Estimativa de esforço em projetos de desenvolvimento de software usando redes bayesianas e análise de ponto de função

Jonatan Ribeiro de Moura*
Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Rua Emídio dos Santos, S/N, Barbalho
Salvador, Bahia
natanojj@gmail.com

Antonio Carlos Santos Souza†
Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Rua Emídio dos Santos, S/N, Barbalho
Salvador, bahia
antoniocarlos@ifba.edu.br

Resumo

A estimativa de *esforço de software* é um processo que envolve diversos fatores internos e externos intrínsecos no desenvolvimento de uma aplicação e mensurá-los em determinados casos não é uma tarefa fácil, por conta dos *inputs* necessários para o aferimento, pois ele depende diretamente de uma base de dados sólida baseada nos históricos de projetos anteriores, incluindo seus artefatos. Com esses dados é possível fazer estimativas e prever o custo necessário para desenvolver determinadas tarefas internas. Pensando nisso, o presente trabalho propõe uma ferramenta que utiliza *redes bayesianas* e *análise de ponto de função* para estimar o *esforço* necessário de uma equipe para desenvolver uma solução lógica.

Palavras-chave

Estimativa de esforço, *Redes bayesianas*,*Análise de ponto de função*, *Estimativa de Custo*

1. INTRODUÇÃO

Ao longo do tempo a tecnologia vem se expandido pelo globo terrestre, juntamente com a necessidade de consumir aplicações lógicas úteis que auxiliem na resolução de problemas do dia a dia em um prazo razoável. O desenvolvimento dessas soluções envolve diversas regras de negócio que o sistema deve contemplar de acordo com cada cliente, sendo intrínseco no processo de construção do *software* questões como custo, risco, tempo e restrições [1, 2, 3, 4, 5].

As empresas responsáveis por criar essas soluções lógicas vêm buscando melhorar o seu processo de desenvolvimento através da engenharia de *software* que tem como objetivo organizar, melhorar a produtividade e a qualidade dos sistemas através da gestão dos projetos a fim de aumentar o índice de entregas úteis [2]. Entretanto, para uma empresa realizar esse processo precisa de mão de obra adequada, ou seja, especializada para desenvolver esse produto e isso também é considerado um custo que fará parte no preço do produto.

Para o desenvolvimento dessas soluções lógicas é inevitável a definição desse preço para propor ao cliente que deseja obter esse produto. Um bom começo para o cálculo desse preço é a elaboração de uma *estimativa de custo* para realizar esse trabalho e essa estimativa envolve quanto de esforço será necessário para finalizar cada atividade e o gasto total do trabalho, no entanto existem diversos fatores e variáveis que influenciam o valor de um projeto de *software* e ele não é só composto do custo de desenvolvimento mais o lucro para a empresa desenvolvedora [5]. Uma das técnicas que pode ser utilizada para fazer a estimativa de esforço seria a utilização de redes bayesianas em conjunto com análise de ponto de função para mensurar o tamanho do *software* e utilizá-lo como um dos diversos *inputs* da rede bayesiana.

Segundo Fenton [6] a expressão "*estimativa de custo*" é normalmente utilizada para definir diversas estimativas, contudo é mais utilizada como sinônimo de "*estimativa de esforço*". Garantir a precisão de uma estimativa é muito difícil principalmente na fase inicial do projeto de desenvolvimento de software, por conta das incertezas como as habilidades das pessoas envolvidas no projeto que são normalmente desconhecidas, da decisão entre as diversas tecnologias de desenvolvimento que vão ser utilizadas e das diversas variáveis intrínsecas ao processo [5]. A falta de controle e gestão adequada dessas variáveis durante a fase de planejamento e desenvolvimento de *software* podem levar a atrasos, aumento no orçamento e até no cancelamento dos projetos [7].

Os dados da pesquisa divulgado pelo Grupo Standish [8] no ano de 2015 ratifica os estudos do Autor Jones [7]. A pesquisa mostrou que 45% dos projetos foram entregues com atraso ou acima do orçamento e 19% dos projetos falharam, ou seja, foram cancelados antes da conclusão.

A maneira mais eficaz de diminuir os atrasos e cancelamento dos projetos de desenvolvimento de *software* é fazer o gerenciamento de todo o ciclo de vida do produto que será desenvolvido, utilizar abordagens que promova a qualidade do *software* [7, 8] , utilizar métricas de estimativa de esforço,

*Graduando em Análise e Desenvolvimento de Sistemas.
†Doutor em Ciências da Computacão e Professor do Curso de Análise e Desenvolvimento de Sistemas.

dentre outras [7].

Uma abordagem que pode ser utilizada para promover a qualidade é o AADSP *Adaptive Approach for Deployment of Software Process*, proposta pelo LABRASOFT - Laboratório de Desenvolvimento de *software*, que baseia-se no modelo do MPS.BR [9]. Para promover uma abordagem de qualidade o AADSP conta com um módulo para gerenciamento de testes proposto por [10] que utiliza redes Bayesianas e conta com o gerenciamento de requisitos capaz de rastreá-los durante todo o ciclo de desenvolvimento de *software*, proposto por [11].

Com bases nas informações divulgadas pelo Grupo Standish [8] e contida no trabalho de Jones [7] o presente trabalho tem como contribuição a criação de um modelo de rede Bayesianas para a estimação de esforço que utiliza como um dos seus parâmetros o tamanho do *software* medido em pontos de função, para melhorar o direcionamento do tempo gasto com as atividades de desenvolvimento de *software*, gerar artefatos contendo dados para auxiliar na estimativa de esforço de projetos futuros e uma ferramenta onde o modelo proposto foi integrado.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 Medição de software

As métricas fazem parte de um fator preponderante para a construção do *software*, pois elas são maneiras de medir o produto que está sendo desenvolvido e mensurar seu processo numa tentativa de melhorá-lo, aumentando sua qualidade e auxiliando as futuras tomadas de decisão ao longo do projeto [12]. Com a medição de *software*, torna-se possível permitir a avaliação da aquisição de ferramentas novas que auxiliem no desenvolvimento do produto fazendo com que haja uma evolução na produção, estimativa do investimento, quantidade de pessoas necessárias, tempo necessário e avaliação dos benefícios [12, 4, 13], entretanto de acordo com José [13] apesar das métricas trazerem esses benefícios, elas por si só não garantem soluções imediatas, no entanto as experiências de processos anteriores juntamente com métricas robustas podem trazer à tona o que de fato está acontecendo com o projeto por conta dos seus indicadores advindos da medição, que pode levar a uma noção do que pode ocorrer com a produção do escopo futuro. Segundo Roger [13] para ter métricas confiáveis juntamente com seus indicadores é necessário que elas advenham de:

- Dados históricos confiáveis que possam ser facilmente adquiridos através de processos seguros
- Processos devidamente segmentados de acordo com as técnicas de estimação
- Técnicas íntegras que garantam o manuseio adequadamente
- Técnicas objetivas as quais evitem a ambiguidade
- Técnicas precisas que evitem a subjetividade dos dados coletados
- Técnicas confiáveis cujo método de obtenção possa gerar resultados idênticos toda vez que os mesmos dados sejam colocados sobre testes
- Técnicas padronizadas que garantam uma unidade de medida única

A medição do *software* pode ser caracterizada em medidas diretas que são consideradas quantitativas e as indiretas que são qualitativas. As medidas diretas são realizadas em termos de atributos observáveis, como: tamanho, custo e esforço [14, 4, 12], esse tipo de métrica pode também ser definida como " uma métrica que não depende da medida de qualquer outro atributo" [15], ainda segundo a norma do padrão IEEE 1061 as medidas diretas são importantes, porque elas são presumidamente válidas e outras métricas são validadas em termo dela, as métricas diretas também podem ser referidas como sinônimo de aferimentos "fundamentais" [14].

As métricas indiretas podem ser derivadas de outras métricas, o contraste entre medição direta e indireta é que a direta pode ser considerada como uma função cujo domínio possui apenas uma variável e a indireta como uma função cujo domínio é uma n-tupla [16]. Alguns exemplos de métricas indiretas são a complexidade, confiabilidade, facilidade de manutenção, qualidade, eficiência, manutenibilidade [12, 14, 4]. De acordo com o Autor Fenton [17] as medições derivadas mais usadas na engenharia são:

- Produtividade do programador (tamanho do código em relação ao tempo de programação)
- Densidade do defeito do módulo (erros em relação ao tamanho do módulo)
- Estabilidade dos requisitos (números de requisitos inicias sobre o número total de requisitos)
- Deterioração do sistema (esforço gasto na correção de falhas sobre o total do esforço do projeto)

Segundo o Autor Pressman [12], esse tipo de medida pode ser dividido em categorias ou domínios, como:

- Orientadas a objeto
- Orientadas a tamanho
- Orientadas a caso de uso
- Orientadas a função

2.2 Métricas Funcionais ou orientadas a função

Métricas funcionais são métricas que buscam medir a funcionalidade ou utilidade do *software*, levando em consideração o panorama do usuário[1] com base no que foi solicitado e recebido de retorno, fazendo uma junção da complexidade e do tamanho do produto que será desenvolvido [18, 19]. Essa medida é capaz de fornecer um método padronizado para mensurar as diversas funções de um produto lógico. Uma das técnicas mais utilizadas é a análise de pontos de função [19, 3], essa medida se relaciona diretamente com os requisitos que o *software* deve atender [20]. A empresa percussora dessa técnica foi a IBM, através de um funcionário chamado Albrecht tendo sua publicação oficial em 1979. Mais tarde, foi criado um grupo internacional de usuários do ponto de função(IFPUG), o qual era responsável por estabelecer os

[1]Segundo a ISO/IEC 14143-1:2007 - A palavra "usuário" no contexto da medição do tamanho funcional significa qualquer interação que o *software* mensurado faça, podendo ser ela pessoas tais como administradores, usuários finais ou até mesmos outros softwares, hardwares etc.

padrões, promover seu uso e evolução [3, 18, 19]. Existem cinco padrões difundidos mundialmente para medição do tamanho funcional [3].

- IFPUG (ISO/IEC 20926)[2]
- COSMIC-FFP (ISO/IEC 19761)[3]
- NESMA (ISSO/IEC 24570)[4]
- MARK II (ISO/IEC 20968)[5]
- FISMA (ISO/IEC 29881)[6]

2.2.1 IFPUG

É o padrão mais antigo de uma organização que não visa o acúmulo de capital e endossa dois tipos de metodologia para dimensionamento de *software*. Ela é responsável pela definição do manual de práticas de contagem de pontos de função (CPM) que é reconhecido pela indústria para a análise de pontos de função (APF). Ela também fornece um fórum para troca de informações e de incentivo ao uso da métrica de análise de pontos de funções. O grupo busca sempre facilitar o intercâmbio de ideais com a finalidade de melhorar as técnicas de medição de *software* [20];

2.2.2 COSMIC

Assim como o IFPGU, o COSMIC é uma organização que não visa o acúmulo de capital [21].Ela foi fundada por Alain Abran e Charles Symons. Os seus fundadores desenvolveram uma nova forma de dimensionamento de sistemas em tempo real, logo após ficou comprovado que esse método poderia também ser utilizados em projetos de sistemas de informação, e já nos dias atuais pode ser usado para mensurar também *software* de infraestrutura e híbridos, entretanto ele não é tão usado na comunidade internacional comparado ao IFPUG [3, 21].

2.2.3 NESMA

É uma associação Holandesa que conecta pessoas e empresas envolvidas em tornar as soluções lógicas mensuráveis. Ela fornece informações valiosas sobre as métricas de *software* e participa também dos comitês das outras organizações relacionadas, o COSMIC e IFPUG. A NESMA reconhece três maneiras de análise de ponto de função (APF), que são [22]:

- APF detalhada
- APF de alto nível ou APF estimada
- APF indicativa

As APFs estimada e indicativa não requerem requisitos detalhados do usuário, ainda segundo a NESMA [22] o tamanho funcional desses métodos é próximo ao tamanho funcional determinado pelo APF detalhada, o que deixam eles adequados para serem utilizados no início do ciclo de vida do *software*, A NESMA possui seu próprio manual para contagem, entretanto ela cobra pela sua aquisição, a primeira versão do seu manual foi feita em 1990 com base no IFPUG [3].

[2]http://www.ifpug.org/
[3]https://cosmic-sizing.org/
[4]https://nesma.org/
[5]http://www.uksma.co.uk
[6]https://www.fisma.fi/

2.2.4 MARK II

Foi implementado por Charles Symons, assim como os fundadores da NESMA ele também se inspirou em Albrecht um dos fundadores do IFPUG, entretanto sua técnica só ganhou proporção no mundo da medição de *software* quando Charles publicou sua técnica na revista IEEE em 1988, a técnica tem seus manuais públicos e é mantida por uma associação do Reino Unido [3].

2.3 Contagem de ponto de função segundo o IFPUG

Para dar início ao processo de contagem do APF para a medir o tamanho funcional do *software* é necessário reunir toda a documentação disponível que descreva as funcionalidades entregues pela aplicação lógica. Se a documentação estiver incompleta, uma consulta a especialistas que estão diretamente relacionados com o projeto deve ser feita a fim de minimizar as imprecisões na documentação. Alguns documentos necessários são diagramas de classes, modelos de dados e objetos, diagramas de fluxo de dados, requisitos, etc. Após reunir toda a documentação, deve-se determinar o escopo, a fronteira da contagem e a identificação dos requisitos funcionais do usuário. Com o escopo determinado e requisitos identificados é feita a medição das funções, o cálculo do tamanho funcional e o preenchimento da documentação com os resultados obtidos [23, 3]. A figura 6, mostra a visão geral do fluxo de contagem dos pontos de função de acordo com o IFPUG.

2.3.1 Tipo de Contagem

Nesse processo é identificado o tipo de contagem de acordo com o escopo, tendo como base o objetivo que pode ser projeto de desenvolvimento, projeto de melhoria e aplicação, que é determinado de acordo com o propósito da contagem. Segundo o Autor Vazquez [3, 23] existem três tipos:

- **Projeto de desenvolvimento:** Trata-se da medida de funcionalidades de um programa que será entregue ao usuário final, tendo sua instalação pela primeira vez.
- **Projeto de melhoria:** Trata-se da medida de funcionalidade das alterações de um programa já existente, sendo elas adições, exclusões e modificações.
- **Aplicação:** Trata-se da medida de funcionalidade de um *software* já instalado, denota a baseline das funções atuais providas ao usuário.

2.3.2 Identificação da fronteira de aplicação

A definição da fronteira é feita após a identificação do tipo de contagem. A fronteira identifica a separação das partes internas "aquelas que estão sendo medidas" da parte externa do *software*. De acordo com o comitê de práticas de contagem [23] a identificação dessa fronteira tem que ser elaborada mediante o ponto de vista do usuário. A fronteira também deve ser baseada na perspectiva de negócio que envolve os dados lógicos mantidos pela aplicação.

