CENTRO UNIVERSITÁRIO UNIEVANGÉLICA
CAMPUS ANÁPOLIS
ENGENHARIA DE COMPUTAÇÃO

FABIANA CRISTINA DE SOUSA

ANÁLISE DE TÉCNICAS PARA ESTIMATIVA DE ESFORÇO DE TEMPO EM PROJETO DE SOFTWARE DESENVOLVIDOS NA FÁBRICA DE TECNOLOGIAS TURING - UNIEVANGÉLICA

Anápolis - GO
2018 - 01

FABIANA CRISTINA DE SOUSA

ANÁLISE DE TÉCNICAS PARA ESTIMATIVA DE ESFORÇO DE TEMPO EM PROJETO DE SOFTWARE DESENVOLVIDOS NA FÁBRICA DE TECNOLOGIAS TURING - UNIEVANGÉLICA

Projeto de Pesquisa apresentado ao Curso de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UniEVANGÉLICA - como requisito parcial à obtenção do título de Bacharel em Engenharia de Computação. Orientador: Prof. Esp. Kleber Silvestre Diogo.

Anápolis - GO
2018 - 01

FABIANA CRISTINA DE SOUSA

ANÁLISE DE TÉCNICAS PARA ESTIMATIVA DE ESFORÇO DE TEMPO EM PROJETO DE SOFTWARE DESENVOLVIDOS NA FÁBRICA DE TECNOLOGIAS TURING - UNIEVANGÉLICA

Projeto de Pesquisa apresentado ao Curso de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UniEVANGÉLICA - como requisito parcial à obtenção do título de Bacharel em Engenharia de Computação. Orientador: Prof. Esp. Kleber Silvestre Diogo.

Aprovada em ( ) de ( ) de 2018

BANCA EXAMINADORA

_____

Prof. Esp. Kleber Silvestre Diogo - Orientador

_____

Convidado

_____

Convidado

AGRADECIMENTO

Primeiramente a Deus pelo dom da vida, sabedoria e por me permitir chegar até aqui.

Aos meus pais, Marcilene Nazaré Lemes de Sousa e Paulo Edson de Sousa, que proporcionaram a oportunidade de estudar, apoiaram incondicionalmente, por não me deixarem desistir e contribuíram diretamente nesta caminhada. Sem os senhores eu não teria chegado aqui.

A toda comunidade educativa do Centro Universitário de Anápolis – UniEvangélica, que de forma direta ou indiretamente participaram da minha formação, em especial os professores do curso de bacharelado em Engenharia de Computação.

O Profº. Esp. Kleber Silvestre Diogo pelo apoio, planejamento e elaboração deste trabalho, exercendo seu papel com maior boa vontade possível e que, com toda certeza, foi fundamental para o resultado final.

A todos os membros e orientadores da Fábrica de Tecnologias Turing, especialmente a Kelly Kianny Caixêta e o Leonardo Duarte Amorim membros da FTT e a diretora Profª. Ma. Viviane Carla Batista Pocivi pela disponibilidade do ambiente e auxilio no estudo e aplicação da pesquisa.

RESUMO

O trabalho estabelece técnicas de estimativa de esforço de tempo em desenvolvimento de software, o estudo utilizou o projeto VIRTOO – Sistema de Gerenciamento Acadêmico e Financeiro, que está sendo desenvolvido na Fábrica de Tecnologias Turing (FTT) do Centro Universitário de Anápolis UniEvangélica. O objetivo foi de identificar as técnicas de estimativa de esforço de tempo que se adequassem as particularidades da FTT. Com a utilização da análise das metodologias e características do ambiente o estudo de técnicas de estimativa de software foi direcionado e ocorreu por meio de um comparativo a identificação das técnicas de estimativa de esforço de tempo a serem implantadas na FTT. Com a definição das técnicas *Planning Poker* e *Ideal Days* para o desenvolvimento, e para a análise de requisitos a técnica PERT, as quais foram implantadas na Fábrica de Tecnologia Turing no projeto VIRTOO. A realização deste trabalho permitiu uma melhora significativa no planejamento e estimativas de entregas de valor no prazo estabelecido no para de projetos da FTT.

Palavras-chave: Técnicas de Estimativa de Esforço de Tempo. Desenvolvimento de Software. Fábrica de Software. Planning Poker. Ideal Days. PERT.

ABSTRACT

The paper establishes techniques of time effort estimation in software development, the survey used the VIRTOO project – Academic and Financial Management System, which is being developed at the Turing Technology Plant (FTT) on University center of Anápolis Unievangélica. The objective was identify the techniques of estimating the time effort that suited the particularities of FTT. With the use of the analysis of the methodologies and characteristics of the environment, the study of software estimation techniques was directed and occurred by means of a match the identification of the techniques of estimated time effort to be deployed in FTT. With the definition of Planning Poker techniques and Ideal Days for the development and analysis of requirements The PERT technique, which were deployed at the Turing technology factory in the VIRTOO project. The realization of this work allowed a significant improvement in the planning and estimates of deliveries of value in the period established in the for of projects of FTT.

Key-words: Time Estimating Techniques. Software development. Software Factory. Planning Poker. Ideal Days. PERT.

LISTA DE ABREVIATURA E SIGLAS

AIE	Arquivos de Interface Externas
ALI	Arquivos Lógicos Internos
APF	Análise de Ponto de Função
BFPUG	*Brazilian Function Point User Group*
CE	Consultas Externas
CMM	*Capability Maturity Model*
EE	Entradas Externas
EF	*Environmental Factors*
FS	Fábrica de Software
FTT	Fábrica de Tecnologias Turing
IBM	*International Business Machines*
IEC	*International Engineering Consortiom*
ID	Ideal Day
IFPUG	*International Function Point User Group*
ISO	*International Organization for Standardization*
ISPVida	Instituto Superior Politécnico Vida
ISTEL	Instituto Superior de Teologia Evangélica de Lubango
PERT	*Program Evolution Review Technique*
PCU	Pontos por Caso de Uso
TCF	Fatores de Complexidade Técnica
TD	Tipo de Dados
TDD	*Test-Driven Development*
TR	Tipo de Registro
SE	Saídas Externas

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE GRÁFICOS

SUM/ÁRIO

1 INTRODUÇÃO

A crescente busca por empresas que realizam entregas de projetos dentro do prazo e escopo estimado, vem criando uma constante necessidade aprimorar as técnicas de estimativa utilizada. Quando as estimativas não são alcançadas, há aumento no prazo e por consequência no custo do projeto. Fazer estimativas é lidar com incertezas, tais como mão de obra, equipamentos, serviços terceirizados entre outros. “Existem tantas incertezas que é impossível estimar com precisão os custos de desenvolvimento de sistema durante os estágios iniciais de um projeto”. (SOMMERVILLE, 2011, p. 442)

A Fábrica de Tecnologias Turing (FTT) do Centro Universitário de Anápolis – UniEvangélica é um ambiente que executa engenharia de software, trabalhando com projetos reais.

> “O objetivo das Fábrica de Software deve ser a geração de produtos requeridos pelos usuários ou clientes, com o mínimo de defeito possível e a um preço (ou custo) competitivo e compatível que forneça a margem necessária para os investimentos em manutenção e melhoria da fábrica.” (FERNANDES, 2011).

Atualmente, a FTT está desenvolvendo um sistema de gerenciamento acadêmico e financeiro, nomeado VIRTOO, destinado as faculdades ISTEL (Instituto Superior de Teologia Evangélica no Lubango) e ISPVida (Instituto Superior Politécnico Vida) ambas, situadas em Lubango na Angola. Este sistema tem como intuito auxiliar a gestão da faculdade, funcionando como um portal de acesso para funcionários e acadêmicos.

O desenvolvimento de software está suscetível a diversas incertezas como: a) dimensão de recursos, prazos e esforços para o projeto; b) identificação e clareza no levantamento de requisitos; c) participação adequada dos *stakeholders*[1] na definição de requisitos; d) mudanças de requisitos; e) disponibilidade de recursos em quantidade, experiência e conhecimento adequado; f) alterações nos prazos de entregas; g) limite de recursos de hardware (PINNA; ARAKAI, 2006).

A FTT enfrenta diversos desafios, por ser um ambiente de aprendizagem. A rotatividade de membros é constante, em função de conquistarem cargos em empresas externas. O desnível de conhecimento entre os membros, impacta diretamente o andamento do projeto, de forma que as tarefas estimadas nas *sprints*[2] não cumpram o proposto.

A FTT utiliza a técnica de estimativa PERT (*Program Evalution Review Technique*), apontada por estudos realizados anteriormente:

> “A técnica PERT seria a mais adequada nestas primeiras *Sprints*, pois com ela poderia avaliar três pontos de vista para uma estimativa: Otimista, Mais Provável e Pessimista. Assim, com essa técnica poderia se considerar os níveis de incerteza de entendimentos

1 Traduzindo para português é parte interessada (TRINDADE, 2011).

2 Um time-boxed de um mês ou menos (SCHWABER; SUTHERLAND, 2013).

> de requisitos e de impedimentos nas tecnologias sem deixar de observar a produtividade. Por ser uma técnica estatística, o PERT não deixa de avaliar o conhecimento histórico, pois para recolher as estimativas Otimistas, Mais Prováveis e Pessimistas é necessário olhar para projetos ou bases históricas já desenvolvidas. Portanto, para os envolvidos poderem estimar, foi considerado as experiências durante o treinamento como base para as estimativas, visto que os itens selecionados para a primeira *Sprint* eram semelhantes aos itens desenvolvidos nos treinamentos" (FERREIRA, 2016).

A técnica PERT é considerado como adequada nas primeiras *sprints*, fazendo necessária a inserção de outra técnica de estimativa (FERREIRA, 2016). A fragilidade desta técnica também é causada por não pontuar o tamanho e complexidade em fases avançadas de projeto como durante o desenvolvimento.

A FTT trabalha com entregas constantes ao cliente com necessidade de determinam o tamanho e tempo que será gasto nas entregas de projetos, afim de articular com o cliente prazos e escopo de entregas por *releases*[3]. Quando se determina a entrega de determinado escopo é necessário ser realizado com exatidão, sendo que é um compromisso com o seu cliente a falha no mesmo irá gerar aumento de tempo e recursos da equipe, provocando desgaste com o cliente e na própria equipe. A estimativa é uma necessidade para que as entregas aconteçam no tempo previsto pela equipe.

