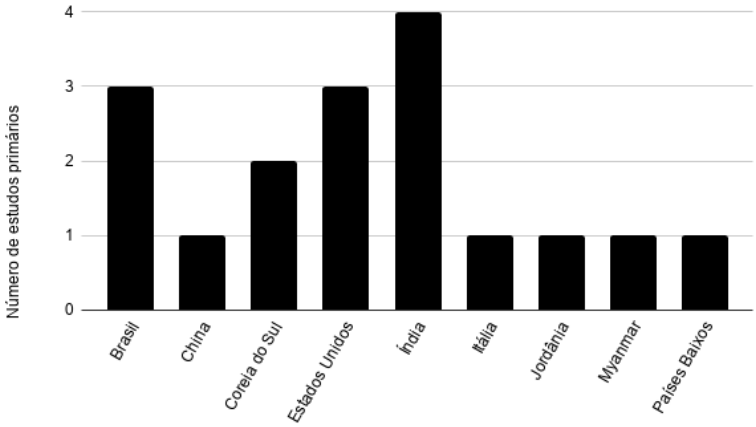

Figura 3. Visão geral dos estudos conforme o país de origem do primeiro autor

Segundo [Lientz and Swanson 1980], existem quatro tipos de manutenção de software: corretiva, adaptativa, perfectiva e preventiva. Observou-se que, referente ao tipo de manutenção a qual seriam empregadas métricas, métodos e ferramentas, cinco estudos não especificaram o tipo de manutenção. Três estudos aplicaram estimativa de esforço relacionado à manutenção corretiva, um estudo focou em manutenção adaptativa, dois estudos focaram em manutenção corretiva e adaptativa, dois focaram em manutenção corretiva e evolutiva e quatro estudos focaram em três tipos de manutenção: corretiva, adaptativa e evolutiva. Conforme ilustrado na Figura 4, pode-se observar a quantidade de estudos considerando separadamente cada tipo de manutenção.

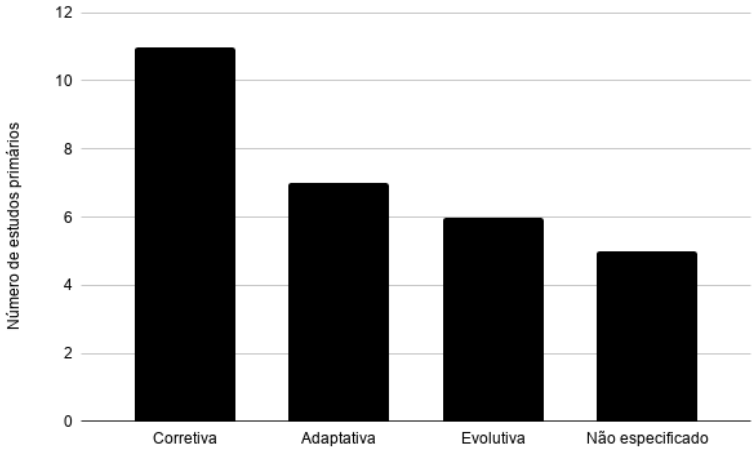

Figura 4. Visão geral dos estudos de acordo com o tipo de manutenção

Para responder a QP, na Tabela 1 são apresentados os métodos utilizados para estimativa de esforço encontrados no MS, juntamente com as categorias nas quais os artigos foram classificados. Observa-se que muitos dos artigos utilizaram métricas (11 de 17) tanto para gerar a estimativa (C_1), quanto para auxiliar outros métodos a gerarem a estimativa (C_1 e C_2). Dos 17 artigos, 13 apresentaram algum método de estimativa de esforço, porém apenas cinco apresentaram apenas o método sem utilizar de métricas ou uma nova ferramenta (C_2). Analisando a categoria C_3, apenas um estudo apresentou um *framework* em conjunto com métodos de estimativa de esforço (C_2 e C_3).

Tendo em vista os estudos que utilizam métricas para o cálculo da estimativa de esforço (categoria 1), observa-se que a métrica mais utilizada é o número de linhas de código (6 de 11 estudos). No entanto, os estudos utilizaram cálculos diferentes dessa métrica: alguns mediram o tamanho do sistema para complementar as demais métricas, outros exploravam a quantidade de linhas alteradas, inseridas ou excluídas para gerar a estimativa. Dos demais estudos, quatro utilizaram pontos de função e apenas um estudo [Chandra et al. 2017] aplicou métricas de orientação a objetos, como número de métodos por classe e profundidade de herança. A Tabela 2 mostra as métricas utilizadas para o cálculo de estimativa de esforço para cada estudo baseado em métricas (C_1).

