**UNIVERSIDADE SAGRADO CORAÇÃO**

**MAICON ANDRE VIEIRA CARVALHO**

# SOFTWARE WEB PARA ESTIMATIVAS DE PROJETOS BASEADAS EM ANÁLISE DE PONTOS DE FUNÇÃO

BAURU
2014

**MAICON ANDRE VIEIRA CARVALHO**

# SOFTWARE WEB PARA ESTIMATIVAS DE PROJETOS BASEADAS EM ANÁLISE DE PONTOS DE FUNÇÃO

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas como parte dos requisitos para obtenção do título de bacharel em Ciência da Computação, sob a orientação do Prof. Dr. Elvio Gilberto da Silva

BAURU
2014

Carvalho, Maicon André Vieira.

C3314s

Software web para estimativas de projetos baseadas em análise de pontos de função / Maicon André Vieira Carvalho. -- 2014.

71 f. : il.

Orientador: Prof. Dr. Elvio Gilberto da Silva.

Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Universidade do Sagrado Coração – Bauru – SP.

1. Métricas de Software. 2. Pontos por função. 3. Qualidade de Software. 4. Software Web. I. Silva, Elvio Gilberto da. II. Título.

**MAICON ANDRE VIEIRA CARVALHO**

# SOFTWARE WEB PARA ESTIMATIVAS DE PROJETOS BASEADAS EM ANÁLISE DE PONTOS DE FUNÇÃO

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, sob a orientação da Prof. Dr. Elvio Gilberto da Silva.

Banca examinadora:

_____

Prof. Dr. Elvio Gilberto da Silva
Universidade do Sagrado Coração

_____

Prof. Ms. Patrick Pedreira Silva
Universidade do Sagrado Coração

_____

Prof. Ms. Alex Setolin Beirigo
Universidade do Sagrado Coração

Bauru, 05 de Dezembro de 2014.

## AGRADECIMENTOS

Primeiramente a Deus, por me iluminar e abençoar minha trajetória, e principalmente nos momentos difíceis.

Agradeço à minha família por acreditarem em mim e por todo o apoio que me deram desde o início.

E aos meus professores que no princípio foi de suma importância para a realização desse trabalho, principalmente ao meu orientador, Prof. Dr. Elvio Gilberto da Silva, pela paciência e motivação incondicional que me proporcionou para a realização deste trabalho.

A todos que de alguma forma ajudaram, agradeço por acreditarem no meu potencial, e nas minhas ideias.

“O sucesso é uma consequência e não um objetivo.” (Gustave Flaubert).

## LISTA DE ILUSTRAÇÕES

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| APF: | Análise de Ponto de Função. |
| AIE: | Arquivos de Interface Externa. |
| ALI: | Arquivos Lógicos Internos. |
| AR: | Arquivo Referenciado. |
| BFPUG: | Brazilian Function Point Users Group. |
| CE: | Consulta Externa. |
| CPM: | Counting Practices Manual. |
| DER: | Diagrama Entidade-Relacionamento. |
| EE: | Entrada Externa. |
| EI: | External Inputs. |
| EIF: | External Interface Files. |
| EO: | External Outputs. |
| EQ: | External Inquiries. |
| FP: | Function Point. |
| HTML: | Hypertext Markup Language. |
| IBM: | International Business Machines. |
| IDE: | Integrated Development Environment. |
| IFPUG: | International Function Point Users Group. |
| ILF: | Internal Logical Files. |
| LOC: | Lines of Code. |
| PHP: | Personal Home Page |
| SE: | Saída Externa. |
| SGBD: | Sistema de Gerenciamento do Banco de Dados |
| SQL: | Structured Query Language, |
| TD: | Tipos de Dados. |
| TR: | Tipos de Registros. |
| UML: | Unified Modeling Language. |
| URL: | Uniform Resource Locator. |
| VFA: | Valor do Fator de Ajuste. |

| | |
|---|---|
| XML: | Entensible Markup Language. |
| WWW: | World Wide Web. |

## RESUMO

O projeto consiste na criação de um protótipo de um software web em PHP para métricas e estimativas em projeto de sistema baseada em Análise de Pontos de Função, que calcule o tamanho de um projeto em Pontos de Função, e apresente as estimativas necessárias que o engenheiro de software precisa: estimativa de tamanho, esforço, prazo e custo. Este protótipo utiliza banco de dados MYSQL, e apresenta uma interface simples e que possibilite os usuários alcancem seu objetivo mais rápido, poupando-se de projeções de cálculos manuais. Também armazena estimativas de projetos passados, para que a partir destes, tenha-se um indicador para estimar futuros softwares, tornando-se a estimativa cada vez mais exata.