Neste contexto, obteve-se o seguinte problema de pesquisa: Qual técnica de estimativa melhor se adequa ao processo do ambiente da Fábrica de Tecnologias Turing, que seja capaz de estimar a entrega de valor de forma mais eficaz?

O objetivo geral deste estudo é analisar e definir técnicas de estimativa de esforço de tempo, visando a melhoria de desempenho nos projetos executados na FTT.

Para alcançar o objetivo proposto as seguintes atividades foram desenvolvidas: a) Estudar os conceitos de metodologias ágeis e técnicas de estimativa de esforço de tempo em projetos de software; b) Realizar um estudo comparativo de técnicas de estimativa de esforço de tempo; c) Identificar a técnica de estimativa que atende às necessidades da FTT, adequando-a se necessário; d) Testar a técnica proposta em um projeto na FTT; e) Descrever as lições aprendidas, após o teste da técnica de estimativa.

Utilizando das metodologias: a) Aprofundamento Teórico; b) Estudo de Conceitos; c) Estudo Comparativo das Técnicas; d) Identificação das técnicas; e) Aplicação da Técnica na FTT; f) Registro de Resultados Após Aplicação.

Antes da aplicação das técnicas de estimativa de esforço de tempo, propostas na execução deste trabalho, a FTT utilizava a técnica de estimativa PERT, para especificação de

3 Quando uma entrega formal é feita ao cliente, no final de uma interação (DANTAS, 2009).

requisitos e desenvolvimento, no planejamento de *sprint*. O trabalho propôs para o planejamento de *sprint* a continuidade da técnica de estimativa PERT na especificação de requisitos e implantação das técnicas de estimativa *Planning Poker* e *Ideal Day* no desenvolvimento.

O trabalho está estruturado em: a) Introdução; b) Fundamentação Teórica; c) Resultado Alcançados; d) Considerações Finais; e) Referências Bibliográficas; f) Anexos.

2 REFERENCIAL TEÓRICO

Os seguintes itens apresentam os tópicos que fundamentam esta proposta, sendo: Métodos Ágeis, Estimativas e Fábrica de Tecnologias Turing.

2.1 METODOLOGIA ÁGIL

Com processos cada vez mais automatizados, os “Software fazem parte de quase todas operações de negócios, assim, novos softwares são desenvolvidos rapidamente para obter proveito das oportunidades e responder às pressões competitivas.” (SOMMERVILLE, 2011). “Criar mudanças requer inovação: desenvolver novos produtos, criar novos canais de venda, reduzir tempo de desenvolvimento de produto, customizar produto para segmentos de mercado cada vez menores.”, (HIGHSMITH, 2012).

Até o início de 1990 os projetos se prendiam mais a fase de planejamento, formalidades para qualidade de segurança e métodos para análise de projetos. Desenvolvedores de software insatisfeitos com estas abordagens pesadas apontam novos ‘métodos ágeis’ (SOMMERVILLE, 2011).

Como alternativas aos métodos tradicionais surge então os métodos ágeis, deixa de priorizar o desenvolvimento orientados a planejamento e passam a ser flexíveis a mudanças, com refatorações sem gerar grandes impactos ao planejamento (SOARES, 2010).

Em (AGILE MANIFESTO, 2001) são apresentados os doze princípios por trás do Manifesto Ágil:

1. A prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado;

2. As mudanças em requisitos são bem-vindas, os processos ágeis tiram proveito das mudanças, visando vantagem competitiva para o cliente;

3. Entrega frequentes do software funcionando, em poucas semanas há poucos meses, de preferência com a menor escala de tempo;

4. Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto;

5. Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho;

6. O método mais eficiente e eficaz de transmitir informações para uma equipe de desenvolvimento é através de conversa ‘face a face’;

7. Software funcionando é a medida primária de progresso;

8. Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.

9. Contínua atenção à excelência de técnica e bom design aumenta a agilidade.

10. Simplicidade, a arte de maximizar a quantidade de trabalho não realizado, é essencial.

11. As melhores arquiteturas, requisitos e designs emergem de equipes auto organizáveis.

12. Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

2.1.1 *Scrum*

“Um *framework* dentro do qual pessoas podem tratar e resolver problemas complexos e adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível.” (IMPROISSI, 2017).

O *Scrum* é utilizado desde 1990, fundamentado no empirismo, onde o conhecimento é adquirido de sua experiência e tomadas de decisões (SCHWABER e SUTHERLAND, 2013).

Baseado nos três pilares: transparência, inspeção e adaptação. E com quatro eventos determinados pelo *framework*: reunião de planejamento, reunião diária, reunião de revisão e reunião de retrospectiva, que visam amparar e garantir que os pilares e o empirismo sejam seguidos (SCHWABER; SUTHERLAND, 2013).

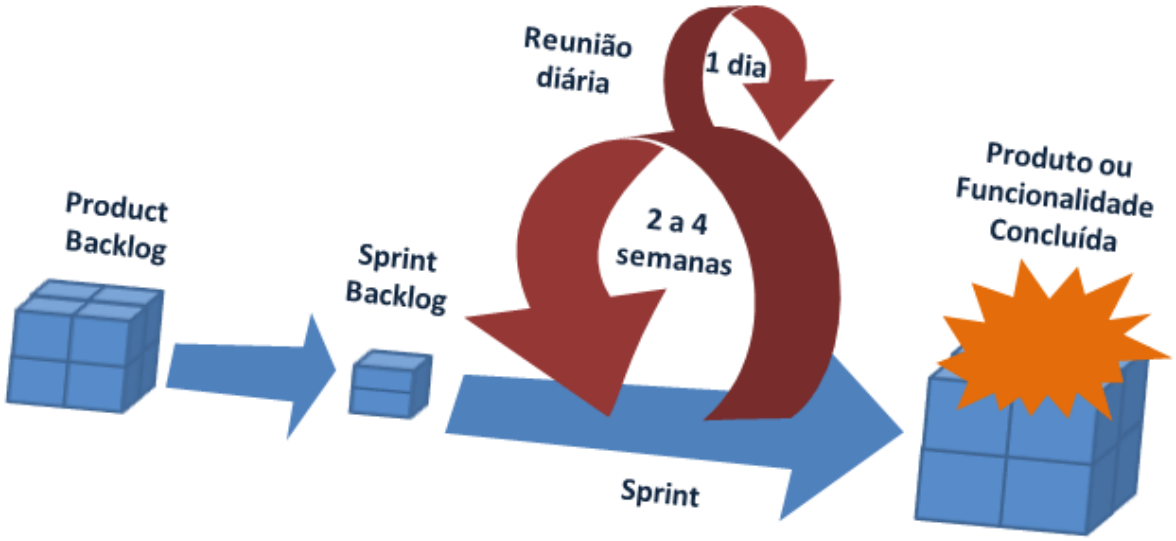
Considerada uma equipe independente o Time *Scrum* é auto organizável e multifuncional, composto por: Time de Desenvolvimento, *Prooduct Owner* e *Scrum Master* (SCHWABER; SUTHERLAND, 2013).

O Time de Desenvolvimento tem a responsabilidade de produzir a entrega de valor que deve ser concluído ao final de cada *sprint*. O *Product Owner* amplia o valor do produto que será desenvolvido, prioriza os itens do *backlog e* representa o cliente dentro da organização. E

o *Scrum Master* garante que todos dentro do Time *Scrum* entendam o processo de desenvolvimento *Scrum* e o siga também é o responsável por resolver impedimentos e conduzir reuniões (SCHWABER; SUTHERLAND, 2013).

No *Scrum* o desenvolvimento é dividido por *time box* de *sprints* de 2 a 4 semanas. Durante cada *sprint* acontece os eventos, com a reunião de planejamento ao início de cada *sprint*, onde será definido os requisitos do *backlog* que devem ser entregues ao final do t*ime box*, a reunião diária, todos os membros relatam as atividades que realizaram no dia anterior, que realizarão no dia e relatam os impedimentos se necessário, na reunião de revisão é apresentado o produto "pronto" e o *Product Owner* avalia se está de acordo com o solicitado e na reunião de retrospectiva, é avaliado o desempenho da *sprint* e exposto por todos quais foram os pontos bons, o que precisa ser melhorado e quais as lições aprendidas (SCHWABER; SUTHERLAND, 2013).

Figura 1 - Processo do *Scrum*.



Fonte: MINDMASTER.

2.2 ESTIMATIVAS

O planejamento de projeto é o início da gerência de projeto, sendo realizada uma estimativa do trabalho, recursos que serão necessários e o tempo de desenvolvimento, então é

determinado um cronograma de execução, sendo estabelecido metas as quais devem ser cumpridas. Deve-se considerar um nível de incerteza, sendo que no planejamento ~~se~~ assumi um compromisso inicial (PRESSMAN, 2011).

Para que a estimativa seja efetiva e condizente com a realidade do ambiente é necessário dedicar bastante atenção as seguintes incertezas em um projeto: complexidade de projeto, relatividade da maturidade da equipe, tamanho do projeto, as incertezas aumentam na proporção de tamanho do projeto e a estrutura, incerteza de dependência e complexidade dos requisitos solicitados (PRESSMAN, 2011).

Nas metodologias ágeis o planejamento acontece em paralelo ao desenvolvimento do projeto, realiza entregas incrementais que são consideradas o progresso e coerentes com a prioridades do cliente. (SOMMERVILLE, 2011)

Para projetos desenvolvidos na metodologia ágil com framework Scrum a estimativa ocorre no planejamento de cada *sprint*, retém-se então a seguintes estimativas para proposta:

2.2.1 PERT

A estimativa *Program Evaluation Review Technique* (PERT) é realizada através de três pontos que se baseia no conhecimento e experiência da equipe. “O método permite compensar com o desenvolvimento de uma estimativa ponderada.” (PETERS; PEDRYCZ, 2001).

Definidos os valores da Estimativa Otimista (O), Estimativa Mais Provável (M) e Estimativa Pessimista (P). Com base nos dados é feito o cálculo da estimativa PERT da tarefa, onde: $PERT = \frac{P+4M+O}{6}$. A estimativa mais provável recebe peso 4, apesar da diferença os outros pontos podem ter grande influência estando muito distantes da estimativa mais provável. Para medir a distância dos elementos calculamos através das médias estatísticas a variância e desvio padrão. A variância tem como fórmula: $Variância = \left(\frac{P-O}{6}\right)^2$. E o Desvio Padrão como a dispersão entre as medias, na fórmula: $DesvioPadrão = \frac{P-O}{6}$. (SANTOS, 2015).