2.3.3 Medição das Funções de Dados

O processo de medir funções de dados dentro do escopo da contagem avalia e identifica os dados que satisfazem os requisitos funcionais do usuário, ou seja, representam as funcionalidades fornecidas pelo sistema ao usuário referente ao armazenamento dos dados são divididas em:

- **Arquivo de Interface Externa (AIE):** São constituídos de informações de controle e dados lógicos que estão do lado de fora da fronteira identificável pelo usuário. Esses dados são mantidos e atualizados por outras aplicações, são apenas referenciados pela aplicação lógica medida.
- **Arquivo Lógico Interno (ALI):** São constituídos de informações de controle e dados lógicos intrínsecos a fronteira da aplicação identificável pelo usuário, esses dados são mantidos pela aplicação que está sendo contada.

Os arquivos lógicos internos e os arquivos de interface externa são classificados segundo sua complexidade que está associada ao número de tipo de dados e tipo de registro como mostra a figura 1. A definição de tipo de dado é comumente chamado de campos, esses campos que são identificados pelo usuário não podem ser repetidos, já a definição de tipo de registro está associada a um subgrupo dos dados, ou seja, são os componentes de um arquivo de interface externa ou arquivo lógico interno.

		Tipos de Registro		
Tipos de Dados		1	2 - 5	> 5
	< 20	Baixa	Baixa	Média
	20 - 50	Baixa	Média	Alta
	> 50	Média	Alta	Alta

Figura 1: Tabela de Complexidade funcional ALI e AIE. Adaptado de [23]

Depois de classificar a complexidade em baixa, média e alta de acordo com o quantitativo de tipos de dados e registro é calculado o tamanho funcional em números de pontos de função para os arquivo lógico interno e arquivo de interface externa como mostra a figura 2.

Tipos de Função	Baixa	Média	Alta
ALI	7 PF	10 PF	15 PF
AIE	5 PF	7 PF	10 PF

Figura 2: Tabela de contribuição dos pontos de funções para os ALI e AIE [3]

2.3.4 *Identificação do tipo das funções de transação*

Para seguir com a medida do tamanho funcional da aplicação lógica é necessário identificar o tipo das funções de transação do *software* que correspondem ao processamento de dado fornecida ao usuário para atender as suas necessidades. Segundo o Autor Vazquez [23, 3] são divididas em:

- **Saída Externa (SE):** Trata-se do processamento de dados que enviam informações para fora da fronteira, tendo cálculos, criações de dados derivados e alteração do comportamento do sistema.
- **Entrada Externa (EE):** Trata-se do processamento de dados ou informações advindas de fora da fronteira cuja a intenção é manter uma ALI e/ou mudar o comportamento do sistema.
- **Consulta Externa (CE):** Trata-se do envio de informações de dentro da fronteira para fora cujo a intenção é exibir os dados para o usuário.

Assim como as funções do tipo de dados, as de transações também precisam ter sua complexidade determinada de acordo com o número de arquivos referenciados e número de tipos de dados como mostra as figuras 3 e 4. Um arquivo referenciado é um registro lógico interno ou arquivo de interface externa lido ou mantido pela função de transação [23, 3];

		Arquivos Referênciados		
Tipos de Dados		< 2	2	> 2
	< 5	Baixa	Baixa	Média
	5 - 15	Baixa	Média	Alta
	> 15	Média	Alta	Alta

Figura 3: Tabela de complexidade para entradas externas adaptado de [23]

.

		Arquivos Referênciados		
Tipos de Dados		< 2	2 - 3	> 3
	< 6	Baixa	Baixa	Média
	6 - 19	Baixa	Média	Alta
	> 19	Média	Alta	Alta

Figura 4: Tabela de complexidade para as saídas externas e consultas externas adaptado de [23]

.

Depois de determinar a complexidade das saídas externas, entradas externas e consultas externas é necessário identificar a contribuição dos pontos de funções das funções do tipo transação de acordo com a figura 5.

Tipos de Função	Baixa	Média	Alta
EE	3 PF	4 PF	6 PF
SE	4 PF	5 PF	7 PF
CE	3 PF	4 PF	6 PF

Figura 5: Tabela de contribuição dos pontos de função das funções do tipo transação [3]

.

2.3.5 *Cálculo do tamanho funcional*

Depois de obter a contribuição em pontos de função das funções de transação e das funções de tipos de dados é necessário efetuar o cálculo final para os tipos de contagem, sendo cada um composto de uma fórmula própria, sendo que

o objeto e o escopo da contagem têm que ser considerados durante a utilização da fórmula.

Projeto de desenvolvimento: A fórmula utilizada para o cálculo desse tipo de projeto engloba o tamanho das funções entregues (ADD) e o tamanho das funções de conversão (CFP). O tamanho das funções entregues correspondem as funcionalidades da aplicação solicitadas pelo usuário, ou seja, representam as funções que satisfazem as necessidades do negócio, essas funções são usadas após a instalação do *software*. As funcionalidades de conversão descrevem as funções disponíveis para mudança dos dados, essas funções geralmente importam dados de um sistema que será substituído por esse que está em desenvolvimento. Depois do uso essas funções são descartadas, pois não serão mais utilizadas.

$$DFP = ADD + AFP \tag{1}$$

Projeto de melhoria: A fórmula utilizada para esse tipo de projeto é diferente do de desenvolvimento, apesar de utilizar o mesmo nome de variável (ADD), o seu significado nesse contexto é diferente, ela corresponde ao tamanho da função incluída pelo projeto de melhoria, já a variável (CFP), tem o mesmo significado do projeto de desenvolvimento, além dessas duas fazem parte da fórmula, o tamanho das funções modificadas (CHGA) e o tamanho das funções excluídas pelo projeto de melhoria (DEL). Segundo o Autor Vazquez [3] o projeto de melhoria envolve apenas manutenções adaptativas que modifiquem as estruturas das funções de tipos de dados e de transação que atendam os novos requisitos do usuário, ou seja, as manutenções preventivas e corretivas não são consideradas um projeto de melhoria segundo o manual do IFPUG, ele ainda define que todas as funções que foram adicionas, alteradas e excluídas colaboram na mesma relevância para o tamanho [23, 3].

$$EFP = ADD + CHGA + CFP + DEL \tag{2}$$

Aplicação: O cálculo do tamanho funcional de uma aplicação pode ser feito de duas maneiras, a primeira é caracterizada pelas funcionalidades de uma aplicação instalada que foram solicitadas pelo usuário, nessa fórmula as funcionalidades de conversão são excluídas, na segunda é utilizada para medição do tamanho funcional de uma aplicação depois de um projeto de melhoria, onde a variável AFPA significa o tamanho da aplicação após a melhoria, AFPB o tamanho da aplicação antes da melhoria, ADD tamanho das funções incluídas pelo projeto de melhoria, CHGA tamanho das funções alteradas pelo projeto de melhoria depois da sua implantação , CHGB tamanho das funções alteradas pelo projeto de melhoria antes do início do projeto

$$AFP = ADD \tag{3}$$

$$AFPA = (AFPB + ADD + CHGA) - (CHGB - DEL) \tag{4}$$

Os pontos de função não determinam de modo direto o custo, produtividade e esforço em um projeto de construção de *software*, entretanto em conjunto com outras variáveis, pode se tornar uma peça fundamental para a derivação da *estimativa de esforço*, produtividade e custo das aplicações lógicas [3].

2.4 Técnicas para a estimativa do *custo de Software*

Segundo os Autores Mendes e Fenton [24, 6], *estimativa de custo* é um processo para determinar a quantidade necessária de esforço, tempo e prazo de um determinado produto lógico que está associado a outros fatores secundários que podem aumentar e diminuir a produtividade impactando no programa e cronograma do *software* que está sendo desenvolvido. Determinar o custo de *software* através de uma técnica especifica não é tão fácil, por conta dos diversos fatores que não fazem parte do *software* em si, ou seja, não estão relacionados diretamente com o custo de desenvolvimento. De acordo com o Autor Boehm [25] alguns métodos de *estimativa de custo* de *software* são:

- **Opinião Especializada:** Este método envolve consultar um ou mais especialistas desde que haja um consenso entre eles
- **Modelos Algorítmicos:** Este método fornece um ou mais algoritmos que produzem uma *estimativa de custo* de *software* baseado em funções com variáveis que são consideradas os principais custos do desenvolvimento
- **Analogia:** Este método envolve o raciocínio por analogia com um ou mais projetos concluídos, a fim de relacionar seus custos para a estimativa de um novo projeto semelhante
- ***Parkinson:*** Um princípio de *Parkinson* ("O trabalho se expande para preencher o volume disponível") é usado para comparar a *estimativa de custo* aos recursos disponíveis
- ***Price-to-win:*** Nessa técnica, a *estimativa de custo* é igualada ao preço necessário para ganhar o cliente ou a um tempo estipulado para ser o primeiro do mercado a entregar o novo produto.
- ***Top-Down:*** Nessa outra técnica uma *estimativa de custo* total para o projeto deriva das propriedades globais do produto do *software*. O custo total é então dividido entre os vários componentes.
- ***Bottom-Up:*** Nesse método cada componente do *software* é estimado separadamente, e os resultados são agregados para produzir uma estimativa para o trabalho geral.

A maioria das técnicas de aferição de esforço requer como base para sua entrada uma estimativa do tamanho do produto em conjunto com os dados históricos de projetos da organização. Esse agregado de referências tem que ser de programas semelhantes ao qual deseja-se estimar para determinar quanto esforço foi empregado anteriormente [26, 27].

O número de projeto pode variar de 1 a n, entretanto quanto mais projetos similares maior o índice de acurácia, o que reforça o custo necessário gasto para o desenvolvimento de soluções lógicas, contudo se a empresa não tiver um conjunto de dados, pelo simples fato de não ter adotado práticas de coletas de métricas e elaboração de documentação e ou não ter nenhum projeto análogo ao qual se deseja estimar ainda sim é possível aferir o esforço do *software* utilizando dados de produtividade analisados por *benchmarking* [27].

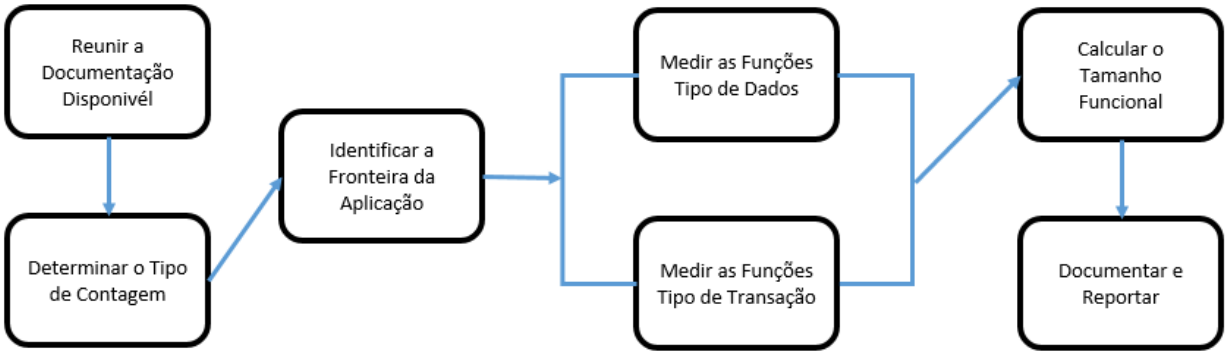


Figura 6: Visão geral do fluxo de contagem dos pontos de função de acordo com O IFPUG adaptado de [23]

.

Algumas empresas no setor de desenvolvimento utilizam abordagem maduras como o COCOMOII, Metodologia de Putnam, Metodologia de regressão, dentre outras. Essas abordagens são baseadas em estudos feitos com um número expressivo de projetos concluídos de diversas empresas, algumas delas como a .ISBSG [7] possui uma base de dados com diversos projetos pertencentes a diversas empresas da área de desenvolvimento de *software* para serem utilizados em estimativa, *Benchmarking*, gerenciamento de projetos, planejamento de infra-estrutura, gestão de *outsourcing*, conformidade de padrões e apoio orçamental [28]. Seus dados são separados em grupos diferentes tais como, tipo da linguagem de desenvolvimento, tipo do *software* desenvolvido, setor que será implantado ou se a solução lógica está sendo construída do zero ou se está sendo feita uma refatoração, etc. Essa instituição também possui uma ferramenta de consulta de dados de produtividade para auxiliar as empresas nas estimativas de taxa de entrega, esforço de trabalho, velocidade de entrega e duração do projeto [28].

2.4.1 Modelos Algorítmicos

A técnica de modelagem algorítmica é feita através de uma fórmula matemática para pressupor os custos do desenvolvimento de *software* intrínseco ou não ao tamanho, em conjunto com outras variáveis como linguagem utilizada para o desenvolvimento, tipo de *software* desenvolvido, expertise da equipe envolvida no projeto, tempo para o desenvolvimento do projeto, dentre outras [12]. Esse modelo algorítmico pode ser obtido de diversas formas e as fórmulas que compõem esse modelo podem ser ajustadas de acordo com os projetos que estão sendo desenvolvidos ou que já estão concluídos, na maioria das fórmulas seu ajuste é feito por variáveis exponenciais. Esse modelo baseia-se na derivação da seguinte fórmula geral matemática, Onde x1, x2, x3. . . xn é o vetor dos fatores de custo [29].

[7]Software *Benchmarking* Standards Group é uma organização sem fins lucrativos que tem o objetivo de promover o uso de dados do setor de TI para melhorar os processos e produtos de *software*

$$EF(custo) = f(x_1, x_2, x_3, x_4, x_5, ..., x_n) \quad (5)$$

COCOMO II Constructive Cost Model II é um modelo algorítmico que permite estimar o custo e esforço de desenvolvimento de *software*. Ele é dividido em 3 submodelos que se adequam de acordo o tipo do *software* mensurado. Esses modelos são denominados de design antecipado, composição de aplicação e pós-arquitetura. Esse modelo teve sua publicação em 1995 sendo desenvolvido por Boehm para substituir seu predecessor COCOMO que não era adequado para fazer estimativas sobre processos de desenvolvimento rápido, reengenharia, softwares orientados a objetos dentre outras [30, 31].

Modelo de Composição: Tem como finalidade estimar o esforço necessário empregado no desenvolvimento de *softwares* que utilizam ferramentas de engenharia assistida por computador, segundo o Autor Boehm [30, 31] esses projetos são bastantes diversificados e rápidos para a composição de componentes interoperáveis, sendo constituídos de objetos construtores de middleware, GUI , gerenciador de banco de dados, etc. E também são construtores de componentes específicos de domínios como pacotes de controle financeiro, médicos, industriais, etc. Esse modelo é baseado em pontos de objeto sendo eles contagem de telas, relatórios e linguagem da aplicação. São ponderados em níveis como simples, médio e difícil. A fórmula utilizada para esse modelo é mostrada abaixo.

$$EF = (NOP \cdot (1 - \%REUSO/100))/PROD \quad (6)$$

Onde o esforço **EF** é dado por:

NOP: o número de pontos de objetos do sistema mensurado.

