1 Introdução

Estimar o esforço necessário para desenvolvimento de software é uma atividade fundamental para o gerenciamento do projeto e também uma das tarefas mais desafiadoras. Estimativas errôneas podem levar a uma alocação de recursos abaixo ou acima do necessário. No primeiro caso, as consequências são atrasos nas entregas, aumento dos custos e a insatisfação do cliente (Idri & Abnane, 2017), além do esgotamento da equipe que foi reduzida porque o esforço foi subestimado (Tronto, Silva, & Sant'Anna, 2008). Quando são alocados mais recursos que o necessário, tempo e dinheiro podem ser desperdiçados e os projetos futuros postergados (Tronto *et al*., 2008; González-Carrasco *et al*., 2012).

Para produzir estimativas precisas é importante ter um conjunto robusto de informações sobre o contexto do novo projeto e sobre projetos históricos semelhantes, o que nem sempre está disponível. No caso das metodologias ágeis, ainda é preciso levar em conta os conceitos não tradicionais da abordagem (Ziauddin & Zia, 2012). Por outro lado, quando uma estimativa precisa é feita, ela se torna apta a compor a base de conhecimentos para estimações futuras e, com isso, espera-se que o tempo de estimação e as chances de erros diminuam, tornando o processo cada vez mais refinado. Ainda assim, existem fatores de incerteza aos quais todo projeto está sujeito e eles também influenciam o desempenho desta atividade (McConnell, 2006). Nesse cenário, uma variedade de mecanismos são implementados para estimar o esforço e diversos fatores influenciam o resultado do cálculo.

A estimativa por especialista é o principal método utilizado para estimar o esforço (Karna & Gotovac, 2014) e depende principalmente da experiência do profissional. Por outro lado, diversos trabalhos propõem alternativas, como, modelos de aprendizagem de máquina, redes neurais, análises de regressão e outras abordagens estatísticas e paramétricas. Algumas delas são altamente dependentes de dados históricos, tendo a exatidão do seu resultado ditada pela semelhança com o novo projeto ou pela qualidade das variáveis de quantificação utilizadas.

Outras técnicas funcionam bem sem tais informações ou as utilizam de forma opcional. No geral, as técnicas estão sujeitas a um nível de erro e sua aplicabilidade depende fortemente do tipo de projeto que se deseja estimar (Azzeh, Neagu, & Cowling, 2008; Bilgaiyan, *et al*., 2017). Assim, esforços também têm sido empregados para estabelecer uma forma de selecionar a melhor prática para realizar a estimação (Peischl *et al*., 2009) ou para selecionar o melhor conjunto de dados de entrada para determinados modelos (Azzeh, *et al*., 2008; Azzeh, 2011; González-Carrasco, *et al*., 2012).

Estimativas de esforço são importantes para todos os tipos de projetos, sejam elaborados com metodologia ágil ou tradicional (cascata). Além disso, a maioria dos métodos de estimativa não faz distinção da metodologia do projeto, tais como redes neurais, análises de regressão e estimativas por analogias. Bilgaiyan *et al*. (2017) mostraram que tanto projetos com metodologia ágil como cascata apresentam taxas de sucesso parecidas e o que parece aumentar de fato a taxa de sucesso é um ambiente de estimativas acuradas.

Apesar da importância de ter estimativas acuradas, não foi encontrada na literatura uma revisão sistemática de trabalhos sobre estimativas de esforço de desenvolvimento de software, independente de abordagem metodológica do projeto. Porém, os trabalhos de Bilgaiyan *et al*. (2017) e de Britto, Usman e Mendes (2014) apresentam revisões da literatura, e em ambos o foco se dá em metodologias ágeis.

Neste cenário, este trabalho tem como objetivo apresentar o estado da arte das técnicas de estimação do esforço de desenvolvimento em projetos de software, independente da abordagem metodológica do gerenciamento do projeto. Para atingir este objetivo, realizou-se um estudo exploratório nas bases de dados científicas. O restante do trabalho está organizado da seguinte maneira: a próxima seção apresenta técnicas de estimativa utilizadas em projetos, em especial na área de sistemas de informação. Em seguida são descritos os procedimentos metodológicos aplicados nesta pesquisa e os resultados obtidos. Por fim, é feita a discussão dos resultados e a conclusão do trabalho.

2 Revisão Bibliográfica

Estimar o esforço necessário para desenvolvimento de um software permite estimar o prazo e os custos do projeto. Por essa razão, é uma atividade fundamental para o gerenciamento do projeto. A revisão bibliográfica realizada nesta pesquisa busca apresentar técnicas para estimativa de prazo e custo em projetos. A apresentação dessas técnicas busca apresentar as técnicas mais utilizadas no cotidiano das organizações.

2.1 Técnicas para Estimativa de Prazos

A definição e a utilização de bons processos de gerência de custos e prazos de projetos são de grande importância. O objetivo desses processos é fornecer diretrizes para a realização das estimativas de um projeto e direcionar as atividades de acompanhamento e controle, de forma a auxiliar na proximidade entre os valores estimados e os reais. Como consequência, a utilização de processos para gerência de custos e prazos bem definidos auxilia na realização de estimativas com menor margem de erro, tornando os desvios dos projetos menos frequentes e menores, favorecendo o sucesso de um maior número de projetos de software. Entretanto, destaca-se que a conclusão de um projeto no prazo e orçamento previstos não é um indicador suficiente do sucesso de um projeto (Barcellos & Travassos, 2004).

O gerenciamento do prazo do projeto inclui os processos necessários para gerenciar o término pontual do projeto. Esses processos incluem a definição de atividades, a estimação dos recursos e duração das atividades, e elaboração do cronograma. Eles interagem entre si e podem se sobrepor (PMI, 2017).