Com a estimativa de atividades do projeto podemos estimar a duração de desenvolvimento do mesmo. A estimativa requer: a) identificar os pontos críticos do projeto, b) definir a estimativa PERT de cada atividade, c) calcular a variância de cada atividade, d) calcular o desvio padrão do projeto, com a fórmula, $RaizdaSomaVariância =$

$\sqrt[2]{VarA + VarB + VarC + VarD}$ e) e calcular a estimativa final por: $EstimativaProjeto = SomaPERT + RaizdaSomaVariância$ (SANTOS, 2015).

> [...] a análise PERT apresentada neste artigo é uma ferramenta poderosa para estimar projetos, até mesmo em suas fases mais iniciais quando existem várias lacunas no detalhamento de escopo. Ela possibilita maior aproximação e entendimento do gerente de projetos sobre o contexto de cada atividade durante a fase de planejamento, onde será possível a extração de informações bastante relevantes para o projeto, [...]. (SANTOS, 2015).

"Essa é apenas uma estimativa grosseira do custo do projeto, visto que só o tamanho do projeto foi considerado." (CARVALHO; CHIOSSI, 2001).

2.2.2 Delphi

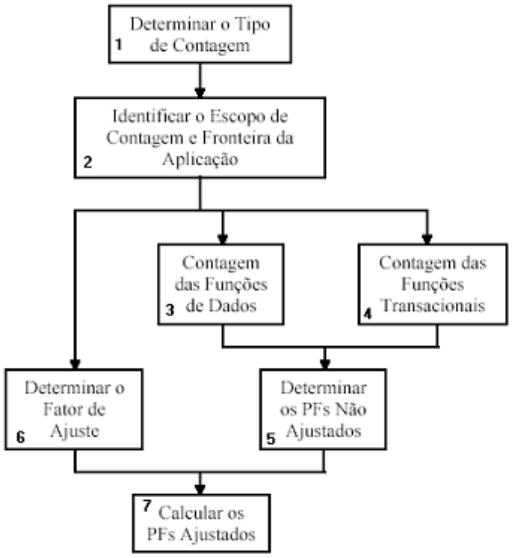
Delphi é um método onde é determinado a estimativa por especialistas, deve se apresentar aos especialistas quais serão as funcionalidades e os mesmos discutem e preenchem uma ficha indicando sua percepção do esforço que será necessário para o desenvolvimento. Fornecendo o tempo otimista, provável e pessimista. Então é feito um resumo das estimativas do grupo e estimativas individuais (PETERS; PEDRYCZ, 2001).

O processo é repetido até que cheguem a um consenso, então é tomado uma média das estimativas individuais ponderadas e colocados na seguinte fórmula: $Est = \frac{LIE+4xEMP+LSE}{6}$, sendo: Est: estimativa, LIE: limite inferior de estimativa, EMP: estimativa mais provável e LSE: limite superior de estimativa. A variância das estimativas individuais é definida como: $Est = \frac{LSE+LIE}{6}$, sendo: Est: estimativa, LIE: limite inferior de estimativa e LSE: limite superior de estimativa (PETERS e PEDRYCZ, 2001).

2.2.3 Análise De Ponto De Função (APF)

APF, uma métrica internacional, visa calcular o tamanho do software. É calculado somente os requisitos funcionais, com base na visão do usuário, desconsiderando a linguagem de programação. A contagem consiste nas seguintes características: independe de tecnologia utilizada, auxilia o desenvolvimento de resultados consistentes, baseada na visão do usuário final, tem total significado para o usuário final, utiliza-se de estimativas, passível automação e relativa subjetividade por refletir a visão do usuário. (LICHIRGU, 2016).

Figura 2 – Análise de Ponto po Função.



Fonte: VAZQUEZ; SIMÕES; ALBERT (2003)

O primeiro passo é definir o tipo de contagem que será realizada, entre: a) contagem de projeto de desenvolvimento, é mensurado as funcionalidades fornecidas aos usuários finais no momento de sua instalação, pode ser incluso a conversão de dados necessários para a implantação do software, b) contagem de projeto de melhoria/manutenção, é mensurado as modificações em um software pronto, o objetivo é manter a contagem de pontos de função sempre atualizada e c) contagem de aplicação – mede a funcionalidade oferecida pelo usuário na aplicação já instalada e em produção com funcionalidade atual medida (BONFIM; ANDRADE, 2015).

Então é identificado o escopo da contagem visando: a fronteira de aplicação, as relações do sistema com funcionalidade fora dela, atribuições de dados e processos suportadas pelo sistema que está em contagem. A fronteira determina o que é externo da aplicação como: as interfaces conceituais, aplicações e usuários externos. O escopo define o tamanho de software, como um conjunto ou subconjunto (VAZQUEZ; SIMÕES; ALBERT, 2003).

Conta-se então as funções de dados e funções de transações não ajustadas como: a) função de dados em arquivos lógicos internos (ALI) e arquivos de interface externas (AIEs), b) funções de transação em entradas Externas (EE), saídas externas (SE) e consultas Externas (CE) (BONFIM; ANDRADE, 2015).

As funções serão classificadas por tipos de dados, de forma que possa ser identificado pelo usuário e logicamente relacionados, em: a) ALIs são aplicações que armazena dados mantidos, que podem ser utilizados por um ou mais processos elementares da aplicação, é um grupo de dados ou informações de controle e são mantidos na fronteira. b) AIEs é o armazenamento de dados referenciados de um ou mais processos elementares dentro da

fronteira de aplicação, ele deve ser um grupo de dados ou informações referenciadas e lidas pela aplicação, está conceitualmente fora da fronteira de aplicação e não é mantido pela aplicação. Os ALI e AIE são classificados com relação a sua complexidade funcional (baixa, média e alta) tendo como base (VAZQUEZ; SIMÕES; ALBERT, 2003):

Define quais as funções do tipo de transação, com relação em: EE processo elementar, que processa dados ou informações de controle recebido de fora da fronteira da aplicação, mantém um ou mais ALI e/ou modificar o comportamento do sistema. SE processo elementar que apresenta ao usuário informações geradas por lógica de processamento, é muito mais do que uma recuperação de dados, deve obrigatoriamente realizar um ou mais cálculos, fórmulas matemáticas, dados derivados, manter um ou mais arquivos lógicos e alterar o comportamento do sistema. CE é um processo elementar que envia dados para fora da fronteira de aplicação, deve apresentar ao usuário por meio de lógica de processamento que apenas recupere os dados ou informações de controle. Após contagem é determinado a complexidade das funcionalidades (BONFIM; ANDRADE, 2015).

A contagem de pontos de função não ajustada é realizada através da complexidade das funcionalidades, são contadas separadamente os tipos de dados e tipos de transição com baixa, média e alta complexidade, está contagem não é considerada um valor final por não ter sido ajustada (VAZQUEZ; SIMÕES; ALBERT, 2003).

Determinamos então o VAF (Valor do Fator de Ajuste) é baseado em 14 características gerais que refletem funções que afetam a aplicação de maneira geral: a) comunicação de dados, b) processamento distribuído, c) performance, d) configuração altamente utilizada, e) volume de transações, f) entrada de dados on-line, g) eficiência do usuário final, h) atualização on-line, i) complexidade de processamento, j) reutilização, k) facilidade de instalação, l) facilidade de operação, m) múltiplos locais e n) facilidade de mudanças (VAZQUEZ, SIMÕES e ALBERT, 2003). Tais características são classificadas por nível de influência no projeto/aplicação, podendo variar de 0 a 5.

Por último calculamos o número dos pontos de função que é a contagem final da análise de ponto de função de acordo com o tipo de contagem (VAZQUEZ, SIMÕES e ALBERT, 2003).

2.2.4 Pontos Por Caso De Uso (PCU)

A métrica Pontos por Caso Uso (ou Use Case Points), criada em 1993 por Gustav Karner, a fim de mensurar o tamanho do projeto no momento da especificação, baseia-se nos casos de uso e é uma métrica voltada para desenvolvimento orientado a objeto além do caso de uso são calculados os TCF (Fatores de Complexidade Técnica) do projeto, os EF (Fatores Ambientais) que determinam a eficiência do projeto e o nível de experiência dos profissionais (CELEPAR, 2006).

Os passos para contagem do PCU são: a) Contar os atores e atribuir o grau de complexidade; b) Contar os casos de uso e atribuir o grau de complexidade; c) Soma o total de atores e casos de uso, obter o PCU não ajustado; d) Determinar a TCF; e) Determinar a complexidade do EF; g) Calcula o PCU ajustado (ANDRADE; OLIVEIRA, 2004).

Deve considerar que o PCU não ajustado é calculado na fórmula: $PCUNA = PA + PUC$, onde PCUNA: pontos de caso de uso não ajustado, PA: soma dos pesos dos atores e PUC: soma dos pesos dos casos de uso. E o PCU ajustado é calculado na fórmula: $PCU = PCUNAxTCFxEF$, onde: PCU: pontos por caso de uso ajustado, PCUNA: pontos por caso de uso não ajustados, TCF: complexidade do fator técnico e EF: Complexidade do Fator Ambiental (ANDRADE e OLIVEIRA, 2004).

2.2.5 *Planning Poker*

"O *planning poker* foi definido e nomeado pela primeira vez por James Grenning, em 2002, e mais tarde popularizado por Mike Cohn, no livro *Agile Estimating and Plannin.*" (BERNARDO, 2014). É um jogo de cartas e ao mesmo tempo um exercício, onde utiliza do consenso para estimar. Sendo possível designar a quantidade de esforço que será necessária para determinado trabalho (RITTER, 2014).

Os membros da equipe recebem um conjunto de cartas, com determinada sequência, conhecido como *story point* com valores: 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100. Também existem as cartas de 'interrogação', 'infinito' e em alguns modelos, a carta 'café'. As cartas zero (0), infinito (∞), interrogação (?) e "café" serão utilizadas de forma que: a) zero (0): Quando a estória de usuário é tão simples que os membros não consideram que haja necessidade de esforço e que será realizado em minutos, e em vez de usar a carta ½, utiliza a carta 0, b) infinito (∞): Sendo o oposto de zero, está estória de usuário é considerada tão grande que os membros não consideram ser possível estimar, precisa ser melhor entendida ou quebrada em estórias de usuário menores, c) interrogação (?): Utilizada se, o membro mesmo com a explicação e

discussão sobre a estória de usuário não conseguiu compreender o suficiente para estimar, sendo necessário discutir melhor a estória de usuário até que toda equipe compreenda e d) café: Sinaliza que algum membro precisa de uma pausa da estimativa, podendo o *Scrum Master* definir o tempo de intervalo (BERNARDO, 2014).

