Estimativa de Software: Análise das Técnicas e Experiências de Profissionais Freelancers.

Gustavo Henrique Martins Vieira[1]

[1]Instituto de Ciências Exatas e Informática –
Pontifícia Universidade Católica de Minas Gerais (PUC MG)
Ed. Fernanda. Rua Cláudio Manoel, 1.162, Funcionários, Belo Horizonte – MG – Brasil

ghmvieira@sga.pucminas.br

Abstract. *One of the biggest challenges encountered in the software development process is to complete projects with quality, within the deadlines and budgets provided. To deal with this, you can use software effort estimation techniques, as deadlines and budgets can be accurately estimated. This problem becomes even more critical when dealing with development by freelance professionals because there is no technique specifically proposed for this type of professional to estimate their projects. This study aims to investigate the best practices of software estimation developed by freelance professionals. The paper seeks to identify what are the main estimation techniques to be used by developers with different levels of experience and how much the developer experience impacts on the accuracy of this estimate.*

Resumo. *Um dos maiores desafios encontrados no processo de desenvolvimento de software é concluir projetos com qualidade, dentro dos prazos e orçamentos previstos. Para lidar com isso, pode-se utilizar técnicas de estimativa de esforço de software, pois assim os prazos e orçamentos podem ser estimados com precisão. Esse problema se torna ainda mais crítico ao se lidar com o desenvolvimento por profissionais freelancer, pois não há uma técnica especificamente proposta para esse tipo de profissional estimar seus projetos. Este estudo tem por objetivo investigar as melhores práticas de estimativa de software desenvolvido por profissionais freelancers. O trabalho busca identificar quais são as principais técnicas de estimativa a serem utilizadas por desenvolvedores com diferentes níveis de experiência e o quanto a experiência do desenvolvedor impacta na precisão dessa estimativa.*

Bacharelado em Engenharia de *Software* - PUC Minas
Trabalho de Conclusão de Curso (TCC)

Orientador acadêmico (TCC I): Laerte Xavier - laertexavier@pucminas.br
Orientador do TCC II: Rodrigo Baroni - profrodrigobaroni@gmail.com

Belo Horizonte, 10 de junho de 2022.

1. Introdução

Um dos aspectos mais importantes de um projeto de software é estimar o quanto de esforço o software demanda. A estimativa de esforço é necessária para produzir uma

estimativa de custos e um cronograma [Tsui et al. 2022]. Para isso, existem alguns modelos bastante conhecidos, tais como: Modelo de Custo Construtivo (COCOMO, do inglês *Constructive Cost Model*), a Análise por Pontos de Função (APF ou FPA, do inglês *Function Point Analysis*) e os Pontos de Casos de Uso (PCU ou UCP, do inglês *Use Case Points*). A métrica Pontos por Casos de Uso (PCU) foi proposta com o propósito de estimar recursos para projetos de software orientados a objeto, modelados por meio de especificação de Casos de Uso [Devmedia 2009]. A métrica Pontos de Função é uma medida de tamanho funcional de projetos de software, considerando as funcionalidades implementadas, sob o ponto de vista do usuário [Devmedia 2009]. O modelo básico COCOMO mede o esforço de desenvolvimento e custo em função do tamanho do software expresso em linhas estimadas de código (LOC, do inglês *Lines Of Code*). Neste trabalho, as técnicas de estimativa serão estudadas no contexto do desenvolvimento *freelancer* que são pessoas autônomas que têm uma associação de curto prazo baseada em tarefas com o empregador e, portanto, não fazem parte da força de trabalho da empresa [Gupta et al. 2020]

Um dos problemas mais enfrentados no desenvolvimento de software é conseguir estimar o projeto corretamente. A prática cotidiana mostra que muitas organizações de software ainda propõem custos de software irreais, trabalham dentro de cronogramas apertados e terminam seus projetos com atrasos e orçamentos superfaturados [Trendowicz and Jeffery 2014]. Diferente do desenvolvimento tradicional, não existe um modelo de estimativa de esforço de software especificamente proposto para estimar o esforço no contexto de desenvolvimento *freelancer* [Senevirathne and Wijayasiriwardhane 2020]. Diante disso, a estimativa se torna uma parte ainda mais crítica do desenvolvimento *freelancer*, pois não há uma forma do desenvolvedor prover uma estimativa precisa para o cliente. Por isso, o problema que este trabalho busca resolver **é a escassez de estudos sobre quais métodos de estimativa de esforço desenvolvedores *freelancers* de diferentes níveis de experiência devem usar**.