**Palavras-chave**: Pontos de Função. Estimativas. Métricas. Software Web.

## ABSTRACT

The project is to create a prototype of a web software in PHP for metrics and estimates in system design based on Function Point Analysis, which calculates the size of a project in Function Points, and to submit the estimates that the engineer software needs: size estimation, effort, time and cost. This prototype uses MySQL database, and provides a simple interface that enables users to reach their goal faster, saving up projections of manual calculations. It also stores estimates from past projects, so that from these, has an indicator for estimating future software becoming increasingly accurate estimate.



**Keywords:** Function Points. Estimates. Metrics. Software Web.

**SUMÁRIO**

# 1 INTRODUÇÃO

O uso de sistemas de informação é essencial para todas as pessoas e empresas nos dias de hoje. Por isso, o planejamento e a gerencia do software é de grande importância para desenvolver um sistema com qualidade e com menos falhas.

O gerenciamento do software é um fator para distinção entre o desenvolvimento profissional e a programação amadora. É necessário o gerenciamento do projeto porque os sistemas estão sempre sujeitos ás restrições de orçamento e prazos. Por isso o gerente do projeto tem a função de estimar o custo e o prazo da entrega do sistema que contribua para as metas da empresa. (SOMMERVILLE, 2003).

Um sistema bem planejado e gerenciado tem mais qualidade, e por isso atrai mais clientes. Para Koscianski e Soares (2007), a qualidade de um produto tem o propósito de satisfazer o cliente. Por isso a qualidade não é uma entidade abstrata, e sim um objetivo concreto, e por isso que cliente e desenvolvedores tem que sempre procurar a qualidade no seu produto.

Questões como preço e tempo de entrega do produto são duas variáveis que tem grande interferência na qualidade de um software, por isso o planejamento, estimativa de esforço de desenvolvimento e a métrica de software são fundamentais em um sistema.

Para Pressman (1995) medir um sistema tem o objetivo de indicar a qualidade do produto, avaliar a produtividade dos programadores, estimar o preço e o tempo de entrega, justificar os pedidos por novas ferramentas ou treinamento adicional.

Ao avaliar os recursos e o custo no desenvolvimento de um sistema é possível se estimar a quantidade de esforço e a quantidade e experiência da equipe alocada no projeto. Também pode ser monitorado o tamanho e estabilidade do projeto. (VAZQUEZ et al., 2013).

Existem várias maneiras de se medir um software, de acordo com Marques (2011), as métricas de software podem ser classificadas em medidas diretas e medidas indiretas. As medidas diretas são aquelas que representam atributos observáveis, tais como custo, esforço, número de linhas de código, tempo de execução e número de

defeitos. Já as medidas indiretas são aquelas relacionadas com a funcionalidade, qualidade, complexidade e facilidade de manutenção do sistema.

A análise de ponto de função (APF) é uma métrica indireta que mede um software através das funcionalidades do sistema. Ela mede as funções visíveis do sistema e não as internas, como por exemplo, o número de linhas de código.

A APF é independente da linguagem de programação usada, tornando-se ideal para aplicações que usam linguagens convencionais e não procedimentais, e se baseia em dados que são mais conhecidos no começo do projeto, tornando essa técnica mais atraente no momento da estimativa. (PRESSMAN, 1995).

De acordo com Vazquez et al. (2013), o escopo de ponto de função define quais funções serão incluídas na contagem, se ele abrangerá parte do sistema ou sistema inteiro. A APF pode medir todas as funcionalidades do sistema, apenas as funcionalidades usadas pelo cliente, ou ainda, apenas algumas funções específicas. A APF mede o número de entradas externas (EE), saídas externas (SE), consultas externas (CE), arquivos lógicos internos (ALI) e arquivos de interface externas (AIE).

Por isso, o cálculo do ponto de função exige um software de interface simples e que possibilite os usuários alcançarem seu objetivo mais rápido.

