Capítulo

8

Como estimar um Software?

Métricas para aferição de Esforço, Prazo e Custo de um Produto de Software

Washington Henrique Carvalho Almeida, Fernando Escobar, Luciano de Aguiar Monteiro, Aislan Rafael Rodrigues Souza e Sérgio Sierro Leal

Abstract

This chapter deals with a recurring question for students, professionals and entrepreneurs in information technology area: how to estimate effort, time and cost related to software development? An overview of usable metrics will be presented for the proper estimation, based on the necessary effort, of the time and cost for systems development. Besides, basic concepts about metrics and a pricing process based on the CEK model will be presented, a proposal for using Canvas, EAP and Kanban.

Resumo

Este capítulo versa sobre uma dúvida recorrente para alunos, profissionais e empreendedores da área de tecnologia da informação: como estimar o esforço, prazo e custo relacionados ao desenvolvimento de um software? Serão apresentados conceitos básicos sobre métricas, listadas as principais características inerentes à formação de preço de software e as métricas para sua correta precificação. Além disso, será detalhado o Modelo CEK, que utiliza Canvas, EAP e Kanban, para orientar o detalhamento do exemplo de precificação de um projeto de software.

8.1. Introdução

O projeto de sistemas é uma atividade da Engenharia de Software estreitamente relacionada com o planejamento e o resultado desta é um produto de software. Todavia, a indústria de desenvolvimento prima por dois aspectos: softwares com qualidade e com um custo mensurável, estas duas perspectivas são contempladas por meio de um conceito usado em diversas engenharias, a medição. Por exemplo, para se compreender como está a qualidade de um código desenvolvido deve-se avaliar a porcentagem de erro por linha

de código e aplicando esse mesmo raciocínio, para se estimar o tamanho de um software ou o seu custo, deve-se definir e empregar alguma métrica.

A área de conhecimento da Engenharia de Software responsável pelos aspectos de medidas de sistemas é a de métrica de software e seu foco principal é usar métricas quantitativas para gerenciar o ambiente de desenvolvimento de software na organização, fornecendo uma visão interna da qualidade do produto de software [O'Regan 2014].

Um questionamento extremamente importante e oportuno em uma fase inicial de planejamento é o que medir em um projeto de sistemas, considerando que o termo "métrica de software" pode abranger muitas atividades, todas envolvendo algum grau de mensuração de software. Neste sentido, [Fenton e Bienan 2014] evidenciaram em seus estudos as principais métricas, elencadas a seguir.

- Modelos e medidas de estimativa de custos e esforços
- Coleção de dados;
- Modelos e medidas de qualidade;
- Modelos de confiabilidade;
- Métricas de segurança;
- Métricas estruturais e de complexidade;
- Avaliação de maturidade de capacidade;
- Gerenciamento por métricas; e,
- Avaliação de métodos e ferramentas.

Dentro dos modelos e medidas de estimativa de custos e esforços, uma das principais dificuldades de desenvolvedores, mesmo os mais experientes, é como estimar o custo de um software. Essa problemática impacta diretamente no planejamento do projeto de sistemas, especificamente em relação ao *lead time* (tempo total entre o início e fim para desenvolvimento de um item do *backlog* do projeto), a quantidade de profissionais envolvidos no projeto e, em decorrência, seu custo final. É importante destacar que um planejamento equivocado nesta fase implicaria no insucesso no projeto e prejuízos financeiros para a empresa.

Como medida para mitigar esta problemática, o presente capítulo visa apresentar uma visão geral sobre as principais métricas utilizadas para estimar software com base no esforço necessário, do prazo e do custo. A abordagem proposta apresentará as principais características da formação de preço de software, incluindo abordagem ágil.

Complementarmente, será contextualizada uma forma de precificação utilizando o Modelo CEK, a partir da integração das ferramentas Canvas de Projeto, Estrutura Analítica do Projeto (EAP) e Kanban, na qual o projeto é norteado desde a sua concepção, planejamento e acompanhamento de sua execução, de forma colaborativa, ágil, controlável e efetiva [Escobar et al. 2019]. O Modelo CEK, proposto pelos autores, foi apresentado pela primeira vez no VII Escola Regional de Computação Ceará, Maranhão, Piauí (ERCEMAPI 2019).

Concluindo, é exemplificada a proposta a partir do exemplo de um projeto de software, o Projeto Moto Easy e a criação de seu mapa de custos.

Esse capítulo está organizado da seguinte forma: 10.1 – Introdução; 10.2 – Definições; 10.3 – Principais Características da Formação de Preço de Software; 10.4 –

Métricas para Precificação; 10.5 – Modelo CEK; 10.6 – Exemplo de aplicação; e 10.7 – Conclusão.

8.2. Definições

Medir software é uma premissa essencial em projetos de sistemas. Desenvolvedores medem as características de seu software para entender se os requisitos são consistentes e completos, se o design é de alta qualidade e se o código está pronto para ser lançado. Gerentes de projeto medem atributos de processos e produtos para fazer projeções de quando o software estará pronto para entrega e se o orçamento será excedido; complementarmente, organizações usam medições de avaliação de processos para selecionar fornecedores de software [Fenton and Bieman 2014].

Gerentes de projetos possuem como proposição básica que o resultado de um projeto é único e exclusivo; esta proposição, por conseguinte, se aplica aos projetos de desenvolvimento de sistemas. Cada projeto de software compartilha desafios distintos relacionados à tecnologia, pessoas e cronogramas. [Bhardwaj and Rana 2016] destacam em seu estudo, que um dos principais desafios enfrentados pelo gerente de projetos de software é identificar as principais métricas de software para controlar e monitorar a execução do projeto.

Os autores explicam, ainda, que cada projeto de desenvolvimento de software, apesar de único e exclusivo, compartilha algumas métricas comuns que podem ser usadas para controlar e monitorar a execução do projeto. Essas métricas são tamanho de software, esforço, duração do projeto e produtividade. Proporcionam, desta forma, uma visibilidade ao gerente de projeto sobre os seguintes aspectos: o que entregar (tamanho), o que foi entregue no passado (produtividade) e quanto tempo levará para entregar mantida a capacidade atual da equipe (tempo e esforço).

A métrica de software busca definir um método padrão para medir certos atributos do processo, produto ou serviço, que se traduzem em indicadores chave de desempenho (KPI). [Jethani 2013] explica que a medição de atributos ajuda a obter informações valiosas sobre problemas e processos, permitindo identificar e gerenciar riscos, ajudando na detecção e resolução precoces de problemas. A medição de atributos fornece informações que melhoram a tomada de decisão objetiva no prazo.

Uma métrica pode possuir sua telemetria baseada em uma única origem ou pode ser uma combinação de dados de várias fontes. Qualquer ponto de dados rastreado acaba se tornando uma métrica que pode ser usada para medir o desempenho de uma equipe. [Davis 2015] destaca alguns exemplos destas métricas, conforme a seguir.

- Velocidade - o desempenho relativo de sua equipe ao longo do tempo
- Linhas de código alteradas (CLOC) - o número de linhas de código que foram alteradas
- Hora-Extra – quantidade de horas trabalhadas a mais, que não estavam previstas no projeto.

Métricas de Software, conforme abordado anteriormente, possuem vários objetivos, como avaliar a qualidade da aplicação e desempenho da equipe. Todavia, o

foco do presente capítulo é estimar software, a partir do qual, algumas métricas específicas devem ser analisadas com mais detalhes.