REUSO: trata-se do percentual dos pontos de objetos reutilizados pela aplicação.

PROD: representa a produtividade do software que está em construção.

Design Antecipado: É baseado na exploração de arquiteturas de sistemas alternativos e conceitos de operações,

tendo como base os requisitos do projeto que será desenvolvido, gerando uma estimativa aproximada, utilizando pontos de função das entradas, saídas, interação com usuário e arquivos do sistema [30, 31], a sua fórmula de esforço é derivada da equação (5) sendo:

$$EF = A \cdot Tamanho^{B} \cdot M \tag{7}$$

Onde o esforço **EF** é dado por:

A: constante predefinida para 2.45 que pode ser calibrada baseada em dados Históricos.

Tamanho: variável que representa o aferimento do *software* que será desenvolvido em pontos de função posteriormente transformado em linhas de código

B, variável composta por cinco fatores que pode ser ajustado com os dados históricos

M, variável composta por 17 fatores que irá se adequar de acordo com o tipo de projeto que vai ser desenvolvido.

Pós-arquitetura esse modelo do COCOMO II é utilizado quando o processo de desenvolvimento de *software* está definido, trata-se de uma extensão do modelo de *design* antecipado tendo como base para a função de cálculo multiplicadores de esforço e fatores de escala .

$$EF = A \cdot Tamanho^{1.01+\sum_{j=1}^{5} SF_j} \cdot \prod_{i=1}^{17} EM_i \tag{8}$$

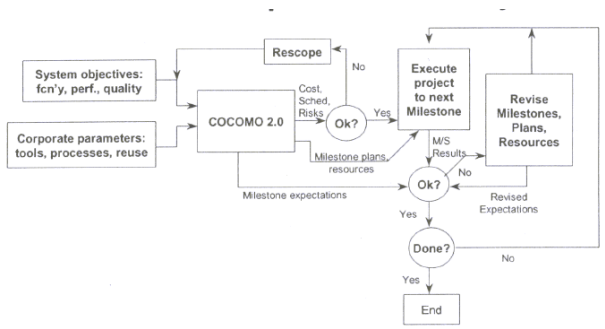


Figura 7: Fluxo do Cocomo fator de experiência II [32]

.

O fluxo da figura 7 demostra o fluxo do processo de estimativa com COCOMO II que tem como entrada os objetivos do sistema e os parâmetros corporativos. Se tudo estiver em conformidade, inicia-se a execução do projeto para o próximo marco, se não volta o processo para análise dos requerimentos(escopo) se o resultado advindo do marco for satisfatório ele encerra o ciclo com a saída dos resultados obtidos, entretanto se o marco não for satisfeito vai sofrer uma reanálise.

O *Overview* da figura 8 mostra as entradas necessárias que o COCOMO II utiliza para fazer a estimativa de custo, as saídas que é gerada após o cálculo e os dados que são utilizados para a calibração do modelo.

Modelo de Puntnam(Slim) segundo os Autores Boehm e Borade [31, 29] o modelo de Puntnam é baseado no número de projetos definidos e de como está distribuída a mão de obra. Esse modelo é baseado no ciclo de vida do *software* em uma distribuição da curva de Rayleigh para conseguir, estimar o cronograma, taxa de defeitos e esforço do projeto, a entrada para aferimento é feita com o fator de produtividade em conjunto com o tamanho do *software*, sua equação é dada como:

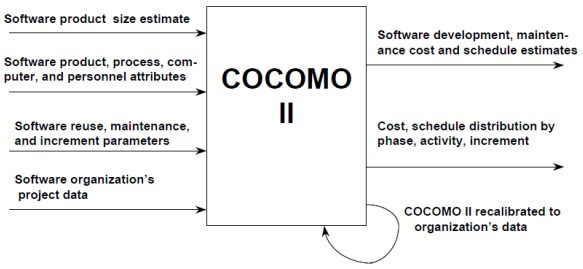


Figura 8: Overview CocomoII [32]

.

$$S = E \cdot EF_{pa}^{1/3} \cdot t_d^{4/3} \tag{9}$$

Onde **S** é o tamanho do sistema em linhas de código.

E, Variável que trata das restrições da equipe de desenvolvimento, que pode ser obtida através de dados históricos

$\mathbf{EF}_pa$, Variável que trata do esforço em pregado em pessoa/ano para o desenvolvimento da aplicação.

$\mathbf{t}_d$, variável que trata do tempo do desenvolvimento em anos

A partir da permuta das variáveis do primeiro membro com o segundo pode-se obter o esforço e tempo de desenvolvimento. Apesar de [31, 29] abordarem o modelo de Puntnam, eles não deixam claro quais são essas variáveis e como é feito o cálculo para obtê-lá a partir dos dados históricos de restrições.

2.4.2 *Opinião especializada*

Este modelo de estimativa é baseado no conhecimento dos profissionais especialistas envolvidos em projetos anteriores similares ao projeto que está para ser desenvolvido. Essa técnica é adotada por conta da falta de dados empíricos. De acordo com o Autor Boehm [31] a experiência desses profissionais podem não corresponder com o esperado, pois esse conhecimento obtido não elimina o fato dessa opinião ser equivocada e fazer com que a estimativa seja super ou sub estimada, De acordo com Borade [29] existem melhores práticas para adotar esse método como algumas que estão descritas abaixo:

- Evitar informações irrelevantes e não confiáveis
- Fazer listas de verificação de estimativa
- Combinar estimativas de especialistas diferentes
- Avaliar a incerteza da estimativa

Técnica Delphi Essa técnica baseada em opinião especializada é fundamentada como um guia para ajudar uma equipe envolvida em um determinado processo, em que os participantes a princípio emitem suas opiniões e ou avaliações individuais sobre um assunto, sem ter conhecimento das dos demais participantes, esses dados são coletados e tabulados [31], depois dessa fase inicial os especialistas envolvidos participam de uma nova rodada para opinar sobre o mesmo assunto, a partir dai todos tem conhecimento das respostas dos seus companheiros, com o fim dessa nova rodada os dados coletados são refinados e os participantes entram em consenso para chegarem ao denominador comum. Boehm

usou essa técnica para estimar os fatores de ajuste de esforço e estimar as taxas de remoção de defeitos do ciclo de vida do *software* no COQUALMO[8].

2.4.3 Modelo baseado na analogia

Esse modelo enfatiza a estimativa com base no silogismo de n-projetos de *softwares* finalizados associando ao projeto que será desenvolvido desde que o produto lógico tenham característica semelhantes. segundo o Autor Martin [33, 34] o processo de conjectura por analogia tem vantagens significativas em relação a outros métodos, pois:

- Evitam os problemas que são comumente relacionados ao processo de elicitação quanto a extração e codificação do conhecimento
- Conseguem lidar com casos de falhas, em conjunto de domínios poucos compreendidos
- Facilitam a aceitação e compreensão dos envolvidos, pois o método deriva-se da forma de raciocínio mais semelhante ao humano.

Para fazer a analogia é preciso identificar o grau de similaridade entre os projetos que é estabelecido através dos marcos dos projetos, como aplicação de domínio, método de desenvolvimento, número de interfaces, dentre outros, esses marcos podem ser flexíveis. As analogias são comumente encontradas através da medição da distância euclidiana em espaço n-dimensional, onde as dimensões correspondem aos marcos, ou seja, os requisitos dos *softwares* [34, 33].A fórmula para aferimento da similaridade é dada como:

$$SIM(C_1, C_2, P) = \frac{1}{\sqrt{\sum_{1\in} Feature - dissimilarity(C_{1j}, C_{2j})}} \quad (10)$$

Onde P é o conjunto de n recursos, C1 e C2 são casos de dissimilaridade característicos dos projetos.

Após a identificação do grau de similaridade um especialista faz o julgamento desse grau o qual o leva a fazer a estimativa de acordo com sua experiência, segundo Martin [33] essa técnica consiste de uma forma geral sistemática de uma opinião especializada, pois depois da analogia o parecer final nada mais é que um julgamento especialista.

2.4.4 Estimativa com Price to Win

O custo do produto lógico é dado tendo em vista o melhor preço para ganhar o contrato, ou seja a *estimativa de esforço* é dada de acordo com o orçamento do cliente e não por conta das funcionalidades do *software*, o que pode acarretar em diversos problemas ao longo do desenvolvimento, aumentando as chances de atraso na entrega e no aumento de horas extras [35].

2.4.5 Modelos de estimativa baseados em IA

2.4.5.1 Estimativa com Lógica Fuzzy.

De acordo com os Autores Mittal e Ahlawat [36, 37], a lógica *fuzzy* trata-se de uma metodologia, para resolver problemas que são complexos demais para serem entendidos quantitativamente. Baseados na teoria dos conjuntos *fuzzy*. O uso de conjuntos difusos é conhecido como lógica *fuzzy*, esse conjunto é definido por uma expressão de associação que relaciona cada ponto do conjunto *fuzzy* a um número real pertencente ao intervalo fechado entre o 0 e 1, que é denominado de grau de associação [36, 37]. A *estimativa de esforço* a partir da lógica *fuzzy* se dá pelo fato do tamanho do projeto em alguns casos não poderem ser estimados por completo no inicio do desenvolvimento, podendo ser tomado como um número de *fuzzy* triangular, tendo a incerteza da entrada que é o tamanho estimado, produzira também uma incerteza na sua saída, por conta disso o esforço é aferido em termos do cálculo da técnica de defuzzificação [36]. A saída única, estimativa *fuzzy* de esforço, pode ser calculada como:

$$E = \frac{W_1 \cdot (a\alpha^b) + W_2 \cdot (am^b) + W_3 \cdot (a\beta^b)}{W_1 + W_2 + W_3} \quad (11)$$

Sendo que a parcela da equação mostrada abaixo trata-se da média ponderada do valor otimista esperada pelo especialista responsável pela estimação do esforço

$$(a\alpha^b) \quad (12)$$

E as outras parcelas abaixo são as estimativas mais povoável e a pessimista respectivamente esperadas pelo especialista responsável pela estimação do esforço

$$(am^b) \quad (13)$$

$$(a\beta^b) \quad (14)$$

Onde w1,w2 e w3 são os pesos da estimativa otimista, mais provável e pessimista de acordo com o especialista responsável pela estimação do esforço.

2.4.5.2 Estimativa com Redes Neurais.

Esse modelo para aferição de esforço, utiliza-se da metodologia de aprendizado de máquina e reconhecimento de padrões para fazer a *estimativa de esforço* no desenvolvimento de aplicações lógicas, assim como a maioria dos modelos, a rede neural também pode ser ajustada, ou seja, treinada com dados históricos para produzir resultados melhores [38]. As redes neurais são desenvolvidas a partir de um *layout* de neurônios que são denominados de nós da rede, sendo dispostos em formas de camadas e de sinapses que são consideradas os pesos para a aferição, esses neurônios que compõem a rede imitam os neurônios biológicos. Após identificar os nós que serão os principais fatores para a estimativa tem que ser definido a função de estimativa ponderada entre os nós e o algoritmo de treinamento. Segundo Nassif [38] , para fazer o treinamento com precisão é necessário um conjunto grande de dados com estruturas intermediarias de qualquer complexidade. O esforço previsto utilizando esse técnica é definido através do nó de saída e essa saída é dada com base na fórmula de ativação abaixo:

$$E = f[\sum_{i=1}^{n} w_i x_i - w_0] \quad (15)$$

onde **xi** são entradas de neurônios, ou seja os atributos para a estimativa **wi** são os pesos e **f** é a função de ativação.

2.4.5.3 Estimativa com *Redes bayesianas*.

[8]Constructuve Qualityity MOdel, modelo que prevê a densidade de defeitos residuais por unidades de tamanho do *software* para calibrar o modelo COCOMO II

As *redes bayesianas* são estruturas gráficas que utilizam a teoria das probabilidades, da ciência e estatística usadas para simbolizar o conhecimento sobre um domínio incerto onde existe uma relação de causa e efeito sendo definidas por nós[9] e arcos[10], onde cada nó antecessor exerce influência sobre os nós sucessores. Sua base é caracterizada pela teoria de Thomas Bayes onde define a possibilidade de um evento ocorrer ou não, dado que outro evento tenha ocorrido, seu teorema pode ser descrito através das seguintes fórmulas:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{16}$$

Onde P(A|B) é a probabilidade de A ocorrer sendo que B tenha ocorrido, P(B|A) é a probabilidade de B ocorrer sendo que A tenha ocorrido, P(A) e P(B) são respectivamente a probabilidade do evento A e B. Como não se conhece P(B) pode-se utilizar a teoria da probabilidade de intersecção de dois eventos que é dada por:

$$P(A \cap B) = P(A|B) \cdot P(B) \tag{17}$$

Passando P(B) para o primeiro membro tem-se que.

$$P(B) = \frac{P(A \cap B)}{P(A|B)} \tag{18}$$

Pare explicar podemos usar esse exemplo hipotético da probabilidade de extrair uma carta de um baralho comum de 52 cartas e obter um Ás, sabendo que ela é uma carta de ouros. Tem-se que A é a probabilidade de se ter um Ás e B é a probabilidade de se ter um carta de ouros. Como só exite 1 ás de ouros no baralho, tem-se que a probabilidade de retirar ele é

$$P(A \cap B) = \frac{1}{52} \tag{19}$$

Como exitem 13 cartas de ouros das 52 cartas disponíveis no baralho a possibilidade de tirar uma carta de ouros é dada por:

$$P(B) = \frac{13}{52} \tag{20}$$

Logo a possibilidade de se ter um Ás de Ouros é ;

$$P(A|B) = \frac{1}{13} \tag{21}$$

Como supracitado uma rede Bayesiana é um grafo composto de nós e arcos, a figura 9 representa esses elementos básicos que formam as *redes bayesianas*, os nós podem sem chamados de:

- Pais quando tem-se um ou mais nós abaixo interligando-o pelos arcos
- Filho quando possuir a cima um ou mais nós interligando-o pelos arcos
- Folha quando o nó possuir pais e não possuir filhos

O grafo presente na rede é modelado de maneira direcionada e acíclica sendo representado pela abreviação **DAG**[11].No modelo **DAG** o significado de acíclico se dá pelo fato dos nós serem orientados de maneira que um filho não possa passar características das suas variáveis para seu pai, entretanto um pai pode passa-las para seus nós filhos.

Figura 9: Representação gráfica dos elementos de uma rede Bayesiana [23]

.

Na modelagem de rede Bayesiana cada nó tem uma tabela de probabilidade associada a ele, onde estão representadas as probabilidades condicionais dos nós filhos que estão relacionas com os valores dos seu pais, a figura 10 mostra o exemplo de uma rede Bayesiana com os nós e as tabelas de probabilidades.