Quanto à categoria 2, ou seja, técnicas e abordagens para o cálculo da estimativa, observa-se que o método mais usado foi a análise de regressão linear (5 de 13 estudos), o qual permite chegar a um valor estimado a partir de outros valores analisados de uma base de dados históricos. Além disso, quatro estudos utilizaram rede neural, o que acabou

Tabela 1. Visão geral dos estudos primários e seus métodos utilizados

Artigo	**Categoria**	**Métodos**
[Niessink and Van Vliet 1997]	C1, C2	Pontos de Função e Analogia
[Leung 2002]	C2	Analogia com o método vizinho virtual (AVN)
[Ahn et al. 2003]	C1	Pontos de Função
[Hayes et al. 2004]	C1	Modelo adaptável baseado em métricas
[De Lucia et al. 2005]	C1, C2	Regressão linear multivariada
[Song et al. 2007]	C1, C2	Modelo probabilístico baseado em rede Bayesiana
[Shukla and Misra 2008]	C1, C2	Rede Neural
[Tenório Jr et al. 2008]	C1, C2	Regressão Linear pelo Menor Quadrado (LSLR) e Pontos de Função
[Thaw et al. 2010]	C2	Mapa auto-organizado em Rede Neural
[Bharathi and Shastry 2011]	C2	Rede Neural
[Nguyen et al. 2011]	C1	COCOMO
[Alomari et al. 2014]	C1, C2	Análise de regressão e fatiamento do código
[Miguel et al. 2016]	C2, C3	Similaridade textual, reputação dos desenvolvedores e análise de regressão
[Chandra et al. 2017]	C1, C2	Análise de regressão e Support Vector Machine (Univariada e Multivariada)
[Hira and Boehm 2018]	C1	COSMIC Pontos de Função com SNAP
[Lélis et al. 2018]	C2	Reputação dos desenvolvedores
[Pillai and Madhukumar 2019]	C2	Julgamento de especialista

Tabela 2. Visão geral dos estudos primários e suas métricas utilizadas

Artigo	**Métricas**
[Niessink and Van Vliet 1997]	Pontos de Função
[Ahn et al. 2003]	Pontos de Função
[Hayes et al. 2004]	Linhas e operadores alterados
[De Lucia et al. 2005]	Tamanho do sistema (KLOC), número de tarefas
[Song et al. 2007]	Experiência do mantenedor, tamanho do software, características estruturais
[Shukla and Misra 2008]	Complexidade, número de linhas, número de arquivos
[Tenório Jr et al. 2008]	Pontos de Função
[Nguyen et al. 2011]	Linhas inseridas, modificadas e excluídas
[Alomari et al. 2014]	Tamanho total do sistema, tempo de atraso, intervalo de tempo entre versões, número de *hashes* modificado
[Chandra et al. 2017]	Número de métodos por classe, acoplamento, falta de coesão, número de filhos, profundidade da herança
[Hira and Boehm 2018]	Pontos de Função

se tornando o segundo método mais aplicado. Dos demais métodos podemos citar: julgamento de especialista na atividade, similaridade textual e analogia que fazem a busca de atividades já realizadas para assim gerar a estimativa de esforço, dentre outros métodos apresentados na Tabela 1.

Ao analisar a categoria 3, referente a ferramentas e *frameworks*, obteve-se apenas um estudo que se enquadrou nesta categoria [Miguel et al. 2016]. O estudo apresentou um *framework* que além de gerar as estimativas segundo diversas abordagens, apresenta também seis tipos de visualizações em formato de diagramas para ajudar no acompanhamento das atividades de manutenção. Dentre as visualizações, se destacam a visualização da reputação dos desenvolvedores ao longo do tempo, e uma representação em regressão linear do esforço real utilizado nas atividades de manutenção.

4. Ameaças à Validade

A *validade interna* é relacionada ao tratamento dos resultados obtidos. Durante a seleção dos artigos, o fator humano pode ter tendenciado a pesquisa e, consequentemente, afetado os resultados do MS. Para mitigar esta ameaça, toda a condução do MS foi realizada em pares (primeiro e segundo autor) e documentada via planilha compartilhada, a qual foi inspecionada posteriormente por um revisor (terceiro autor).