- Tamanho do software é a medida da funcionalidade do software que está sendo entregue ou espera-se que seja entregue. O tamanho do software é uma medida numérica dos requisitos de sistemas que são definidos qualitativamente pelo usuário, depende apenas do que entregar e não de como entregar. O tamanho do software é uma medida numérica dos requisitos funcionais [Bhardwaj and Rana 2016].
- Esforço é o tempo acumulado gasto por toda a equipe do projeto. O esforço geralmente mede horas pessoais, meses pessoais ou dias pessoais, mas o horário pessoal é a unidade mais adequada e inequívoca, já que outras unidades requerem conversão de hora para dia ou mês. Essa conversão não é padrão porque em alguns países (países especialmente desenvolvidos) há 8 horas de trabalho por dia, enquanto em alguns países (países em desenvolvimento) são mais de 8 horas [Bhardwaj and Rana 2016].
- Tempo representa a duração do calendário do projeto. Ele informa as datas esperadas de início e término do projeto. Não há relação linear entre tempo e esforço. O gerente de projeto deve entender que podemos reduzir a duração geral do projeto adicionando mais membros da equipe, mas além de certo ponto, o aumento no tamanho da equipe não resultará na redução da duração do projeto, mas aumentará a duração do projeto [Bhardwaj and Rana 2016].

Projetos ágeis de desenvolvimento de software possuem algumas características específicas, pois os requisitos podem ser alterados ao longo da evolução do projeto, tornando-se desta forma necessário usar métricas distintas ou adaptadas dos projetos de desenvolvimento tradicionais. Para [Padmini, Dilum and Perera 2015], embora algumas das métricas tradicionais possam ser adaptadas para uso no desenvolvimento ágil, algumas não são adequadas. Isso ocorre devido ao processo ágil de desenvolvimento ser iterativo e incremental, que está disposto a incorporar mudanças ao longo do processo.

Em uma Revisão Sistemática (RS) de [Canedo and Da Costa 2018], a questão principal versou sobre métricas e métodos usados para fazer estimativas de esforço, prazos e custos para o planejamento ágil de projetos de software. Os autores chegaram às métricas apresentadas a seguir.

- Story Point é uma métrica usada no gerenciamento e desenvolvimento ágil de projetos para determinar (ou estimar) o esforço (dificuldade) de implementar uma determinada história. Nesse contexto, uma história é uma necessidade comercial específica atribuída à equipe de desenvolvimento de software;
- Estimativa de esforço utiliza o Planning Poker, uma das técnicas mais populares para equipes ágeis, no planejamento e estimativa de esforço antes de iniciar cada iteração;
- Estimativa de tamanho são agrupadas com as técnicas de estimativa de esforço porque, no contexto do desenvolvimento ágil, a estimativa de esforço geralmente é derivada das estimativas de tamanho e dos cálculos de velocidade.

[Canedo and Da Costa 2018] ainda evidenciaram em sua RS dois trabalhos bem inovadores: (1) na área de **estimativas de projetos ágeis**, que sugere uma estrutura que

depende do uso da lógica fuzzy e visa ajudar na produção de estimativas precisas, propondo-se modificar o método de pontos de caso de uso (UCP) para torná-lo adequado para o desenvolvimento ágil de software, nomeando o novo método como a versão interativa do UCP (iUCP); e, (2) na área de **previsão de esforço** derivado de por mudanças nos requisitos de software. O modelo integra a análise do impacto das mudanças com o modelo de estimativa de esforço COCOMO, para melhorar a precisão das estimativas de esforço decorrentes de mudanças em projetos de desenvolvimento ágil de software.

8.3. Principais Características da Formação de Preço do Software

Em regra, quando estamos em processo de formulação do preço de um software para algum cliente, costumamos calcular apenas o custo do desenvolvimento e o lucro do desenvolvedor, conforme [Sommerville 2019]. Entretanto, nem sempre o relacionamento entre o custo e o preço final costuma ser tão simples assim.

De acordo com [Pereira 2004], precificar software é uma arte, os produtos de software podem ser vendidos por unidade/licença, pelo número de usuários simultâneos, número de servidores, por upgrades (atualização de versão), dentre outras formas. A estratégia do preço a ser utilizada é impactada pelas características de como o software será vendido, além de impactar a competividade entre as empresas fabricantes de software.

O processo de precificação requer das empresas uma constante análise e monitoramento dos fatores que influenciam o ambiente interno e externo. Podemos observar alguns desses fatores na Tabela 1.

Tabela 1 - Fatores que afetam a precificação. Fonte: [Sommerville 2019] adaptado.

Fator	Descrição
Termos contratuais	Um cliente pode permitir que o desenvolvedor mantenha a propriedade do código-fonte, o que possibilitaria o reuso em outro projetos.
Incerteza da estimativa de custos	Quanto mais incerto o cenário, uma organização pode aumentar o percentual de contingência além do seu lucro.
Saúde financeira	Empresas com problemas financeiros podem baixar seu preço para ganhar um contrato. É melhor baixar o lucro ou trabalhar em lucro zero do que a falência.
Oportunidade de mercado	Pode haver cotação baixa do preço para entar em um novo segmento de mercado, por exemplo na área pública, conseguir um atestado de capacidade poderá dar chance de novos contratos futuros com melhores percentuais de lucro.
Volatilidade de requisitos	Se houver mudança de requisitos previsíveis, a organização pode baixar seu preço e cobrar altos preços para incorporar mudanças posteriores.

Durante o processo de precificação, devemos levar em consideração aspectos organizacionais, econômicos, políticos e comerciais, além daqueles mostrados na Tabela 1. Possíveis riscos podem alterar o preço do software, seja para cima ou para baixo. Além disso, devemos levar em consideração que, a decisão sobre o preço do projeto deve ser

realizada em grupo, envolvendo as equipes de marketing e vendas, e demais gerências envolvidas na concepção do software.

O preço de um software costuma ser definido a partir de uma proposta preliminar, que será formulada com base nos fatores apresentados na Tabela 1. A partir disso, as negociações acontecem entre cliente e fornecedor, com o objetivo de estabelecer a especificação detalhada do software.

Em muitos projetos de desenvolvimento de software, os fatores fixos não costumam ser os requisitos do projeto, mas sim, requisitos para a precificação. Para que o preço não exceda o custo esperado pelo cliente, os requisitos podem ser alterados [Sommerville 2019].

Por exemplo, digamos que uma empresa esteja oferecendo um contrato para o desenvolvimento de um sistema de controle de frotas de veículos, seja para uma companhia de frete ou para uma companhia que possua uma frota de veículos própria, no qual o objetivo seja acompanhar e monitorar as atividades rotineiras.

Neste ponto, ainda não foi possível levantar, de forma detalhada, todos os requisitos necessários para a concepção do sistema, mas mesmo assim, a empresa apresenta uma proposta com um valor estimado em R$ 800 mil, imaginando que seja um valor competitivo perante as demais fabricantes de software concorrentes, e que esteja dentro do orçamento do cliente.

Uma vez o contrato celebrado entre o cliente e o fornecedor, os requisitos detalhados do sistema são negociados, para que a funcionalidade básica seja entregue. Durante essa etapa, custos adicionais são estimados devido a outros requisitos adicionais, caso extrapole o preço inicial, mas não prejudique a funcionalidade básica negociada durante a celebração do contrato, esses requisitos adicionais podem ser financiados em um futuro orçamento. Dessa forma, o cliente evita que seu orçamento seja consumido mais do que o inicialmente alocado, não aumentando os custos para a compra do software necessário.