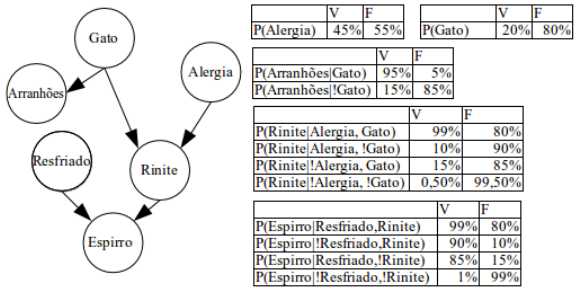


Figura 10: Representação gráfica de uma rede bayesiana com as tabelas de probabilidade dos nós [39]

.

O esforço através da rede bayesiana é definido com os valores das probabilidades dos nós. Uma característica desse modelo segundo Sérgio [39] é que ele permite a equipe usar informações dos dados da amostra a priori de maneira logicamente consistente ao fazer inferências, pois o teorema de Bayes produz uma distribuição de pós-dados, ou seja, posteriori que servem como parâmetros para o modelo, isso pode ser definido como um processo de aprendizagem, pois a distribuição posteriori é determinada pelas variações das informações a priori. Se a variância dessas informações forem menores que a variação das informações da amostragem um peso maior será atribuído as informações a priori, entretanto se a variação da amostra for menor que a variança da informação a priori, então um peso maior é atribuído a amostra, fazendo com que a estimativa posteriori fique mais próxima dos dados da amostra [39] De um modo geral as probabilidades a priori são incondicionais que estão associadas de maneira simples as informações da amostra, enquanto as posteriori são condicionais aos dados da amostra e as informações a priori. A figura 11 mostra a definição dos dados posteriori e a priori no teorema de bayes.

2.4.6 *estimativa de esforço no Scrum*

Projetos Ágeis segundo a Agile Alliance é um conjunto de praticas definidas através de um manifesto que baseia-se

[9]Os nós são as estruturas da rede que retratam os fatores do problema que se quer resolver
[10]Os arcos são estruturas que retratam a correlação de dependência probabilísticas entre os nós
[11]Directed Acyclic Graph

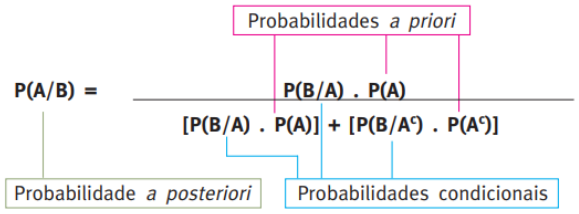


Figura 11: Relação da Probabilidade a Posteriori dado as probabilidades a priori [39]

.

em 12 princípios para auxiliar a implementação e execução de projetos com agilidade através de diferentes abordagens que podem ser selecionadas de acordo com a necessidade do projeto. As abordagens mais conhecidas segundo Rafael [40] são o SCRUM e O XP.

O *Scrum* é uma metodologia que define a gestão de projetos de *software* [40]. Os projetos que são baseados nessa metodologia são compostos de ciclos de desenvolvimento interativos, denominados de *Sprints* [40]. As *Sprints* tem uma data de inicio e de fim que podem variar de acordo com o ambiente em que esta sendo adotada. O *Scrum* define um conjunto de atividades que são as funcionalidades que serão implementadas no projeto, essas atividades são denominadas de *Product Backlog*, que são segmentas em grupos de atividades menores,recebendo o nome de *Backlog da Sprint*. Quando a *Sprint* inicia-se o *Product Owner*[12] prioriza os itens do *Product Backlog* em uma reunião chamada de *Sprint Planning Meeting*. A equipe responsável pelo desenvolvimento escolhem as atividades que serão capazes de desenvolver, essas atividades são colocadas no *Backlog da Sprint* [40].

Quando a *Sprint* inicia-se o *Scrum Team*[13] que é formado pela equipe de desenvolvimento se reúnem todos os dias para o compartilhamento de conhecimento, identificação das dificuldades e impedimentos que sugiram no dia anterior [40] Normalmente quando existe algum impedimento o *Scrum Master*[14] auxilia na resolução. Quando o ciclo de uma *Sprint* termina a equipe envolvida no projeto fazem duas reuniões a *Sprint Review Meeting* onde são apresentadas as implementações feitas pelo time, logo após é feita uma reunião de *Sprint Retrospective*, onde são abordados os pontos positivos e negativos da *sprint* [40].

A *estimativa de esforço* mais utilizada nos projetos desenvolvidos utilizando a metodologia do *Scrum* é o *Planning de Poker* [41]. A figura 12 mostra o processo de estimação do *Planning Poker* de acordo com o Autor Mendez [41].

Planning Poker: De acordo com Gandomani e Mendez [42, 41] esse método é baseado no consenso de um grupo de especialistas sendo proposto inicialmente por J.Grenning. Onde as estimativas são definidas por um jogo de cartas, onde toda equipe envolvida no processo de desenvolvimento de *software* participa, cada carta utilizada no *Plannig* tem

[12]É o responsável pela tomada de decisão no decorrer da Sprint

[13]É formado pelos responsáveis que são encarregados do desenvolvimento das atividades definidas no Product Bakclog.

[14]Responsável por facilitar e ajudar a equipe de desenvolvimento durante a Sprint

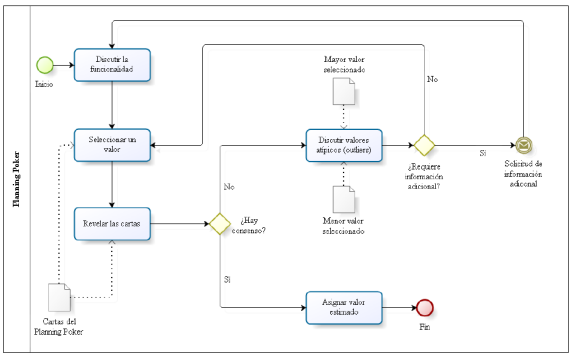


Figura 12: Processo de estimação do Planning Poker [41]

.

um peso associado que é denominado de pontos de história, esses pontos representam as funcionalidades dos requisitos dos clientes e ajudam a decidir quanto trabalho será necessário durante a *Sprint*. Cada Pessoa vai se concentrar na quantidade de histórias de usuários que conseguem entregar. Os recursos são discutidos através de perguntas direcionadas ao representante do cliente, após a discussão do recurso cada participante escolhe uma carta para a representação de sua estimativa, com isso cada um revela a carta que tem. Se houver um consenso em relação aos estimadores, o peso da carta se torna a estimativa, no entanto se isso não ocorrer são discutidos novamente os estimadores até todos chegarem a um consenso.

2.4.7 *Rede Bayesiana Proposta*

A rede proposta para fazer a *estimativa de esforço* mostrada na figura 13 foi elaborada para ser implantada na ferramenta que será descrita nas próximas seções. A rede é composta por 8 nós principais que serão os responsáveis pela aferição e aprendizado da rede, para ter uma acurácia significativa é necessários que o modelo seja carregado com dados validos de n-projetos em desenvolvimento, concluídos e ou com a opinião de um especialista, pois o mesmo foi definido com dados hipotéticos.

A estimativa inicia-se com a seleção dos requisitos que irão fazer parte da *Sprint* ou de um modulo em desenvolvimento, após a definição dos requisitos, será necessário fazer a medição dos mesmos, se não existir um módulo similar ou requisito semelhante com o tamanho aproximado definido, O cálculo ainda poderá ser feito, pois a ferramenta proposta também dispõem de um módulo para estimativa de tamanho de *software*, através da técnica de ponto de função, explicada no inicio da segunda seção. A rede parte do pressuposto que todo requisito está associado a um tamanho funcional, uma ou mais linguagem de desenvolvimento, uma equipe de desenvolvimento, que possui uma determinada experiência. Os nós da rede bayesianas são definidas como:

- **Backlog da Sprint:** Onde estão definidos os requisitos que serão estimados pela rede, esses requisitos tem tês níveis prioridade baixa, que significa que o desenvolvimento e aferição do requisito no momento não é relevante, então seu peso é pouco considerado, prioridade média que considera seu desenvolvimento considerável e prioridade alta, o que torna o requisito imprescindível.

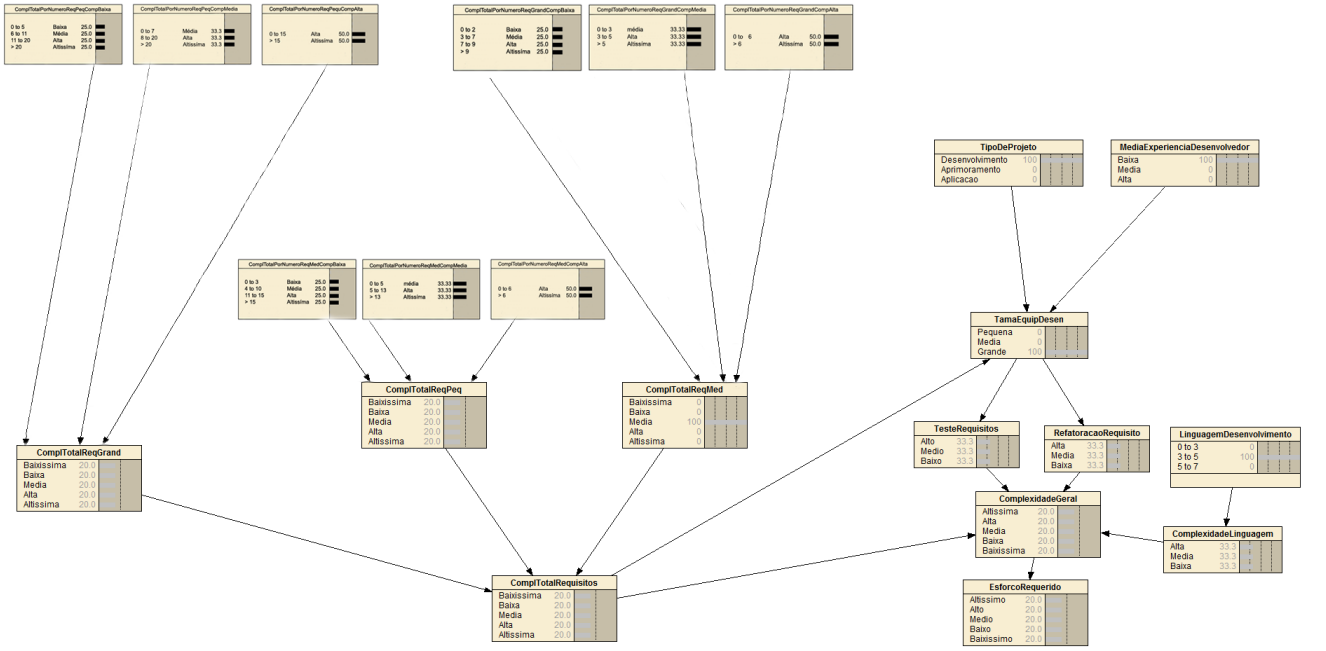


Figura 13: RepMOde Bayesiana Proposta, elaboração Própria

.

- **Tamanho da equipe:** Cada requisito sera desenvolvido por um número de desenvolvedores, sendo considerado uma equipe pequena a priori quando tiver menor que 2, equipe média quando for maior que 2 e menor que 4 a priori e equipe grande quando for maior que 5.

- **Tamanho do requisito** Cada requisito terá um tamanho associado a ele expresso em pontos de função, sendo que a priori será considerado pequeno quando for menor igual a 40 pontos de função, médio quando for maior que 40 menor que 80 e grande quando for maior que 80 pontos de função.

- **Experiencia Equipe:** Cada desenvolvedor vai ter um peso relacionado a sua experiência, sendo definido como pouca experiência se os desenvolvedores envolvidos tiverem menos de 2 anos de carreira, experiente se o desenvolvedor possuir mais de 2 anos e menos que 6 anos de carreira, muito experiente se ele tiver mais de 7 anos de carreira

- **Quantidade de linguagens** Será considerada baixa se o requisito for desenvolvido até com 2 linguagens, média se possuir três linguagens e alta se existir mais de 3.

- **Complexidade do tamanho da equipe:** Esse nó será representado pela complexidade do tamanho da equipe que vão variar de acordo com a medida normalizada entre os valores dos desvios reais e estimados.

- **Complexidade do tamanho funcional:** Esse nó será representado pela complexidade do tamanho funcional que, vão variar de acordo com a medida normalizada entre os valores dos desvios reais e estimados.

- **Complexidade da experiência do desenvolvedores:** Esse nó será representado pela complexidade da experiência dos desenvolvedores, que vão variar de acordo com a medida normalizada entre os valores dos desvios reais e estimados.

- **Complexidade da linguagem:** Esse nó será representado pelo grau de complexidade das linguagens de desenvolvimentos, que vão variar de acordo com a medida normalizada entre os valores dos desvios reais e estimados.

- **Complexidade dos Requisitos:** Esse nó será representado pela complexidade dos requisitos tendo como entrada as complexidades de cada nó supracitados, sendo que se for maior que 0,8 terá uma complexidade alta os requisitos, se for maior que 0,5 e menor que 0,8 terá uma complexidade média e se for menor igual a 0,5 terá uma complexidade baixa.

- **Esforço imprevisto:** Esse nó trata-se das horas gastas com imprevistos durante o desenvolvimento do requisito, sendo considerado baixa se for menor que uma hora, média se for maior que uma e menor que 2 e alta se for maior igual a duas horas.

- **Esforço dos testes:** Esse nó trata-se das horas gastas com os testes dos requisitos durante o desenvolvimento, sendo considerado alto se for maior igual a 2 horas, médio se for menor que 2 e maior que uma e baixo se for menor que uma hora.

- **Esforço com refatoração:** Esse nó trata-se das horas gastas com a refatoração de requisitos sendo considerados baixa se for menor igual a 3 horas, média se for maior maior que 3 e menor que 4, e alta se for maior igual a 4 horas.

- **Esforço Implementação:** É nó é representado pelo esforço gasto para o desenvolvimento do requisito sem levar em consideração as horas gastas com os testes, imprevistos e refatoração, esse nó é definido como produtividade alta se o tempo de desenvolvimento for menor que 20 horas dada uma *sprint* com 40 horas, média se for maior que vinte e menor que 30 horas e baixa se for maior igual a 30 horas.

- **Esforço da documentação:** Esse nó representa o tempo gasto com o esforço na documentação final de todos os requisitos desenvolvidos, sendo baixa se for menor igual a 5 horas, médio se for maior igual a 6 horas e menor igual a 7 e alto se for maior que 7 horas.

Os valores e pesos fornecidos aos nós das redes referenciados a cima são hipotéticos, sendo necessário fazer a calibração com dados concretos ou opinião especializada.

3. TRABALHOS CORRELATOS

Nesta presente seção serão apresentados os trabalhos correlatos na área de *estimativa de esforço*. Na literatura existe diversos trabalhos feitos cada um com sua especificidade e tecnologias similares ou diferentes pra fazer o aferimento de um projeto de *software*.