Funcionamento do *Planning Poker*: O *Product Owner* apresenta aos membros a estória de usuário de maneira clara e objetiva. É discutido os critérios de aceitação e tiram as dúvidas da atividade com o *Product Owner*. Com exceção do *Product Owner*, cada membro apresenta a quantidade de *story points* necessários para a atividade selecionando a carta do baralho em suas mãos. Depois de todos estarem com a carta selecionada em mãos, todos viram ao mesmo tempo. Em caso de valores diferentes, os membros apresentam as justificativas dos valores mais alto e mais baixo. E votado novamente até que o grupo chegue em um acordo. Caso não consiga se chegar a um acordo, o *Scrum Master* pode interferir, solicitando ao time que entre em acordo onde os membros considera a maioria, média ou confiar em determinado participante que já desenvolveu o mesmo trabalho da estória de usuário (BERNARDO, 2014).

"Em um projeto, as partes interessadas costumam ter dificuldade para entender o conceito de horas ou dias 'ideais'.". Ressaltando a importância de se utilizar *story point* e não volume de horas (VIGNADO, 2016).

"Enquanto um membro mais experiente pode dizer que termina o desenvolvimento da *user story* em apenas 1 hora, um menos experiente pode informar que levará 10." (VIGNADO, 2016).

"O uso de *story points* elimina este problema pois separamos o tamanho da *user story* do tempo que realmente levará para desenvolver." (VIGNADO, 2016).

2.2.6 *Ideal Days*

Utilizado para realizar estimativa de forma ágil, o *Ideal Days* é aplicado no planejamento de interações, tendo como destaque o cálculo da velocidade com base nas horas gasta pela equipe para implementar um ID - Ideal Day (RITTER, 2015).

Ideal Days (em português, Dias Ideais), ao estimar em dias ideais, retiramos todas as atividades adicionais da equipe como reuniões, telefonemas, envio de e-mails entre outras e conta-se apenas a sua dedicação a atividade proposta. Ao fazermos estimativas baseadas a tempo não temos um histórico consistente, pois com a evolução da equipe uma atividade que, por exemplo, demorava 5 horas passa ser executada agora e realizada em 4 horas, o que torna inviável utilizar a base anterior (KLEIN, 2014).

Para efetuar o cálculo de dias ideais utilizamos $\boldsymbol{DE = \frac{IED}{1 - IED_{REAL}\%}}$, onde, DE: quantidade de dias estimados para concluir tarefa, IED: prazo necessário para implementar o item (este prazo é definido pela equipe) e $\boldsymbol{IED_{REAL}\%}$: percentual que indica quanto tempo o desenvolvedor estará dedicado a atividade (MARTINS, 2007).

Como uma equipe não se pode expor quem executa a atividade em maior tempo, é realizado a dinâmica em grupo assim o membro busca a melhor forma para realizar a entrega de valor (RITTER, 2015).

Exemplo de caso prático: Uma equipe com 4 horas diárias, tem o seu valor de referência como 1 *Idela Day* = 4 horas, e com velocidade média de um ID = 10 horas, que foi retirado de dados históricos de especialistas. Retira-se a seguinte lista do *backlog*, com valores já atribuídos pelo *Planning Poker*: a) RF01 – Implementar carrinho de compras – 0,5 ID; b) RF02 – Cadastrar livros – 0,3 ID; c) RF03 – Consultar livros – 0,3 ID; d) RF04 – Implementar recomendação automática de livros – 0,4 ID (RITTER, 2015).

Ao utilizar o conceito de Pilha, os membros retiram a tarefa a ser executada e alocado a quantidade de horas ao final do dia. Sendo de total importância que todas as horas utilizadas na atividade sejam alocadas, para que ao final de *sprint* seja determinado o *Ideal Days* concluídos e o seu tempo de execução (RITTER, 2015).

O valor do requisito é definido apenas uma vez e não é modificado, somente o esforço que depende do recurso alocado. No exemplo não temos o histórico determinando o esforço do recurso, utilizamos então empiricamente o valor de 10 horas. E calcula o esforço em: $\boldsymbol{Esforço = tamanhoxvelocidade}$, então o RF01 terá Esforço = 0,5 x 10 = 5 horas (RITTER, 2015).

Em casos que as tarefas não serão finalizadas em um dia, são quebradas para ser evidenciado o trabalho realizado no dia. Em tabela apresenta os dados em evidência: (RITTER, 2015).

Tabela 1 – *Ideal* Days: Dados apurados

Requisito	Tamanho Previsto	Recurso	Horas Reais
RF01	0,5 ID	Recurso 01	4,5 horas
RF02	0,3 ID	Recurso 02	2,5 horas
RF03	0,1 ID	Recurso 01	1,5 horas
RF04	0,4 ID	Recurso 02	3 horas

TOTAL	1,3 ID		11, 5 horas

Fonte: RITTER (2015)

Após o fim da s*print* consideramos a tabela a seguir e nela são evidenciados recursos, os quais o recurso 01 implementou o RF01 e RF 03 e o recurso 02 implementou o RF02 e RF04, dados em tabela (RITTER, 2015):

Tabela 2 – *Ideal* Days: Dados Consolidados

	Velocidade Inicial	ID Previsto	ID Realizado	Horas Real
Recurso 01	10	0,6	0,6	6 horas
Recurso 02	10	0,7	0,7	5,5horas

Fonte: RITTER (2015)

Para determinar a velocidade temos que considerar o tempo gasto em retrabalhos, então usamos a fórmula: $\boldsymbol{velocidade = horas_{Realizadas} + \frac{horas_{retrabalho} X1,3}{IDRealizados}}$, obtendo Recurso 01 = 6 + 0/0,6 = 10 horas e Recurso 02 = 5,5 + 0/0,7 = 7,8 horas, tendo como média da equipe 8,9 horas. No *sprint* posterior será utilizada a média de velocidade 8,9 horas/*point*, desconsiderando o valor anterior. (RITTER, 2015).

A produtividade considerada a quantidade extraída do número de *Ideal Days* entregues em cada *sprint*, neste caso 1,3 ID. Sendo considerado este valor para alocar a quantidade de *Ideal Days* que serão trabalhados na próxima *sprint* e assim sucessivamente são coletados e alocados a quantidade de *Ideal Days* em cada *sprint*. Para se obter a média de produtividade durante a *Release* calculamos na fórmula: $ProdutividadeDaRelease = \frac{ID_{Realizados}}{NúmeroDeSprints}$ (RITTER, 2015).

2.3 FÁBRICA DE TECNOLOGIAS TURING (FTT)

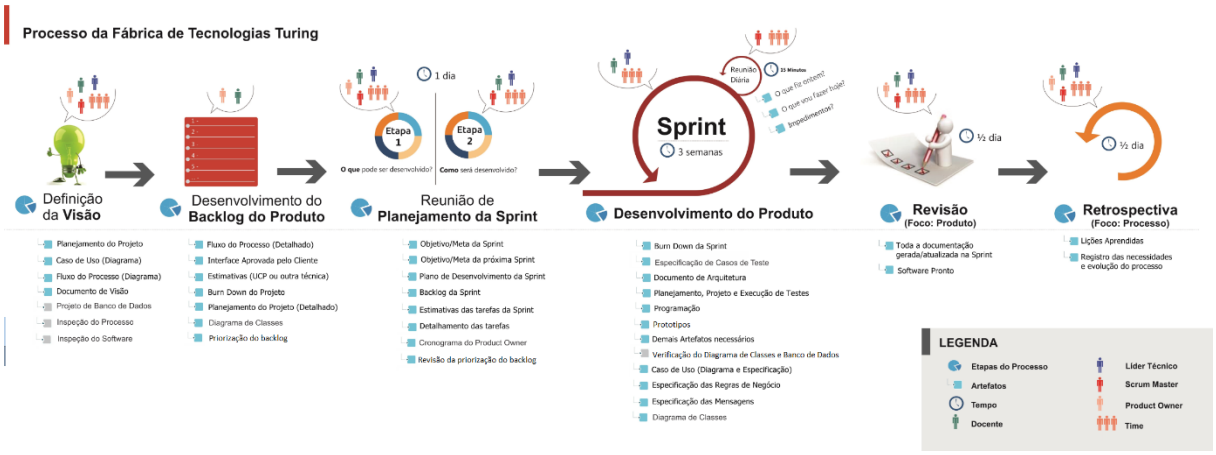
"O paradigma de Fábrica de Software (FS) busca a obtenção de qualidade e produtividade no desenvolvimento e manutenção de software por meio de padronização de processos, reúso de artefatos e controle do sistema de produção" (MOURA, 2008).

Fundada em 2006, a Fábrica de Tecnologias Turing (FTT) é um ambiente de desenvolvimento de software vinculada aos Cursos Bacharelados de Computação da UniEvangélica, que oferece aos graduandos uma experiência em fábrica de software (TURING, 2017).

Os alunos membros da FTT são constantemente incentivados a se aperfeiçoarem, ganhando destaque em recrutamentos de profissionais por empresas externas. Com uma rotatividade alta da equipe, há a necessidade de diversas seleções e treinamento para novos membros, dando a oportunidade de mais alunos fazerem parte da FTT e terem uma vivencia real antes do mercado de trabalho.

A FTT visa em um ambiente de constante melhorias, considerado fundamental o desenvolvimento empírico em seus projetos. Utilizando de tecnologias atuais no mercado como Métodos Ágeis e *frameworks Scrum* e *Open Up*, tem o seu processo de desenvolvimento definido, conforme apresenta a Figura 1.

Figura 3 – Processo de Desenvolvimento da FTT.



Fonte: Repositório FTT (2017).