Como vimos no exemplo acima, o custo de um projeto costuma ser fracamente relacionamento ao preço indicado para o cliente, devido a isso, uma estratégia que é comumente usada é “preço para ganhar” [Sommerville 2019]. Definir “preço para ganhar” significa que uma empresa tem uma ideia do preço que o cliente espera pagar e, com base nisso, uma oferta de contrato é feita com base no preço esperado pelo cliente. Depois disso, os custos adicionais para outros requisitos são estimados, para serem adicionados, ou não, na composição do contrato.

8.3.1. Modularidade e Custo do Software

De acordo com [Pressman 2016], modularidade é a manifestação mais comum da separação por interesses. Assim, um software baseado em modularidade é dividido em componentes separadamente especificados e localizáveis, algumas vezes denominados módulos, que são integrados para satisfazer os requisitos de um problema.

Um software monolítico, cuja definição remete a um grande software composto de um único módulo, não costuma ser entendido de forma fácil por um engenheiro de

software. O número de variáveis e complexidade geral torna o entendimento de um software monolítico quase impossível.

Como consequência, caso o engenheiro de software decida seguir um modelo monolítico no desenvolvimento do software, a entrega para o cliente dependerá que todas as funcionalidades e requisitos estejam implementados. Seguir esse tipo de modelo demanda um maior tempo de desenvolvimento, algo intimamente relacionado aos custos que serão necessários para finalizar o produto acordado durante a concepção do contrato. Quanto maior o tempo, maior serão os custos, maior será o preço repassado ao cliente, o que pode tornar o contrato inviável, levando a uma possível rescisão.

[Pressman 2016] nos mostra que, em quase todos os casos, devemos dividir o projeto em vários módulos para facilitar a compreensão e, consequentemente, reduzir os custos necessários para construir o software esperado pelo cliente. Entretanto, é preciso ter cuidado com o número de módulos que serão utilizados para dividir o software que está em desenvolvimento. A Figura 1 demonstra a relação entre modularidade e custo do software.

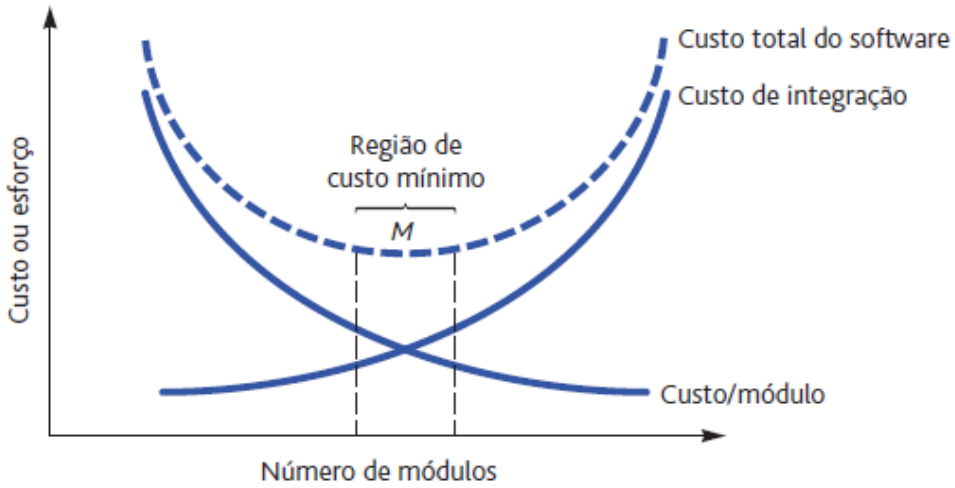

Figura 1 - Modularidade e custo do software. Fonte: [Pressman 2016].

Conforme a Figura 1, o esforço que será demandado para o desenvolvimento de um respectivo módulo individual do software tende a reduzir conforme o número total de módulos aumenta, dividindo o volume de trabalho na codificação. Além disso, é possível entregar ao cliente, de forma gradativa, os requisitos estabelecidos em contrato, conforme os módulos vão sendo finalizados, validados e testados. Dessa forma, segundo [Pressman 2016], dado um mesmo conjunto de requisitos, mais módulos significa um tamanho individual menor.

Por outro lado, é possível analisar um outro cenário a partir da Figura 1. Um ponto importante ao se utilizar a modularidade ao desenvolver um software é a capacidade de integração entre os módulos. Conforme a quantidade de módulos aumenta, maior o esforço associado à integração dos módulos.

Diante disso, ao modularizar um projeto, deve haver um equilíbrio, de modo que o desenvolvimento possa ser planejado mais facilmente, incrementos de software possam ser definidos e entregues, as mudanças possam ser mais facilmente acomodadas, os testes

e a depuração possam ser conduzidos de forma mais eficaz e a manutenção no longo prazo possa ser realizada sem efeitos colaterais graves [Pressman 2016].

8.3.2. Defeitos de Software nos Custos

Um ponto importante que pode impactar nos custos do desenvolvimento de um software se relaciona a possíveis falhas e defeitos. [Pressman 2016] pontua que ambos indicam um problema de qualidade que é descoberto após o software ser liberado para os usuários (ou para outra atividade estrutural dentro da gestão de qualidade).

Para mitigar isso, deve-se manter revisões técnicas, com o objetivo de encontrar erros durante o processo de desenvolvimento, para que não se tornem defeitos depois da liberação do software. Caso seja necessário realizar ajustes em alguma parte do software, esse esforço será refletido em aumento nos custos de desenvolvimento. Como será pontuado adiante, mudanças, de qualquer natureza, geram novos custos, mesmo que seja uma mudança com o objetivo de realizar alguma correção no software.

De acordo com [Pressman 2016], as revisões de software são como um "filtro" para a gestão de qualidade, isso significa que elas são aplicadas em várias etapas durante o processo de engenharia de software e servem para revelar erros e defeitos que podem ser eliminados.

A descoberta de falhas de projeto durante as revisões técnicas ajuda a eliminar grande percentual dos possíveis erros, evitando novos custos com correções, ou alterações, em partes do software que já foram liberadas para uso do cliente e dos usuários.

Conforme ainda pontuado por [Pressman 2016], detectar e eliminar um grande percentual de possíveis erros do software antes da sua liberação, reduz substancialmente o custo das atividades subsequentes dentro do projeto.

8.3.3. Desenvolvimento Ágil na Formação de Preço do Software

Ainda segundo [Pressman 2016], a engenharia de software ágil combina filosofia com um conjunto de princípios de desenvolvimento, os quais têm-se um foco na satisfação do cliente e a entrega incremental antecipada; equipes de projeto pequenas e altamente motivadas; métodos informais; artefatos de engenharia de software mínimos; e, acima de tudo, simplicidade no desenvolvimento geral.

A implementação do desenvolvimento ágil prioriza a entrega mais do que a análise e o projeto, além de uma comunicação ativa e contínua entre desenvolvedores e clientes. Entretanto, desenvolvimento ágil não significa que não será criado nenhum tipo de documento ou documentação, significa que apenas os documentos que vão ser consultados mais adiante no processo de desenvolvimento são criados [Pressman 2016].

Durante o desenvolvimento de um projeto de software, pode acontecer de ser difícil, ou impossível, prever como esse software irá evoluir com o tempo. Por exemplo, no caso de um aplicativo móvel, pode ser difícil prever futuros requisitos do cliente, conforme o uso do aplicativo é disseminado entre os usuários. Pode acontecer também, das condições do mercado mudarem rapidamente, a presença de novas empresas concorrentes, novos softwares com a mesma funcionalidade básica, assim como, as

necessidades do cliente podem mudar, todos esses fatores afetam na forma como o software será desenvolvido e nos custos necessários para esse projeto.

