Instituto Federal da Paraíba
Unidade Acadêmica de Informação e Comunicação
Mestrado Profissional em Tecnologia da Informação

Relatório Técnico

Estimativa de Esforço de Software baseada em Métricas

Disciplina
Tópicos Especiais em RSD

Professora
Crishane Azevedo Freire
crishane@ifpb.edu.br

Aluno
Savyo Igor da Nóbrega Santos
savyo.santos@academico.ifpb.edu.br

João Pessoa - PB
Março de 2020

ÍNDICE

1. Introdução

Neste relatório, são apresentados e descritos os resultados da pesquisa sobre o levantamento bibliográfico do tema "Estimativa de Esforço de Software baseada em Métricas".

As atividades do estudo envolveram atividades de metodologia diversas, tais como: definição de *strings* de busca para facilitar a pesquisa, busca de artigos sobre o tema proposto, leitura superficial dos trabalhos encontrados e, finalmente, registro dos artigos mais relacionados com o tema aqui neste documento.

Este documento foi estruturado como segue. Nesta seção, foram descritas as informações introdutórias do estudo. A Seção 2 contém os objetivos gerais e específicos. Na Seção 3, são elucidados os trabalhos encontrados e que são relevantes para a pesquisa, compondo a fundamentação teórica do estudo. A Seção 4 finaliza o documento com as considerações finais, seguido das referências bibliográficas.

2. Objetivos

A pesquisa ora documentada teve como objetivo geral realizar um levantamento do estado da arte sobre o tema "Estimativa de Esforço de Software" e apontar questões em aberto nessa área.

Dentre os objetivos específicos da referida pesquisa podem ser mencionados: (i) definir *strings* de busca para nortear a pesquisa; (ii) analisar os trabalhos dos últimos cinco anos sobre o tema proposto; e (iii) verificar quais as lacunas existentes no estado da arte, a fim de que se defina um problema de pesquisa.

3. Fundamentação Teórica

De acordo com Sommerville (2015), o processo de **medição de software** tem o objetivo de quantificar alguns atributos de um produto ou de um processo de software. Comparando essas informações, é possível tirar conclusões sobre a qualidade do software medido. Além disso, a medição permite mensurar se mudanças organizacionais (a adoção de novas ferramentas ou metodologias, por exemplo) estão sendo positivas ou não para os processos de desenvolvimento utilizados. Nesse caso, são feitas medições antes e depois da mudança a fim de verificar se ela foi positiva ou não para a organização.

A **estimativa de esforço de desenvolvimento de um software**, por sua vez, vem sendo utilizada desde os anos 50 nos projetos existentes e, com o aumento da complexidade das máquinas

nos anos 80, outras técnicas foram surgindo para aperfeiçoar essa estimativa (VERA; OCHOA; PEROVICH, 2018). Hoje ela é um aspecto importante e requisito de todas as empresas de desenvolvimento de software (KAUSHIK; TAYAL; YADAV, 2019). Em outras palavras, estimar o esforço é um meio de avaliar o total de custo necessário para executar um projeto ou produto.

Essas estimativas de esforço e custo são comumente baseadas na predição do **tamanho** do sistema que será desenvolvido (ABRAHAO; INSFRAN, 2008). No entanto, para projetos de melhoria, o seu tamanho pode ser uma referência para o cálculo do esforço a ser feito pela equipe.

Portanto, estimar o custo e esforço de um software é crucial no seu desenvolvimento, seja para projetos de sistemas novos ou melhorias de um já existente, pois é a partir dessa estimativa que é possível se ter uma ideia do tamanho e do tempo que precisará ser gasto no sistema que será desenvolvido ou evoluído e, assim, com esta projeção, o projeto será contratualmente regido (WANDERLEY, 2015). Uma alta estimativa de custo e esforço pode indicar que o tamanho do software é grande. E uma baixa estimativa pode ser um indicativo que o projeto é pequeno.