A estimativa da duração das atividades usa informações sobre o escopo do projeto e sobre os recursos disponíveis. Existem cinco técnicas básicas para estimar a duração das atividades

(1) **Opinião especializada**. É guiada por informações históricas a partir de projetos anteriores similares. Pode também ser usada para determinar se seria recomendável combinar diferentes métodos de estimativas e como reconciliar as diferenças entre eles;

(2) **Estimativa análoga**. A estimativa análoga usa parâmetros tais como duração, orçamento, tamanho e complexidade de um projeto anterior similar como base para a estimativa dos mesmos parâmetros ou medidas para um projeto futuro. É usada quando há uma quantidade limitada de informações detalhadas sobre o projeto. Apesar de ser menos dispendiosa e consumir menos tempo que outras técnicas, ela é menos precisa e requer que a equipe do projeto possua habilidade técnica necessária;

(3) **Estimativa Paramétrica**. A estimativa paramétrica utiliza uma relação estatística entre dados históricos e outras variáveis para calcular uma estimativa para parâmetros da atividade, tais como custo, orçamento e duração. Esta técnica pode produzir altos níveis de precisão dependendo da sofisticação e dos dados básicos colocados no modelo;

(4) **Estimativas de três pontos**. A precisão das estimativas de duração da atividade pode ser aperfeiçoada considerando-se as incertezas das estimativas e riscos. Este conceito se originou com a Técnica de Revisão e Avaliação de Programa (PERT), que usa três estimativas para definir uma faixa aproximada para a duração de uma atividade. Esta técnica calcula a duração esperada de uma atividade (Te) usando uma média ponderada dessas três estimativas:

$$Te = (To + 4Tm + Tp) / 6$$

Onde Tm é a duração mais provável, To é a duração otimista baseada no melhor cenário, e Tp é a duração pessimista baseada no pior cenário;

(5) **Análise das Reservas**. As estimativas podem incluir reservas para contingências para considerar incertezas. À medida que informações mais precisas sobre o projeto se tornam disponíveis, a reserva pode ser usada, reduzida ou eliminada.

Em projetos de software é comum determinar o tamanho do software, a partir deste, determinar o esforço exigido, e, em seguida, o prazo necessário. Segundo Abran (2006), as estimativas de esforço podem ser geradas a partir de pontos de função. Este último depende de diversos atributos do projeto, entre os quais se destacam: tamanho do software, plataforma computacional, complexidade da aplicação, confiabilidade desejada para o software, experiência da equipe, metodologia de desenvolvimento utilizada, nível de testes requerido, estilo de interface com o usuário, grau de reutilização desejado, e disponibilidade de ferramentas de software. Como consequência, a estimativa de esforço e de prazo podem ser obtidas com mais precisão a partir de banco de dados histórico de projetos com dados sobre os atributos de cada projeto.

2.2 Técnicas para Estimativa de Custos

O gerenciamento dos custos do projeto inclui os processos envolvidos em estimativas, orçamentos e controle dos custos. Estimar os custos de um projeto significa desenvolver uma estimativa de custos dos recursos monetários necessários para executar as atividades do projeto (PMI, 2017). A estimativa é um prognóstico baseado na informação conhecida num determinado momento, e inclui a identificação e a consideração das alternativas de custo para iniciar e terminar o projeto. Compensações de custos e riscos devem ser consideradas.

As estimativas de custos devem ser refinadas durante o curso do projeto, e sua precisão aumentará conforme o projeto progride no seu ciclo de vida. Portanto, a estimativa de custos é um processo iterativo que representa uma avaliação quantitativa dos custos prováveis do projeto.

Estimar os recursos do projeto envolve a determinação da disponibilidade e quantidade necessária de pessoal e material para executar as atividades. Entretanto, outros aspectos devem ser considerados: (1) inclusão ou não de custos indiretos; (2) custos de mitigação de riscos devem ser computados; (3) fatores ambientais, tais como condições de mercado; e (4) ativos de processos organizacionais que incluem políticas e modelos de estimativa de custos, informações históricas e lições aprendidas.

A estimativa de custos do projeto usa ferramentas e técnicas semelhantes às ferramentas e técnicas usadas na estimativa dos prazos do projeto, ou seja, opinião especializada, estimativa análoga, estimativa paramétrica, estimativas de três pontos e análise das reservas. Adicionalmente, pode-se utilizar a estimativa *bottom-up*, que é um método para estimar custos a partir de pacotes de individuais trabalho, que em seguida são resumidos em níveis mais elevados.

Muitas vezes, quando faltam dados históricos para a estimativa de custos, é interessante optar por uma estimativa baseada em modelo paramétrico. Segundo Futrell, Donald, e Shafer (2002), entre os modelos paramétricos mais amplamente utilizados para a determinação do esforço destaca-se o COCOMO II (*Constructive Cost Model*). Este modelo estima o esforço, o prazo e a equipe média para as fases de elaboração e implementação do software.

3 Método da Pesquisa

A pesquisa proposta neste trabalho caracteriza-se por ser um estudo bibliográfico e exploratório. Segundo Cervo, Bervian e Silva (2007), a pesquisa exploratória visa oferecer informações sobre o objeto de estudo. Dessa forma, esta pesquisa explora as bases de dados científicas para identificar técnicas de estimação do esforço em projetos de software.

3.1 Coleta de Dados

A identificação de técnicas de estimação do esforço em projetos de software foi realizada por meio de uma revisão sistemática de literatura (RSL). A RSL é um método para identificar e analisar trabalhos disponíveis nas bases de dados científicas e responder a questões de pesquisa (Baptista & Campos, 2017). Para realizar a RSL foi utilizado o protocolo proposto por Kitchenham e Charters (2007). Esse protocolo estabelece estratégias de pesquisa para a estruturação do trabalho, e para identificação e avaliação dos materiais encontrados. O protocolo foi realizado em duas fases: planejamento e seleção dos trabalhos.

Planejamento. É constituído por dois itens: definição do objetivo e do protocolo de pesquisa. O objetivo desta RSL é identificar técnicas de estimação do esforço de desenvolvimento em projetos de software. A definição do protocolo foi realizada em três etapas apresentadas a seguir:

(1) Questões de pesquisa. As questões de pesquisa visam atender ao objetivo da RSL. Assim, as questões definidas para esta RSL foram:

Q1. Qual o papel e o impacto das estimativas de software no gerenciamento do projeto?