Para um projeto de software pequeno e bastante simples que envolve uma equipe de uma a três pessoas, estimar o esforço envolvido no desenvolvimento do projeto é relativamente fácil. Tanto os requisitos funcionais como os não funcionais do projeto são menores em número e complexidade, mesmo assim ainda não é uma tarefa trivial. Para sistemas grandes e complexos, capturar e entender os requisitos por si só pode ser desafiador. Estimar o esforço total e chegar a um cronograma de projeto confiável sob essas difíceis condições é uma das principais razões por trás de tantos fracassos em projetos de software. As estimativas e cronogramas de esforço imprecisos para sistemas grandes e complexos são muitas vezes extremamente otimistas, trazendo expectativas irreais em ambos, clientes e desenvolvedores desses sistemas [Tsui et al. 2022]. Portanto, é importante entender como desenvolvedores autônomos estimam seus projetos e investigar quais as melhores formas de se fazer essa estimativa sem uma ferramenta própria. Dessa forma, pode-se conseguir uma estimativa mais precisa e reduzir riscos dos projetos.

O objetivo deste trabalho é investigar as melhores práticas para se estimar o esforço/custo de um software *freelancer*. Para atingir esse objetivo são propostos os seguintes objetivos específicos: (i) descobrir como desenvolvedores autônomos de diferentes níveis de experiência estimam seus projetos; (ii) medir a eficácia dessas estimativas; (iii) relacionar a medição com as técnicas mais comumente utilizadas por cada nível de

experiência de desenvolvedores *freelancers*.

Espera-se descobrir quais são os modelos de estimativa mais utilizados entre os desenvolvedores autônomos e, a partir dessa descoberta, orientar os desenvolvedores quais são os modelos com maior taxa de sucesso a serem utilizados para cada nível de experiência. Com esses resultados, espera-se que este trabalho contribua com o processo de estimativa de custo/esforço de um software, tornando as estimativas mais precisas para desenvolvedores e clientes.

O restante deste trabalho está organizado em 4 seções. Na Seção 2, apresenta-se a fundamentação teórica. A Seção 3 contempla os trabalhos relacionados. Já a Seção 4 possui os materiais e métodos propostos.

2. Fundamentação Teórica

Nesta seção, são detalhados os principais conceitos e técnicas que estão relacionados à estimativa de esforço no desenvolvimento de software. São eles: o que é uma estimativa de software, o que são técnicas de estimativa e algumas dessas técnicas.

2.1. Estimativas de Software

Desde a década de 80, várias abordagens têm sido propostas e aperfeiçoadas com o objetivo de estimar o esforço empregado no desenvolvimento e o tamanho de um software. De acordo com Vazquez et al. (2003), o processo de estimativa de um projeto envolve, basicamente, três atividades:

1. Estimar o tamanho do produto a ser gerado;
2. Estimar o esforço empregado na execução do projeto;
3. Estimar a duração do projeto.

Para obter as respostas aos questionamentos iniciais de tempo e custo a partir da aplicação de um processo de estimativa, é necessário decidir a unidade para medir o tamanho do produto.

O processo de estimativa inicia-se a partir dos primeiros níveis de abstração dos requisitos do projeto. Para ser completo e eficiente, deve-se levar em consideração as experiências, os dados históricos de projetos passados e os recursos que cercam os projetos. A cada etapa realizada, as estimativas obtidas devem passar por um processo de aprovação antes de serem registradas na base histórica. Além de poderem ser utilizadas como fonte de informação para projetos futuros, podem ser utilizadas nas estimativas das etapas seguintes. Re-estimar deve ser uma atividade constante durante todo o ciclo de vida do desenvolvimento do software. Na etapa final do processo, com o produto concluído, as medidas reais de tamanho servem também para a validação e o aprimoramento de todo o processo de estimativa.

O termo estimativa é definido como uma medida aproximada da escala e carga do projeto necessária para o processo de estimativa de desenvolvimento. Após a verificação, recomendações técnicas e comerciais são feitas, planos detalhados e cronogramas são apresentados e o projeto é monitorado de forma eficaz.