Durante o processo de estimativa do esforço, é comum que se fale numa "curva de incerteza" existente no processo (LEDERER; PRASAD, 1989). De modo geral, nos estágios iniciais de desenvolvimento, a quantidade de incerteza na estimativa é alta e, com isso, é produzida uma medição com baixa qualidade. À medida que o projeto evolui, o valor estimado melhora, portanto sua incerteza diminui.

De um modo geral, existem dois tipos de métodos de estimativa de esforço: **os métodos heurísticos**, que se concentram em uma experiência humana (um estimador), e **os métodos algorítmicos**, que consistem em fórmulas matemáticas para calcular o total de custo/esforço necessário. Além desses dois tipos, existem outros métodos propostos na literatura, tais como o método de regressão, analogia e *soft-computing* (AMIT; VERMA, 2013). No entanto, os métodos heurísticos e algorítmicos são mais usados pela literatura e serão abordados nas subseções a seguir.

3.1 Método Heurístico

O **método heurístico** de estimativa se concentra na utilização de conhecimentos e experiência de especialistas. O chamado estimador e o especialista de determinado domínio estimam o custo/esforço total necessário do desenvolvimento do software. Alguns métodos heurísticos utilizados na literatura são a opinião especializada de

especialistas, a técnica Delphi (baseada em consenso) e a estrutura de detalhamento do trabalho (dividir a carga de trabalho em tarefas simples para que facilite a estimativa) (AMIT; VERMA, 2013).

3.2 Método Algorítmico

O **método algorítmico** consiste em formulações matemáticas para realizar a estimativa. É mais utilizado na literatura devido a sua consistência e por não depender do fator humano, que tende a ser subjetivo. Alguns dos métodos algorítmicos mais citados na literatura são a Análise de Pontos de Função (APF), o COCOMO II e a Estimativa Baseada em Casos de Uso (AMIT; VERMA, 2013) (VERA; OCHOA; PEROVICH, 2018) (HAI; NHUNG; THAI, 2019).

COCOMO II (*Constructive Cost Model*) (BAIK; BOEHM, 2000) é a segunda versão do modelo de estimativa COCOMO, que envolve o uso de equações matemáticas baseadas em pesquisa e dados históricos, e utilizam como entrada a quantidade de linhas de código do projeto de software e a avaliação de outros aspectos relevantes para a estimativa chamados de *cost drivers*.

A metodologia de modelagem em sete passos do COCOMO II é mostrada na Figura 1.

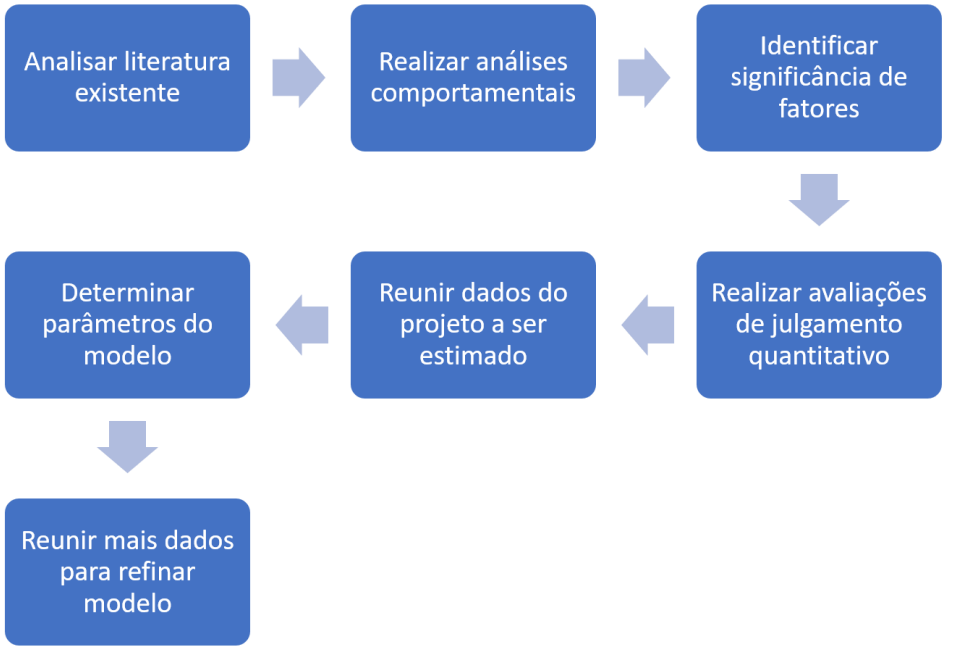

Figura 1 – Modelo em sete passos do COCOMO II [adaptação de (BAIK; BOEHM, 2000)].