Q2. Quais os principais obstáculos à estimação acurada de esforço em projetos de software?

Q3. Quais as oportunidades de pesquisa inexploradas sobre estimação de esforço em projetos de software?

(2) Identificação dos estudos. Foi realizada uma busca ampla nas bases de dados por estudos que respondam às questões de pesquisa propostas. Esta RSL utilizou a base de dados Web of Science (https://apps.webofknowledge.com) e Scopus (https://www.scopus.com). A definição das palavras chaves considerou os termos "*Software Project Management*" e "*Effort Estimation*",

(3) Critérios de seleção dos estudos. Adotaram-se critérios de inclusão, exclusão e qualidade para a seleção dos artigos. Os trabalhos foram incluídos quando atenderam a todos os critérios de inclusão, mas eliminados ao satisfazer um dos critérios de exclusão. Os critérios definidos foram:

• Critérios de inclusão: (I1) área de estudo; e (I2) artigos publicados no período de 2007 a 2021.

• Critérios de exclusão: (E1) trabalho não é apresentado na sua totalidade em inglês; (E2) trabalho não é artigo em periódico ou conferência revisada por pares; (E3) artigo é de *workshop*, *lecture notes*, *work in progress* ou artigo curto; (E4) artigo não está disponível eletronicamente ou com restrições de acesso; e (E5) artigo não é um estudo primário.

• Critério de qualidade: (C1) foi utilizado a classificação de relevância do artigo baseado no motor de busca das bases de dados.

Seleção de Artigos. A busca na base foi realizada no primeiro semestre de 2021. A aplicação do termo de busca na base de dados *Web of Science* retornou 108 e na base Scopus 43 artigos. Todos trabalhos retornados pela Web of Science também foram retornados pela Scopus e por esse motivo, a etapa de aplicação de filtros foi feita inteiramente na Scopus. Aplicando os critérios de exclusão foram excluídos 10 artigos. Usando o critério de qualidade foram selecionados 15 trabalhos. Na última fase, avaliaram-se os 15 artigos pelo abstract, o que acabou resultando na exclusão de três trabalhos que não se adequaram ao objetivo desta pesquisa. Os 12 trabalhos selecionados nesta RSL estão apresentados no apêndice A.

4 Apresentação dos Resultados

Diversos trabalhos investigam a importância e o impacto das estimativas de esforço para o projeto. Segundo Hosni & Idri (2017), o sucesso do projeto está diretamente relacionado à acurácia das estimativas. Essa relação é tão forte que, segundo (Bilgaiyan, *et al.*,

2017), 15% dos projetos falham por causa de estimativas erradas e 60% deles excedem o orçamento pelo mesmo motivo. Segundo (Laqrichi *et al*., 2015), o excedente no orçamento fica entre 30% e 40%, também para cerca de 60% dos projetos, chegando até a 80% deles. Já segundo González-Carrasco *et al*. (2012), 2 a cada 3 projetos que falham tiveram estimativas ruins.

As estimativas influenciam o planejamento do projeto (Tronto *et al*., 2008) e, consequentemente, a alocação de recursos (Azzeh, 2011), a tomada de decisões (Laqrichi *et al*., 2015) e os prazos (Tronto *et al*., 2008; Azzeh, 2011; Laqrichi *et al*., 2015). Diante de tamanha importância, diversas técnicas foram desenvolvidas para estimar o esforço de desenvolvimento de software, mas todas elas usam abordagens em comum como redes neurais, análise de regressão e seleção de projetos análogos. A maioria dos autores categorizam esse tipo de trabalho em 3 ou 4 categorias: 1) baseado em aprendizado de máquina, 2) baseado em analogias, 3) baseado em métodos estatísticos e dados históricos e 4) baseado em julgamento de especialistas.

4.1 Métodos baseados em Aprendizado de Máquina

Métodos de estimação baseados em aprendizagem de máquina utilizam dados de projetos anteriores e técnicas de aprendizagem de máquina para aprender a estimar para um novo projeto, conseguindo realizar a atualização do modelo de predição ao longo do tempo (Idri, Zakrani, & Zahi, 2010; Park & Baek, 2008; Setiono, *et al*. 2010), o que reduz o gap entre a informação do modelo e a informação mais recente disponível (Tronto, *et al*., 2008). Além disso, esses métodos são notavelmente eficazes em mapear relacionamentos complexos, uma vez que possibilitam inúmeras alternativas de design (Ahmed & Muzaffar, 2009; Tronto *et al*., 2008; González-Carrasco *et al*., 2012). Alguns exemplos de técnicas empregadas para extrair funcionalidades dos dados são Redes Neurais, Lógica *Fuzzy* e Análise de Regressão.

Geralmente, diferentes designs de redes neurais são explorados e o melhor é escolhido para avaliação, contudo, a estrutura básica é a mesma. Tronto *et al*., (2008) propôs uma rede que tem uma camada, com 23 neurônios, na qual 17 variáveis, tais como o tamanho do *dataset*, tamanho do software e restrições de tempo, foram usadas para treiná-la. Na rede de Laqrichi, *et al*. (2015), o número de nós é de até duas vezes o número de entradas (variáveis para estimar), o que permite gerar uma grande variedade de redes. Os autores também avaliaram diferentes arquiteturas, mas apenas a que apresentou a melhor performance de generalização foi apresentada. Nesse caso de estudo, foram montados 9 neurônios da primeira camada, quatro na segunda e um na saída. Já González-Carrasco *et al*. (2012) usaram um algoritmo genético para determinar o número de neurônios e de camadas. A abordagem também usa lógica *fuzzy* para processar os dados que servem de entrada para a rede neural.