Benyon (2011) diz que para um sistema tenha um alto grau de usabilidade, ele tem que ter as seguintes características:

- Apresentar uma menor quantidade de esforço para acessar determinado resultado;
- Mostra informações adequadas e organizadas de forma apropriada;
- Tem que ser seguro de operar no contexto que será usado;
- Terá um alto grau de utilidade no sentido de que fará as coisas que os usuários querem que sejam feitas.

Além disso, o sistema irá guardar informações de projetos passados para estimar o esforço de uma equipe no desenvolvimento de software futuros.

# 2 OBJETIVOS

## 2.1 OBJETIVO GERAL

Desenvolver um protótipo de um software web para métricas e estimativas em projeto de sistema, baseado em Análise de Pontos de Função, visando facilidade e confiabilidade, para que o engenheiro de software possa apenas executá-lo e gerenciá-lo, não necessitando de projeções manuais.

## 2.2 OBJETIVOS ESPECÍFICOS

- Desenvolver uma pesquisa bibliográfica sobre Métricas de Software;
- Aprofundar-se no conhecimento específico de Análise de Pontos de Função;
- Planejar o software e fazer a sua modelagem;
- Desenvolver um sistema web com uma interface clara e de fácil utilização utilizando a linguagem PHP;
- Programar todas as variáveis necessárias que a análise de pontos de função utiliza, a fim de deixar o cálculo e a estimativa com maior exatidão e confiabilidade.
- Criar um software que permita armazenar o histórico de estimativas de projetos para utilizar-se com um indicador nos próximos trabalhos.
- Testar o software com base no desenvolvimento do próprio projeto.

# 3 SOFTWARE

Os softwares são um conjunto de instruções que quando executados fornecem características, funções que manipulam informações para se produzir um resultado. (PRESSMAN, 1995).

Pressman (2011) complementa que software é um produto que profissionais desenvolvem e dão suporte em longo prazo. Abrange programas executáveis em computadores ou em outras arquiteturas eletrônicas.

Um sistema consiste em conjuntos de programas, bancos de dados, arquivos de configuração, manuais que descrevem a estrutura do projeto. Para Staa (1983), softwares são compostos por documentação, dados, códigos e procedimentos, e tem como objetivo instruir máquinas e pessoas na realização de conjunto de tarefas de processamento de dados. Sistemas são ferramentas que tem como objetivo transformar dados em resultados confiáveis e oportunos.

Muitas pessoas associam o termo software aos programas de computador, porém, esta é uma visão muito restritiva, já que software não é apenas um programa, mas também toda documentação, e configurações necessárias para que o programa opere de forma correta. (SOMMERVILLE, 2003).

Atualmente, sistemas de informação abrange diversas áreas em nosso cotidiano e é praticamente impossível uma empresa e indústria existir sem o uso da tecnologia. É essencial que haja nas empresas um investimento na área da tecnologia e em software de qualidade.