Como [Pressman 2016] sugere, não é possível definir os requisitos completamente antes do início do projeto. É preciso ser ágil o suficiente para garantir uma resposta, mantendo o ambiente de negócios fluido, algo que implica em mudanças constantes. Qualquer tipo de mudança pode sair cara – particularmente se for mal controlada e mal gerenciada. Nesse ponto, a metodologia ágil procura reduzir os custos da mudança no processo de software.

Segundo [Jacobson 2002], a agilidade se tornou um sinônimo para descrever um processo de desenvolvimento de software moderno. Uma equipe ágil precisa ser rápida e capaz de responder de modo adequado às mudanças. Mudanças no software que está sendo criado, mudanças nos membros da equipe, mudanças devido a novas tecnologias, mudanças de todos os tipos que poderão ter um impacto no produto que está em construção ou no projeto que cria o produto. Ainda de acordo com [Jacobson 2002], os engenheiros de software devem ser rápidos, caso queiram assimilar as rápidas mudanças citadas anteriormente.

O processo de desenvolvimento de software tradicional apresenta os custos relacionados às mudanças de forma não linear, os quais costumam aumentar conforme o avanço do projeto. Observe na Figura 2, uma curva preta contínua. Conforme pontuado por [Pressman 2016], os custos crescem rapidamente, e o tempo e os custos necessários para assegurar que a mudança seja feita, sem efeitos colaterais, não serão insignificantes.

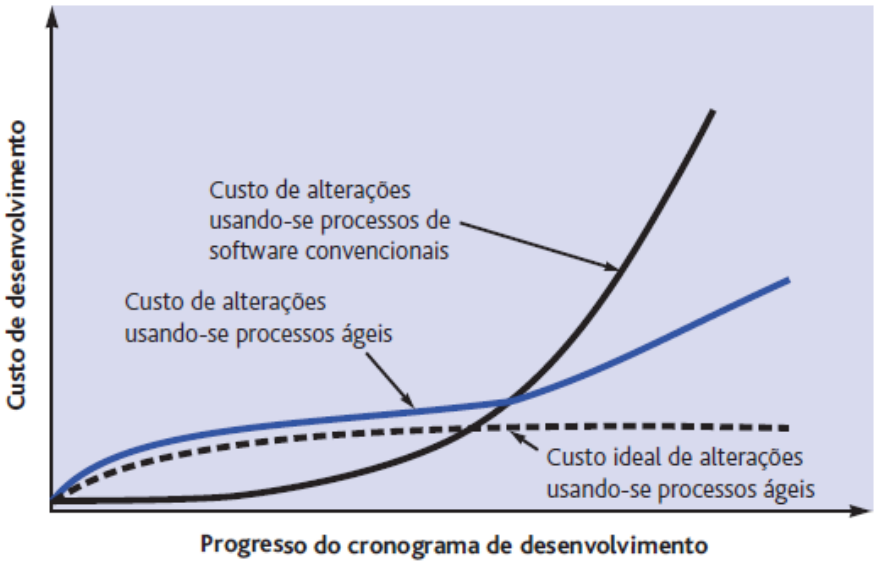

Figura 2 - Custos de mudanças como uma função do tempo em desenvolvimento. Fonte: [Pressman 2016].

O processo de desenvolvimento ágil procura, por meio da aplicação de sua metodologia, dar um direcionamento na forma como lidar com as constantes mudanças durante o desenvolvimento do software. A utilização de uma metodologia ágil ajuda a "achatar" a curva dos custos de mudanças, conforme é possível observar na Figura 1, por meio da curva azul contínua. Assim, como apontado por [Pressman 2016], a equipe de

software consegue assimilar as alterações, realizadas posteriormente em um projeto de software, sem um impacto significativo nos custos ou no tempo.

Percebe-se que é importante ter em mente que possíveis mudanças podem ocorrer durante o desenvolvimento de um software, e avaliar como isso pode impactar no preço já aprovado pelo o cliente, ou seja, no orçamento já alocado para o projeto.

8.4. Métricas para precificação

No dia-a-dia das empresas inúmeras métricas são utilizadas para precificar o custo de desenvolvimento, na pesquisa de [Almeida and Furtado 2019] são levantadas, por meio de pesquisa a empresas desenvolvedoras de software, quais métricas são utilizadas com esses objetivos, ou seja, precificar o custo envolvido no desenvolvimento. As métricas reportadas são: Homem-Hora (HH), Pontos de Função (PF), Unidade de Serviço Técnico (UST), Hora de Serviço Técnico (HST) e outras, que no final são variações ou combinações dessas métricas.

HH é a métrica mais simples e amplamente utilizada em toda a indústria de software, baseada apenas no cálculo de quanto tempo é gasto, com uso da unidade hora, para executar uma determinada tarefa. Alguns estudos mais recentes mostram que, para o cálculo do homem-hora, são usadas a opinião de especialistas, dentro daquele escopo que está sendo desenvolvido e, para dirimir variações grotescas, podem ser utilizadas técnicas como a Delphi.

PF não é uma métrica direta de esforço, mas a partir da medição funcional atrelada à produtividade e tecnologia utilizada com base histórica, pode se chegar ao esforço envolvido e ao custo. Há inúmeros estudos que relatam uma má precisão da utilização da métrica para cenários de manutenção ou sustentação de software.

UST é uma métrica utilizada em contratos de governo na qual há a terceirização do desenvolvimento de software. Para compor a UST, é criada uma unidade de medida as vezes vinculada à hora e são criados pacotes de trabalho que podem ser verificados. A partir dos pacotes de trabalho, é montado um catálogo de serviço previamente definido e são solicitadas as atividades baseadas neste catálogo.

HST é aplicada de maneira similar à UST, mas já define previamente a hora como unidade de medida.

8.5. O Modelo CEK

De acordo com [Escobar et al. 2019], o Modelo CEK, fundamentado na abordagem por projetos aplicada à engenharia de software, por meio da integração das ferramentas de gestão do Canvas de Projeto, da EAP e Kanban, foi proposto como uma das possíveis respostas aos problemas do desenvolvimento de software.

Considerado pelos próprios autores como um método híbrido de gerenciamento de projetos, que combina elementos das abordagens tradicional e ágil, de forma a mitigar os problemas relacionados aos projetos de desenvolvimento de software, em especial às deficiências no levantamento de requisitos e estabelecimento de objetivos, e às falhas de comunicação entre equipe e clientes, tendo como foco a ideação, planejamento e monitoramento da execução, reduzindo desperdícios, apoiando a transformar ideias em projetos colaborativos, ágeis, controláveis e efetivos [Escobar et al. 2019]. O Modelo

CEK está esquematizado na Figura 3, com destaque para as interações entre as ferramentas que o compõem.