Apesar da flexibilidade de design das redes neurais, o que permite adaptação para diferentes contextos e o testes de várias alternativas, esse tipo de método é tendenciado pelos dados e é difícil de generalizar e de calibrar (Bilgaiyan, et al., 2017). Por conta disso, informações contextuais não são consideradas, diferentemente de um método de estimação por expert, por exemplo, onde o especialista consegue interpretar e utilizar tais dados em seu julgamento. Para superar essa carência, Karna & Gotovac (2014) propõem uma rede neural artificial que tenta estimar como um expert faria. São utilizadas 3 fontes de dados: o projeto, o gerenciamento da aplicação e a equipe responsável pela estimação. Também são empregados

técnicas de decomposição de features, de mineração de dados e 18 preditores. Contudo, apenas um único projeto é avaliado.

Redes Neurais também deixam de considerar muitos fatores que não são linearmente dependentes (Bilgaiyan, et al., 2017) e o seu resultado é difícil de justificar (Laqrichi, et al., 2015). O método de Regressão Linear, por sua vez, é um método paramétrico simples e por isso facilita o entendimento do processo de estimação. Basicamente, o método busca encontrar uma relação linear entre variáveis predictoras. As técnicas de Regressão Stepwise e Regressão Stepwise Forward foram aplicadas em (Tronto, et al., 2008). A primeira transforma as 17 variáveis consideradas pelo modelo enquanto a segunda seleciona as mais relevantes. Com esses valores calculados para 6 datasets, o esforço foi calculado em total de homem-horas para desenvolver os projetos. Já Tronto, et al. (2007) usaram a técnica de Regressão Stepwise Backward para calibrar o seu modelo de Regressão Stepwise. No geral, essa abordagem é sensível a valores fora da curva, mas Briand & Wieczorek (2002) propuseram uma técnica que consegue contornar esse problema.

Redes Neurais e Modelos de Regressão Linear foram comparados em (Tronto, et al., 2007; Tronto, et al., 2008). Apesar de serem ambas baseadas no mesmo tipo de abordagem, uma técnica não substitui a outra, uma vez que fatores como a natureza da função de custo também influenciam no sucesso da estimação. Além disso, datasets mais homogêneos podem favorecer a técnica de regressão enquanto um dataset desproporcional pode prejudicar a sua performance. As redes, por sua vez, lidam melhor com não-linearidade (Tronto, et al., 2007; Tronto, et al., 2008). Segundo Tronto, et al., (2007), pode ser vantajoso utilizá-las em conjunto.

4.2 Métodos baseados em Analogias

Partindo da premissa que a história se repete, projetos anteriores que foram concluídos de forma bem sucedida também podem ser utilizados para estimar o esforço para desenvolver um novo (Azzeh, 2011; Azzeh, *et al*., 2008; González-Carrasco, *et al*., 2012; Hosni & Idri, 2017; Idri & Abane, 2017; Keung, 2009). Os métodos de seleção de analogias selecionam projetos similares e extraem características (features) relevantes para então gerar uma estimativa de esforço para um novo projeto com base em dados históricos. O primeiro grande desafio é selecionar bem os projetos e as features que irão compor a estimação, seja por uma média simples ou um cálculo mais sofisticado. Em seguida, é necessário definir um número adequado de analogias para calcular o resultado final. Alguns estudos sugerem usar um número fixo (Briand, et al., 1999; Kocaguneli, et al., 2007; Mendes, et al., 2003; Walkerden & Jeffery), enquanto outros utilizam algoritmos (Azzeh, 2011; Mendes, *et al*., 2003), opinião especializada baseada em objetivos (Kocaguneli, *et al*., 2007) e *threshold* (Idri, Abran, & Khoshgoftaar, 2001).

O algoritmo de otimização de abelhas foi usado por Azzeh (2011) para gerar combinações do número de projetos semelhantes e o de funcionalidades, além de pesos para cada funcionalidade. Tais soluções foram usadas em uma função para minimizar a diferença entre os projetos históricos e o novo. O estudo de Azzeh *et al*. (2008), por sua vez, utilizou todos os projetos que não possuíam valores faltando, totalizando 477 projetos de dois *datasets*. Para selecionar o melhor subconjunto de funcionalidades, uma técnica baseada em analogia *Fuzzy* de cinco passos foi aplicada: 1) selecionar um subconjunto; 2) performar a

Fuzzificação dos dados; 3) avaliar a similaridade entre cada par de clusters no subconjunto; 4) avaliar a similaridade entre todos os clusters no subconjunto; e 5) escolher o melhor subconjunto, ou seja, o que tem o menor grau de similaridade (Ross, 2004). De forma semelhante, Idri e Abane (2017) propõem uma abordagem de lógica *Fuzzy* que escolhe apenas os projetos considerados como "altamente similares". Para tal, cada projeto é descrito em vários atributos, os atributos são medidos e classificados pelo nível de similaridade individual e global com o novo projeto e então os valores são estabelecidos em um intervalo [0,1], onde a proximidade a 1 é considerada como alta similaridade. Assim, todas as funcionalidades do projeto são usadas.

Por sua vez, Hosni e Idri (2017) propõem um método denominado *Ensemble Effort Estimation* (EEE), ou estimativa de esforço de conjunto, que combina duas técnicas de analogia: analogia clássica e subespaços aleatórios. A primeira permite selecionar os projetos e as funcionalidades e montar soluções. Foram buscadas as cinco analogias mais próximas (Azzeh & Elsheikh, 2017; Li, *et al*., 2007; Mendes, *et al*., 2003), usando a combinação de 4 estratégias de atribuição de pesos às features, tais como média e mediana, e 5 métodos para calcular a similaridade (por exemplo, distância Euclidiana e distância de Minkowski), totalizando um espaço de soluções de 100 (5 x 4 x 5). A aplicação de subespaços aleatórios permitiu encontrar a melhor combinação entre os métodos descritos anteriormente para os projetos de cada *dataset*.