A *validade externa* é relacionada às chances da generalização dos estudos obtidos. Este MS extraiu estudos de quatro grandes bases de busca a partir de uma *string* de busca abrangente, com apenas dois termos. Em consequência dessa busca ampla, uma grande quantidade de estudos foram retornados no início da busca. No entanto, não foi utilizado um estudo de controle. Dessa forma, há a possibilidade de existirem outros estudos relevantes e que não foram incluídos no MS.

5. Considerações Finais

Neste trabalho foi realizado um mapeamento sistemático a partir das diretrizes propostas por [Kitchenham 2004], com o principal objetivo de encontrar quais métodos, métricas ou ferramentas estavam sendo utilizadas na manutenção de software para realizar estimativa de esforço de atividades. Deste modo, foram apresentados e discutidos os 17 estudos selecionados do mapeamento, os quais foram divididos em três categorias: C_1 – Métricas, C_2 – Métodos e C_3 – Ferramentas.

Os resultados apontam que os métodos são mais utilizados que as métricas, apesar dos dois poderem ser usados em conjunto. Podemos citar que a métrica mais utilizada foi linhas de código, podendo ser usada para medir o tamanho total do sistema, ou identificar linhas alteradas, incluídas ou excluídas. Já o método mais utilizado foi a análise de regressão linear, o que representa que muitos estudos preferem utilizar dados históricos para gerar novas estimativas. Quanto às ferramentas, obteve-se apenas um *framework* proposto, composto por seis tipos de visualizações para facilitar para o usuário o acompanhamento das atividades.

Entre os estudos é possível analisar que poucos levaram em consideração a opinião dos especialistas no projeto quando se trata de estimativas, os quais podem fornecer dados baseados em suas experiências. Diante dos estudos resultantes do MS, observa-se que todos apresentaram um método que aprimora a estimativa, porém apenas o ar-

tigo [Miguel et al. 2016] apresentou uma maneira de ajudar o usuário a gerar essas estimativas. Assim, uma lacuna de pesquisa seria o desenvolvimento de *frameworks* ou ferramentas que facilitassem ao usuário a geração das novas estimativas de esforço para manutenção de software.

Portanto, a principal contribuição deste trabalho é a obtenção de uma visão geral das pesquisas que estavam sendo realizadas de estimativa de esforço e que focassem nas atividades de manutenção de software. Em vista disso, como trabalho futuro, pretende-se realizar um survey, o qual teria como objetivo identificar o que os desenvolvedores levam em consideração no momento de estimar uma nova atividade e quais métodos estão sendo utilizados na indústria. Desse modo, visa-se aprimorar o método de geração de estimativa e deixá-la mais próxima do contexto atual do projeto.

Referências

Ahn, Y., Suh, J., Kim, S., and Kim, H. (2003). The software maintenance project effort estimation model based on function points. *Software maintenance and evolution: Research and practice*, 15(2):71–85.

Alomari, H. W., Collard, M. L., and Maletic, J. I. (2014). A slice-based estimation approach for maintenance effort. In *30th International Conference on Software Maintenance and Evolution*, pages 81–90.

Bharathi, V. and Shastry, U. (2011). Neural network based effort prediction model for maintenance projects. In *11th International Conference on Software Process Improvement and Capability Determination*, pages 236–239.

Chandra, D., Choudhary, M., and Gupta, D. (2017). Prophecy of software maintenance effort with univariate and multivariate approach: By using support vector machine learning techniquewith radial basis kernel function. In *6th International Conference on Computing, Communication and Automation*, pages 876–880.

De Lucia, A., Pompella, E., and Stefanucci, S. (2005). Assessing effort estimation models for corrective maintenance through empirical studies. *Information and Software Technology*, 47(1):3–15.

Erlikh, L. (2000). Leveraging legacy system dollars for e-business. *IT Professional*, 2(3):17–23.

Hayes, J. H., Patel, S. C., and Zhao, L. (2004). A metrics-based software maintenance effort model. In *8th European Conference on Software Maintenance and Reengineering*, pages 254–258.

Hira, A. and Boehm, B. (2018). COSMIC function points evaluation for software maintenance. In *11th Innovations in Software Engineering Conference*, page 4.

Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26.

Lélis, C. A., Miguel, M. A., Araújo, M. A. P., David, J. M. N., and Braga, R. (2018). AD-reputation: a reputation-based approach to support effort estimation. In *Information Technology – New Generations*, pages 621–626. Springer.

Leung, H. K. (2002). Estimating maintenance effort by analogy. *Empirical Software Engineering*, 7(2):157–175.

Lientz, B. and Swanson, E. (1980). *Software maintenance management: a study of the maintenance of computer application software in 487 data processing organizations*. Addison-Wesley.

Miguel, M. A., Araujo, M. A. P., David, J. M. N., and Braga, R. (2016). A framework to support effort estimation on software maintenance and evolution activities. In *12th Brazilian Symposium on Information Systems: Information Systems in the Cloud Computing Era*, pages 232–239.

Nguyen, V., Boehm, B., and Danphitsanuphan, P. (2011). A controlled experiment in assessing and estimating software maintenance tasks. *Information and Software Technology*, 53(6):682–691.

Niessink, F. and Van Vliet, H. (1997). Predicting maintenance effort with function points. In *13th International Conference on Software Maintenance*, pages 32–39.

Pillai, S. and Madhukumar, R. (2019). An experiment to improve expert judgment software estimation through work breakdown structure. *Innovative Technology and Exploring Engineering*, 8(7):2278–3075.

Shukla, R. and Misra, A. K. (2008). Estimating software maintenance effort: a neural network approach. In *1st India Software Engineering Conference*, pages 107–112.

Song, T.-H., Yoon, K.-A., and Bae, D.-H. (2007). An approach to probabilistic effort estimation for military avionics software maintenance by considering structural characteristics. In *14th Asia-Pacific Software Engineering Conference*, pages 406–413.

Tenório Jr, N. N., Ribeiro, M. B., and Ruiz, D. D. (2008). A quasi-experiment for effort and defect estimation using least square linear regression and function points. In *32nd Annual IEEE Software Engineering Workshop*, pages 143–151.

Thaw, T., Aung, M. P., Wah, N. L., Nyein, S. S., Phyo, Z. L., and Htun, K. Z. (2010). Comparison for the accuracy of defect fix effort estimation. In *2nd International Conference on Computer Engineering and Technology*, pages 550–554.

Vazquez, C. E., Simões, G. S., and Albert, R. M. (2013). *Análise de Pontos de Função: Medição, estimativas e gerenciamento de projetos de software*. Érica, São Paulo.

APÊNDICE B – QUESTIONÁRIO AVALIATIVO

Avaliação sobre o EstimAi

* O EstimAi é um ferramenta para geração de estimativas de esforço de modo automático.

As questões da pesquisa são referentes a sua opinião pessoal, não existem respostas corretas ou erradas. Sinta-se livre e confortável para responder da maneira que achar melhor.

Muito obrigada por participar desta pesquisa, sua opinião é muito importante.

*Obrigatório

1. Ao marcar a opção abaixo você concorda em participar livremente dessa pesquisa: *

Marque todas que se aplicam.

- [ ] Concordo em participar

2. Eu achei interessante a ideia de geração automática das estimativas de esforço: *

*Essa questão é referente apenas à ideia em si de geração de estimativas, e não sobre a ferramenta proposta.

Marcar apenas uma oval.

	1	2	3	4	5	6	
Discordo totalmente	◯	◯	◯	◯	◯	◯	Concordo totalmente

3. Eu achei que a ferramenta gerou estimativas precisas (próximas ao real esforço utilizado): *

Marcar apenas uma oval.

	1	2	3	4	5	6	
Discordo totalmente	◯	◯	◯	◯	◯	◯	Concordo totalmente

4. Caso a estimativa gerada não tenha sido igual ao número de horas utilizado, por que você acha que isso aconteceu:

5. Eu acho que a ferramenta seria útil e que iria agregar na hora de escolher as estimativas para as atividades: *

Marcar apenas uma oval.

	1	2	3	4	5	6	
Discordo totalmente	◯	◯	◯	◯	◯	◯	Concordo totalmente

6. Qual a sua opinião sobre a ferramenta:

O que não gostou, o que gostou, o que poderia melhorar...

Este conteúdo não foi criado nem aprovado pelo Google.

Google Formulários