2.2. Técnicas de Estimativas

Para se chegar à estimativa do esforço necessário para o desenvolvimento de software, existem muitas técnicas propostas na literatura. A seguir são apresentados algumas des-

sas técnicas, sendo elas: Pontos de Função, Pontos por Casos de Uso, *Planning Poker*, Comparação, Decomposição e Técnica de Avaliação e Revisão de Programas.

2.2.1. Pontos de Função

De acordo com o IFPUG (Grupo Internacional dos Usuários de Pontos de Função, do inglês *International Function Points Users Group*), Ponto de Função é uma unidade de medida padronizada usada para representar o tamanho de um software. O método de medição funciona quantificando os requisitos funcionais do software identificados, categorizando-os em cinco tipos: saídas, consultas, entradas, arquivos internos e interfaces externas. Uma vez que a função é identificada e categorizada em um tipo, ela é então avaliada para complexidade e atribuída uma série de pontos de função. Cada um dos requisitos funcionais do usuário mapeia para uma função de negócio do usuário final, como uma entrada de dados para uma entrada ou uma consulta de usuário para uma consulta. Essa distinção é importante porque tende a fazer com que as funções medidas em pontos de função mapeiem facilmente em requisitos orientados ao usuário, mas também tende a ocultar funções internas (por exemplo, algoritmos), que também exigem recursos para implementar. O número resultante é chamado uma contagem de Pontos de Função.

Devido ao propósito principal de um processo de estimativa ser o de fornecer informações que ajudem no planejamento e no controle dos projetos de software o mais cedo possível durante seu ciclo de vida, a utilização de pontos de função como unidade padrão de tamanho é a mais acertada. Isso se dá por uma série de motivos: softwares bem projetados ou com alto índice de componentização ou de reutilização de código têm seus indicadores baseados em LOC penalizados. Por exemplo, um sistema com 50.000 LOC, mas que exigiu um alto esforço para o desenvolvimento de uma arquitetura bem elaborada, irá apresentar uma estimativa final inferior à de outro projeto similar e de funcionalidade equivalente, porém com arquitetura menos trabalhada com 60.000 LOC. Já com pontos de função, essa distorção não seria observada, uma vez que não existiria diferença relevante no tamanho estimado de ambos os projetos [Carvalho 2001].

A análise de pontos de função fornece aos usuários métricas objetivas e com visão de longo prazo, de forma que a relação entre usuários e desenvolvedores pode ser transformada por meio dos requisitos de métodos de medição de tamanho de função padronizados, e os dados podem ser interpretados para permitir para melhor entendimento. A influência é representada pelo ponto de função, e o usuário define o valor aproximado para o propósito de medição ou ponto de função a fim de entender a diferença de medição [Carvalho 2001].

2.2.2. Pontos por Casos de Uso

A técnica Pontos por Casos de Uso (PCU) foi proposta por Gustav Karner em 1993 com o propósito de estimar recursos para projetos de software orientados a objeto, modelados por meio de especificação de Casos de Uso. A métrica é de fácil aplicação e não requer muito tempo de treinamento ou experiência prática. No entanto, o PCU somente pode ser aplicado em projetos de software cuja especificação tenha sido expressa em casos de uso. Além disso, como não existe um padrão único para a escrita de uma especificação de

caso de uso, diferentes estilos na escrita do caso de uso ou na sua granularidade podem levar a resultados diferentes na medição por PCU. Assim, a métrica se torna subjetiva. E ainda, devido ao processo de medição do PCU ser baseado em casos de uso, o método não pode ser empregado antes de concluída a análise de requisitos do projeto. Na maioria das vezes, existe a necessidade de se obter uma estimativa antes da finalização desta etapa. Nesses casos, esta métrica não é recomendada para utilização como unidade de medida das estimativas de tamanho de projetos de software, sendo melhor aproveitada em etapas mais avançadas do projeto. [Devmedia 2009]

2.2.3. Outros Métodos de Estimativa

Nesta seção, são apresentadas outras técnicas de estimativa que podem ser usadas por desenvolvedores:

- *Planning Poker*: A utilização dessa técnica é bem simples e pode ser utilizada em qualquer projeto. Para isso, é necessário um baralho com os números das cartas baseado na sequência de Fibonacci (1, 2, 3, 5, 8, 13, 21, 34 e 55). São necessários ao menos dois desenvolvedores que vão ler os requisitos e escolher uma carta que representa a quantidade de horas que ele levará para concluí-la. Se a tarefa for maior que isso, ela é considerada grande demais e deve ser quebrada em mais de uma tarefa. Ambos mostram as cartas e se forem diferentes os tempos, eles devem explicar os motivos. Em seguida, refazem o processo e se ainda sim forem divergentes, o maior valor é escolhido.