A estimação por analogias é uma tentativa de simulação da estimação por expert. Assim como a estimação por expert depende do background de quem estima (Azzeh, *et al*., 2008; Bilgaiyan, *et al*., 2017; Laqrichi, *et al*., 2015), a estimação por analogias depende do *dataset* utilizado. Todavia, outras técnicas podem ser usadas em conjunto para atenuar o problema, como o *Case-Based Reasoning* (CBR), que lida bem com *datasets* barulhentos (Azzeh, 2011). Além do mais, é uma técnica vantajosa para mapear relacionamentos e fácil de entender pelos usuários (Azzeh *et al*., 2009; Li *et al*., 2007; Walkerden & Jeffery, 1999). Também já foi demonstrado que a sua acurácia aumenta quando outra técnica é utilizada em conjunto, como lógica *fuzzy* e algoritmos genéticos (Idri, Amazal, & Abran, 2015).

4.3 Métodos baseados em Modelos Estatísticos

Alguns métodos são baseados em equações elaboradas com base em dados históricos Tais equações compõem um modelo que calcula o esforço em função de parâmetros que influenciam esse esforço. Alguns exemplos incluem (Boehm, Abts, & Chulani, 2000; Hsu *et al*., 2010; Wang, Song, & Shen, 2007). Tais métodos enfrentam alguns desafios para alcançar uma estimação acurada, como a necessidade de considerar sua própria performance, suas experiências e técnicas, demonstrada por Kitchenham e Linkman (1997). Tais métodos precisam ser adaptados para cada projeto que se deseja estimar e o seu contexto (Tronto *et al*., 2008), mas devido à forte dependência aos dados históricos e, consequentemente, às organizações provedoras dos dados, se torna difícil realizar a adaptação (Ahmed & Muzaffar, 2009).

Para superar essas e demais limitações que qualquer método de estimação está sujeito, pode-se recorrer à utilização de mais de uma técnica, o que torna a estimação mais confiável e gera resultados mais precisos (MacDonell & Shepperd, 2003). Dessa forma, Hsu *et al*. (2010) propõem 3 combinações lineares diferentes para converter os resultados de diferentes técnicas

em uma estimativa de esforço final. A primeira combinação atribui o mesmo peso para todos os métodos. A segunda combinação atribui pesos maiores para os métodos cujo resultado está mais próximo da mediana, pois a mediana, em alguns casos, possui um potencial de moderador maior que a média, logo, pode ajudar a reduzir o erro na predição. A última combinação atribui pesos maiores para métodos que performaram melhor dentro de um critério significativo para a acurácia da estimação. A técnica consiste em selecionar os critérios mais decisivos para a acurácia da estimação (Bailey & Basili, 1981; Hsu & Huang, 2007) e, por meio de uma fórmula definida pelos autores, calcular um peso maior para os métodos que os favorecem. Dentre as 3 combinação, a última apresentou os melhores resultados.

Uma combinação de métodos estatísticos também é feita por Wang *et al*. (2007). O Modelo Cinza (Ju-Long, 1982), já utilizado para prever preços de ações (Wang, 2003), desempenho de pavimento (Shen & Du, 2004), dentre outros, é utilizado pelos autores para prever o esforço de desenvolvimento em sistemas de software. Para adaptar o Modelo Cinza para o problema, é necessário fazer algumas transformações. Adicionalmente, os autores usam uma correção de tendência, um parâmetro utilizado na predição. Os dados são treinados e a média de 80% dos dados medianos é usada como tendência. Tal tecnologia permite aumentar a acurácia do resultado. Uma vantagem do Modelo Cinza é que ele é concebido para lidar com cenários onde parte das informações é desconhecida. Devido a essa característica, os autores adaptaram o modelo para fazer a estimativa por fases, ao invés de estimar o esforço do projeto inteiro. Assim, o modelo consegue usar estimativas de fases iniciais para estimar fases posteriores e conseguiu mostrar resultados expressivos. Além disso, o método deixa aberta a possibilidade de desenvolver formas de corrigir as tendências, o que aumentaria ainda mais a precisão da estimativa.

4.4 Métodos baseados em Julgamento de Expert

A estimação baseada na intuição humana e na experiência e no nível de conhecimento do estimador ainda é a forma mais comum de prever o esforço de desenvolvimento de software (Britto *et al*., 2014; Jørgensen, 2004; Jørgensen & Grimstad, 2010). Tais métodos são mais fáceis de implementar (Karna & Gotovac, 2014), dado sua flexibilidade para lidar com informações em diferentes formatos (Moløkken & Jørgensen, 2005), e são mais precisos (Jørgensen, 2004). A facilidade para absorver informações contextuais que o expert possui é uma vantagem que outras estratégias, como modelos estatísticos, não possuem (Laqrichi, et al., 2015).

Ainda assim, essa forma de estimação também está sujeita a inconveniências, como erros humanos, decisões tendenciosas, influência de fatores externos como o ambiente organizacional do expert, além da bagagem de conhecimento acumulada pelo expert em projetos anteriores (Bilgaiyan, et al., 2017). O tipo de projeto também influencia o resultado final, sendo os projetos de tamanho pequeno e médio os mais indicados para a sua utilização (Boroujen, 2014). Apesar da facilidade para interpretar dados, um expert pode tomar decisões diferentes em contextos distintos e influenciados por tais fatores, mesmo tomando por base a mesma informação (Jørgensen, 2004; Jørgensen & Grimstad, 2010), o que torna o método menos confiável e difícil de replicar o resultado (Tronto, et al., 2008).

Contudo, o método pode ser mais útil se usado como outros modelos, como foi feito em (Karna & Gotovac, 2014) e exerce uma grande influência em outras técnicas, como a estimação por analogias, que tenta simular a estimação por expert além de outros métodos como Planning Poker e Técnica Delphi que usam a estimativa humana em algum nível para chegar ao consenso e compor o resultado final. Em todos esses casos, os dados históricos exercem um papel fundamental, uma vez que exercem uma medida de produtividade que irá influenciar as decisões do expert ou dos membros que julgam o esforço das atividades.