Em [43] utiliza-se da técnica de lógica *fuzzy* para fazer a *estimativa de esforço*. Para calibrar o seu modelo proposto, ele utilizou uma base de dados da NASA63. Depois da calibração, para fazer um novo aferimento o usuário teria que entrar com os parâmetros do projeto, sendo um deles o tamanho do produto em linhas de código. Apesar do modelo proposto em [43] ser calibrado com uma boa base de dados, o fato de utilizar O LOC [15] como forma de mensuração de acordo com [18] não é uma boa abordagem, pois em projetos orientados a objetos teria que fazer um conversão o que interferiria significativamente no resultado final.

Em [44] aborda a técnica de *Story Points* para fazer a *estimativa de esforço* em projetos que usam a metodologia ágil. Esse modelo de acordo com Gandomani [42] é um excelente abordagem, pois além de estimar o esforço para realização de uma determinada atividade é capaz de avaliar a complexidade e os ricos inerentes ao seu desenvolvimento, sendo possível minimizá-los com antecedência. Apesar de ser uma excelente técnica, os *Story Points* não são armazenados para uma consulta posterior.

Em [45] aborda o uso de *redes bayesianas* para a estimativa de *softwares* para a *Web*, a calibração da rede proposta é feita com uma base de 150 projetos. Segundo Emília [45] o projeto obteve o resultado esperado dentro do domínio proposto conseguindo uma acurácia significativa, entretanto o presente trabalho proposto pode ser aplicado em mais de um domínio de desenvolvimento de *software*.

Os trabalhos correlacionados de um modo geral oferecem excelente abordagem através de suas devidas técnicas. O presente trabalho traz um diferencial, pois além de fazer a estimativa, o *software* proposto pode também fazer a estimativa do tamanho com *análise de ponto de função* e a rede proposta leva em considerações alguns fatores que os outros trabalham não consideram, como o esforço gasto em testes, na refatoração e os imprevistos que podem vim a ocorrer durante a fase de desenvolvimento.

[15]Linhas de código

4. DESENVOLVIMENTO DA FERRAMENTA

4.1 Requisitos Funcionais

Nas subsecções seguintes, serão descritas algumas das principais funcionalidades do sistema desenvolvido. Devido ao tamanho do sistema só listaremos alguns requisitos do *software*.

4.1.1 Autenticação

Para entrar no sistema o usuário deverá inserir seus dados de *login* e *password*. Durante a instalação do sistema, o primeiro acesso será feito com um usuário padrão préconfigurado na base de dados, com o perfil de administrador. O usuário padrão terá acesso a todas funcionalidades do sistema. O *login* desse usuário padrão para ter acesso ao sistema deve ser "admin" e a senha "admin".

4.1.2 Sair do Sistema

Os usuários logados no sistema poderão efetuar sua saída em qualquer página que ele estiver, bastando-lhe clicar no botão de *loggof*.

4.1.3 Cadastro de Usuários

Para um novo usuário ter acesso ao sistema deverá ser efetuado o cadastro, por um gestor que tenha o perfil de administrador. Para o cadastro ser efetuado com sucesso, todas as informações do usuário necessárias para acessar o sistema deverão ser preenchidas.

4.1.4 Consulta de Usuários

Todos os usuários cadastrados do sistema poderão ser consultados pelos gestores que tenham o perfil de administrador. Na tela de consulta é possível filtrar as informações dos usuários.

4.1.5 Cadastrar Perfis de Acesso

O usuário que possui o perfil de administrador poderá cadastrar o perfil de acesso dos demais usuários. Desse modo deve-se selecionar as permissões e ações que os usuários do sistema poderão desempenhar.

4.1.6 Cadastrar Clientes

Os usuários autenticados, com permissão de acesso a esse requisito, poderão cadastrar novos clientes que solicitarem o desenvolvimento de uma aplicação lógica. Para o cadastro ser efetuado com sucesso, todas as informações do cliente deverão ser preenchidas.

4.1.7 Cadastrar Projetos

Os usuários autenticados, com permissão de acesso a esse requisito, poderão cadastrar novos projetos que serão mensurados pela ferramenta. Para o cadastro ser efetuado com sucesso todas as informações do projeto deverão ser preenchidas.

4.1.8 Consultar Projetos

Todos os projetos cadastrados no sistema deverão ser listados e exibidos na tela para os usuários autenticados que tenham permissão de acesso a esse requisito. Na tela de consulta é possível filtrar as informações dos projetos.

4.1.9 Cadastrar Linguagens de Programação

Todas linguagens de programação utilizadas pela empresa deverão ser cadastradas no sistema junto com o valor da hora, a quantidade de ponto de função por hora e o valor do ponto de função.

4.1.10 Cadastrar Desenvolvedores

Todos os desenvolvedores envolvidos no projeto que será mensurado deverão ser cadastrados no sistema. O salário e as horas de trabalho dos programadores devem ser preenchidas.

4.1.11 Avaliar experiência dos desenvolvedores

Todos os programadores devem ter uma nota referente a sua experiência de desenvolvimento em projetos anteriores. Quanto mais anos ele tiver de carreira maior será sua nota.

4.1.12 Consultar e Editar Desenvolvedores

O sistema deve fornecer uma tela para consultar os desenvolvedores da empresa onde o *software* será implantado. Nessa mesma tela deverá conter dois botões: Um para editar os dados do desenvolvedor e outro para excluir seus dados.

4.1.13 Cadastrar Requisitos

Os requisitos do projeto devem ser cadastrados no *Product Backlog*. Para o cadastro ser efetuado com sucesso todas as informações inerentes aos requisitos deverão ser preenchidas. Os requisitos cadastrados devem possuir um identificador único gerado automaticamente pelo sistema. Os identificadores não poderão ser alterados.

4.1.14 Cadastrar *Sprints*

Todas as atividades que fazem parte de um projeto devem ser divididas em *Sprints* . Antes de uma *Sprint* ser iniciada a mesma deve ser cadastrada no sistema. O tempo de duração, a data e hora de início da *Sprint* devem ser registradas, assim como a data e hora de termino. As *Sprints* cadastradas devem possuir um identificador único gerado automaticamente pelo sistema. Os identificadores não poderão ser alterados.

4.1.15 Vincular requisitos a *Sprint*

Os requisitos que forem ser implementados pelos desenvolvedores da *Sprint* vigente, devem ser vinculados ao *Backlog da Sprint*. Caso algum requisito não seja implementado na *Sprint* vigente, o sistema deve fornecer um mecanismo para desvincular o requisito da *Sprint*.

4.1.16 Vincular programadores aos requisitos do projeto da *Sprint*

Os desenvolvedores devem ser vinculados aos requisitos que estiverem desenvolvendo na *Sprint*. O tempo total gasto para o desenvolvimento do requisito deve ser registrado. O sistema deve fornecer um mecanismo para desvincular o programador, pois ele poderá ser remanejado para outro projeto.

4.1.17 Cadastrar as funções de dados

Nesse requisito será possível cadastrar as funções de dados do *software* que será mensurado. Os dados devem ser categorizados em arquivos lógicos internos e arquivos lógicos externos. O mapeamento dos arquivos devem ser identificados caso existam. As funções de dados não poderão ser excluídas, enquanto o processo de medição não for concluído.

4.1.18 Determinar a complexidade das funções de dados

Esse requisito determinará a complexidade das funções de dados cadastrados. A complexidade deve ter seu respectivo peso associado à sua categoria. As categorias devem estar dividas em baixa, média e alta. Cada complexidade deve ter um identificador único gerado pelo sistema.

4.1.19 Mensurar as funções de dados

Esse requisito possibilitará a medição das funções de dados dos projetos ou requisitos mensurados. Para mensurar será necessário que as funções de dados tenham sido cadastradas previamente no sistema em conjunto com suas complexidades.

4.1.20 Cadastrar as funções de transação

Nesse requisito será possível cadastrar as funções de transação dos projetos mensurados. Os dados devem ser categorizados em entradas externas, saídas externas e consultas externas. Os possíveis dados que serão processados e interligados as funções de transação devem ser identificados, caso eles existam.

4.1.21 Determinar a complexidade das funções de transação

Esse requisito determinará a complexidade de cada entrada externa, saída externa e consulta externa cadastradas. Sua complexidade funcional deve ser dividida em baixa, média e alta. Cada complexidade deve ter um identificador único gerado pelo sistema.

4.1.22 Medir as funções de transação

Esse requisito possibilitará a medição das funções de transação dos projetos ou requisitos mensurados. Para mensurar será necessário que as funções de transação tenham sido cadastradas previamente no sistema em conjunto com suas complexidades.

4.1.23 Calcular o tamanho funcional

Esse requisito possibilitará o cálculo do tamanho funcional do requisito ou *software* mensurado. Para realizar o cálculo as funções de dados e de transação devem estar cadastradas no sistema e as suas respectivas complexidades determinadas.

4.1.24 Calcular a estimativa de esforço

Esse requisito possibilitará o cálculo da estimativa de esforço do desenvolvimento de um ou mais requisitos ou de um *software*. Para o cálculo ser efetuado com sucesso, todos os dados necessários devem estar cadastrados no sistema. A única exceção que pode não estar cadastrada no sistema é o tamanho funcional, pois o usuário pode inserir diretamente esse dado para o cálculo, caso o usuário tenha feito o cálculo utilizando outra ferramenta.

4.2 Requisitos Não-Funcionais

4.2.1 Usabilidade

O sistema deve possuir uma interface de fácil utilização, sendo intuitiva para os usuários e administradores da ferramenta. Os ícones presentes nas telas devem ser autoexplicativos

4.2.2 Disponibilidade

O sistema deve oferecer uma alta disponibilidade em conjunto com o servidor da aplicação e do banco de dados. Todas as operações de inclusão, edição, exclusão, consulta e geração de relatórios só podem serem feitas com o sistema *online*.

4.2.3 Integridade

O sistema deve validar todas os dados fornecidos pelos usuários através de formulários, a fim de manter a integridade das informações na base dados.

4.2.4 Confidencialidade

A senha de acesso do sistema deve ser mascarada, quando os usuários digitarem. O banco de dados não deve armazenar a senha explicitamente, a mesma deve passar por uma função *Hash* equivalente a SHA-256.

5. CASOS DE USO

Alguns dos principais casos de uso do Sistema são demostrado através do diagrama de caso de uso presente na figura 24, que está presente no anexo B.

6. ARQUITETURA DO SISTEMA

A arquitetura utilizada para implementar o sistema foi o MVC demostrado na figura 25, que está presente no anexo C.

7. TECNOLOGIAS UTILIZADAS

- **JSF 2.2:***Java server Faces* (JSF) Trata-se de um *Framework* para a criação de interfaces de usuários alicerçada em componentes para soluções lógicas que utilizem a *web*.
- ***Primefaces:*** Trata-se de um *Framework* de código fonte aberto baseado em JSF para criação de páginas da *web* mais rica.
- **Servidor para web:** *Apache Tom Cat 8.*
- ***Primefaces:*** Trata-se de um *Framework* de código fonte aberto baseado em JSF para criação de páginas da *web* mais rica.
- ***Hibernate:*** Trata-se de um *Framework* para o mapeamento de objeto-relacional utilizando-se arquivos XML ou *Annotation*.
- **SGDB *Data Base Management System:*** *SQL-Server* 2014.
- **Jayes 2.0.1:** Trata-se de uma API de código fonte aberto para criação de Redes Bayesianas. Mantida pela empresa *Eclipse Foundation*.

8. PROJETO EM ESTUDO

O modelo de rede Bayesiana e a ferramenta proposta foi implementada no setor de desenvolvimento da empresa COGEL. Essa empresa desenvolve soluções *web* para o gerenciamento das atividades internas, externas e aplicativos *mobile* para o público externo. A metodologia ágil utilizada no ambiente de desenvolvimento é o *Scrum* e a *Sprint* de desenvolvimento tem duração de 30 dias úteis corridos. A ferramenta proposta foi utilizada no período de 03/06/2019 à 20/09/2019, totalizando duas *Sprints* e meia. A *Sprint* é dividida em ciclos, sendo que cada ciclo dura cinco dias, totalizando seis ciclos por *Sprint*, sendo que a ferramenta foi testada durante 15 ciclos. O ideal proposto pelo gerente de projetos em conjunto com o time *Scrum* é que os desenvolvedores entreguem funcionalidades a cada ciclo, entretanto se não for possível fazer a entrega de uma funcionalidade em um ciclo, a atividade é estendida para o próximo ciclo. Para fazer a estimativa de esforço do projeto que estava em desenvolvimento durante os 15 ciclos das *Sprints*, foi necessário realizar a aprendizagem da rede Bayesiana utilizando a precisão preditiva com bases na análise dos dados específicos de projetos anteriores desenvolvidos pela própria empresa, esses dados foram utilizados em forma de *DataSet*. Possibilitando, portanto, a coleta dos resultados do projeto que foi submetido para a estimativa de esforço. Para saber maiores detalhes sobre como é feito a formulação do procedimento de aprendizagem de redes Bayesianas e estimação dos valores dos parâmetros ausentes usando precisão preditiva, consultar [46]. A base de dados da empresa é composta de 42 projetos, desses projetos aproximadamente 73,81% foram feitos para a *Web* e os outros 26,19% para *smartphones*. Dos 42 projetos que foram desenvolvidos pela empresa somente 31 tiveram parâmetros coletados para serem utilizados como *DataSet*. Os 31 projetos que foram utilizados como parâmetro correspondem aos projetos de desenvolvimento para a plataforma *Web*, com o total de 25.373 pontos de função. A figura 14 mostra as tecnologias usadas para o desenvolvimento dos projetos da empresa, o número de pontos de função e o tamanho médio dos pontos de função por projetos.

Linguagem /Tecnologia	Qantidade de Projetos	Número de PF	Tamanho Médio Projeto PF
PHP	25	21625	865
Java(JSF)	4	3425	857
C#(.NET)	2	323	162

Figura 14: Tabela com tamanho dos projetos desenvolvidos pela empresa em PF - Elaboração Própria

.

9. DISCUSSÃO E RESULTADOS

Além da precisão preditiva, os 31 projetos utilizados para validar a rede Bayesiana proposta foram tratados com uma previsão pontual, que é calculada usando o método descrito em [47]. Este método calcula a distribuição de probabilidade conjunta para utilização em estimativas de esforço usando a distribuição de crenças [48], que permite a discretização do esforço em variáveis como é mostrado na figura 15 que representa a discretização do nó estimativa de esforço da rede Bayesiana proposta em horas.

A estimativa foi feita em um projeto *web* que estava sendo desenvolvido por seis funcionários, sendo que cinco eram contratados e um era estagiário. Os cálculos de estimativa de esforço foram feitos antes do início de cada *Sprint* e de cada ciclo. No termino de cada ciclo e de cada Sprint o esforço real empregado nas atividades eram calculadas. O número de pontos de função do projeto eram de 435 PF.

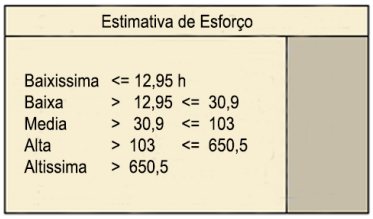


Figura 15: Nó principal da estimativa de esforço, utilizando a aproximação discreta do esforço de acordo com a distribuição de [48]

.