Os passos mostrados na Figura 1 envolvem:

1. Analisar a literatura existente com relação aos fatores que podem afetar a quantificação das estimativas;
2. Realizar análises comportamentais para determinar o efeito dos níveis de fator nas quantidades a serem estimadas;
3. Identificar o quão significantes são os fatores nas quantidades a serem estimadas;
4. Realizar avaliações de julgamento quantitativo, a fim de formular uma primeira versão;
5. Reunir dados do projeto e determinar a significância estatística dos vários parâmetros existentes;
6. Determinar um conjunto de parâmetros do modelo proposto;
7. Reunir mais dados para refinar o modelo.

Análise de Pontos de Função (APF) (ALBRECHT, 1979) é um modelo versátil de estimar o custo e esforço do desenvolvimento de software. Essa técnica quantifica as funcionalidades existentes no produto com base no ponto de vista do usuário, analisando seu projeto lógico e especificação funcional, independente da tecnologia envolvida.

Atualmente, a APF é mantida como padrão internacional através da norma ISO/IEC 20.9261 (ISO/IEC 20926, 2002) e pelo *International Function Point User Group* (IFPUG), responsável por definir suas regras de contagem. Além disso, existem roteiros de métricas elaborados por sistemas de cada país que visam complementar a estimativa de projetos que não estão contemplados no manual do IFPUG (IFPUG, 2010). No Brasil, o órgão responsável por esses manuais complementares é o Sistema de Administração dos Recursos de Tecnologia da Informação (SISP), que foi criado em 2011 com o objetivo de organizar a operação, controle, supervisão e coordenação dos recursos de tecnologia da informação da administração direta, autárquica e fundacional do Poder Executivo Federal (SISP, 2018).

O modelo de contagem por APF é mostrado na Figura 2.

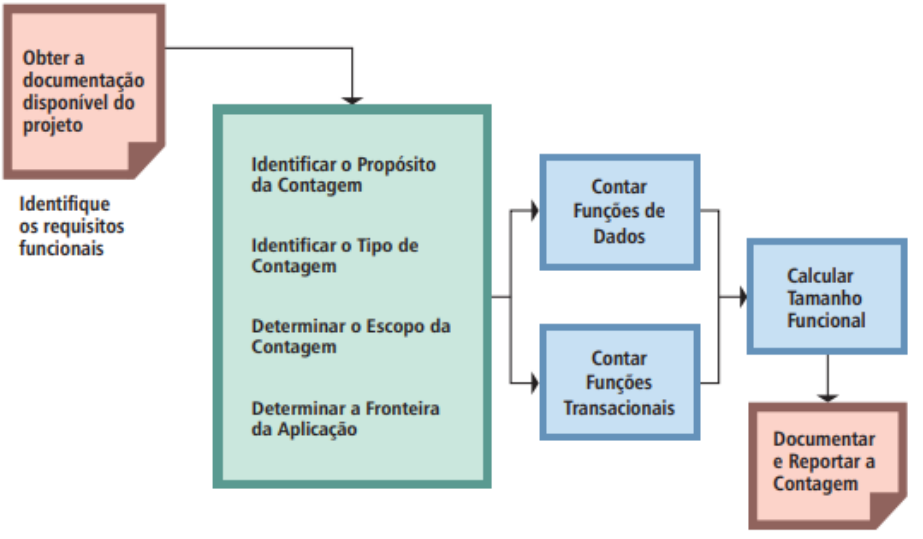

Figura 2 – Modelo de Análise de Pontos de Função (SISP, 2018).

Como pode ser visto na Figura 2, o processo de estimativa utilizando APF se inicia com a análise da documentação disponível do projeto a ser estimado, visando a identificação dos seus requisitos funcionais.