Este processo defini a estrutura de desenvolvimento seguida pela FTT, contemplando: a) Definição da Visão que ocorre ao início do projeto com a execução do Planejamento do Projeto, Diagrama de Caso de Uso, o Diagrama de Fluxo do Processo, Documento de Visão e possíveis execuções de Projeto de Banco de Dados, Inspeção do processo e Inspeção de Software; b) Desenvolvimento do *Backlog* do Produto após a Definição da Visão com a execução do Fluxo de Processo (Detalhado), Interface Aprovada pelo Cliente, Estimativa, Burn Down do Projeto, Planejamento do Projeto (Detalhado), Diagrama de Classe e Priorização de *Backlog*; c) Reunião de Planejamento que ocorre após o Desenvolvimento do *Backlog* do Produto e se repete antes do início de cada *sprint* com a execução do Objetivo e Meta da *Sprint*; Objetivo e Meta da Próxima *Sprint*, Plano de Desenvolvimento da *Sprint*; *Backlog* da *Sprint*, Estimativa das Tarefas da *Sprint*, Detalhamento das Tarefas, Cronograma do Product Owner e Reunião da Priorização do *Backlog*; d) Desenvolvimento do Produto ocorre durante a *sprint* que acontece após o planejamento de *sprint* com a execução do Burn Down da *Sprint*, Execução dos Casos de Teste, Documento de Arquitetura, Planejamento de Projeto e Execução de Testes,

Programação, Protótipos, Demais atividades Necessárias, Validação do Diagrama de Classe e Banco de Dados, Especificação do Caso de Uso, Especificação das Regras de Negócio, Especificação das Mensagens e Diagrama de Classe; e) Revisão ocorre após cada *Sprint* com a execução da Análise dos Documentos Gerados na *Sprint* e Software do Produto; f) Retrospectiva ocorre após cada *Sprint* com o debate das Lições Aprendidas e Registro das Necessidade e Evoluções do Processo (TURING, 2017).

A FTT realiza o desenvolvimento em *sprint* de 15 dias letivos, com uma variação de horas de trabalho decorrida da rotatividade de membros e diferença carga horaria trabalhada por membro. Os estagiários possuem 6 horas diária de trabalho que totalizam 90 horas por *sprint*, os bolsistas possuem 5 horas diárias de trabalho que totalizam 75 horas e os voluntários possuem 4 horas diárias de trabalho que totalizam 60 horas.

3 RESULTADOS ALCANÇADOS

A fim de melhor análise das técnicas estudas, utilizamos de um *checklist* com os principais fatores a serem analisados para determinar as técnicas a serem aplicadas na Fábrica de Tecnologias Turing.

Tabela 3 – *Checklist*: Comparativo entre técnicas de estimativa.

	Técnica de Estimativa					
Fatores de análise	**PERT**	**DELPHI**	**APF**	**PCU**	***Planning Poker***	***Ideal Days***
Utilizada em Metodologias Ágeis?	X	X	X	X	X	X
Aplicada no *Scrum*?	X		X	X	X	X
Realizada a cada *srint*?	X		X	X	X	X
Não à necessidade de especialistas?					X	X
Aplicado antes da especificação de requisitos?	X	X				
Aplicado a partir da especificação de requisitos?	X	X	X	X	X	X
Acompanha a atual velocidade da equipe?						X
Estima o tamanho do requisito?			X	X	X	
Estima o esforço de tempo dos requisitos?	X		X	X		X

Fonte: Elaborado pelo autor.

A FTT passa por constante rotatividade da equipe, tendo grandes oscilações na sua velocidade média de desenvolvimento, este então é o fator primordial na comparação e diferenciação das técnicas. Além da velocidade todos os outros fatores do *checklist* foram de suma importância para identificar qual técnica de estimativa de esforço de tempos a utilizar em projeto de softwares desenvolvidos na Fábrica de Tecnologias Turing.

Foi identificado que a técnica de estimativa que mais se adequa para ser implantada na FTT é a *Ideal Days* que acompanha a velocidade da equipe, usada como parâmetro de velocidade média para a próxima *sprint*. A técnica de estimativa *Ideal Days*, calcula a quantidade de tempo estimada para o desenvolvimento do produto “pronto” com a utilização do *story point*, definido pela técnica de estimativa *Planning Poker*. Percebe então a necessidade de implantar as duas técnicas, a *Planning Poker* para fornecer o tamanho dos requisitos e a *Ideal Days* para calcular a quantidade de horas estimadas e qual a velocidade média da equipe por *sprint*.

Por fator de necessidade de estimar a especificação de requisitos notou-se a necessidade de uma técnica diferente, pois o *Planning Poker* e *Ideal Days* necessitam da especificação dos requisitos pronta e aprovada, para ~~que~~ a análise de complexidade acontecer de forma assertiva. Então, as estimativas da análise de requisitos continuarão ~~a~~ sendo realizadas através da técnica PERT, a qual atende a necessidade de estimar a especificação de requisitos e utiliza parâmetros de acordo com o tempo gasto pela equipe.

Portanto, na estimativa das *sprints,* a Análise de Requisito e Teste de Inspeção utilizará a técnica PERT e no Desenvolvimento e Teste Funcional as técnicas *Planning Poker* e *Ideal Days.*

A partir da definição de proposta, é dado início a aplicação no ambiente da FTT.

3.1 Aplicação Na Fábrica De Tecnologias Turing

Definidas as técnicas de estimativa de esforço de tempo que mais se adequaram ao ambiente, aconteceu a validação das mesmas com os responsáveis pela gestão de projetos da FTT, consideram o consentimento e aprovação do P.O, *Scrum Master*, Líder de Requisitos e Professores Orientadores.

Após aprovação transcorreu um workshop para treinamento dos membros da FTT, o mesmo decorreu no próprio ambiente da Fábrica de Tecnologias Turing e foi composto por uma apresentação do formato a ser utilizado, expondo pontos a serem observados durante a estimativa de forma participativa os membros questionaram sobre pontos como fariam para decidirem a complexidade, com a explicação que se decorre de uma base de comparação com demais requisitos, com o workshop alcançaram uma visão didática de como a técnica é aplicada e a função e responsabilidade que teriam no processo de estimativa, também foi apresentado um manual de ‘Técnicas de Estimativa de Esforço de Tempo da FTT’ (ANEXO A), o qual é destinado a orientá-los sobre a utilização das técnicas no ambiente da FTT.

3.1.1 Resultado da Primeira *Sprint*

A primeira *sprint* de acompanhamento iniciou com o P.O. apresentando, através do *backlog* definido com a equipe, os requisitos a serem trabalhados no *sprint*. Para desenvolvimento foi definido: Lançar Falta, Lançar Nota, Relatórios, Critério de Avaliação Padrão e Mantenedora. E para Análise de Requisitos: Turma, Matricular Aluno, Realizar Pagamento, Gerar Dívida e Gerenciar Cobrança.

Na estimativa do desenvolvimento utilizou *Planning Poker* e *Ideal Days* em que ocorreu a entrega das cartas de *story point* para os membros ao iniciar a estimativa identificamos dificuldade dos membros diferenciarem o *story point* do tempo gasto nos requisitos, por estarem acostumados a estimar o tempo por PERT. Como ocorreu no caso estimarem o *story point* do Lançar Falta consideraram 8 pontos e estimaram Lançar Nota com a metade dos pontos, com a justificativa que os dois eram muito parecidos e usariam o código já produzido, com algumas modificações logo, foi realizado uma intervenção, para que levassem em consideração somente a complexidade e não a reutilização do código, somente então, os membros determinaram a mesma complexidade a ambos. Como não havia parâmetro para estipular a velocidade de tempo para produção dos *story point*, realiza uma média ponderada com base no autoconhecimento dos desenvolvedores, chegando a velocidade de 3,3 pontos/hora. Então temos:

Tabela 4 - Estimativa Primeira *Sprint* de Acompanhamento do Desenvolvimento.

Estimativa Primeira *Sprint* de Acompanhamento do Desenvolvimento			
Prioridade	Requisito	*Planning Poker*	*Ideal Days*
1º	Lançar Falta	8pts	26h 30m
2º	Lançar Nota	8pts	26h 30m
3º	Relatórios	20pts	66h
4º	Critério de Avaliação Padrão	33pts	108h 50m
5º	Mantenedora	3pts	9h 50m

Fonte: Elaborada pelo autor.

Na estimativa de análise de requisitos a técnica de estimativa foi PERT, os membros dividiram entre eles os requisitos a serem desenvolvidos e a estimativa foi feita de forma individual, ou seja, cada requisito foi estimado por seu responsável. A estimativa final do requisito seguiu os seguintes parâmetros, o responsável pelo requisito apontou os três tempos estimados para especificar e o líder de teste apontou os três tempos estimados para a inspeção do requisito, assim obtivemos a soma das estimativas dos tempos otimistas, mais prováveis e pessimistas para cada requisito, e foi utilizado a fórmula PERT para chegar a estimativa final dos requisitos.

Tabela 5 – Estimativa da Primeira *Sprint* de Acompanhamento da Análise de Requisitos

Estimativa da Primeira *Sprint* de Acompanhamento da Análise de Requisitos					
Prioridade	Requisito	Temp. Pessimista	Temp. Otimista	Temp. Provável	PERT
1º	Turma (Refatorar: Frequência e Avaliação Incluir: Critério de Avaliação)	59	32	44	44,5
2º	Matricular Aluno	64	32	52	50,66
3º	Realizar Pagamento	49	32	38	38,83
4º	Gerenciar Dívida	44	26	36	35,6

5º	Gerenciar Cobrança	44	26	36	35,6

Fonte: Elaborada pelo autor.

No início da *sprint* de intervenção as técnicas de estimativa que haviam sido implantadas identificaram a necessidade da mudança de tecnologia utilizada no desenvolvimento, pois precisavam que o desenvolvimento ocorresse com mais agilidade e a tecnologia utilizada não permitia. Então a equipe deixou de utilizar o Thymeleaf e o orientador definiu que a equipe priorizasse o estudo do Angular 2, a tecnologia a ser utilizada no sistema. Com este impedimento o planejamento da equipe de desenvolvimento se tornou inalcançável, pois a equipe precisou priorizar o estudo do Angular e a definição da estrutura do sistema, já que a mesma interfere em todo o desenvolvimento. As estimativas se tornaram inalcançáveis e o fechamento não foi o esperado com nenhum dos requisitos definidos. A equipe de requisitos seguiu o seu curso do planejamento conseguindo terminar todas as atividades na *sprint*. Não ouve acompanhamento de tempo real gasto durante a sprint.

3.1.2 Resultado Segunda *Sprint*

Na segunda *sprint* aconteceu a inserção de novos membros que tornou necessário realizar novamente o workshop para alinhamento da equipe. Após workshop o PO apresentou o atual *backlog* e junto a equipe definiram os requisitos a serem produzidos. A equipe de desenvolvimento foi responsável pelos requisitos: Manter Unidade, Manter Mantenedora, Manter Instituição, Manter Horário, Manter Instituição de Escolaridade, Frequência Padrão, Avaliação Padrão, Manter Disciplina, Manter Emolumentos, Manter Banco, Manter Agência/Conta e Plano de Pagamento. E a equipe de análise de requisitos: Turma, Bloco/Piso/Sala, Critério de Avaliação e Relatório.