9.1 Resultados obtidos durante os 6 cilos da primeira Sprint

A figura 18 contida no Anexo A mostra os resultados dos seis ciclos da primeira *Sprint*.

- **1º ciclo da primeira *Sprint*:** A estimativa de esforço total calculada para o desenvolvimento das atividades foram de 301,37 horas de esforço, sendo que o esforço real para desenvolver as atividades foram de 185,147 horas.

- **2º ciclo da primeira *Sprint*:** A estimativa de esforço total calculada para o desenvolvimento das atividades foram de 266,93 horas de esforço, sendo que o esforço real para desenvolver as atividades foram de 141,735 horas.

- **3º ciclo da primeira *Sprint*:** A estimativa de esforço total calculada para o desenvolvimento das atividades foram de 308,673 horas de esforço, sendo que o esforço real para desenvolver as atividades foram de 210,102 horas.

- **4º ciclo da primeira *Sprint*:** A estimativa de esforço total calculada para o desenvolvimento das atividades foram de 145,271 horas de esforço, sendo que o esforço real para desenvolver as atividades foram de 95,204 horas.

- **5º ciclo da primeira *Sprint*:** A estimativa de esforço total calculada para o desenvolvimento das atividades foram de 145,271 horas de esforço, sendo que o esforço real para desenvolver as atividades foram de 201,42 horas.

- **6º ciclo da primeira *Sprint*:** A estimativa de esforço total calculada para o desenvolvimento das atividades foram de 260,156 horas de esforço, sendo que o esforço real para desenvolver as atividades foram de 203,248 horas.

9.2 Resultados obtidos durante os 6 cilos da segunda Sprint

A figura 19 no anexo A mostra o gráfico das estimativas de esforço, junto com o esforço real empregado para o desenvolvimento dos ciclos da segunda *Sprint*.

- **1º ciclo da segunda*Sprint*:** A estimativa de esforço total calculada para o desenvolvimento das atividades foram de 240,596 horas de esforço, sendo que o esforço real para desenvolver as atividades foram de 184,129 horas.

- **2º ciclo da segunda *Sprint*:** A a estimativa de esforço total calculada para o desenvolvimento das atividades foram de 172,487 horas de esforço, sendo que o esforço real para desenvolver as atividades foram de 135,72 horas.

- **3º ciclo da segunda *Sprint*:** A estimativa de esforço total calculada para o desenvolvimento das atividades foram de 215,124 horas de esforço, sendo que o esforço real para desenvolver as atividades foram de 177,394 horas.

- **4º ciclo da segunda *Sprint*:** A estimativa de esforço total calculada para o desenvolvimento das atividades foram de 239,19 horas de esforço, sendo que o esforço real para desenvolver as atividades foram de 203,486 horas.

- **5º ciclo da segunda *Sprint*:** A a estimativa de esforço total calculada para o desenvolvimento das atividades foram de 228,21 horas de esforço, sendo que o esforço real para desenvolver as atividades foram de 190,205 horas.

- **6º ciclo da segunda *Sprint*:** A estimativa de esforço total calculada para o desenvolvimento das atividades foram de 125,099 horas de esforço, sendo que o esforço real para desenvolver as atividades foram de 99,496 horas.

9.3 Resultados obtidos durante os 6 cilos da terceira Sprint

A figura 20 no anexo A mostra o gráfico das estimativas de esforço, junto com o esforço real empregado para o desenvolvimento dos ciclos da terceira *Sprint*.

- **1º ciclo da terceira *Sprint*:**a estimativa de esforço total calculada para o desenvolvimento das atividades foram de 254,747 horas de esforço, sendo que o esforço real para desenvolver as atividades foram de 212,569 horas.

- **2º ciclo da terceira *Sprint*:** A estimativa de esforço total calculada para o desenvolvimento das atividades foram de 220,035 horas de esforço, sendo que o esforço real para desenvolver as atividades foram de 173, 752 horas.

- **3º ciclo da terceira *Sprint*:** A estimativa de esforço total calculada para o desenvolvimento das atividades foram de 199,232 horas de esforço, sendo que o esforço real para desenvolver as atividades foram de 159,723 horas.

- **4º ciclo da terceira *Sprint*:** A estimativa de esforço total calculada para o desenvolvimento das atividades foram de 207,38 horas de esforço.

- **5º ciclo da terceira *Sprint*:** A estimativa de esforço total calculada para o desenvolvimento das atividades foram de 93,595 horas de esforço.

- **6º ciclo da terceira *Sprint*:** A estimativa de esforço total calculada para o desenvolvimento das atividades foram de 235,72 horas de esforço.

Os dados dos esforços reais empregados para o desenvolvimento das atividades dos ciclos da terceira a sexta *Sprint*, não foram calculados, pelo fato das atividades da *Sprints* não terem sido ainda desenvolvidas durante a confecção desse trabalho.

A figura 16 mostra a tabela com os esforços estimados e os esforços requeridos para o desenvolvimento das atividades durantes os ciclos das 3 *sprints*. No anexo A esta disponibilizado essa mesma tabela.

1ª Sprint	Ciclos de desenvolvimento	Estimativa de Esfoço Horas	Esforço requerido Horas
	1º Ciclo de desenv.	301,37	185,47
	2º Ciclo de desenv.	266,93	141,735
	3º Ciclo de desenv.	308,673	210,102
	4º Ciclo de desenv.	145,271	95,204
	5º Ciclo de desenv.	287,95	201,42
	6º Ciclo de desenv.	260,156	203,248

2ª Sprint	Ciclos de desenvolvimento	Estimativa de Esfoço Horas	Esforço requerido Horas
	1º Ciclo de desenv.	200,596	184,129
	2º Ciclo de desenv.	167,487	135,72
	3º Ciclo de desenv.	200,124	177,394
	4º Ciclo de desenv.	279,19	203,486
	5º Ciclo de desenv.	243,21	190,205
	6º Ciclo de desenv.	130,099	99,496

3ª Sprint	Ciclos de desenvolvimento	Estimativa de Esfoço Horas	Esforço requerido Horas
	1º Ciclo de desenv.	244,747	212,569
	2º Ciclo de desenv.	230,035	173,752
	3º Ciclo de desenv.	199,232	159,723
	4º Ciclo de desenv.	207,38	
	5º Ciclo de desenv.	93,595	
	6º Ciclo de desenv.	235,72	

Figura 16: Tabela das estimativas calculada pela ferramenta durante os ciclos de desenvolvimento

.

A figura 17 mostra a tabela com o esforço total estimado e requerido para o desenvolvimento das atividades da *Sprint*. No anexo A está disponibilizado essa mesma tabela

Sprints	Estimativa Esforço em Horas	Esforço Requerido em Horas
Sprint 1	1570,35	1037,179
Sprint 2	1220,706	990,43
Sprint 3	1210,709	
Sprint 4		

Figura 17: Tabela das estimativas Totais das Sprints calculada pela ferramenta

.

A estimativa de esforço total de cada *Sprint* e do esforço real das duas primeiras *Sprints* são mostradas na figura 20 no anexo A

10. CONCLUSÃO E TRABALHOS FUTUROS

O presente trabalho apresentou uma ferramenta para a estimação de esforço e mensuração do tamanho de *software*, que utiliza uma rede Bayesiana integrada ao sistema para fazer a estimativa de esforço e um módulo que utiliza a análise de ponto de função para mensurar o tamanho do *software*. Segundo os Autores [7] e Standish [8] a estimativa de esforço é tida como um dos fatores importantes para ajudar na melhoria do processo de desenvolvimento de *softwares* e diminuição das falhas de projetos, entretanto as estimativas devem ser baseadas em projetos anteriores para produzirem resultados aceitáveis [7].

Como trabalhos futuros, podem ser desenvolvidas as estimativas para o cálculo da duração em meses do projeto, a velocidade de entrega das atividades e taxa de entrega do projeto, haja vista que essas métricas são derivadas da estimativa de esforço. Os desenvolvimentos dessas funcionalidades ajudariam por sua vez a estimativa de qualidade do *software* em conjunto com outras variáveis inerente ao processo.

11. REFERÊNCIAS

[1] B. W. Boehm, Clark, Horowitz, Brown, Reifer, Chulani, R. Madachy, and B. Steece, *Software Cost Estimation with Cocomo II with Cdrom*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1st ed., 2000.

[2] I. Sommerville, *Engenharia de software*. Rua Nelson Francisco, SP, BR: Pearson Education Brasil, 8st ed., 2007.

[3] C. E. Vazques, G. S. Simões, and R. M. Albert, *Análise de Pontos de Função Medição, Estimativas e Gerenciamento de Projetos de Software*. Rua São Gil, Tatuapé SP, BR: Editora Érica Ltda, 10st ed., 2010.

[4] T. C. Abreu, L. S. Mota, and M. A. P. Araújo, "Métricas de software como utilizá-las no gerenciamento de projetos de software," *Engenharia de Software*, no. 21, pp. 50–55.

[5] I. Sommerville, *Engenharia de Software*. Rua Nelson Francisco, SP,BR: Pearson Education Brasil, 9st ed., 2011.

[6] N. E. Fenton and P. S. Lawrence, *Software Metrics: A Rigorous and Pratical Approach*. Boston,MA,USA: PWS Publishing Co., 2st ed., 1998.

[7] C. Jones, *The Economics of Software Quality*. Addison - wesley Professional, 1st ed., 2011.

[8] S. Group, "Chaos report 2015." https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf, 2015. Accessed JUN 17, 2019.

[9] A. S. Carlos, C. Dias, and M. Macedo, *AADSP Gerência De Requisitos*. São Paulo/SP: Editora Ixtlan, 1st ed., 2018.

[10] v. S. Santos, "T-aadsp test: Um módulo de gerenciamento de testes que utiliza redes bayesianas para priorização de casos de teste," Master's thesis, 2019.

[11] C. Dias, "T-aadsp requirements - gerenciamento de requisitos com base na abordagem aadsp," Master's thesis, 2018.

[12] R. S. Pressman, *Engenharia de Software*. NY, USA: The McGraw-Hill Companies, 7st ed., 2011.

[13] J. C. José, "Introdução a metrica de software." https://www.devmedia.com.br/introducao-a-metricas-de-software/36856. Accessed FEV 15,2019.

[14] TLCBrazil, "Métricas de software." https://www.ibm.com/developerworks/community/blogs/tlcbr/entry/metricas_de_software?lang=en. Accessed FEV 16,2019.

[15] "Ieee standard for a software quality metrics methodology," *IEEE Std 1061-1998*, p. 2, Dec 1998.

[16] C. Kaner and W. P. Bond, "Software engineering metrics: What do they measure and how do we know?," in *10TH INTERNATIONAL SOFTWARE METRICS SYMPOSIUM, METRICS 2004*, 2004.

[17] N. Fentron and J. Bieman, *Software Metrics: A Rigorous and Pratical Approach, Third Edition*. Boca Raton, FL, USA: CRC Press, Inc., 3st ed., 2014.

[18] A. Abran and P. N. Robillard, "Function points analysis: An empirical study of its measurement processes," vol. 22, (Piscataway, NJ, USA), pp. 895–910, IEEE Press, Dec. 1996.

[19] R. C. da Cunha Ferreira and C. Hazan, "Uma aplicação da análise de pontos de função no planejamento e auditoria de custos de projetos de desenvolvimento de sistemas." http://www.fattocs.com/files/pt/livro-apf/citacao/RenatoCesardaCunhaFerreira-ClaudiaHazan-2011.pdf. Accessed FEV 15, 2019.

[20] K. Lawrence, "Introdução ifpug." http://www.ifpug.org/introduction/. Accessed FEV 15,2019.

[21] "Método cosmic a segunda geração na medição de tamanho funcional." https://cosmic-sizing.org/cosmic-fsm/. Accessed FEV 15,2019.

[22] "Sobre a nesma." https://nesma.org/. Accessed FEV 15,2019.

[23] C. de Práticas de Contagem, *Manual de Práticas de Contagem de Pontos de Função*. Princeton Junction, 4.3.1 ed., 2010.

[24] A. D. S. F. Mendes, "Estimativa de custos de software, roteiro e dicas para estimar projetos," *Revista Espaço Acadêmico*, pp. 98–102, Maio 2014.

[25] B. W. Boehm, *Software Engineering Economics*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1st ed., 1983.

[26] K. Peters, "Software project estimation," *Computer Communication and Informatics*, pp. 2–3.

[27] H. Claudia and A. von Staa, *Análise e Melhoria de um Processo de Estimativas de Tamanho de Projetos de Software*. Rua Marquês de São Vicente, RJ,BR: PUC-Rio Departamento de Informática, 2005.

[28] "Sobre o isbsg." https://www.isbsg.org. Accessed FEV 15,2019.

[29] J. G. Borade, "Software project effort and cost estimation techniques," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, pp. 730–739, August 2013.

[30] B. Boehm, B. Clark, E. Horowitz, C. Westland, R. Madachy, and R. Selby, "Cost models for future software life cycle processes: Cocomo 2.0," *Annals of Software Engineering*, vol. 1, pp. 57–94, Dec 1995.

[31] B. Boehm, C. Abts, and S. Chulani, "Software development cost estimation approaches – a survey," *Annals of software Engineering*, vol. 10, pp. 177–205, Nov 2000.

[32] B. Boehm, R. Valerdi, J. Lane, and W. Brown, "The cocomo ii suite of software estimation models," *The Journal of Defense Software Engineering*, vol. 5, pp. 20–25, April 2005.

[33] M. Shepperd, C. Schofield, and B. Kitchenham, "Effort estimation using analogy," in *Proceedings of the 18th International Conference on Software Engineering*, ICSE '96, (Washington, DC, USA), pp. 170–178, IEEE Computer Society, 1996.

[34] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *IEEE Trans. Softw. Eng.*, vol. 23, pp. 736–743, nov 1997.

[35] H. Leung and Z. Fan, "Software cost estimation," in *Handbook of Software Engineering and Knowledge Engineering*, vol. 2, (Pennsylvania,AC,USA), pp. 307–234, University of Pittsburgh, May 2002.

[36] H. Mittal and P. Bhatia, "Optimization criteria for effort estimation using fuzzy technique," *CLEI electronic Journal*, vol. 10, no. 1, pp. 1–11, 2007.

[37] D. Ahlawat and R. Chawla, "software development effort estimation using fuzzy logic framework : An implementation," *Ijacte, ird India*, vol. 4, pp. 15–21, 06 2015.

[38] A. B. Nassif, M. Azzeh, L. F. Capretz, and H. Danny, "Neural network models for software development effort estimation: A comparative study," *Springer London*, vol. 28, pp. 2369–2381, 2016.

[39] S. D. Pena, "Bayes: O cara," *Ciência Hoje*, vol. 38, pp. 22–29, Julho 2006.

[40] K. S. Rubin, *Scrum Essencial: Um guia prático para o mais popular processo ágil*. Alta Books, 1st ed., 2018.