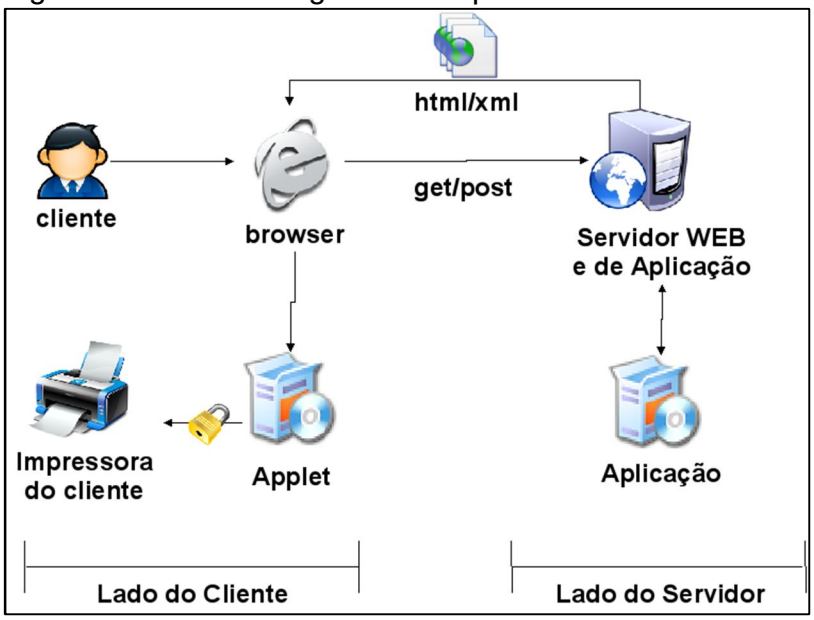
## 3.1 SOFTWARE WEB

No início da World Wide Web (WWW) os ‘websites’ consistiam em um conjunto de arquivos e hipertexto que apresentavam informações de texto, imagens e gráficos bem limitados. Com passar do tempo, o Hypertext Markup Language (HTML) foi evoluindo com o uso de ferramentas e tecnologia de desenvolvimento, como o Entensible Markup Language (XML), Java, Personal Home Page (PHP) e outras, que permitiram a interação do computador do cliente com o servidor. As aplicações web se

tornaram ferramentas sofisticadas que se integram com bancos de dados, e outros sistemas de grande porte. (PRESSMAN, 2009).

Em um sistema online, conforme ilustra a Figura 1, o usuário digita uma Uniform Resource Locator (URL), ou clica em um botão, ou link na página em um browser qualquer (Internet Explorer, Chrome, Firefox ou outros), e o navegador faz uma solicitação ao servidor. A solicitação pode ser GET ou POST. O servidor recebe a solicitação da página, e gera uma resposta, que é enviada ao navegador que processa o HTML e apresenta ao cliente.

Figura 1 - Uma visão geral da arquitetura cliente servidor




Fonte: Nobrega Filho (2009).

De acordo com Deitel, P. e Deitel, H. (2008), software web provê vários benefícios como redução de demandas no departamento interno, aumenta a acessibilidade do sistema fora do escritório, e facilita a manutenção do software em grande escala. Em vez de ser instalado na máquina local, o sistema é instalado no servidor Web e é acessado pelos clientes como um serviço de internet, e as atualizações são diretas no servidor e mantêm a mesma atualização em toda a organização.

Para Pressman (2009), aplicativos web são softwares de computador no sentido de que são uma coleção de instruções executáveis, e ao mesmo tempo fornecem informações e funcionalidade para o usuário final.

Por fim, sistemas na web são ferramentas complexas e inovadoras, que não precisam de instalação nem de arquivos de configuração, e não necessitam de espaço em disco. Além disso, reduzem o custo da infraestrutura, pois é necessário somente internet para acessar o software, e o aplicativo fica disponível em qualquer hora e local, desde que o usuário tenha conexão com a internet.

# 4 ENGENHARIA DE SOFTWARE

A Engenharia de Software é uma área que tem como objetivo o desenvolvimento de um sistema sem falhas, entregue no prazo, dentro do orçamento previsto, e que atenda às necessidades do cliente. Para atingir esses objetivos são necessárias técnicas apropriadas ao analisar o projeto, desenvolvimento e manutenção do sistema. (SCHACH, 2009).

De acordo com Sommerville (2003), engenharia de software é uma disciplina, cujo objetivo é o desenvolvimento de sistema que contenha uma boa relação de custo e benefício.

O autor ainda detalha que:

> A engenharia de software é uma disciplina de engenharia relacionada com todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até sua manutenção, depois que este entrar em operação. (SOMMERVILLE, 2003, p. 5).

O uso de ferramentas e técnicas de engenharia na análise do sistema é de grande importância para que tarefas, dados, pessoas e tecnologia estejam alinhadas para criação de um aplicativo de qualidade.

Para Pressman (1995), engenharia de software é um conjunto de etapas que envolvem métodos, ferramentas e procedimentos, conhecido como paradigmas da engenharia de software. Esses paradigmas são escolhidos de acordo com a natureza do projeto.

De forma detalhada os paradigmas da engenharia de software são:

- Métodos: envolvem várias tarefas para se construir um sistema que incluem: planejamento e estimativa, análise de requisito do sistema, projeto da estrutura de dados, arquitetura de programa e algoritmo de processamento, codificação, teste e manutenção.
- Ferramentas: são os instrumentos ou sistema utilizado para realizar uma tarefa da melhor maneira.
- Procedimentos: são a combinação de ferramentas e métodos. Os procedimentos definem a sequência em que os métodos serão aplicados,

os produtos, os controles que ajudam a qualidade, e a coordenação de mudanças, e possibilita um processo de desenvolvimento claro, eficiente, visando garantir ao desenvolvedor e seus clientes, a produção de um software de qualidade.