4.5 Seleção do Método

Diante de tantos fatores que podem influenciar a exatidão da previsão de esforço, se faz necessário selecionar bem o método que será utilizado para tal finalidade. Uma aplicação de Sistemas de Recomendação foi proposta por Peischl *et al*. (2009) para indicar o método de estimação mais adequado para o usuário. O sistema não necessita de dados do projeto, o que pode ser um problema, já que, como foi demonstrado pelos trabalhos revisados, o tipo do projeto, o seu tamanho e o contexto influenciam no resultado final, mas tais informações podem ser usadas se estiverem disponíveis. Uma base de conhecimento construída por meio de entrevistas e de revisões bibliográficas é utilizada para calcular o score para cada método, que varia de acordo com as informações inseridas pelo usuário. Apesar de escolher bem entre métodos diferentes, os autores ressaltam que o método não é tão preciso para o uso diário e também não explica o resultado final. Ademais, não foi avaliado se o método escolhido era de fato o mais indicado.

5 Discussão dos Resultados

Este trabalhou fez uma investigação sobre o estado da arte na área de estimação de esforço de software. O critério de qualidade utilizado para a seleção dos artigos considerou a classificação, por relevância, da ferramenta de busca. A discussão foi guiada pelas três perguntas da RSL, cujas respostas encontradas são apresentadas a seguir.

5.1 Papel e Impacto das Estimativas de Software no Gerenciamento do Projeto (Q1)

A diversidade de trabalhos no campo de estimação de esforço de desenvolvimento de software, sobre os quais essa revisão se debruçou, reforçam a importância de apoio para a execução dessa atividade, a importância dela para o processo de desenvolvimento e a persistência de desafios para essa tarefa. Alguns desses trabalhos apresentam números significativos de insucesso em projetos relacionados a erros nessa fase (Bilgaiyan, et al., 2017; González-Carrasco, et al., 2012).

A grande importância da estimação se dá pelo efeito que ela tem em outras atividades do projeto, como o planejamento, a alocação de recursos e o monitoramento e controle. Além disso, essa atividade influencia diretamente a tomada de decisão e os riscos do projeto, sendo fator crucial para aumentá-los ou mitigá-los.

A estimação inapropriada traz consequências negativas tanto quando o esforço é subestimado quanto quando é superestimado. No primeiro cenário, ela pode causar atrasos, mais custos, menor qualidade e insatisfação entre diversos stakeholders. No segundo caso, ela pode causar atrasos em projetos futuros e desperdício de tempo e dinheiro.

5.2 Principais Obstáculos à Estimação Acurada de Esforço em Projetos de Software (Q2)

PP2 - Quais os principais obstáculos à estimação de esforço acurada em projetos de software que persistem atualmente?

Uma série de fatores influencia o resultado da estimação de esforço e eles estão intrinsecamente relacionados ao tipo do método que é utilizado. Não existe um consenso na literatura sobre um método universalmente eficaz para estimar qualquer projeto e por isso é possível encontrar diferentes tipos de abordagens, como aprendizado de máquina, seleção de analogias e métodos de regressão. Assim, o primeiro desafio para a estimação é escolher o método adequado.

Além da escolha do método, influenciam no resultado final os dados que são utilizados, o tipo do projeto e o contexto. Existem ainda, outras fontes de incerteza para a estimação, como a falta de informação sobre o projeto, que raramente são consideradas pela maioria das ferramentas analisadas. Tais fatores representam mais fontes de obstáculo para a acurácia da técnica.

Os dados utilizados, por exemplo, podem ser imprecisos ou válidos apenas para alguns projetos e ainda precisam ser adaptados para o projeto que se deseja estimar. O contexto do projeto inclui relacionamentos que nem sempre são considerados pelos modelos de estimação, mas que podem influenciar no esforço final. Ademais, alguns métodos não são indicados para determinados tipos de projeto, como os baseados em expert que não são indicados para projetos grandes.

5.3 Oportunidades de Pesquisa Inexploradas sobre Estimação de Esforço em Projetos de Software (Q3)

Alguns trabalhos sugerem que a combinação de mais de uma técnica de estimação pode ser mais vantajosa que uma técnica única. Além de não haver consenso sobre um método aplicável para todas as situações, alguns trabalhos conseguiram obter bons resultados combinando seleção de analogias, regressão linear, entre outras. Uma possível direção de pesquisa seria aprofundar tais combinações e expandir esse tipo de metodologia para as demais técnicas já propostas pela literatura.

Outro ponto de investigação de importância aparente se dá no impacto das incertezas sobre a execução da estimação. A maioria dos trabalhos revisados sequer as cita. Acredita-se que considerar tais fontes na estimação, de forma a propor a mitigação delas, por exemplo, o processamento dos dados realizado em (Karna & Gotovac, 2014), pode proporcionar uma maior acurácia.

Por fim, trabalhos futuros que foquem em indicar um método adequado para estimar um projeto serão de grande importância, uma vez que existe uma gama de ferramentas para estimar, mas nem todas podem ser utilizadas por todos os projetos. A ausência de trabalhos, com exceção de poucos trabalhos (Peischl *et al.*, 2009), reforça a proeminência dessa direção de pesquisa.

6 Conclusão e Trabalhos Futuros

Este trabalho apresentou uma revisão bibliográfica na área de estimação de esforço de desenvolvimento de software. Os trabalhos encontrados foram divididos em três categorias de método de estimação: baseados em aprendizado de máquina, baseados em seleção de analogias e modelos estatísticos. Cada categoria foi explorada e suas características, vantagens, desvantagens e indicações foram relatadas. Uma quarta categoria de método de estimação, baseada em especialistas, foi incluída na revisão. Essa inclusão foi feita pois tais métodos são os mais utilizados atualmente e são constantemente referenciados por outros trabalhos das demais categorias que buscam minimizar seus pontos fracos. A importância do método baseado em especialistas foi verificada, pois outras abordagens tentam simulá-lo por meio de técnicas como aprendizado de máquina.