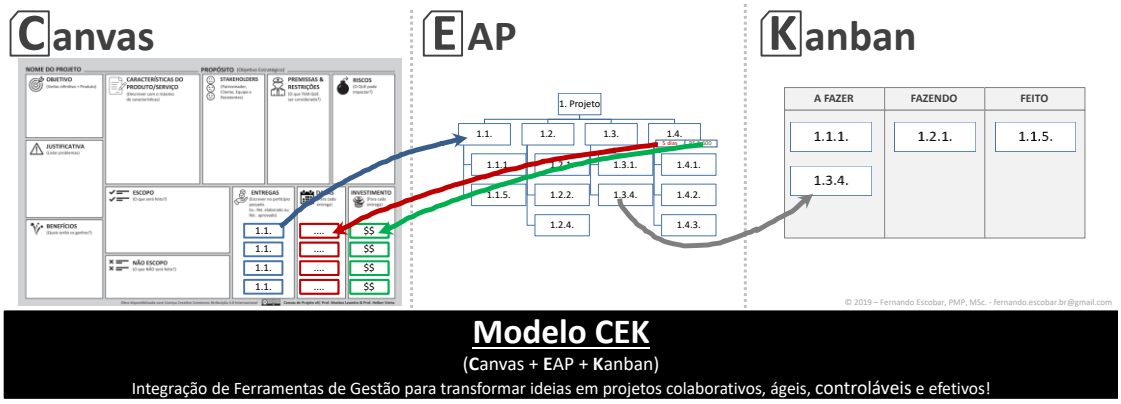

Figura 3 - O Modelo CEK

Proposto por [Leandro and Vieira 2018], o Canvas Project Design é resultado da síntese entre modelos teóricos, boas práticas e diversas sessões de tentativa e erro, incorporando conceitos, abordagens e características do Design Thinking, da Programação Neurolinguística, dos Guias PMBOK e PRINCE2, das Abordagens Ágeis e do Princípio MECE (Mutuamente Exclusivo, Coletivamente Exaustivo). A Figura 4 ilustra o Canvas Project Design.

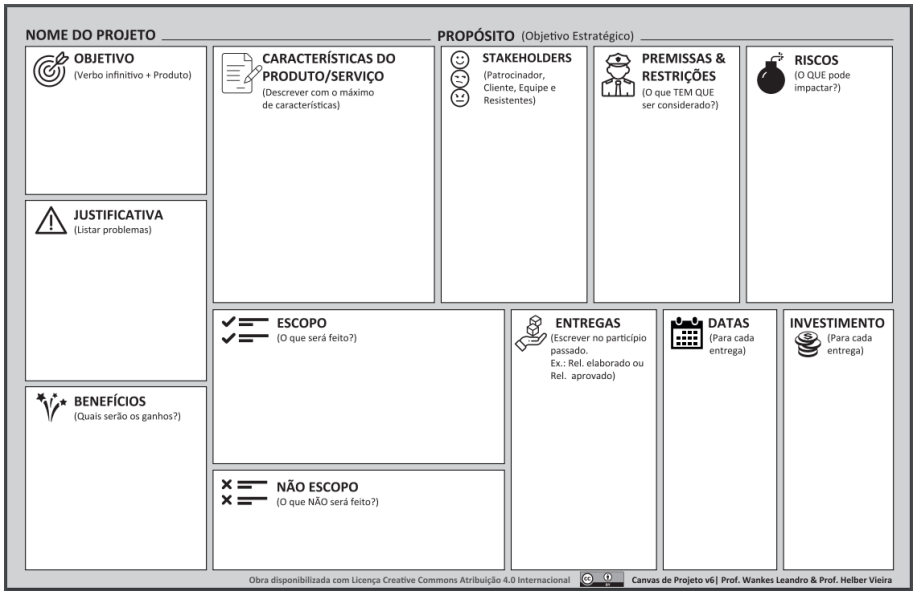

Figura 4 - Canvas Project Design. Fonte: [Leandro and Vieira 2018]

Fundamentado na simplicidade do modelo do Canvas Project Design, com sua abordagem visual, participativa e colaborativa, o Modelo CEK adota o Canvas Project

Design para sua aplicação prática, como ferramenta de gestão para apoiar equipes na fase de concepção de seus projetos [Escobar et al. 2019].

Complementar à etapa de concepção, abordando aspectos relativos ao planejamento, segundo [Escobar et al. 2019], também fundamentado em uma abordagem visual e colaborativa, o Modelo CEK acolhe o processo de criação da EAP.

De acordo com o PMBOK 6 [PMI, 2017], o trabalho relacionado à criação da EAP é o processo de, a partir da técnica da Decomposição, dividir e subdividir o escopo do projeto e suas entregas em partes menores e mais facilmente gerenciáveis, até o nível do pacote de trabalho. A visão estruturada do que deve ser entregue é fundamental para o planejamento e a efetividade do projeto. A Figura 5 apresenta a estrutura de uma EAP.

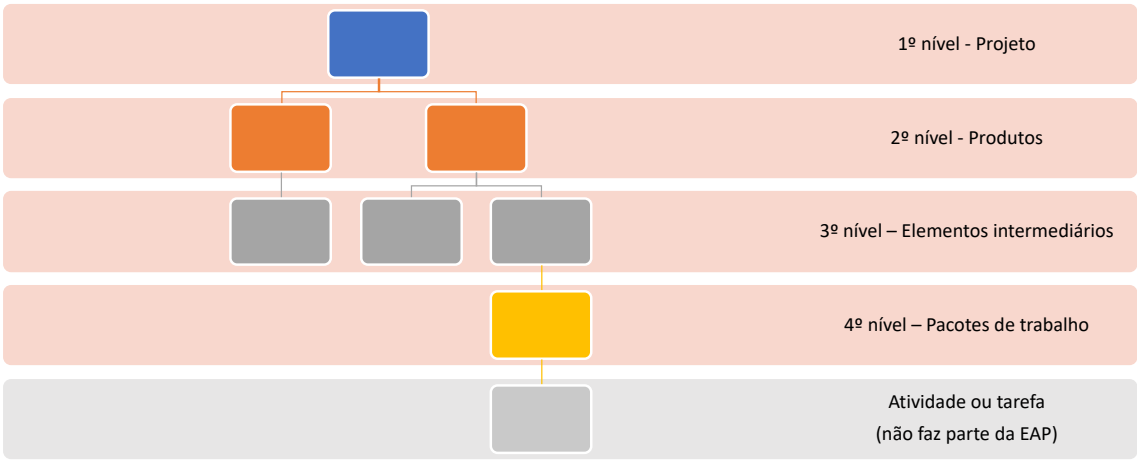

Figura 5 - Estrutura da EAP. Fonte: [Escobar et al. 2019]

Segundo [Escobar et al. 2019], na abordagem proposta pelo Modelo CEK, ao aplicar a técnica de Decomposição, dividindo as Entregas (resultados intermediários) idealizadas no âmbito do Canvas, até o nível do pacote de trabalho, é possível estimar duração e custos. Complementar, ainda segundo os autores, com a técnica da Agregação, as estimativas de duração e custos são somadas nos níveis superiores da EAP, até derivar nos respectivos blocos de Datas e Investimentos do Canvas Project Design. A Figura 6 exemplifica a abordagem para agregação de custos, desde os pacotes de trabalho (nível mais baixo), passando pelos elementos intermediário, o 2º nível da EAP e o projeto como um todo.