A próxima fase é a identificação do propósito da contagem, que fornece uma resposta para uma questão de negócio a ser resolvida, por exemplo: necessidade de dimensionar um novo projeto da empresa x.

Com base no propósito da contagem, podem ser definidos o tipo e o escopo dela. O tipo pode ser um projeto de melhoria, de desenvolvimento ou de uma aplicação instalada. O escopo compreende o conjunto de funcionalidades que serão contadas, por exemplo: apenas as funcionalidades de *front-end*.

O próximo passo é definir a fronteira da aplicação da contagem, que é feita de acordo com a visão do usuário e estabelece os limites das aplicações envolvidas no projeto que será contado (SISP, 2018).

Após a designação da fronteira da aplicação, ocorre a medição propriamente dita das funções de dados e transacionais. As funções de dados são divididas entre Arquivos Lógicos Internos (ALI) e Arquivos de Interface Externa (AIE), cada um com seus Registros Lógicos (RL) e Itens de Dados (ID). As funções de transação, por sua vez, constituem de processos elementares distintos, podendo ser classificadas como Entradas Externas (EE), Saídas Externas (SE) e Consultas Externas (CE), cada uma com seus Arquivos Lógicos Referenciados (ALR) e Itens de Dados (ID).

Em seguida a essas definições, pode-se calcular o tamanho funcional do projeto estimado e, com isso, documentar a contagem realizada.

A Estimativa Baseada em Casos de Uso (ANDA; ANGELVIK; RIBU, 2002), por sua vez, é uma técnica que explora o modelo e a descrição dos casos de uso de um sistema para que seja estimado o seu tamanho. Ela depende de uma padronização dos casos de uso, pois a contagem se baseia em transações identificadas em cada fluxo de eventos das funcionalidades do projeto.

Em linhas gerais, os *Use Case Points* (ou Pontos por Caso de Uso) podem ser calculados com base no modelo mostrado na Figura 3 a seguir.

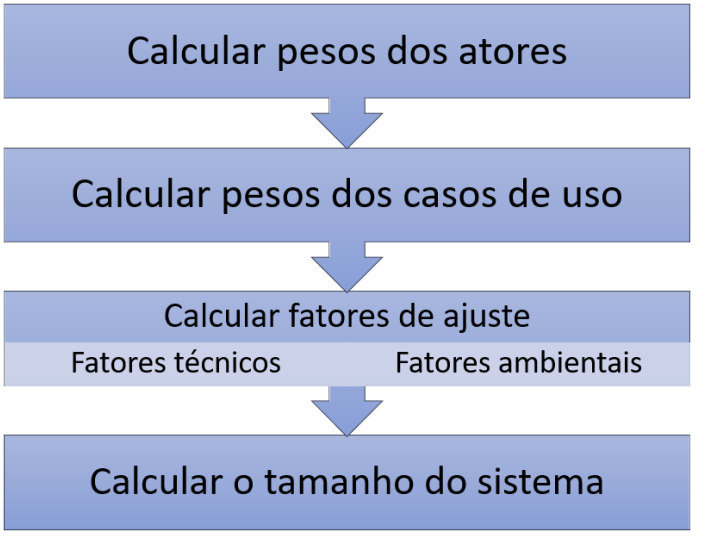

Figura 3 – Modelo da Estimativa Baseada em Casos de Uso

De acordo com a Figura 3, o processo de inicia classificando os atores do sistema de acordo com o nível de complexidade (simples, médio ou complexo).

O próximo passo consiste na contagem dos casos de uso e sua classificação com base no grau de complexidade quanto ao número de classes e transações.

Após as duas fases de classificação, o método de ajuste é aplicado no sistema e consiste em calcular fatores técnicos, cobrindo uma série de requisitos funcionais do sistema, e fatores de ambiente, requisitos não-funcionais associados ao processo de desenvolvimento, por exemplo: experiência da equipe, motivação e estabilidade do projeto.

Finalmente, o tamanho do projeto é obtido com base na relação existente dos pontos e complexidades calculados nas fases anteriores.