- Comparação: Geralmente é utilizada por profissionais bastante experientes e que já participaram de diversos projetos. Assim, a técnica de comparação pode ser empregada utilizando dados de projetos anteriores como base de planejamento atual. Para isso acontecer da forma correta, é preciso manter uma documentação consistente onde seja possível buscar informações de prazos anteriores.

- Decomposição: Essa técnica é bem simples, porém pode ser complexa e demorada quando se trata de projetos grandes. A ideia é dividir as atividades dentro do projeto até atingir tarefas mínimas que possam ser estimadas com maior exatidão. Pode-se citar o exemplo do cadastro de clientes, deve-se dividir em cadastrar, editar e excluir dessa forma, fica mais fácil de estimar o tempo real de cada parte. Para saber o tempo total do cadastro de cliente, basta somar todas as estimativas.

- PERT (Técnica de Avaliação e Revisão de Programas, do inglês *Program Evaluation and Review Technique*) ou Técnica dos três pontos: Para calcular a estimativa, basta utilizar a fórmula PERT que aplica um peso maior para a estimativa mais provável sem deixar de considerar a pessimista e a otimista. PERT = (Pessimista + 4 x Provável + Otimista) / 6 Exemplo: Expectativa otimista são 20 dias, a expectativa pessimista é de 35 dias e a provável são 25 dias, aplicando esses valores a fórmula: 35+25*4+20/6=25,83 Nesse caso, a estimativa PERT para a tarefa é de 25,83 dias. E pode-se calcular o Desvio Padrão, que indica o quanto a duração calculada na fórmula ainda poderá variar para mais ou para menos. Desvio Padrão:

Pessimista – Otimista / 6 Seguindo o exemplo, tem-se: Desvio Padrão: 35 – 20 / 6 = 2,5 dias Ou seja, a atividade pode variar 2,5 dias para mais ou para menos.

3. Trabalhos Relacionados

Nesta seção, são apresentados alguns trabalhos relacionados. Estes trabalhos envolvem a aplicação de técnicas de estimativa e avaliação em softwares desenvolvidos por *freelancers*, a análise de aplicações de técnicas de estimativa em manutenções de software em geral.

De acordo com Senevirathne and Wijayasiriwardhane (2020), a previsão de esforço leva em consideração vários fatores, que são diferentes no desenvolvimento *freelancer* e no desenvolvimento tradicional. Outra questão que atrapalha a previsão de esforço no desenvolvimento *freelancer* é que não há modelos de estimativa de esforço especificamente propostos para estimar o esforço de desenvolvimento de software autônomo. Diante desses fatores, o trabalho destes autores busca estender o modelo de estimativa de esforço baseado em pontos de caso de uso para o desenvolvimento de software *freelancer*. Para isso, foram coletados detalhes de projetos de software de código aberto de vários desenvolvedores de software *freelancer* e foram calculados os pontos de caso de uso para cada um desses projetos com base nos diagramas de caso de uso. Foram testados quatro modelos de adaptação para os valores estimados: a mediana da analogia mais próxima, a média ponderada de classificação inversa da analogia mais próxima, a média da analogia mais próxima e a analogia mais próxima. Por fim, concluiu-se que a "Média ponderada de classificação inversa da analogia mais próxima"teve o melhor resultado. O artigo se relaciona a este trabalho ao analisar a estimativa de software no contexto do desenvolvimento *freelancer*. A diferença é que o objetivo do artigo citado é criar uma nova técnica de estimativa a ser usada pelos desenvolvedores *freelancers*, enquanto que o objetivo deste trabalho é entender como o tempo de experiência influencia na precisão da estimativa.