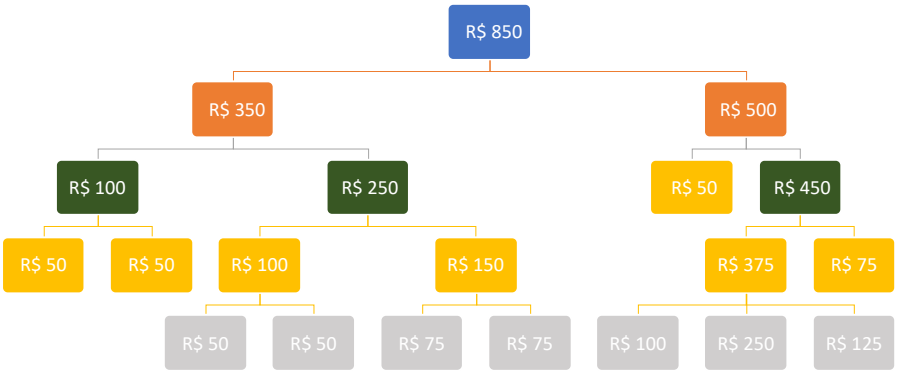

Figura 6 - Exemplo de abordagem para agregação de custos. Fonte: [Escobar et al. 2019]

Para controlar a execução do projeto, por meio do acompanhamento da execução dos pacotes de trabalho, definidos de forma colaborativa com o Canvas e com a EAP,

buscando a efetividade dos projetos de software, de acordo com [Escobar et al. 2019], o Modelo CEK adota o Kanban, buscando reduzir desperdícios, melhorar a comunicação, agilizar o fluxo de entrega de forma contínua em um sistema puxado.

De acordo com [Khaled Yacoub et al. 2016], o Kanban possui cinco princípios fundamentais: (1) visualizar o fluxo de trabalho; (2) limitar o trabalho em andamento; (3) medir e gerenciar o fluxo; (4) tornar explícitas as políticas do processo; e (5) usar modelos para reconhecer melhorias e oportunidades. Esses princípios são aplicados a partir de um Quadro Kanban, que permite visualizar o fluxo de atividades do processo em várias colunas. A Figura 7 ilustra um exemplo de Quadro Kanban.

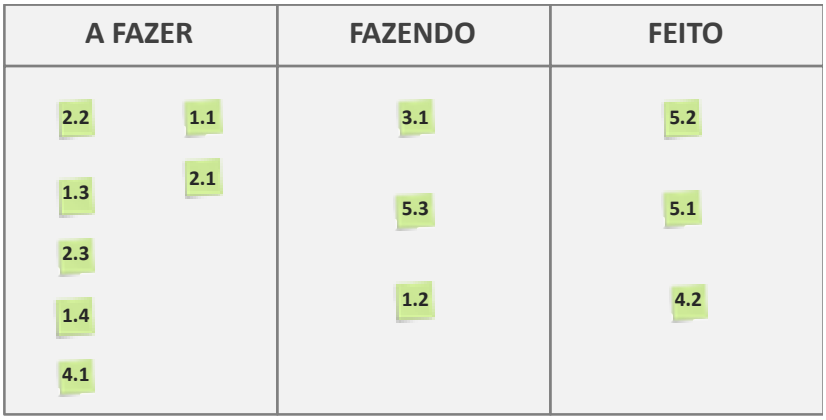

Figura 7 - Quadro Kanban. Fonte: [Escobar et al. 2019]

O Modelo CEK adota a sistemática do Quadro Kanban, com os Pacotes de Trabalho da EAP que ainda não foram priorizados, sendo colocados na coluna A FAZER (backlog do projeto); os Pacotes de Trabalho que já foram priorizados e que tiveram seu trabalho iniciado (limitados a 1 Pacote de Trabalho por equipe ou responsável – em atenção ao limite do Work In Progress, um dos princípios do Kanban) são movidos para a coluna FAZENDO, até que sejam concluídos e movidos para FEITO – à medida que os cartões são movidos para a coluna FEITO, novos espaços são liberados para a coluna FAZENDO, em um sistema puxado e de fluxo contínuo [Escobar et al. 2019].

8.6. Exemplo – o Projeto Moto Easy

Para exemplificar a proposta, fundamentada no Modelo CEK, será apresentado o mapa de custos criado para o Projeto Moto Easy.

Em diversas cidades do Brasil existe um serviço peculiar de transporte de usuários realizados através de motos. Os prestadores destes serviços são denominados mototaxistas. Esses profissionais são prestadores de serviço autônomos que são contratados diretamente pelo cliente. O Moto Easy irá realizar a gestão de corridas desta categoria, integrando os usuários aos profissionais por meio de uma solução desenvolvida na plataforma mobile e aplicação web para gestão dos serviços.

Observe que a formação de custos pode ser aplicada a qualquer tipo de projeto, independente da abordagem de desenvolvimento se tradicional (cascata ou waterfall) ou ágil. Entretanto, é pertinente observar que na abordagem ágil, a questão do retrabalho (e

de seu custo derivado) que tem que ser estimada e monitorada, pois a variação pode ser grande o que impactará no custo final.

8.6.1. O Projeto Moto Easy

A Figura 8 apresenta o Canvas Project Design preenchido para o Projeto Moto Easy, até a definição das Entregas, que irão compor o 2º nível de nossa EAP. Ao final, com o exercício de decomposição, estimativa de prazo e custo, e agregação, será possível derivar as Datas e Investimento para cada uma das Entregas.

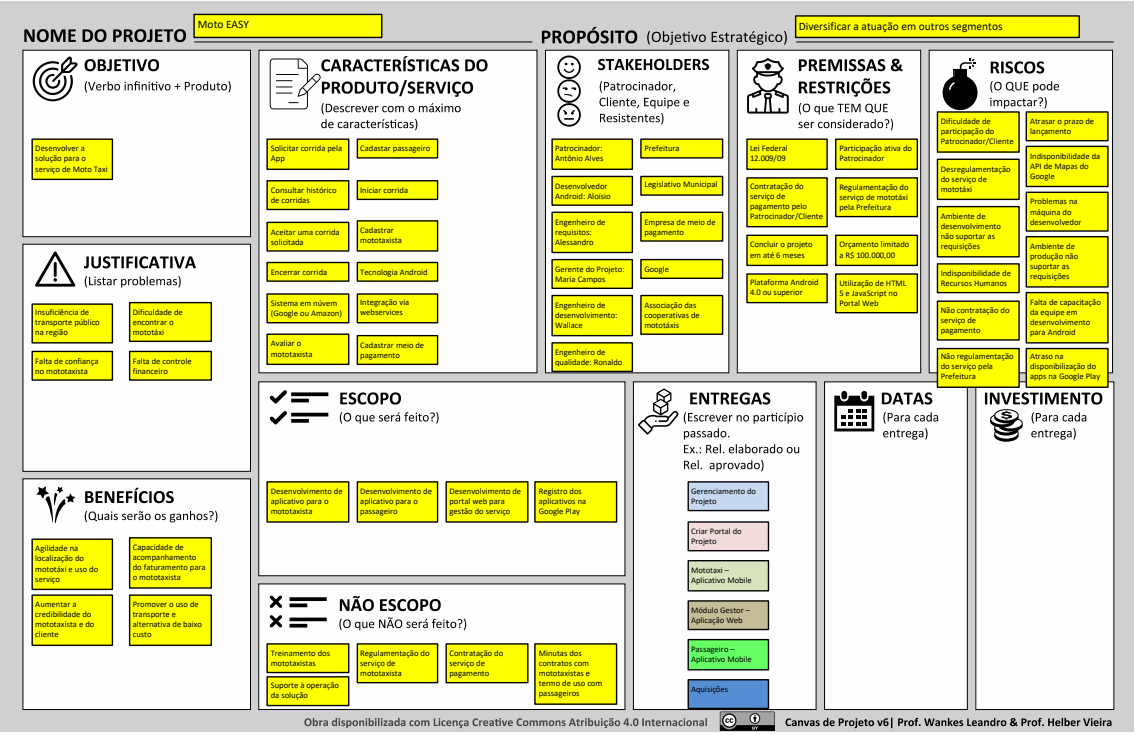

Figura 8 - o Canvas para o Projeto Moto Easy

O projeto foi realizado para entregar os produtos relacionados a seguir, diretamente vinculados às necessidades identificadas durante o levantamento realizado inicialmente com o Cliente:

- desenvolvimento de aplicativo para dispositivos móveis na plataforma Android destinado aos usuários de transporte, que será utilizado para a realização de chamadas aos serviços prestados pelos motos taxistas.
- desenvolvimento de aplicativo para dispositivos móveis na plataforma Android, para uso pelos profissionais prestadores do serviço de transporte de passageiros, os mototaxistas.
- desenvolvimento de sistema na plataforma web, para utilização pelo Cliente na gestão dos serviços oferecidos por meio da solução construída com o projeto.