4. Trabalhos Relacionados

Na presente seção, serão discutidos os trabalhos encontrados sobre o tema "Estimativa de Esforço de Software" que foram encontrados após definição das seguintes *strings* de busca:

- *Effort Estimation;*
- Software Effort Estimation;
- *Functional Point Analysis (FPA).*

Utilizando as *strings* de busca acima, foram encontrados diversos trabalhos na literatura. Após a aplicação de alguns critérios de exclusão (considerar apenas trabalhos recentes – nos últimos 5 anos – e *full paper*), chegou-se aos estudos que serão mostrados a seguir.

4.1 Mapeamentos/Revisões Sistemáticas e Surveys

Hai, Nhung e Thái (2019) escreveram um capítulo no livro *Computational Statistics and Mathematical Modeling Methods in Intelligent Systems* dando uma revisão sobre estimativa de esforço de software baseada em Análise de Pontos de Função (APF). O artigo foca em algumas limitações e possíveis soluções para elas.

Os autores elencaram os principais aspectos que podem ser considerados para melhorar a precisão da estimativa de software baseada em APF.

São eles:

- pesos e complexidades;
- métodos independentes da tecnologia;
- ajuste no cálculo do ponto de função.

Sobre o primeiro aspecto, são mostrados estudos que evidenciam uma melhora no problema de permitir que a mesma função de dados ou de transação tenha o mesmo valor utilizando diferentes combinações de Registros Lógicos (RL), Arquivos Lógicos Referenciados (ALR) e Itens de Dados (ID). Outros foram dedicados a propor melhorias para tornar a APF mais detalhada, no entanto o custo é alto para se realizar isso.

Sobre o segundo aspecto, são mostrados estudos que envolvem certos tipos de técnicas no processo de desenvolvimento de software - mas não utilizados para todos. Portanto, os autores sugerem que se crie

um método correspondente para o exemplo técnico relevante. Isso exclui a generalização do método.

Sobre o último aspecto, é evidenciado que as propostas de melhoria relacionadas ao ajuste do tamanho funcional, com a adição de muitos fatores adicionais, como adição de segurança, podem tornar o processo de estimativa mais complicado e mais caro. Sobretudo quando o software requer apenas requisitos funcionais ou quando é aplicado num ambiente específico, como DW/Data Mart.

Vera, Ochoa e Perovich (2018) elaboraram uma revisão sistemática da literatura sobre métodos de estimativa de esforço de software entre 2000 e 2017, com o objetivo de levantar um estado da arte sobre o tema e identificar as principais variáveis utilizadas para classificar esses métodos. Essa análise visa auxiliar a tomada de decisão de empresas de software na escolha do melhor processo de estimativa dos seus produtos (ou a junção de dois ou mais processos para melhorar essa estimativa). De acordo com o estudo realizado pelos autores, não foi possível encontrar um método amplamente aceito no mercado, pois existem muitos critérios relevantes e quase independentes para classificar as técnicas avaliadas no trabalho.

Sehra *et al.* (2017) desenvolveram uma pesquisa para identificar quais áreas de pesquisa foram mais utilizadas no tocante da estimativa de esforço de software nos trabalhos entre 1996 e 2016. Além disso, foram levantadas cerca de 60 tendências de pesquisa dentro da área. A metodologia utilizada pelos autores foi o método estatístico *Latent Dirichlet Allocation* (LDA) aplicado a um conjunto de dados de 1178 artigos publicados sobre estimativa de esforço. Os resultados mostraram além das 60 tendências de pesquisa dentro da área, as 12 principais áreas de pesquisa sobre o tema, que são mostradas a seguir:

- Validação de modelos de estimativa;
- Estudos e mapeamentos sistemáticos;
- Estimativa de aplicações específicas;
- Julgamento de especialistas;
- Estimativa de aplicações Web;
- Fatores que afetam as estimativas;
- Projeto de seleção de dados;
- Métricas de tamanho;
- Estimativa por analogia;
- Técnicas de *machine learning*;
- Junção de vários modelos/técnicas; e

- Estimativa dinâmica de esforço.