Na estimativa da equipe de desenvolvimento utilizou o *Planning Poker* e *Ideal Days*, foram entregues as cartas de *story point* aos membros e ~~a~~ cada requisito apresentado era discutida e determinada a quantidade de *story points*. Novamente por não ter parâmetro fizemos uma média com os desenvolvedores de quanto tempo consideravam necessário para desenvolver um determinado requisitos e dividimos pela quantidade de points determinada a ele, chegando a velocidade de 2,4 point/hora de desenvolvimento, obtendo a estimativa:

Tabela 6 – Estimativa da Segunda *Sprint* de Acompanhamento do Desenvolvimento

Estimativa da Segunda *Sprint* de Acompanhamento do Desenvolvimento		
Média de velocidade: 2,4 points/horas		
Requisitos	Points	Tempo Estimado
Manter Unidade	9	21,6

Manter Mantenedora	9	21,6
Manter Instituição	9	21,6
Manter Horário	13	31,2
Manter Instituição de Escolaridade	5	12
Frequência Padrão	5	12
Avaliação Padrão	5	12
Manter Disciplina	4	9,6
Manter Emolumentos	8	19,2
Manter Banco	5	12
Manter Agência/Conta	8	19,2
Plano de Pagamento	13	31,2

Fonte: Elaborada pelo autor.

Para a estimativa dos requisitos de análise, o PO apresentou cada requisitos e o mesmo foi divido pelos membros da equipe, assim cada membro estimava o requisito a qual era responsável. O cálculo para definição da estimativa inclui o tempo gasto na equipe de teste, como realizado na primeira *sprint*, chagando a seguinte estimativa:

Tabela 7 – Estimativa da Segunda *Sprint* de Acompanhamento Análise de Requisitos

Estimativa da Segunda *Sprint* de Acompanhamento Análise de Requisitos				
Técnica PERT	Estimativa			
Requisito	Pessimista	Otimista	Provável	PERT
Turma	42	28	31	32,33
Bloco/Piso/Sala	30	23	26	26,17
Critério de Avaliação	20	16	18	18,00
Matrícula	39	26	31	31,50
Relatório	21	12	18	17,50

Fonte: Elaborada pelo autor.

Ao iniciar a *sprint*, os desenvolvedores perceberam que a arquitetura utilizada na antiga tecnologia de desenvolvimento não se adequava a atual, como a arquitetura inadequada pode complicar e, consequentemente, causar o fracasso do projeto, precisou priorizar a atualização da arquitetura de acordo com a tecnologia empregada e assim continuar o desenvolvimento. Com a entrada de novos membros os antigos precisaram despor de tempo para auxiliá-los, e isso não foi levado em consideração no planejamento, logo, não foi possível atingir o desenvolvimento dos requisitos “pronto” como proposto no planejamento, mas a arquitetura foi definida e estruturada, não permitindo que esse tipo de impedimento voltasse a ocorrer.

Alguns requisitos foram iniciados na *sprint*, mas não atingiram a definição de pronto do planejamento da *sprint* e por falta de utilizar uma ferramenta para controle de horas reais, não foi possível relacionar a quantidade de pontos já realizadas com a quantidade de tempo

gasto, assim o fechamento da *sprint* apresentou o que foi feito até o momento em cada requisito de desenvolvimento.

Os requisitos de análise foram realizados conforme planejado, mas também por falta de ferramenta para controle de tempo gasto não foi possível relacionar o tempo estimado com o tempo real gasto.

No fechamento foi proposto a implantação de utilizar a ferramenta GitLab para controle das horas gastas, a mesma ferramenta já é utilizada como repositório de arquivos e gestão do desenvolvimento.

3.1.3 Resultado Terceira *Sprint*

Na terceira *sprint* com a equipe alinhada não se fez necessário o treinamento das técnicas de estimativa novamente, então seguiu com o processo de planejamento, o P.O. apresentou o *backlog* priorizado e junto a equipe entraram no consenso dos requisitos a serem especificados e desenvolvidos na *sprint*. O desenvolvimento ficou responsável pelos requisitos: Endereço, *Test-Driven Development* (TDD), Horário, Swagger, Província/Município e Atribuir Valor ao Curso. E a análise de requisitos: Matricular Aluno, Gerenciar Dívida, Gerenciar Cobrança, Manter Turma, Relatório, Fechamento de Disciplinas, Permissões de Usuário, Manter Encargos, Manter Bolsas, Histórico do Aluno, Manter Tipo de Bolsa e Resumo Acadêmico do Aluno.

A estimativa dos requisitos para desenvolvimento foram estimados por *Planning Poker* e *Ideal Days*, teve início com a entrega das cargas de *story point*, então o P.O. apresentou os requisitos a serem estimados e os membros apresentaram a quantidade de pontos que julgavam correta, quando necessário os *story points* com maior diferença eram justificados até chegarem a um consenso para os pontos necessários no desenvolvimento de cada requisitos, como no *sprint* anterior as horas reais não foram registradas, foi utilizado a mesma média de velocidade da *sprint* anterior e chegou a seguinte estimativa:

Tabela 8 – Estimativa da Terceira *Sprint* de Acompanhamento do Desenvolvimento

Estimativa da Terceira *Sprint* de Acompanhamento do Desenvolvimento		
Média de velocidade: 2,4 points/horas		
Requisitos	Points	Tempo Estimado
Endereço	13	31,2
TDD	8	19,2
Horário	8	19,2
Swagger	5	12
Província / Município	3	7,2

Atribuir Valor ao curso	20,3	48,7

Fonte: Elaborada pelo autor.

Para a análise de requisitos a estimativa foi realizada pela técnica PERT, como nas demais *sprints* cada membro estimou os requisitos que eram responsáveis e o teste estimou o tempo necessário para testar cada requisitos, alcançando a seguinte estimativa para a *sprint*:

Tabela 9 – Estimativa da Terceira *Sprint* de Acompanhamento da Análise de Requisitos

Estimativa da Terceira *Sprint* de Acompanhamento da Análise de Requisitos				
Técnica PERT	Estimativa			
Requisito	Pessimista	Otimista	Provável	PERT
Matricular aluno	15	10	12	12,17
Gerenciar Dívida	35	25	30	30
Gerenciar Cobrança	40	30	35	35
Manter Turma	32	26	30	29,67
Relatório	25	20	22	22,17
Fechamento de Disciplina	20	15	17	17,17
Permissões de Usuário	33	22	26,7	26,97
Manter Encargos	4	2	3,5	3,33
Manter Bolsa	4	2	3,5	3,33
Histórico do Aluno	4	2	3,5	3,33
Manter Tipo de Bolsa	3	2	2	2,17
Resumo Acadêmico do Aluno	3	2	2	2,17

Fonte: Elaborada pelo autor.

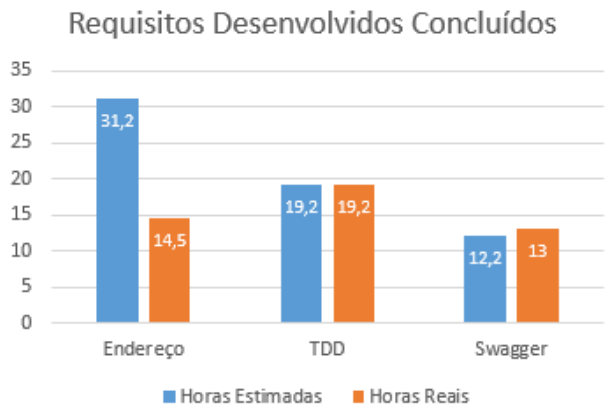
Nesta *sprint*, diferente das demais, utilizou-se a ferramenta GitLab para registro de horas gastas em cada atividade, possibilitando uma análise real de andamento da *sprint* e percepção da assertividade da estimativa da *sprint*.

Houve alguns impedimentos durante o processo, como o afastamento de um membro e algumas dificuldades por falta de experiência, mas a *sprint* seguiu sem mudança do que foi planejado, o que possibilitou que calculasse a viabilidade do uso das técnicas de estimativa *Planning Poker* e *Ideal Days*, as novas técnicas, implantadas no ambiente da FTT.

Com a utilização da ferramenta de acompanhamento de horas foi realizado a comparação de horas planejadas e horas gastas dentro da *sprint*, com a relação de horas gastas versos horas reais apresentada em gráficos para melhor visualização na formulação dos gráficos utilizamos uma conversão por regra de três para que os minutos fossem convertidos na base 100 em vez de 60, para o cálculo do gráfico, como exemplo 1 hora e 30 minutos será 1,50.

A equipe de desenvolvimento planejou 6 requisitos os quais 50% foi concluído com o desenvolvimento da funcionalidade e execução dos testes funcionais.

Gráfico 1 – Requisitos Desenvolvidos Concluídos

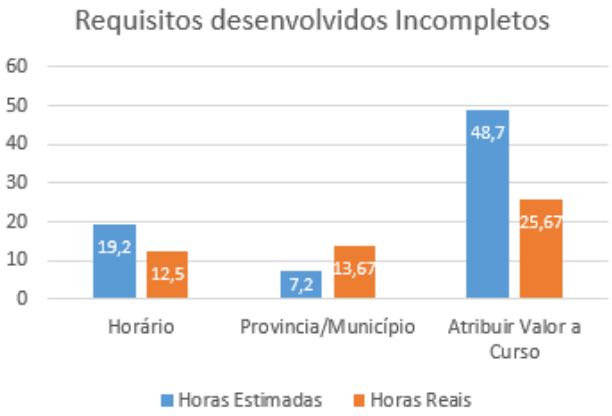


Fonte: Elaborada pelo autor.

O gráfico apresenta os requisitos de desenvolvimento concluídos. Eles alcançaram aproximadamente uma média de 75% de assertividade de sua estimativa no planejamento, com o requisito Endereço com a maior diferença de horas estimadas versos horas reais, desenvolvido por um membro com maior experiência, a oscilação é torna perceptivo o desnível de conhecimento da equipe.