[41] E. R. Méndez, "EstimaciÓn de esfuerzo en proyectos de desarrollo de software con metodologÍas Ágiles," Master's thesis, 2018.

[42] T. J. Gandomani, K. T. Wei, and A. K. Binhamid, "A case study research on software cost estimation using experts' estimates, wideband delphi, and planning poker technique," *International Journal of Software Engineering and Its Applications*, vol. 8, no. 11, pp. 173–182, 2014.

[43] R. Chawla, "Software development effort estimation using fuzzy logic framework- an implementation," *International Journal On Advanced Computer Theory*, vol. 4, pp. 2319–2526, 2015.

[44] E. Coelho, "Effort estimation in agile software development using story points," *International Journal of Applied Information System*, vol. 3, Auust 2012.

[45] E. Mendes, "Predicting web development effort using a bayesian network." https://www.researchgate.net/publication/228795514_Predicting_Web_Development_Effort_Using_a_Bayesian_Network, 04 2007. Accessed FEV 15,2019.

[46] L. Anderson, "Redes bayesianas: Uma introduÇÃo aplicada a credit scoring," Master's thesis, 2010.

[47] P. C. Pendharkar, G. H. Subramanian, and J. A. Rodger, "A probabilistic model for predicting software development effort," *IEEE Transactions on Software*

Engineering, vol. 31, pp. 615–624, July 2005.

[48] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1st ed., 2014.

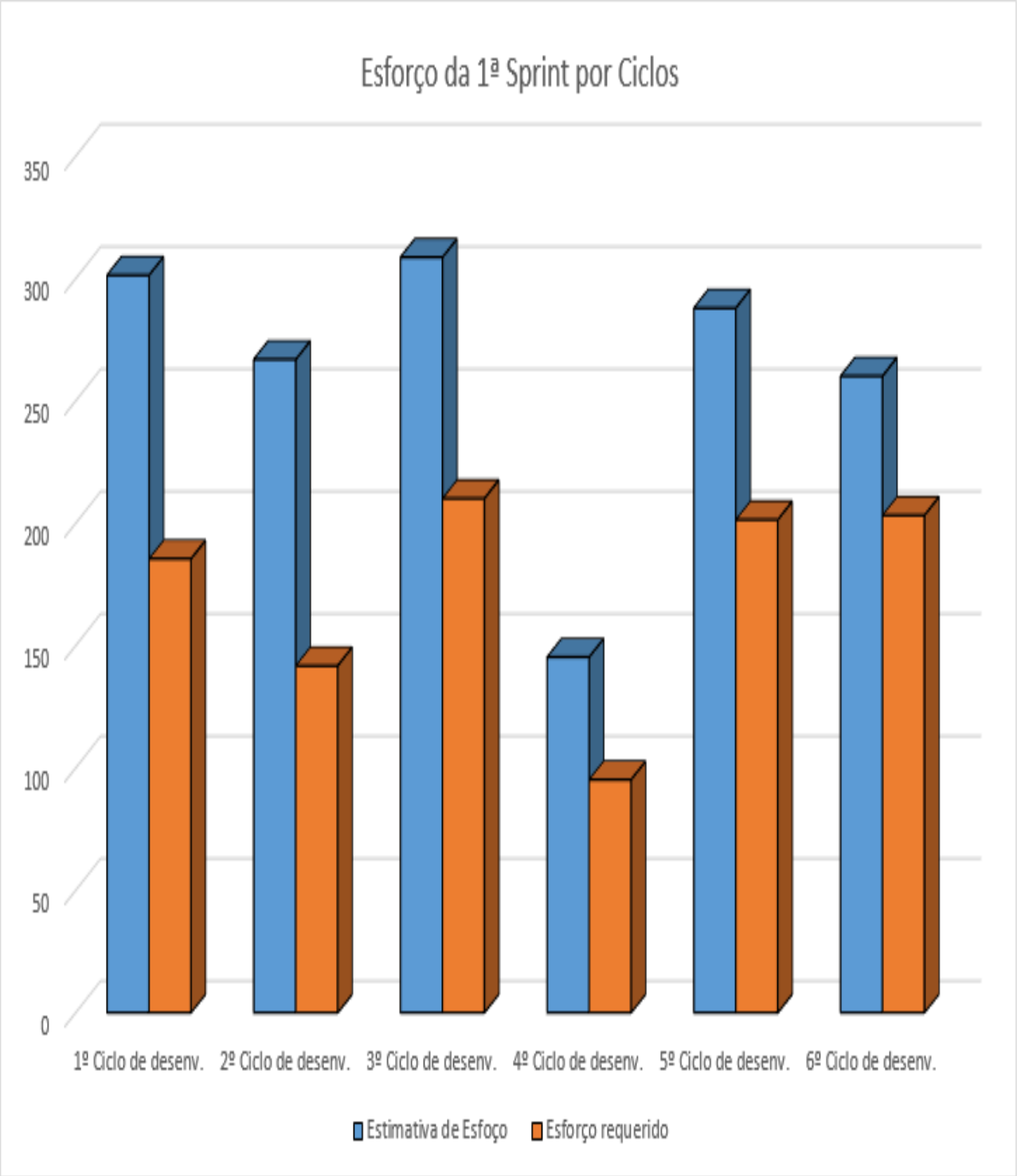


Figura 18: Gráfico com a comparação do esforço estimado, com o esforço real calculado da 1ª Sprint

.

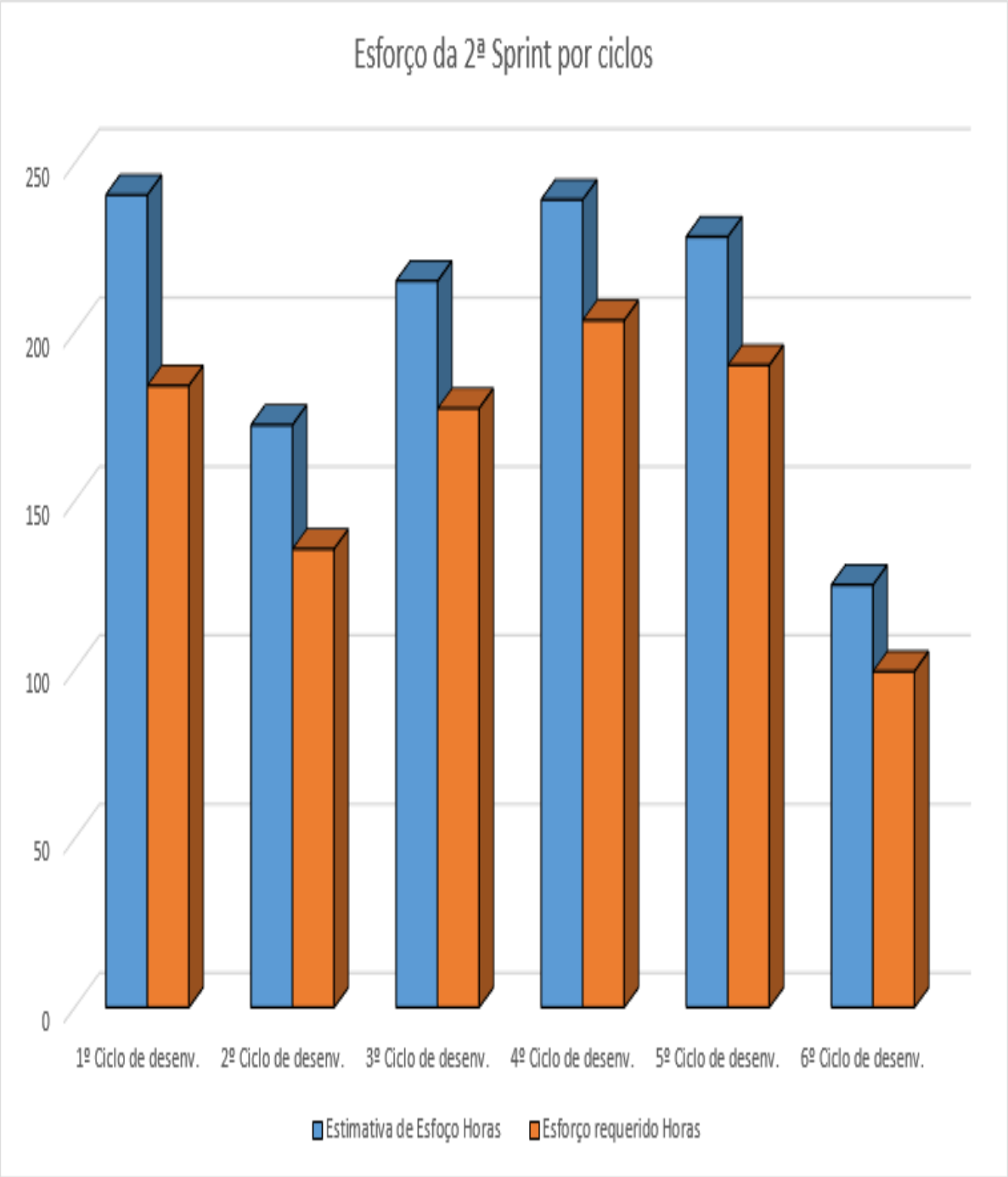


Figura 19: Gráfico com a comparação do esforço estimado, com o esforço real calculado 2ª Sprint

.

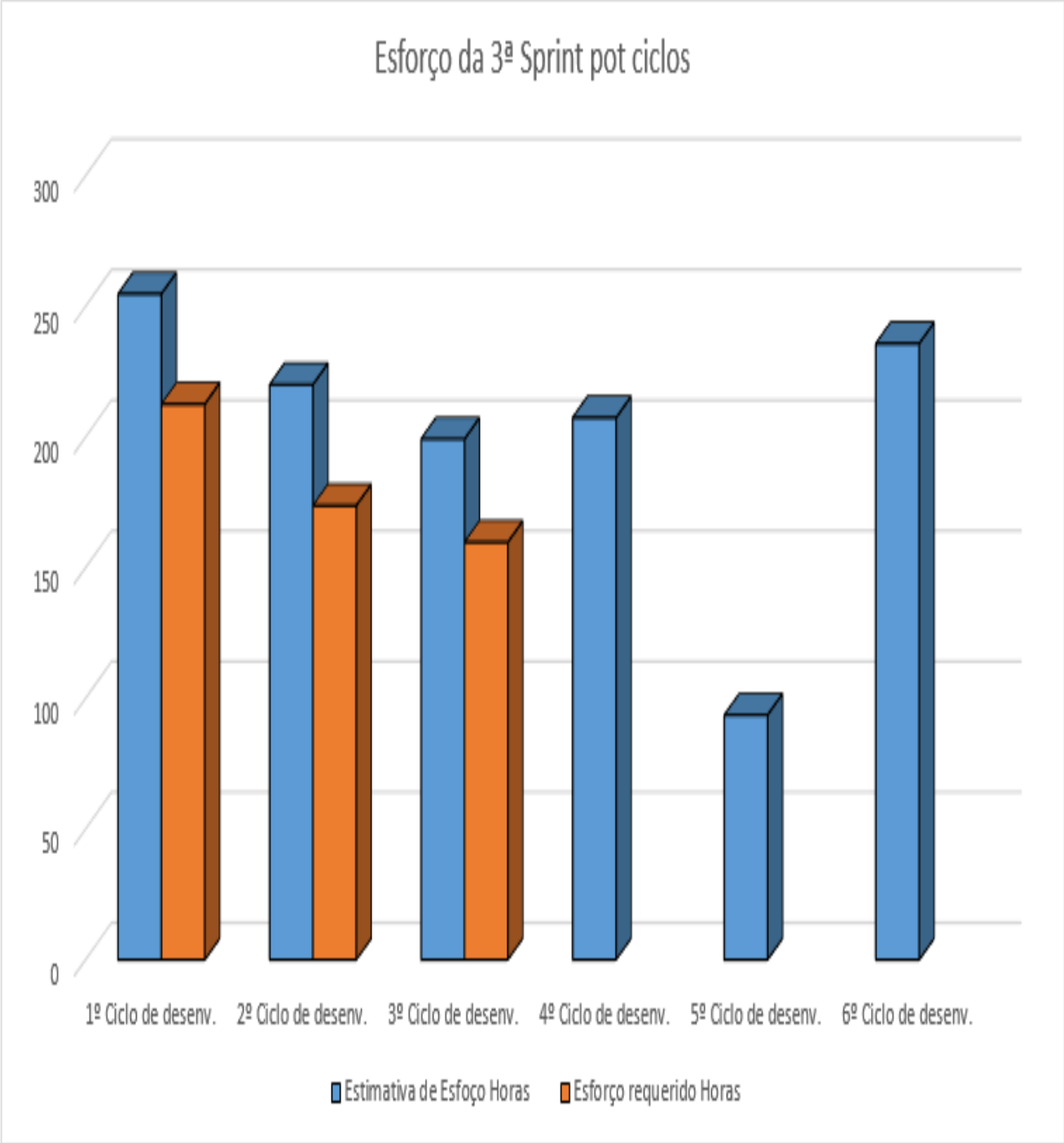


Figura 20: Gráfico com a comparação do esforço estimado, com o esforço real calculado 3ª Sprint

.