## 5 QUALIDADE DE SOFTWARE

De uma forma geral, um sistema apresenta boa qualidade quando produz resultados úteis e confiáveis, de uso claro e fácil, corrigível, modificável, e ainda quando opera em máquinas e ambientes reais, e foi desenvolvido de forma econômica e dentro do prazo e funciona com economia de recursos. Portanto, qualidade de software é um conceito muito mais amplo do que software correto e bem documentado, e requer o uso de técnicas e métodos específicos para ser atingido. (STAA, 1983).

Atualmente, qualidade em um software é associada aos processos do projeto, desenvolvimento e manutenção do sistema, cujo principal objetivo é satisfazer os requisitos e expectativas do cliente.

Para Pressman (1995, p. 724), qualidade é definida como:

> Conformidade com requisitos funcionais, e de desempenho explicitamente declarados, normas de desenvolvimento explicitamente documentadas e características implícitas, que são esperadas em todo software profissionalmente desenvolvidos.

Ainda de acordo com Pressman (1995), qualidade de software é uma combinação de fatores complexos que variam de acordo com diferentes aplicações e clientes que a solicitam.

Atualmente, os sistemas buscam melhorar a qualidade e, consequentemente, reduzir os custos da implementação do software. Dessa forma, o conceito de qualidade de software é algo de difícil definição, pois está diretamente relacionado à satisfação do cliente em relação ao custo, prazo de entrega e produto desenvolvido.

## 6 PLANEJAMENTO E ESTIMATIVA DE SOFTWARE

Para obter um sistema com qualidade, entrega na data prevista e com custos baixos, é importante um bom planejamento. Identificar os processos, gerenciar prazos e custo, antecipar problemas e buscar soluções fazem parte de um planejamento de software adequado.

Um bom planejamento de atividades deve ser bem detalhado, devendo incluir duração e sequencia das atividades referentes aos requisitos do sistema. É importante que se saiba quanto tempo, e o custo que cada atividade ira consumir, ainda que seja de forma estimada. (TONSIG, 2008).

A estimativa é parte importante do planejamento de um sistema, ela fornece dados que permitem prever o número de pessoas na equipe de desenvolvimento, onde também é possível planejar o orçamento e cronograma do projeto.