O gráfico apresenta os requisitos de desenvolvimento que ficaram incompletos. Não foi possível estimar a quantidade de percentagem concluída de horas estimadas versos horas reais, pois necessita a conclusão do requisito para obter este parâmetro, assim temos a quantidade de horas estimadas e a horas reais que já foram gastas, nos dando a percepção de quanto tempo possuem em cada requisito para concluir o requisito estimado.

Gráfico 2 – Requisitos Desenvolvidos Incompletos



Fonte: Elaborada pelo autor.

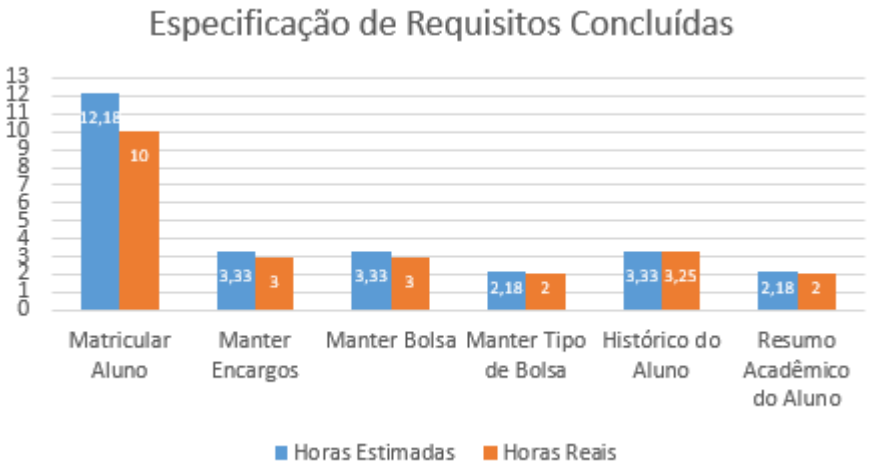
Foi realizado a média de velocidade somente com requisitos de desenvolvimento concluído, assim pegamos os pontos e horas reais dos requisitos Endereço, TDD e Swagger, em que tiveram um total de 26 pontos e 46,7 horas reais resultando em uma velocidade de 1,80

ponto/hora de desenvolvimento. Esta velocidade será utilizada para a próxima *sprint* de desenvolvimento da Fábrica de Tecnologias Turing.

A análise de requisito concluiu 50 % dos 12 requisitos planejados.

Os requisitos concluídos apresentaram aproximadamente 88% de assertividade da estimativa, no gráfico 4 apresenta-se a diferença individual de horas estimadas e horas reais de cada requisito concluído.

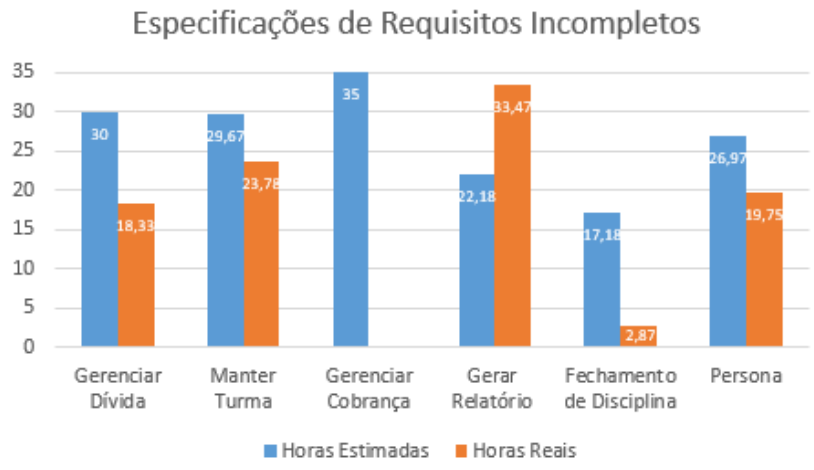
Gráfico 3 – Especificação de Requisitos Concluídas



Fonte: Elaborada pelo autor.

Para os requisitos de análise incompletos, é visto a quantidade de horas estimadas e as horas reais que já foram gastas, mas não podemos afirmar porcentagem que falta pois não tem a quantidade exata do requisito que foi realizada. Assim analisamos somente a diferença de tempo que foi planejado para o requisito e a quantidade que já foi gasta, expressa no gráfico.

Gráfico 4 – Especificação de Requisitos Incompletos



Fonte: Elaborada pelo autor.

4 CONSIDERAÇÕES FINAIS

Como previsto os resultados não seriam de imediato e que a implantação de técnicas de estimativa de esforço de tempo se torna imprescindível para o sucesso do planejamento de projeto de software. A implantação na FTT foi um processo longo e gradual, foi realizado um acompanhamento durante três *sprints*, com o crescimento visível.

Na primeira *sprint* no planejamento percebeu uma maior dificuldade de entendimento dos parâmetros de medida por parte dos membros. Durante a *sprint* a mudança para a equipe de desenvolvimento do que seria feito desviou totalmente do planejado da *sprint e* tornou o primeiro planejamento de desenvolvimento inviável. O impedimento fez a equipe aprimorar o seu formato de planejamento. Para implantação das técnicas de estimativa não poderia ser melhor. É preferível que os imprevistos aconteçam durante esta etapa, pois é mais fácil aprimorar o formato em um período que nem tudo esteja consolidado.

A segunda *sprint* teve a inserção de novos membros e novamente uma dificuldade com os parâmetros de medida, mas dessa vez de forma mais branda. Também ocorreu durante a *sprint* uma alteração no planejamento que considere os riscos da *sprint*, afetando a conclusão dos requisitos planejados, mas diferente da primeira *sprint* foi possível realizar ao menos parte do que foi proposto. Nesta etapa é identificado a necessidade de uma ferramenta para controle das horas gastas durante a *sprint*. Percebeu-se a evolução no planejamento para prever os possíveis riscos, apresentando uma melhora na retenção dos impedimentos.

Na a terceira *sprint* o resultado apresentou de uma maior facilidade de aplicação das técnicas de estimativa no planejamento a equipe já se sentia à vontade com os parâmetros utilizados, a *sprint* correu como planejada, obtendo um resultado de 50% do total das atividades com o desenvolvimento das funcionalidades e testes funcionais concluídos, a porcentagem de 50% é decorrido da adaptação as técnicas de estimativa aos projetos, a assertividade é adquirida com o tempo e experiencia de uso. As horas reais foram registradas através da ferramenta GitLab possibilitou calcular que a equipe de desenvolvimento consumiu média 75% das horas estimadas nos requisitos concluídos e a equipe de análise de requisitos 88% das horas estimadas nos requisitos concluídos.

Com os resultados obtidos é aparente que a implantação das técnicas de estimativa de esforço de tempo atuará como uma ferramenta fundamental no planejamento e gerenciamento dos projetos desenvolvidos na Fábrica de Tecnologias Turing do Centro Universitário de Anápolis – UniEvangélica.

REFERÊNCIAS BIBLIOGRÁFICAS

AGILE MANIFESTO. **Manifesto for Agile Software Development**, 17 February 2001. Disponível em: <http://www.agilemanifesto.org/>. Acessado em 16 de outubro 2017.

ANDRADE, Edméia Leonor Preira de; OLIVEIRA, Káthia Marçal de. **Aplicação de Pontos de Função e Pontos de Caso de Uso de Forma Combinada no Processo de Gestão de Estimativa de Tamanho de Projetos de Software Orientado a Objetos.** Disponível em: <http://www.ip.pbh.gov.br/ANO7_N1_PDF/IP7N1_andrade.pdf> Acesso em: 02 de outubro de 2017.

BERNARDO, Kleber. Planning Poker: **A técnica baseada no consenso**, 2014. Disponível em: <https://www.culturaagil.com.br/planning-poker-tecnica-baseada-consenso/>. Acesso em 20 de setembro de 2017.

BONFIM, Márcia Regina Guiotti; ANDRADE, José Romildo: **Guia de Contagem de Ponto de Função do Ministério do Planejamento, Orçamento e Gestão (MP). Versão 1.0.** 2015.

CARVALHO, Ariadne M. B. Rizzoni; CHIOSSI, Thelma C. dos Santos. **Introdução a Engenharia de Software.** São Paulo: Unicamp, 2001.

FERNANDES, Aguinaldo Aragon; TEIXEIRA, Descartes de Souza. **Fábrica de Software**. São Paulo: Atlas, 2011.

FERREIRA, Felix Soares; POCIVI, Viviane Carla Batista. **Método para Estimativa em Projeto de Software para Fábrica de Software Acadêmica.** 2016.

HIGHSMITH, Jim. **Gerenciamento Ágil de Projeto.** Rio de Janeiro, 2012.

IMPROISSI, Jorge. **Imergindo nos Valores do Scrum.** Disponível em: <https://www.culturaagil.com.br/imergindo-nos-valores-do-scrum/> Acesso 21 de dezembro de 2017.

LICHIRGU, Ana Gabriela F. Bittencourt. **Contagem de Ponto de Função.** 2016. Disponível em: <http://www.devmedia.com.br/contagem-de-pontos-de-funcao/34390> Acesso em: 25 de setembro de 2017.

MOURA, Luzia. **Definição e Estabelecimento de Processos de Fábrica de Software em uma Organização de TI do Setor Público.** 2008.

MINDMASTER, Educação Profissional. Scrum: A Metodologia Ágil Explicada de Forma Definitiva. Disponível em: <http://www.mindmaster.com.br/scrum/>. Acesso em 20 de Novembro de 2017.

PETERS, James F.; PEDRYCZ, Witold. **Engenharia de Software: Teoria e Prática.** Rio de Janeiro: Campus, 2001.

RITTER, Roger. **Scrum e Planning Poker: Análise de Estimativa de Software.** 2014. Disponível em <https://www.devmedia.com.br/scrum-e-planning-poker-analise-de-estimativa-de-software/31019> Acesso 16 de novembro de 2017.

RITTER, Roger. **Planning Poker e Ideal Days: Técnica de abordagens de Estimativa ágil.** 2015. Disponível em: <http://www.rogerritter.com.br/tag/ideal-day/> Acesso 04 de outubro de 2017.

SANTOS, Cleber. **Estimativa de 3 pontos, um ponto forte**, 2015. Disponível em: < http://pmpath.com.br/estimativa-de-3-pontos-um-ponto-forte/>. Acesso em 29 de agosto de 2017.

SCHWABER, Ken; SUTHERLAND, Jeff. **Guia do Scrum.** Disponível em: <https://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf> Acesso em: 30 de setembro de 2017.