Para Iqbal et al. (2020), a terceirização de desenvolvimento de software está se tornando cada vez mais comum e alguns estudos confirmam que uma grande proporção dos projetos de terceirização de desenvolvimento de software não consegue obter sucesso. Os autores citam que de acordo com investigações sobre as falhas desses projetos, em muitos casos, projetos de terceirização de desenvolvimento de software falham devido a problemas associados ao processo de engenharia de requisitos. Devido a essas questões, o objetivo dos autores é identificar e hierarquizar os problemas comuns do processo de engenharia de requisitos no caso de terceirização de desenvolvimento de software. Para isso, a literatura contemporânea foi avaliada rigorosamente, os problemas enfrentados pelos profissionais foram identificados e três questionários foram organizados e aplicados para profissionais terceirizados experientes. Com os resultados obtidos, os autores chegaram à conclusão que a maioria dos problemas estão relacionados à comunicação e à gestão e coordenação. O artigo se relaciona a este ao analisar problemas em um contexto de terceirização do desenvolvimento de software. A diferença é que o artigo citado busca por problemas gerais ocorridos durante todo o ciclo de desenvolvimento de software, enquanto que este trabalho busca entender problemas relativos à estimativa de esforço no desenvolvimento de software.

De acordo com Jørgensen and Grov (2021), a qualificação do desenvolvedor e o

tipo de contrato escolhido podem influenciar no resultado final do software. Com base nisso, o objetivo dos autores é testar duas hipóteses: a primeira é se o uso de *work-sample testing* (testes que verificam se o comportamento do colaborador e suas habilidades são consistentes com as características exigidas pela especificidade do trabalho) melhora a seleção de desenvolvedores de software qualificados; e a segunda é se o uso de contratos baseados em pagamento por hora leva a melhores resultados de projetos de software do que contratos de preço fixo. Para se chegar ao resultado, 57 desenvolvedores *freelancers* com experiência relevante e boas pontuações de avaliação de clientes anteriores foram convidados a concluir uma tarefa de avaliação de duas horas para se qualificar para um projeto de desenvolvimento de software. Assim sendo, 36 desenvolvedores concluíram a tarefa de *trialsourcing* com desempenho aceitável e, com base em um processo de alocação aleatória estratificada, foram solicitados a apresentar uma proposta baseada em pagamento por hora ou contrato de preço fixo. Os resultados encontrados mostram que embora o uso de *trialsourcing* possa ter impedido a seleção de desenvolvedores com habilidades insuficientes, o desempenho na tarefa de *trialsourcing* dos desenvolvedores selecionados, em grande parte, não previu seu desempenho nos projetos. O uso de pagamentos por hora parece ter levado a custos mais baixos do que os contratos de preço fixo, mas não a uma melhoria no projeto. O artigo citado se relaciona a este trabalho ao avaliar o contexto de desenvolvedores *freelancers*. A diferença está no fato do artigo citado avaliar como o contrato impacta no preço final e na qualidade do software, enquanto este trabalho busca avaliar o quanto a experiência influencia na precisão da estimativa de esforço.

Para Shafiq et al. (2020), a terceirização de desenvolvimento de software é um dos paradigmas de desenvolvimento mais populares na indústria de software e há uma série de desafios associados a esse paradigma, incluindo os desafios relacionados ao processo de engenharia de requisitos. Com essas informações, os autores buscam identificar os fatores de sucesso associados ao processo de engenharia de requisitos no contexto do desenvolvimento terceirizado. Para se chegar ao resultado, 25 fatores de sucesso foram descobertos por meio de uma revisão sistemática de literatura e validados através de um questionário. Os autores também examinaram os desafios da engenharia de requisitos identificados em relação à sua relevância para diferentes tipos e tamanhos de organizações. O artigo citado se relaciona a este ao analisar o contexto do desenvolvimento terceirizado. Diferenciam-se no fato de que o artigo citado avalia aspectos relacionados à engenharia de requisitos enquanto este trabalho busca analisar aspectos relacionados à estimativa de esforço.