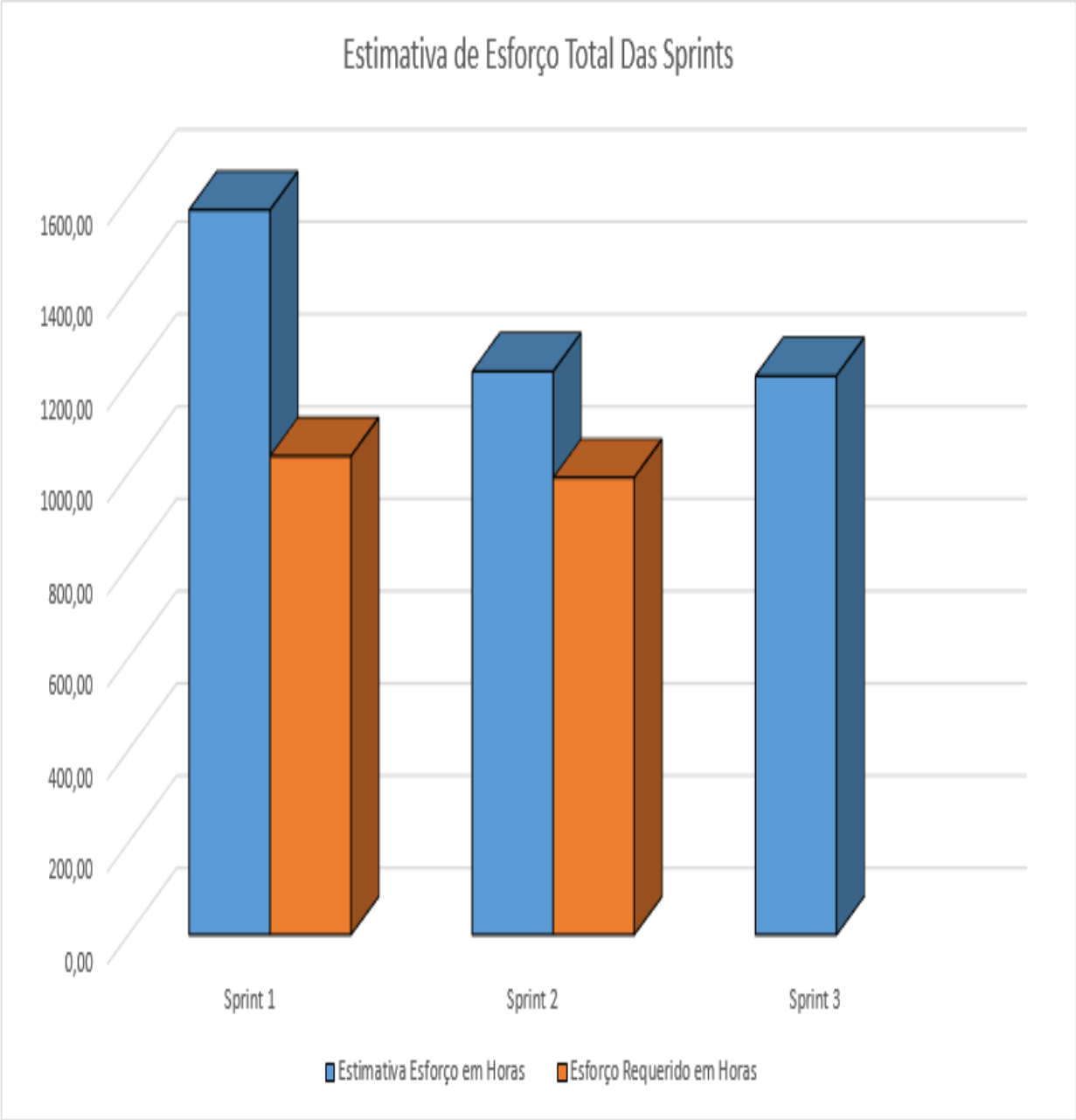


Figura 21: Gráfico com a comparação do esforço estimado, com o esforço real calculado 3ª Sprint

.

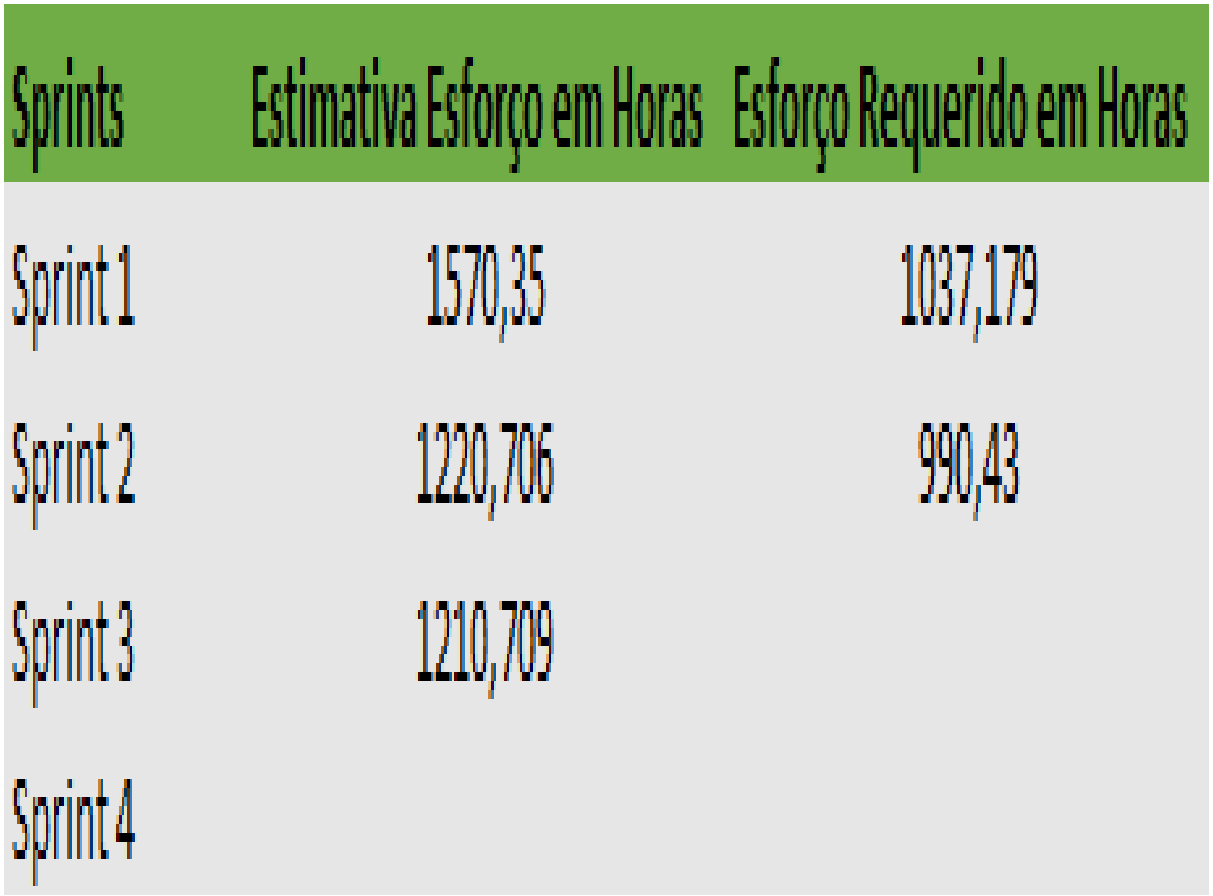
1ª Sprint	Ciclos de desenvolvimento	Estimativa de Esfoço Horas	Esforço requerido Horas
	1º Ciclo de desenv.	301,37	185,47
	2º Ciclo de desenv.	266,93	141,735
	3º Ciclo de desenv.	308,673	210,102
	4º Ciclo de desenv.	145,271	95,204
	5º Ciclo de desenv.	287,95	201,42
	6º Ciclo de desenv.	260,156	203,248

2ª Sprint	Ciclos de desenvolvimento	Estimativa de Esfoço Horas	Esforço requerido Horas
	1º Ciclo de desenv.	200,596	184,129
	2º Ciclo de desenv.	167,487	135,72
	3º Ciclo de desenv.	200,124	177,394
	4º Ciclo de desenv.	279,19	203,486
	5º Ciclo de desenv.	243,21	190,205
	6º Ciclo de desenv.	130,099	99,496

3ª Sprint	Ciclos de desenvolvimento	Estimativa de Esfoço Horas	Esforço requerido Horas
	1º Ciclo de desenv.	244,747	212,569
	2º Ciclo de desenv.	230,035	173,752
	3º Ciclo de desenv.	199,232	159,723
	4º Ciclo de desenv.	207,38	
	5º Ciclo de desenv.	93,595	
	6º Ciclo de desenv.	235,72	

Figura 22: Tabela com as estimativas dos ciclos das três Sprints calculada pela ferramenta

.



Sprints	Estimativa Esforço em Horas	Esforço Requerido em Horas
Sprint 1	1570,35	1037,179
Sprint 2	1220,706	990,43
Sprint 3	1210,709	
Sprint 4		

Figura 23: Tabela com as estimativas totais das Sprints calculada pela ferramenta

.

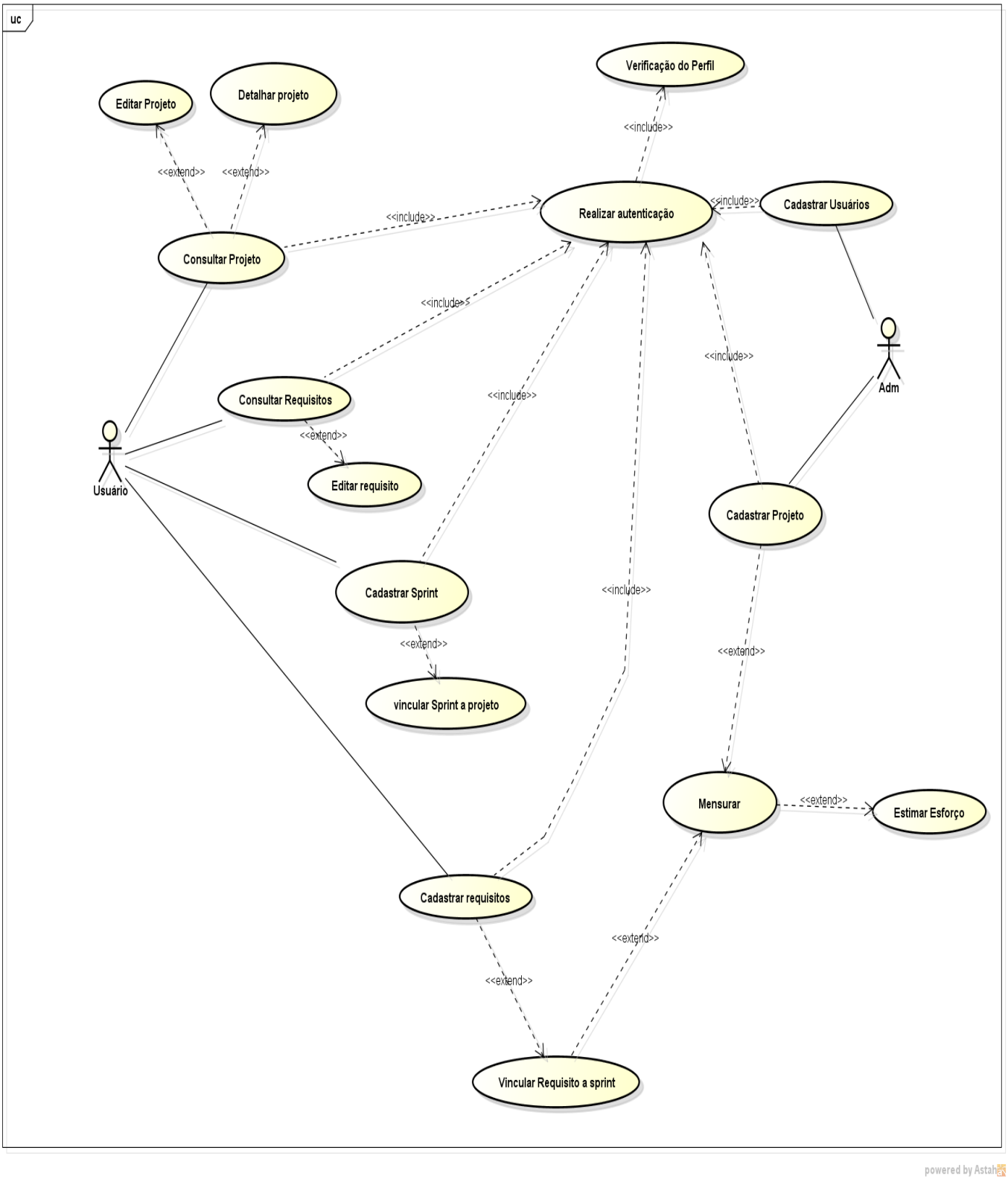


Figura 24: Diagrama de caso de uso

.

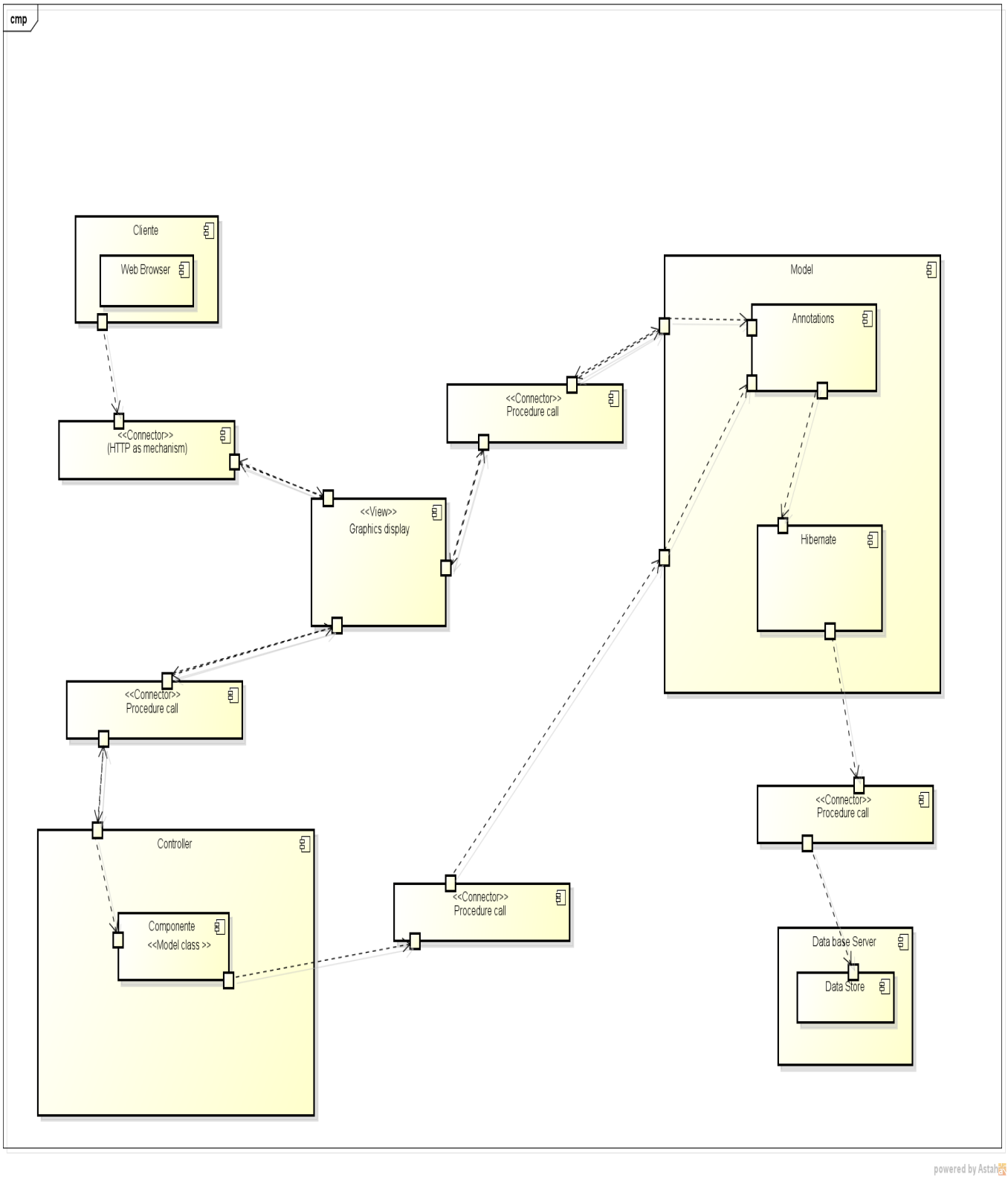


Figura 25: Diagrama da Modelagem Arquitetural

.