Wanderley (2015) realizou um mapeamento sistemático sobre técnicas de estimativa de esforço e, após esse trabalho, escolheu três delas, que utilizam Pontos de Função como base para realizar estimativas, para realizar uma análise comparativa em projetos que utilizam métodos ágeis. As técnicas comparadas foram as seguintes:

- *Extending Function Point Analysis;*
- *Function Point Analysis and Cost Estimation in An Agile Development Environment;*
- *Agile Estimation Using Functional Metrics*.

Após um experimento, em três cenários distintos, os resultados mostraram que a abordagem *Agile Estimation Using Functional Metrics* apresentou o melhor desempenho.

4.2 Estudos, Trabalhos e Modelos

Kaushik, Tayal e Yadav (2019) elaboraram um estudo que investiga a aplicação de uma técnica híbrida de estimativa de esforço que utiliza a arquitetuta de redes neurais *Deep Belief Network* (DBN) juntamente ao modelo *Antlion Optimizer* (ALO). Essa técnica foi usada para previsão de esforço em ambientes de desenvolvimento de software ágeis e não ágeis. No total, a aplicação da técnica foi feita em 4 conjuntos de dados do repositório de projetos não ágeis e 3 conjuntos de dados de projetos ágeis. Além de fornecer um intervalo de previsão de esforço para lidar com a incerteza da estimativa, os resultados apontaram que a técnica híbrida foi melhor que as outras indicadas no estudo.

Abdelali, Mustapha e Abdewahed (2019) investigaram o uso do modelo de floresta aleatória na estimativa de esforço de software. Eles propuseram uma abordagem de conjuntos de árvores ótimas para a estimativa e os resultados apontaram que a abordagem proposta funcionou bem em comparação com modelos aleatórios de floresta e árvores de regressão.

Pandey, Litoriya e Pandey (2019) realizaram um experimento onde várias técnicas de Inteligência Artificial (IA) aplicadas na estimativa de esforço foram usadas num conjunto de dados de análise de software para aplicativos móveis. O objetivo do trabalho foi identificar a técnica que melhor se encaixa nesse contexto de aplicações móveis. As técnicas comparadas foram algoritmos genéticos, regressões lineares múltiplas, redes neurais perceptron de múltiplas camadas e abordagem de previsão Naive. A análise mostrou que os

algoritmos genéticos apresentaram melhor desempenho que as outras abordagens nesse contexto.

Ertugrul *et al.* (2019) propõem um ajuste de desempenho para modelos de previsão de esforço de desenvolvimento de software baseados em *machine learning*. Neste estudo, os algoritmos de aprendizado de máquina foram investigados em conjunto com técnicas de transformação de recursos, seleção de recursos e ajuste de parâmetros para estimar com precisão o esforço de desenvolvimento, e um novo modelo foi proposto como parte de um sistema especialista.

Tanveer, Vollmer e Engel (2017) avaliaram o uso da análise de impacto de mudança dentro da estimativa de esforço de desenvolvimento de software aplicada em metodologia ágil. Os resultados mostraram que os praticantes do experimento feito acharam o modelo fácil de usar e tinham interesse em usar no seu processo de estimativa, além de reconhecerem que ele fornece mais objetividade à estimativa se comparado a natureza subjetiva de métodos heurísticos (julgamento de especialistas).

Azzeh e Nassif (2016) projetaram um modelo híbrido que utiliza máquinas de vetores de suporte e redes neurais para auxiliar os modelos que utilizam apenas Pontos de Casos de Uso e, às vezes, a produtividade para estimar o esforço de desenvolvimento do software. O modelo proposto apresentou bons resultados nos conjuntos de dados aplicados, o que fez com que os autores concluíssem que o uso de modelos e técnicas de IA podem auxiliar os modelos de estimativas existentes.