De acordo com Khan et al. (2021), os projetos de GSD (Desenvolvimento de Software Global, do inglês *Global Software Development*) compreendem vários direcionadores de custos críticos que afetam o custo geral do projeto e a sobrecarga orçamentária. Por isso, há a necessidade de ampliar o modelo de estimativa existente nesse contexto para reduzir os riscos associados à sobrecarga de custos. Motivado por isso, os autores visam ampliar o modelo de estimativa COCOMO com direcionadores de custo do contexto do desenvolvimento de software global, para obter estimativas mais eficientes. Para se chegar ao resultado, foram realizadas as seguintes atividades: identificação de direcionadores de custos; categorização de direcionadores de custos; formação de métricas; atribuição de valores; alteração da equação do modelo base. Além disso, a eficácia do modelo proposto foi validada por meio da avaliação de especialistas, estudos de caso e Magnitude das Estimativas Relativas. As estimativas obtidas são eficientes, quantificadas e abrangem

aspectos do contexto em questão. Portanto, pode-se superar o risco geral do projeto GSD implementando o modelo. O artigo citado se relaciona a este ao estudar a estimativa de software no contexto do desenvolvimento global de software. Diferenciando-se no fato de que o artigo citado busca estender o modelo COCOMO para ser usado no contexto em questão, enquanto este trabalho busca entender como a experiência de um desenvolvedor *freelancer* afeta a precisão de sua estimativa.

4. Materiais e Métodos

O estudo proposto neste trabalho é composto por três etapas: a primeira é classificada como pesquisa qualitativa, a segunda como pesquisa quantitativa de natureza experimental e a terceira visa relacionar o resultado obtido nas duas primeiras etapas. Busca-se, por meio de um questionário, descobrir como os desenvolvedores *freelancers* estimam seus projetos e, após isso, por meio de de métricas quantitativas, busca-se identificar os impactos da experiência do desenvolvedor *freelancer* ao se estimar um projeto. Para tanto, são avaliadas quais modelos são mais utilizados pelos desenvolvedores, seu tempo de experiência e quais as taxas de erro das estimativas dos desenvolvedores.

4.1. Procedimentos

Este trabalho é composto por três etapas. Na primeira etapa, é utilizado um questionário que tem como objetivo investigar quais são as técnicas de estimativa mais utilizadas pelos desenvolvedores *freelancers* e quais mais utilizadas por nível de experiência. Para a segunda, é realizado um experimento composto pelas seguintes atividades: selecionar um pequeno software, fazer a estimativa dele utilizando a técnica de pontos de função junto a um especialista e acompanhar os desenvolvedores *freelancers* enquanto eles estimam esse mesmo software utilizando uma técnica escolhida pelo autor. A etapa tem como objetivo coletar métricas relacionadas ao processo de estimativa de software, tais como: eficácia da estimativa (tempo estimado pelo desenvolvedor em relação ao tempo estimado pelo especialista) e facilidade de uso da técnica. Em seguida, os resultados do experimento são relacionados às respostas obtidas com o questionário, com o objetivo de identificar, para cada nível de experiência, qual a técnica mais utilizada e qual a que obteve melhores resultados.

As etapas são realizadas com desenvolvedores *freelancers* e alunos do curso de Engenharia de Software da Pontifícia Universidade Católica de Minas Gerais (PUC Minas) que trabalham com desenvolvimento *freelancer*. O questionário será livre para qualquer desenvolvedor *freelancer* que quiser responder, já o experimento será executado com cerca de 6 a 9 desenvolvedores para cada nível de experiência. Esses desenvolvedores são separados em 3 grupos de acordo com seu tempo de experiência, sendo eles: 3 anos, de 3 a 6 anos e acima de 6 anos. O detalhamento do questionário encontra-se no Apêndice A.

Para a realização do experimento, inicialmente, os desenvolvedores informam seu nível de experiência e quais técnicas costumam utilizar. Os desenvolvedores são classificados de acordo com seu tempo de experiência. Então, o desenvolvedor faz a estimativa do software selecionado utilizando de uma técnica escolhida pelo autor dentre as mais citadas no questionário para aquele tempo de experiência. Após a estimativa feita, ela é comparada com a estimativa feita junto a um especialista para se avaliar a taxa de erro/eficácia da sua estimativa.

4.2. Etapas da Pesquisa

A execução da pesquisa consiste nas seguintes etapas:

1. Disponibilização e divulgação do questionário;
2. Seleção do software/requisito a ser estimado;
3. Estimativa do software/requisitos junto a um especialista;
4. Seleção dos desenvolvedores *freelancers* para participarem do experimento;
5. Realização do experimento;
6. Comparação dos resultados obtidos com o experimento com a estimativa realizada pelo especialista;
7. Comparação dos resultados do experimento com as respostas do questionário;
8. Escrita e revisão final do artigo.