A Figura 9 exemplifica a EAP para o Projeto MotoEasy.

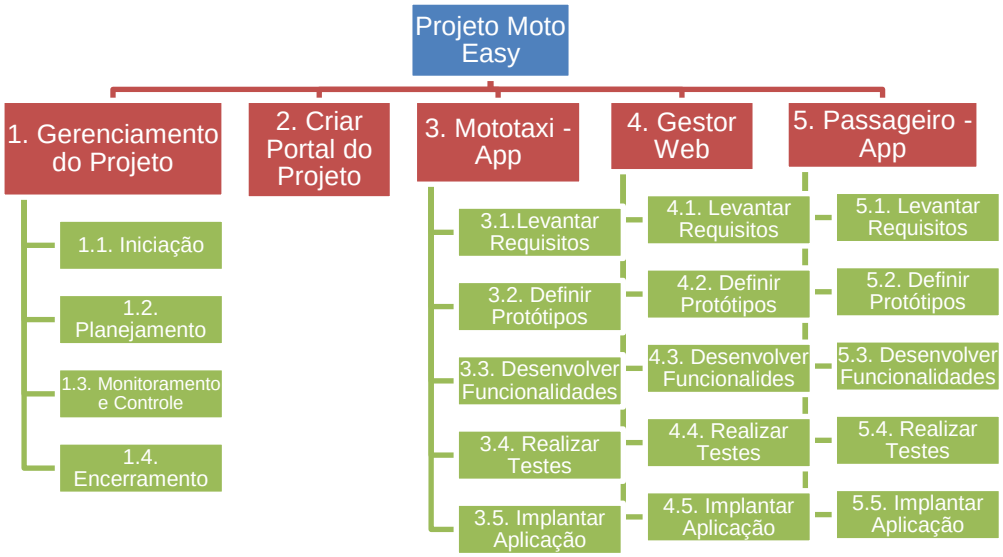

Figura 9 - EAP para o Projeto Moto Easy

8.6.2. O Mapa de Custos

Então definida a EAP, os custos têm que ser levantados numa visão *bottom-up*, partindo dos pacotes de trabalho. Assim foram feitos e estão apresentados nas tabelas a seguir, começando pelo gasto com pessoal.

Existem diversas formas de precificar, mas as mais utilizadas fazem uma composição de todos os custos do projeto e um percentual de lucro, assim foi realizada a estrutura de custos baseadas nos seguintes itens: material permanente, serviços, aquisição de software, pessoal, reserva técnica e lucro.

Na Tabela 2 é apresentado o gasto com material permanente para o projeto. É importante relatar o contexto: o projeto foi desenvolvido *part-time* por alunos de mestrado e foi necessária a aquisição de alguns materiais para facilitar a codificação e testes do software, mas em um cenário de uma *startup* devem ser contabilizados todos os demais custos envolvidos.

Tabela 2. Gasto com material permanente.

Descrição	Quantidade	Valor unitário	Valor total
Celular Samsung Galaxy S7	1	R$ 2.800,00	R$ 2.800,00
Celular Motorola G4	2	R$ 1.100,00	R$ 2.200,00
Notebook Dell Vostro	1	R$ 2.800,00	R$ 2.800,00
TOTAL GERAL			R$ 7.800,00

Na Tabela 3 são demostrados os custos com aquisição de licenças de software. Os demais softwares são todos baseados em licenças gratuitas ou softwares livres, não caracterizando custos adicionais.

Tabela 3. Custos com licenças de software

Descrição	Quantidade	Valor unitário	Valor total
Licença Mensal Adobe Creative Cloud	8	R$ 175,00	R$ 1.400,00
Corel Draw X8	1	R$ 2.399,00	R$ 2.399,00
TOTAL GERAL			R$ 7.800,00

Os serviços descritos na Tabela 4 compõem os gastos com serviços, que abrangem a contratação de serviços em nuvem, conexão de internet e gastos com energia elétrica da equipe envolvida.

Tabela 4. Gastos com serviços

Descrição	Quantidade	Quantidade de meses	Valor Unitário	Valor total
Assinatura Internet Plano 4G	7	8	R$ 110,00	R$ 6.160,00
Hospedagem para estrutura Amazon	1	8	R$ 300,00	R$ 2.400,00
Energia Elétrica *home office*	7	8	R$ 40,00	R$ 2.240,00
TOTAL GERAL				R$ 10.800,00

E, para finalizar a formação de custos, a folha de pagamento pessoal, colocando um valor médio na hora de cada membro da equipe, como se tratava de uma equipe multidisciplinar, inerente a uma equipe ágil, cada um dos membros atuou no papel e o que variou foi o tempo de alocação em cada função, essa estrutura pode variar conforme o projeto e a estrutura de custos de cada empresa.

Tabela 5. Folha de pagamento pessoal Projeto Moto Easy

	Valor Hora	R$ 75,00						
Função	**Horas trabalhadas no projeto**	**Valor Bruto**	**INSS Retenção**	**INSS Patronal**	**Contribuição para terceiros**	**Risco ambiental do trabalho**	**Encargos**	**Prolabore líquido**
			11%	20%	5.8%	1%		
Gerente de projeto	95	R$ 7.125,00	R$ 783,75	R$ 1.425,00	R$ 413,25	R$ 71,25	R$ 2.693,25	R$ 6.341,25
Engenheiro de requisitos	70	R$ 5.250,00	R$ 577,50	R$ 1.050,00	R$ 304,50	R$ 52,50	R$ 1.984,50	R$ 4.672,50
Engenheiro de desenvolvimento	90	R$ 6.750,00	R$ 742,50	R$ 1.350,00	R$ 391,50	R$ 67,50	R$ 2.551,50	R$ 6.007,50
Engenheiro de Qualidade	70	R$ 5.250,00	R$ 577,50	R$ 1.050,00	R$ 304,50	R$ 52,50	R$ 1.984,50	R$ 4.672,50
Engenheiro de desenvolvimento	90	R$ 6.750,00	R$ 742,50	R$ 1.350,00	R$ 391,50	R$ 67,50	R$ 2.551,50	R$ 6.007,50
Engenheiro de desenvolvimento	90	R$ 6.750,00	R$ 742,50	R$ 1.350,00	R$ 391,50	R$ 67,50	R$ 2.551,50	R$ 6.007,50
Engenheiro de desenvolvimento	90	R$ 6.750,00	R$ 742,50	R$ 1.350,00	R$ 391,50	R$ 67,50	R$ 2.551,50	R$ 6.007,50
TOTAL GERAL FOLHA DE PAGAMENTO							R$ 16.868,25	R$ 39.716,25

Todos esses custos devem ser agregados de forma a compor a Tabela 6, com uma visão geral, colocando um percentual de 10% de Reserva Técnica, para cobrir riscos inerentes ao projeto, e 35% de lucro. Em contratações com o governo, os lucros são limitados, pois já existem legislações que proíbem lucros exorbitantes, mas na esfera privada, onde estão inseridas a maioria das startups, o lucro pode ser ajustado. Observe que na venda final para o cliente, esses custos e planilhas não são apresentados, o que está estruturado nesse trabalho é formação dos preços.