SOARES M. dos S.; **Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software**. Unipac-Universidade Presidente Antônio Carlos, Faculdade de Tecnologia e Ciências de Conselheiro Lafaiete, 2010.

SOMMERVILLE, Ian. **Engenharia de Software.** 9. ed. São Paulo: Pearson Prentice, 2011.

TURING, Fábrica de tecnologias. **Processo FTT.** 2017.

VAZQUEZ, Carlos Eduardo; SIMÕES, Guilherme Siqueira; ALBERT, Renato Machado Albert. **Análise de Pontos de Função.** São Paulo: Erica, 2003.

VIGNADO, Alexandre Luis. **Afinal como estimar usando o Planning Poker?**, 2016. Disponível em: <http://www.vignado.com.br/afinal-como-estimar-usando-o-planning-poker/>. Acesso em 20 de setembro de 2017.

ANEXO A – Manual de Estimativas para Planejamento de Sprint da Fábrica de Tecnologias Turing

O Manual de Estimativas para Planejamento de *Sprint* da Fábrica de Tecnologias Turing (FTT) estabelece um padrão a ser seguido para utilização das técnicas de estimativa de esforço de tempo, evidenciadas como mais adequadas a estrutura do ambiente.

As técnicas de estimativas empregadas foram separadas entre as fases de especificação de requisitos e desenvolvimento de requisitos. Como definição a especificação de requisitos é estimada pela técnica *Program Evalution Review Technique* (PERT) e o desenvolvimento de requisitos estimada pelas técnicas *Planning Poker* e *Ideal Days*.

Especificação de Requisitos:

Na FTT a estimativa da especificação de requisitos é realizada para controle de atividades e desempenho dos membros.

Nesta estimativa é necessário ter uma ideia do que será especificado. Por ser uma estimativa realizada sem dados concisos utiliza-se a técnica PERT que é realizada através de três pontos e baseia-se no conhecimento e experiência da equipe. São retirados os valores da Estimativa Otimista (O), Estimativa Mais Provável (M) e Estimativa Pessimista. A partir daí é feito o cálculo da estimativa PERT da tarefa, onde: $\boldsymbol{PERT} = \boldsymbol{P}+\boldsymbol{4M}+\boldsymbol{O}\ \boldsymbol{6}$. A estimativa mais provável recebe peso 4, apesar da diferença os outros pontos podem ter grande influência estando muito distantes da estimativa mais provável. Para medir a distância dos elementos calculamos através das médias estatísticas a variância e desvio padrão. A variância é definida pela fórmula: $\boldsymbol{Vari}$â$\boldsymbol{ncia}$(formatação) = ($\boldsymbol{P}-\boldsymbol{O}\ \boldsymbol{6}$) $\boldsymbol{2}$. E o Desvio Padrão como a dispersão entre as medias, na fórmula: $\boldsymbol{Desvio\ Padr}$ã$\boldsymbol{o}$ = $\boldsymbol{P}-\boldsymbol{O}\ \boldsymbol{6}$(SANTOS, 2015).

Assim a estimativa segue o seguinte roteiro:

1. O P.O. apresenta o *backlog* priorizado dos requisitos a serem especificados e junto a equipe de análise define os requisitos a serem especificados na *sprint*;
2. A equipe de análise define, entre ela, quem será o responsável pelo requisito a ser especificado;
3. Cada requisito definido é estimado nos três pontos (otimista, mais provável e pessimista) pelo responsável da especificação;
4. O líder de teste deve também estimar os três pontos (otimista, mais provável e pessimista) para execução do teste de inspeção em cada requisito;

5. Então é somado para cada requisito as estimativas realizadas pelo responsável do requisito e o líder de teste;
6. Com a soma das estimativas otimistas, mais provável e pessimista, então é realizado o cálculo da técnica PERT;
7. Os dados da estimativa devem ser anexados para análise de resultados no fechamento da *sprint*;
8. Cada membro é responsável por alocar o tempo gasto do requisito na tarefa do GitLab;
9. No fechamento de *sprint* o tempo estimado versos o tempo gastos são comparados e é identificado as causas em casos de não conformidade.
10. Todos os dados devem ser guardados para aprimorar os próximos planejamentos.

Desenvolvimento de Requisitos:

Para a equipe de desenvolvimento a estimativa é utilizada para acompanhamento do desenvolvimento dos membros e controle das demandas de entregas do produto "pronto" estabelecido, então em cada *sprint* são definidos os requisitos a serem trabalhados de acordo com a demanda. A estimativa de desenvolvimento é realizada pelas técnicas de estimativa *Planning Poker* e *Ideal Days*.

O *Planning Poker* é definido pelo entendimento dos membros sobre a complexidade de cada requisito, em que designado a quantidade de esforço necessária em cada atividade, o esforço é alocado em quantidade de story point. É baseado em informações recolhidas dos *stakeholders*, geralmente escrito em estória de usuário ou casos de uso. (espaçamento) Os membros da equipe recebem um conjunto de cartas, com determinada sequência, conhecido como *story point* com valores: 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100. Também existem as cartas de 'interrogação', 'infinito' e em alguns modelos, a carta 'café'. As cartas zero (0), infinito (∞), interrogação (?) e "café" serão utilizadas de forma que: a) zero (0): Quando a estória de usuário é tão simples que os membros não consideram que haja necessidade de esforço e que será realizado em minutos, e em vez de usar a carta ½, utiliza a carta 0, b) infinito (∞): Sendo o oposto de zero, está estória de usuário é considerada tão grande que os membros não consideram ser possível estimas, precisa ser melhor entendida ou quebrada em estórias de usuário menores, c) interrogação (?): Utilizada se, o membro mesmo com a explicação e discussão sobre a estória de usuário não conseguiu compreender o suficiente para estimar, sendo necessário discutir melhor a estória de usuário até que toda equipe compreenda e d) café: Sinaliza que algum

membro precisa de uma pausa da estimativa, podendo o *Scrum Master* definir o tempo de intervalo (BERNARDO, 2014).

Funcionamento do *Planning Poker*: O *Product Owner* apresenta aos membros a estória de usuário de maneira clara e objetiva. É discutido os critérios de aceitação e tiram as dúvidas da atividade com o *Product Owner*.Com exceção do *Product Owner*, cada membro apresenta a quantidade de *story points* necessários para a atividade selecionando a carta do baralho em suas mãos. Depois de todos estarem com a carta selecionada em mãos, todos viram ao mesmo tempo. Em caso de valores diferentes, os membros apresentam as justificativas dos valores mais alto e mais baixo. E votado novamente até que o grupo chegue em um acordo. Caso não consiga se chegar a um acordo, o *Scrum Master* pode interferir, solicitando ao time que entre em acordo onde os membros iram decidir ir pela maioria, média ou confiar em determinado participante que já desenvolveu o mesmo trabalho da estória de usuário (BERNARDO, 2014).

O *Ideal Days* é utilizado para realizar estimativa de forma ágil, é aplicado no planejamento de interações, tendo como destaque o cálculo da velocidade com base nas horas gasta pela equipe para implementar um ID - *Ideal Day* (RITTER, 2015).

Ideal Days (em português, Dias Ideais), ao estimar em dias ideais, retiramos todas as atividades adicionais da equipe como reuniões, telefonemas, envio de e-mails entre outras e conta-se apenas a sua dedicação a atividade proposta. Ao fazermos estimativas baseadas a tempo não temos um histórico consistente, pois com a evolução da equipe uma atividade que, por exemplo, demorava 5 horas passa ser executada agora e realizada em 4 horas, o que torna inviável utilizar a base anterior (KLEIN, 2017).

Para efetuar o cálculo de dias ideais utilizamos $\boldsymbol{DE = \frac{IED}{1 - IED_{REAL}\%}}$, onde, DE: quantidade de dias estimados para concluir tarefa, IED: prazo necessário para implementar o item (este prazo é definido pela equipe) e $\boldsymbol{IED_{REAL}\%}$: percentual que indica quanto tempo o desenvolvedor estará dedicado a atividade (MARTINS, 2007).

Então a estimativa do desenvolvimento de requisitos segue o seguinte roteiro:

1. O P.O. apresenta o *backlog* dos requisitos a serem desenvolvidos priorizados e junto com o time defini os requisitos a serem trabalhados nesta *sprint*;
2. Os requisitos definidos são apresentados pelo P.O. e os membros devem questionar a fim de entender todos os detalhes e complexidade de cada um;
3. Os requisitos são postos a mesa um por um para que seja realizado o story point de cada um pela técnica *Planning Poker*;

4. Os membros recebem as cartas do story point, então o requisito é avaliado por todos, em segredo cada membro define a quantidade de story point que considera necessário para o requisito e então ao mesmo tempo todos apresentam a carta escolhida;
5. Em caso de grandes divergências de quantidade os membros devem defender a carta escolhida, assim o membro que apresentou o menor valor e maior valor dão a justificativa e é discutido entre o grupo, novamente os membros selecionam a carta em segredo e todos mostram simultaneamente, este processo deve se repetir até atingir um consenso, em casos que não é estabelecido um único valor após várias tentativas é realizado uma média entre as quantidades escolhidas;
6. Os story points definidos para cada requisito devem ser anexados em planilha de estimativa;
7. Após a definição de todos os story points então é aplicado a técnica *Ideal Days*, em que é utilizada a velocidade alcançada pela equipe na *sprint* anterior para calculo do tempo a ser estimado para cada requisito a ser desenvolvido;
8. Assim a velocidade da equipe é multiplicada pela quantidade de story point definido em que gera o resultado do tempo estimado para o requisito;
9. Caso não haja histórico de velocidade anterior deve utilizar um requisito para cálculo, em que todos os desenvolvedores palpitam a quantidade de tempo que gastaria neste requisito e com a média de tempo é divido pela quantidade de story point que determinaram, assim se chega a uma velocidade, é indicado que utilize está prática somente na primeira *sprint* do projeto;
10. As estimativas estabelecidas devem ser anexadas também a planilha para comparativo de tempo estimado versos tempo gasto no fechamento da *sprint*;
11. Cada membro é responsável por alocar o tempo gasto do requisito na tarefa do GitLab;
12. No fechamento de *sprint* o tempo estimado versos o tempo gastos são comparados e é identificado as causas em casos de não conformidade.
13. No fechamento da *sprint* deve ser calculado a velocidade atingida pela equipe durante a *sprint*.
14. Todos os dados devem ser guardados para aprimorar os próximos planejamentos.

Este manual deve servir de apoio a execução do planejamento das *sprints*.