As etapas são planejadas de acordo com o cronograma apresentado na Tabela 1.

Tabela 1. Cronograma de atividades

Etapas	Meses						
	Agosto		Setembro		Outubro		Novembro
	1	2	1	2	1	2	1
1	x						
2	x						
3		x					
4		x	x				
5			x	x			
6					x		
7					x		
8						x	x

4.3. Métricas e Avaliação

Conforme descrito na Seção 4.1, neste trabalho são realizadas três etapas. Para estas etapas, são utilizadas métricas relacionadas à experiência do desenvolvedor, à técnica que ele costuma utilizar e à eficácia dessas técnicas. Essas métricas são descritas da seguinte maneira:

- Experiência: É o tempo de experiência do profissional como desenvolvedor *freelancer*. Os desenvolvedores são divididos em grupos de acordo com o tempo de experência, os grupos são: desenvolvedores que tem até 3 anos de experiência no desenvolvimento *freelancer*, desenvolvedores com 3 a 6 anos de experiência e desenvolvedores que têm mais de 6 anos de experiência. Neste trabalho, utiliza-se essa métrica para classificar os desenvolvedores.

- Tempo estimado: É o tempo necessário para se desenvolver um software obtido com a estimativa. Neste estudo, essa métrica é utilizada com o objetivo de avaliar a eficácia da estimativa feita pelo desenvolvedor *freelancer*.
- Eficácia da técnica: É a margem de erro da estimativa do desenvolvedor em relação à estimativa feita pelo especialista. Quanto menor o valor, melhor é a estimativa feita pelo desenvolvedor. Neste trabalho, utiliza-se essa métrica para avaliar quais são as técnicas mais eficazes para cada nível de experiência.

Referências

Carvalho, A. M. (2001). *Introdução à engenharia de software*. Ed. da Unicamp.

Devmedia (2009). Análise de ponto de função.

Gupta, V., Fernández-Crehuet, J. M., and Hanne, T. (2020). Freelancers in the software development process: A systematic mapping study. *Processes*, 8:1215.

Iqbal, J., Ahmad, R. B., Khan, M., Alyahya, S., Nizam Nasir, M. H., Akhunzada, A., and Shoaib, M. (2020). Requirements engineering issues causing software development outsourcing failure. *PloS one*, 15(4):e0229785.

Jørgensen, M. and Grov, J. (2021). A field experiment on trialsourcing and the effect of contract types on outsourced software development. *Information and Software Technology*, 134:106559.

Khan, J. A., Khan, S. U. R., Khan, T. A., and Khan, I. U. R. (2021). An amplified cocomo-ii based cost estimation model in global software development context. *IEEE Access*, 9:88602–88620.

Senevirathne, D. S. and Wijayasiriwardhane, T. K. (2020). Extending use-case point-based software effort estimation for open source freelance software development. In *2020 International Research Conference on Smart Computing and Systems Engineering (SCSE)*, pages 188–194.

Shafiq, M., Zhang, Q., Azeem Akbar, M., Alsanad, A., and Mahmood, S. (2020). Factors influencing the requirements engineering process in offshore software development outsourcing environments. *IET Software*, 14(6):623–637.

Trendowicz, A. and Jeffery, R. (2014). *Software Project Effort Estimation*, pages 349–364.

Tsui, F., Karam, O., and Bernal, B. (2022). *Essentials of software engineering*. Jones & Bartlett Learning.

Vazquez, C. E., SIMÕES, G. S., and Albert, R. M. (2003). Análise de pontos de função. *São Paulo: Érica*.

A. Apêndice A. Questionário

1. A quanto tempo você trabalha com desenvolvimento freelance?
2. Você tem certificação em alguma técnica de estimativa de esforço de software?
3. Qual a técnica de estimativa de tamanho você utiliza?
4. Quais ferramentas de estimativa de tamanho você utiliza?
5. Como você define quantas horas serão gastas no projeto?
6. Qual a quantidade média de projetos você faz por ano?
7. Qual a quantidade média dos projetos que você percebe que serão gastas uma quantidade de horas diferente da estipulada inicialmente?