Com base na análise destes trabalhos, três perguntas de pesquisas foram respondidas, revelando os impactos, obstáculos e oportunidades de pesquisa sobre estimação de esforços no desenvolvimento de projetos de software. Primeiramente, foi evidenciada a importância da estimação do esforço para o processo de desenvolvimento de software e as consequências de uma condução inapropriada dessa atividade. Como demonstrado, cerca de 60% dos projetos excedem o orçamento e cerca de 15% deles falham quando não há uma estimativa correta do esforço. Além de influenciar o orçamento, essa atividade é crucial para a alocação de recursos, tomada de decisão, planejamento, e para o monitoramento e controle do projeto. Além disso, foram identificados obstáculos para a estimação precisa, tais como os dados utilizados, a escolha do método e o tipo do projeto. Por fim, foram apresentadas oportunidades de pesquisa nessa área, como a combinação de técnicas de estimação, o impacto das incertezas e a identificação do método adequado para estimação de cada projeto.

Cabe destacar que esta pesquisa apresenta limitações, das quais se destacam: o número reduzido de trabalhos que atenderam aos critérios da RSL, que pode ter privado a escolha de trabalhos mais atuais. Como trabalho futuro, sugere-se a investigação com foco maior nos riscos do projeto, e como eles podem ser tratados na estimação de esforço em projetos de software. Apesar de ser um fator chave para a acurácia da estimativa, poucos trabalhos o mencionam.

Referências Bibliográficas

Abran, A. (2006). Identification of the Structural Weaknesses of Function Point Metrics. Recuperado em 11 novembro, 2020, de https://bfpug.wordpress.com/artigos/

Ahmed, M. A., & Muzaffar, Z. (2009). Handling imprecision and uncertainty in software development effort prediction: A type-2 fuzzy logic based framework. *Information and Software Technology*, 51(3), 640-654.

Azzeh, M., & Elsheikh, Y. (2017). Learning best K analogies from data distribution for case-based software effort estimation. arXiv preprint arXiv:1703.04567.

Azzeh, M., Neagu, D., & Cowling, P. (2008). Improving analogy software effort estimation using fuzzy feature subset selection algorithm. *Proceedings of the 4th International Workshop on Predictor Models in Software Engineering*.

Azzeh, M., Neagu. D., & Cowling, P. (2009). Software effort estimation based on weighted fuzzy grey relational analysis. *Proceedings of the 5th International Conference on Predictor Models in Software Engineering*.

Azzeh, M. (2011). Adjusted case-based software effort estimation using bees optimization algorithm. *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, Berlin, Heidelberg.

Bailey, J. W., & Basili, V. R. (1981). A meta-model for software development resource expenditures. *Proceedings of the 5th international conference on Software engineering*.

Barcellos, M. P., & Travassos, G. H. (2004). Planejamento de Tempo e Custos em Ambientes de Desenvolvimento de Software Orientados à Organização. *III Simpósio Brasileiro de Qualidade de Software – SBQS*.

Baptista, M. N. & Campos, D. C. D. (2017). *Metodologias de Pesquisa em Ciências Análise Quantitativa e Qualitativa*, 2ª ed. LTC, Rio de Janeiro, Brasil.

Bilgaiyan, S. et al. (2017). A Systematic Review on Software Cost Estimation in Agile Software Development. *Journal of Engineering Science & Technology Review*, 10(4), 51-64 .

Boehm, B., Abts, C., & Chulani, S. (2000). Software development cost estimation approaches - A survey. *Annals of software engineering*, 10(1), 177-205.

Boroujen, I. (2014). A case study research on software cost estimation using experts' estimates, wideband delphi, and planning poker technique. *International Journal of Software Engineering and its applications*, 8(11), 173-182.

Briand, L. C., & Wieczorek, I. (2002). Resource estimation in software engineering. *Encyclopedia of software engineering*, 2, 1160-1196.

Briand, L. C. et al. (1999). An assessment and comparison of common software cost estimation modeling techniques. *Proceedings of the 1999 International Conference on Software Engineering (IEEE Cat. No. 99CB37002)*. IEEE.

Britto, R., Usman, M., & Mendes, E. (2014). Effort estimation in agile global software development context. *International Conference on Agile Software Development*, Springer, Cham.

Cervo, A. L., Bervian, P. A. & Silva, R. (2007). *Metodologia científica*, 6ª ed. São Paulo: Pearson Prentice Hall.

Futrell, R.T., Donald, F. S., & Shafer, L. I. (2002). *Quality Software Project Management*. Prentice-Hall.

González-Carrasco, I., et al. (2012). SEffEst: Effort estimation in software projects using fuzzy logic and neural networks. *International Journal of Computational Intelligence Systems*, 5(4), 679-699.

Hosni, M., & Idri, A. (2017). Software effort estimation using classical analogy ensembles based on random subspace. *Proceedings of the Symposium on Applied Computing*.

Hsu, C., & Huang, C. (2007). Improving effort estimation accuracy by weighted grey relational analysis during software development. *14th Asia-Pacific Software Engineering Conference* (APSEC'07). IEEE.

Hsu, C., et al. (2010). A study of improving the accuracy of software effort estimation using linearly weighted combinations. *IEEE 34th Annual Computer Software and Applications Conference Workshops*. IEEE.

Idri, A., Zakrani, A., & Zahi, A. (2010). Design of radial basis function neural networks for software effort estimation. *IJCSI International Journal of Computer Science*, issues 7(4).

Idri, A., Abran, A., & Khoshgoftaar, T. M. (2001). Fuzzy analogy: A new approach for software cost estimation. *International Workshop on Software Measurement*.

Idri, A., & Abnane, I. (2017). Fuzzy analogy based effort estimation: An empirical comparative study. *IEEE International Conference on Computer and Information Technology (CIT)*. IEEE, 2017.

Idri, A., Amazal, F.A., & Abran, A. (2015). Analogy-based software development effort estimation: A systematic mapping and review. *Information and Software Technology*, 58, 206-230.

Jørgensen, M., & Grimstad, S. (2010). *Software development effort estimation-Demystifying and improving expert estimation*. Simula Research Laboratory. Springer, Berlin, Heidelberg, 381-403.