Tabela 6 - Resumo dos custos do projeto

Descrição	Valor (R$)
Pessoal	67.384,50
Material Permanente	7.800,00
Aquisição de Software	3.799,00
Serviços	10.800,00
Reserva Técnica (10%)	8.978,13
Lucro (35%)	31.424,23
TOTAL	130.186,08

A Tabela 7 apresenta um cronograma de desembolso, uma boa prática de gestão para, por exemplo, a cobrança do cliente por pacotes de trabalho no tempo, o que manterá a saúde financeira da empresa, com o fluxo de capitais.

Tabela 7 - Cronograma de desembolso de recursos financeiros

Mês	Pessoal	Material permanente	Software	Serviços	Outros	Total
Janeiro	3.000,00			1.350,00	5.050,32	9.400,32
Fevereiro	10964,08	7.800,00	3.799,00	1.350,00	5.050,32	28.963,40
Março	10964,08			1.350,00	5.050,32	17.364,40
Abril	10964,08			1.350,00	5.050,32	17.364,40
Maio	10964,08			1.350,00	5.050,32	17.364,40
Junho	8.264,08			1.350,00	5.050,32	14.664,40
Julho	8.264,08			1.350,00	5.050,32	14.664,40
Agosto	4.000,02			1.350,00	5.050,34	10.400,36
TOTAL	67.384,50	7.800,00	3.799,00	10.800,00	40.402,58	130.186,08

Por fim, é apresentado um cronograma de execução do projeto com as horas alocadas proporcionando uma visão no tempo e o total de horas alocadas em cada fase do ciclo de desenvolvimento, compondo o Mapa de custos para o Projeto Moto Easy, conforme Tabela 8.

Tabela 8. Cronograma de execução do projeto

Descrição		**Início previsto**	**Término previsto**	**Status**	**Janeiro**	**Fevereiro**	**Março**	**Abril**	**Maio**	**Junho**	**Julho**	**Agosto**	**Horas trabalhadas**
					Dias úteis trabalhado								**3 Horas/Dia**
Iniciação	Definição do escopo	15/01/2017	04/02/2017	☑	14	3							51
Planejamento	Sprint Planning	05/02/2017	14/03/2017	☑		20	12						96
Execução, Monitoramento e Controle	Primeiro Release	15/03/2017	29/04/2017	☑			15	25					120
	Segundo Release	30/04/2017	14/06/2017	☑					27	12			117
	Terceiro Release	15/06/2017	30/07/2017	☑						14	25		117
	Quarto Release	31/07/2017	15/08/2017	☑							1	13	42
Encerramento	Entrega final e encerramento do projeto	16/08/2017		☑								8	24
Total Geral					14	23	27	25	27	26	26	21	567

8.7. Conclusão

Em uma análise preliminar do assunto a precificação aparenta ser uma tarefa fácil de ser realizada, bastando a composição dos custos e lucro dos itens que fazem parte do escopo do produto. Entretanto, o maior problema da formação de custos são os riscos atrelados ao processo e às atividades de desenvolvimento de software, e isso acaba por impactar nos custos finais.

O desenvolvimento de software parte de um contexto atrelado a riscos que, na maioria das vezes, acabam por ser concretizados, conforme buscou-se demonstrar nesse

trabalho, onde foram apresentados diversos aspectos inerentes a composição de custos e fatores conhecidos, tanto no mercado como na literatura acadêmica, que devem ser considerados na composição de preços de uma organização que constrói soluções de software.

Ainda no escopo desse estudo, foi apresentado um modelo emergente, o Modelo CEK, que agrega ferramentas para o amadurecimento do escopo que, em grande parte, costuma ser o maior problema para as empresas ao receber demandas de seus clientes. A medição e controle em um contexto de mudanças traz também a necessidade de entendimento por parte de quem produz da importância em saber se adaptar a esses cenários. Por isso, pode-se dizer que os maiores problemas enfrentados pela engenharia de software na verdade podem ser descritos como falta de engenharia de software, ou seja, não utilizar os inúmeros processos, métodos e ferramentas já consolidadas nessa área de estudo, que ainda pode ser considerada recente, se comparada a outras engenharias.

Por fim, nesse trabalho fica evidente que, sem foco na qualidade, as empresas e *startups* terão dificuldade para sobreviver. Portanto, não basta mais apenas boas intenções e bons profissionais, pois os custos inerentes às atividades acabam por ser um fator impactante em um mercado altamente competitivo.

8.8. Referências

BHARDWAJ, M.; RANA, A. Key Software Metrics and its Impact on each other for Software Development Projects. **ACM SIGSOFT Software Engineering Notes**, v. 41, n. 1, p. 1–4, 2016.

CANEDO, E. D.; DA COSTA, R. P. Methods and metrics for estimating and planning agile software projects. **Americas Conference on Information Systems 2018: Digital Disruption, AMCIS 2018**, 2018.

DAVIS, C. W. H. **Agile Metrics in Action: Measuring and Enhancing the Performance of Agile Teams**. 1st. ed. USA: Manning Publications Co., 2015.

FENTON, N.; BIEMAN, J. **Software Metrics: A Rigorous and Practical Approach**. 3rd. ed. USA: CRC Press, Inc., 2014.

JETHANI, K. Software metrics for effective project management. **International Journal of Systems Assurance Engineering and Management**, v. 4, n. 4, p. 335–340, 2013.

O'REGAN, G. Software Metrics. In: Undergraduate Topics in Computer Science. Cham: Springer International Publishing, 2014. p. 151–183.

PADMINI, K. V. J.; DILUM BANDARA, H. M. N.; PERERA, I. Use of software metrics in agile software development process. **MERCon 2015 - Moratuwa Engineering Research Conference,** p. 312–317, 2015.

Sommerville, Ian (2019). Engenharia de Software. 10. ed. **Pearson Education**.

Pressman, Roger; Maxim, Bruce (2016). Engenharia de Software, Uma Abordagem

Profissional. 8. **Ed. McGraw-Hill Global Education**

Jacobson, Ivar (2002). A Resounding 'Yes' to Agile Processes—But Also More. **Cutter IT Journal.**

Almeida, W. H. C. and Furtado, F. (2019). Análise sobre métricas nos contratos de fábricas de software no âmbito da administração pública federal. 1. ed. Rio de Janeiro: **Albatroz**. v. 1

ESCOBAR, Fernando et al. (2019). Modelo CEK (Canvas+ EAP+ Kanban): Integração de Ferramentas de Gestão para transformar ideias em projetos colaborativos, ágeis, controláveis e efetivos.

Leandro, W. and Vieira, H. (2018). Canvas de Projeto - Canvas Project Design. 1a. ed. São Paulo: **Riemma**.

PMI (2017). Um Guia do Conhecimento em Gerenciamento de Projetos (Guia PMBOK).

Khaled Yacoub, M., Abdel Athim Mostafa, M. and Bahaa Farid, A. (2016). A New Approach for Distributed Software Engineering Teams Based on Kanban Method for Reducing Dependency. **Journal of Software**, v. 11, n. 12, p. 1231–1241