Nathanael, Hendradjaya e Sunindyo (2015) elaboraram um estudo de estimativa de esforço de software de projetos Big Data baseado em três modelos: COCOMO II, Análise de Pontos de Função e Estimativa Baseada em Casos de Uso. Os autores partiram do problema em se buscar um método e modelo que melhor se adequasse às necessidades de estimativa de esforço vindas de projetos Big Data. Assim, criaram um modelo utilizando as três métricas citadas anteriormente e pode-se concluir que essa junção não foi a melhor combinação para realizar uma estimativa de esforço para o desenvolvimento de software de Big Data. No entanto, há muito espaço para aprimoramento para escolher melhores opções, de uma maneira que possa acomodar as características especiais do software Big Data e o paradigma instável de um processo de desenvolvimento de software.

Reza *et al.* (2015) desenvolveram um trabalho voltado para aplicações Web. O principal objetivo do trabalho foi demonstrar as habilidades dos métodos de estimativa de custos de aplicativos da Web e agrupar com base na medição sub-funcional, medição funcional e cálculo da complexidade, ajudando o supervisor do projeto a entender melhor o software que está sendo desenvolvido.

Todos os trabalhos e estudos mostrados nessa seção contribuíram da melhor maneira possível dentro de cada contexto envolvido e têm seus próprios méritos e deméritos. No entanto, não foi evidenciado, neste estudo, uma técnica padrão que seja comumente aceita para estimativa de esforço, pois a depender do contexto utilizado na pesquisa, um método pode ser melhor aplicado do que outro.

No Quadro 1 a seguir, apresenta-se um sumário dos trabalhos encontrados e que são relevantes para a pesquisa, com os seus autores, método de estimativa utilizado, técnica de apoio (caso tenha) e ambiente utilizado na pesquisa.

Quadro 1 – Sumário dos trabalhos sobre estimativa de esforço de software.

Autores	Método de estimativa	Técnica de apoio	Ambiente
Kaushik, Tayal e Yadav (2019)	Híbrido	Deep Belief Network (DBN) e Antlion Optimizer (ALO)	Projetos ágeis e não ágeis
Abdelali, Mustapha e Abdewahed (2019)	Híbrido	Floresta aleatória	3 ambientes (projetos Web, industriais e de software)
Pandey, Litoriya e Pandey (2019)	Algorítmico (COCOMO)	IA (algoritmos genéticos, regressões lineares múltiplas, redes neurais perceptron de múltiplas camadas e abordagem de previsão Naive)	Aplicações *mobile*
Ertugrul et al. (2019)	Híbrido	Transformação de recursos, seleção de recursos e ajuste de parâmetros	Machine learning
Tanveer, Vollmer e Engel (2017)	Heurístico (julgamento de especialistas)	Análise de impacto de mudança	Projetos ágeis
Azzeh e Nassif (2016)	Híbrido (Pontos de Casos de Uso e Produtividade)	IA (Máquinas de vetores de suporte e redes neurais)	Projetos industriais
Nathanael, Hendradjaya e Sunindyo (2015)	Algorítmico (COCOMO, APF e Estimativa baseada em Casos de Uso)	-	Big Data
Reza et al. (2015)	Híbrido	Medição funcional e não-funcional, e cálculo da complexidade	Aplicações Web

Conforme é mostrado no Quadro 1, observou-se que a maioria dos trabalhos encontrados utiliza métodos de estimativa híbridos, além de usarem técnicas de apoio da Inteligência Artificial, aprendizado de máquina, entre outros. Essa união de técnicas é importante para aumentar a precisão da estimativa de esforço realizada, de modo a se aproximar do que realmente foi o custo e esforço reais do desenvolvimento do software.

5. Considerações Finais

A principal contribuição desta pesquisa é dar um panorama inicial dos trabalhos existentes sobre o tema "Estimativa de Esforço de Software".

Conforme observado a partir das pesquisas citadas na seção anterior, percebe-se que o tema é relevante para a literatura e que possui algumas lacunas a serem exploradas, como a utilização de métodos de Inteligência Artificial para melhorar a precisão da estimativa de esforço, o uso de ambientes com grande quantidade de dados (*Big Data*) ou que utilizem técnicas de visualização.

Referências Bibliográficas

ABDELALI Z., MUSTAPHA H. e ABDELWAHED N. Investigating the use of random forest in software effort estimation. Proceedings. Second International Conference on Intelligent Computing in Data Sciences. Volume 148, p. 343-352, 2019.