Jørgensen, M. (2004). A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, 70(1-2), 37-60.

Ju-Long, D. (1982). Control problems of grey systems. *Systems & control letters*, 1(5), 288-294.

Karna, H., & Gotovac, S. (2014). Modeling expert effort estimation of software projects. *22nd International Conference on Software, Telecommunications and Computer Networks* (SoftCOM). IEEE, 2014.

Keung, J. (2009). Software development cost estimation using analogy: a review. *Australian Software Engineering Conference*. IEEE.

Kitchenham, B., & Linkman, S. (1997). Estimates, uncertainty, and risk. *IEEE Software*, 14(3), 69-74.

Kitchenham, B. & Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. *Technical Report EBSE 2007-001*, Keele University and Durham University Joint Report.

Kocaguneli, E., et al. (2007). TEAK: Learning Better Case Selection Strategies for Analogy Based Software Cost Estimation. *IEEE Transactions on Software Engineering*, 6(1).

Laqrichi, S., et al. (2015). Integrating uncertainty in software effort estimation using Bootstrap based Neural Networks. *IFAC-PapersOnLine*, 48(3), 954-959.

Li, J. et al. (2007). A flexible method for software effort estimation by analogy. *Empirical Software Engineering*, 12(1), 65-106.

MacDonell, S. G., & Shepperd, M. J. (2003). Combining techniques to optimize effort predictions in software project management. *Journal of Systems and Software*, 66(2), 91-98.

McConnell, S. (2006). *Software estimation: demystifying the black art*. Microsoft press.

Mendes, E. et al. (2003). A comparative study of cost estimation models for web hypermedia applications. *Empirical Software Engineering*, 8(2), 163-196.

Moløkken, K., & Jørgensen, M. (2005). Expert estimation of web-development projects: are software professionals in technical roles more optimistic than those in non-technical roles? *Empirical Software Engineering*, 10(1), 7-30.

Park, H., & Baek, S. (2008). An empirical validation of a neural network model for software effort estimation. *Expert Systems with Applications*, 35(3), 929-937.

Peischl, B. et al. (2009). Recommending effort estimation methods for software project management. *IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, vol. 3. IEEE.

PMI. (2017). *A Guide to the Project Management Body of Knowledge – PMBOK Guide*, 6th ed. Pennsylvania: Project Management Institute.

Ross, T. J. (2004). *Fuzzy logic with engineering applications*. John Wiley & Sons. Inc. New York, US.

Setiono, R. et al. (2010). Software effort prediction using regression rule extraction from neural networks. *22nd IEEE International Conference on Tools with Artificial Intelligence. Vol. 2*. IEEE.

Shen, D., & Du, J. (2004). Grey model for asphalt pavement performance prediction. *Proceedings of the 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No. 04TH8749)*. IEEE.

Tronto, I. F. B., Silva, J. D. S., & Sant'Anna, N. (2007). Comparison of artificial neural network and regression models in software effort estimation. *International Joint Conference on Neural Networks*. IEEE, 2007.

Tronto, I. F. B., Silva, J. D. S., & Sant'Anna, N. (2008). An investigation of artificial neural networks based prediction systems in software project management. *Journal of Systems and Software*, 81(3), 356-367.

Walkerden, F., & Jeffery, R. (1999). An empirical study of analogy-based software effort estimation. *Empirical software engineering*, 4(2), 135-158.

Wang, Y. (2003). On-demand forecasting of stock prices using a real-time predictor. *IEEE Transactions on Knowledge and Data Engineering*, 15(4), 1033-1037.

Wang, Y., Song, Q., & Shen, J. (2007). Grey prediction based software stage-effort estimation. *Wuhan University Journal of Natural Sciences* 12(5), 927-931.

Ziauddin, S. K. T., & Zia, S. (2012). An effort estimation model for agile software development. *Advances in computer science and its applications* (ACSA), 2(1), 314-324.

Apêndice A

A tabela 1 apresenta os artigos da RSL em ordem de relevância segundo a base dados Scopus.

Tabela 1:

Artigos selecionados na RSL

#	Título	Autores	Abordagem	Ano
1	Recommending effort estimation methods for software project management	Peischl, B. *et al.*	Sistema de Recomendação	2009
2	An investigation of artificial neural networks based prediction systems in software project management	Tronto, I. F. B., Silva, J. D. S. e Sant'Anna, N.	Aprendizado de Máquina	2008
3	Adjusted case-based software effort estimation using bees optimization algorithm	Azzeh, M.	Seleção de Analogias	2011
4	Improving analogy software effort estimation using fuzzy feature subset selection algorithm	Azzeh, M., Neagu, D. e Cowling, P.	Seleção de Analogias	2008
5	A study of improving the accuracy of software effort estimation using linearly weighted combinations	Hsu, Chao-Jung, *et al.*	Modelo Estatístico	2010
6	Integrating uncertainty in software effort estimation using Bootstrap based Neural Networks	Laqrichi, S., *et al.*	Aprendizado de Máquina	2015
7	Comparison of artificial neural network and regression models in software effort estimation	Tronto, I. F. B., Silva, J. D. S. e Sant'Anna, N.	Aprendizado de Máquina	2007
8	Fuzzy Analogy Based Effort Estimation: An Empirical Comparative Study	Idri, A. e Abnane, I.	Seleção de Analogias	2017
9	Modeling expert effort estimation of software projects	Karna, H. e Gotovac, S.	Aprendizado de Máquina	2014
10	Software effort estimation using classical analogy ensembles based on random subspace	Hosni, M. e Idri, A.	Seleção de Analogias	2017
11	Grey learning based software stage-effort estimation	Wang, Y., Qin-Bao S. e Jun-Yi S.	Modelo Estatístico	2007
12	SEffEst: Effort estimation in software projects using fuzzy logic and neural networks	González-Carrasco, I., et al.	Aprendizado de Máquina	2012

Nota. Fonte: dados da pesquisa.