ABRAHAO S.; INSFRAN E.; A Metamodeling Approach to Estimate Software Size from Requirements Specifications. Software Engineering and Advanced Applications. XXXIV, pp. 465-475. Washington (DC): IEEE Computer Society, 2008.

ALBRECHT A. J.; Measuring Application Development Productivity. In Proc. IBM Applications Development Symposium. GUIDE Int and Share Inc., IBM Corp., Monterey, pp. 83, 1979.

AMIT, VERMA B. Software Development Effort Estimation: A Review. Proceedings of the International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, 2013.

ANDA, B. ANGELVIK, E. RIBU, K. Improving Estimation Practices by Applying Use Case Models, LNCS 2259, pp. 383-397, 2002.

AZZEF M. & NASSIF A. B. A Hybrid Model for Estimating Software Project Effort from Use Case Points. Proceedings. Applied Soft Computing Journal, Elsevier. 2016.

BAIK J; BOEHM B. Empirical Analysis of CASE Tool Effects on Software Development Effort. Center for Software Engineering,2000.

ERTUGRUL E., BAYTAR Z., CATAL C. & MURATLI C. Performance tuning for machine learning-based software development effort prediction models. Proceedings. Turkish Journal of Electrical Engineering and Computer Sciences 27(2), 2019.

HAI V. V., NHUNG H. & THAI, H. H. A Review of Software Effort Estimation by Using Functional Points Analysis. Book. Computational Statistics and Mathematical Modeling Methods in Intelligent Systems, 2019.

KAUSHIK A., TAYAL D. K. & YADAV K. A Comparative Analysis on Effort Estimation for Agile and Non-agile Software Projects Using DBN-ALO, 2019.

IFPUG. Manual de Práticas de Contagem de Pontos de Função (CPM), Versão 4.3.1, 2010.

LEDERER A. L. & PRASAD, J. The Information System Development Cost Estimating Conundrum. Paper from OSRA/TIMS Meeting, 1989.

MEDEIROS E. Desenvolvimento de Software com UML 2.0. Makron Books. São Paulo, 2004.

NATHANAEL E. H., HENDRADJAYA, B. & SUNINDYO, W. D. Study of Algorithmic Method and Model for Effort Estimation in Big Data Software Development. Case Study: Geodatabase. Proceedings. The 5th International Conference on Electrical Engineering and Informatics. Bali, Indonesia, 2015.

PANDEY M., LITORIYA R. & PANDEY P. Validation of Existing Software Effort Estimation Techniques in Context with Mobile Software Applications. Wireless Pers Commun. 2019.

REZA S. M., RAHMAN, M., PARVEZ, M. H., KAISER M. S. & MAMUN S. A. Innovative Approach in Web Application Effort & Cost Estimation using Functional Measurement Type. Proceedings. International Conference on Electrical Engineering and Information & Communication Technology. Bangladesh, 2015.

SEHRA S. K., BRAR Y. S., KAUR N. & SEHRA S. S. Research Patterns and Trends in Software Effort Estimation. Proceedings. Information and Software Technology, 2017.

SISP. Roteiro de Métricas de Software do SISP: versão 2.3 / Ministério do Planejamento, Desenvolvimento e Gestão. Secretaria de Tecnologia da Informação e Comunicação - Setic. – Brasília: MP, 2018.

SOMMERVILLE, Ian. Software Engineering: 10 ed.Pearson, 2015.

TANVEER B., VOLLMER A. M. e ENGEL U. M. Utilizing Change Impact Analysis for Effort Estimation in Agile Development. Proceedings. 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA). Vienna, Austria, 2017.

VERA T., OCHOA S. F. & PEROVICH D. Survey of Software Development Effort Estimation Taxonomies. Universidad de Chile, 2018.

WANDERLEY, E. G. Aplicação de pontos por função em projetos que usam métodos ágeis: uma análise comparativa entre abordagens existentes. Dissertação de mestrado. Universidade Federal de Pernambuco. CIN, Ciência da Computação, 2015.