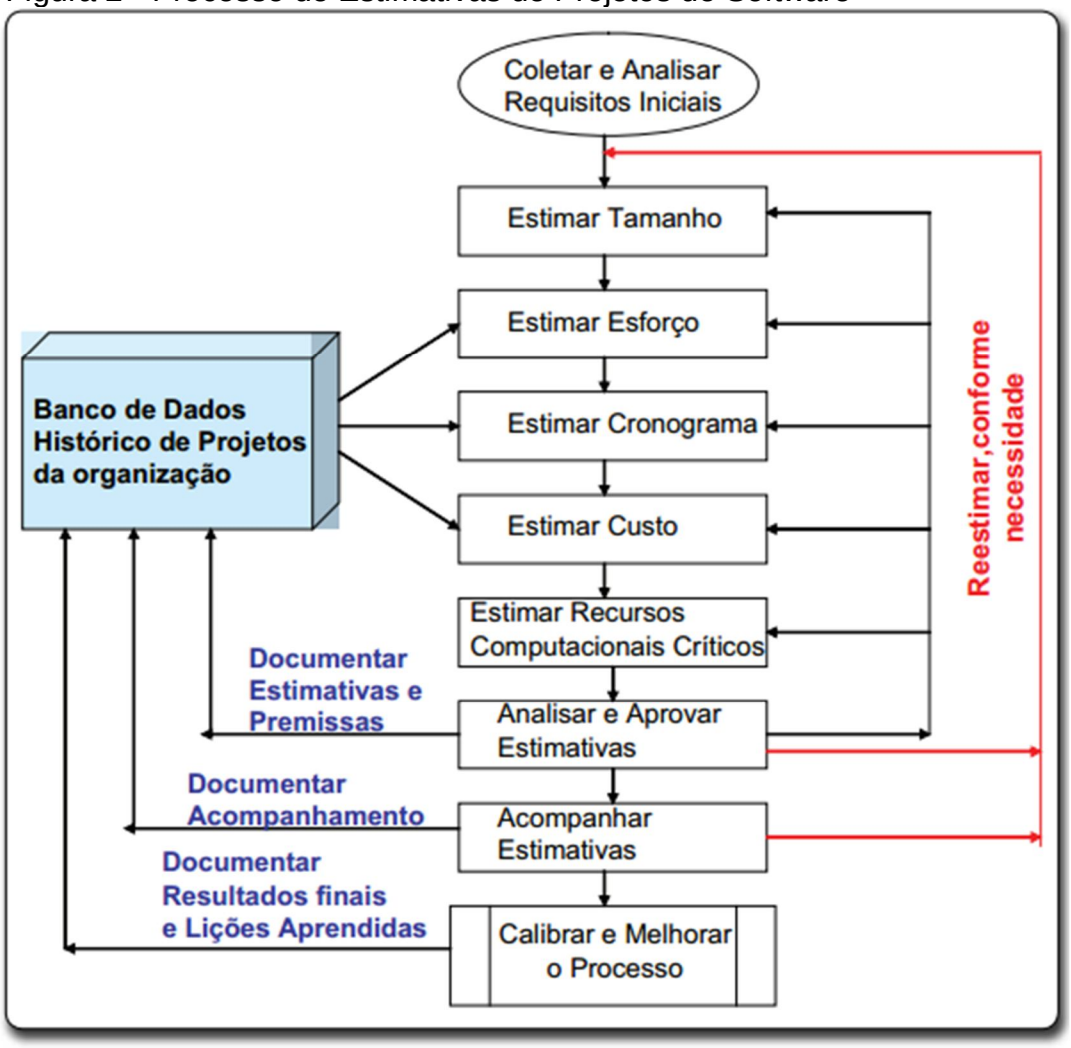
Para Pressman (1995), as estimativas de custo e de esforço de software não são uma ciência exata, pois, existem várias variáveis humanas, técnicas, ambientais e políticas que podem afetar o resultado final de uma estimativa.

Ainda de acordo com o autor anteriormente, para conseguir estimativas confiáveis é necessário seguir uma série de opções:

- Usar técnicas de decomposição relativamente simples para gerar estimativas.
- Desenvolver um modelo empírico para o custo e esforço de desenvolvimento de software.
- Adquirir uma ou mais ferramentas para automatizar o processo de estimativa.
- Atrasar as estimativas até um ponto tardio do desenvolvimento do projeto.

Esta última opção não é adotada na prática, pois conforme ilustra a Figura 2, a estimativa de custo e de esforço deve ser realizada no início do projeto junto com o planejamento. De acordo com Vazquez et al. (2013), o processo de estimava deve-se iniciar a partir da análise de requisito do projeto, e para ser completo e eficiente deve levar em consideração os dados de projetos passados, os recursos disponíveis, os custo e fatores de riscos que cercam o desenvolvimento do sistema.

Figura 2 - Processo de Estimativas de Projetos de Software




Fonte: Vazquez et al. (2013).

Estimativas são realizadas com base em métricas de software. As métricas têm como principal função reunir dados de desempenho do software, e analisar os históricos dos projetos anteriores para utilizar essas informações nas previsões de projetos futuros.

# 7 MÉTRICAS

## 7.1 MEDIDAS E MÉTRICAS DE SOFTWARE

Segundo Fenton (1991 citado por PRESSMAN, 2011), medição é o processo pelo qual números e símbolos são atribuídos a entidades no mundo real para defini-los de acordo com regras claramente estabelecidas. Em várias áreas da ciência e da tecnologia é possível medir atributos consideráveis incomensuráveis, e é claro que essas medidas não são tão refinadas como muitas feitas nas ciências físicas, mas “medir o incomensurável”, é uma ferramenta para melhorar a compreensão de entidades particulares e de extrema importância na engenharia de software.

Embora métricas de sistemas sejam imperfeitas, ainda assim podem proporcionar uma maneira de avaliar a qualidade em um conjunto de regras claramente definidas. Elas também proporcionam uma visão objetiva do projeto antes de seu desenvolvimento, isso permite descobrir e corrigir problemas antes que se tornem problemas catastróficos. (PRESSMAN, 2011).

Sommerville (2011) destaca que métrica de software é uma característica do sistema, da documentação ou de um processo de desenvolvimento que pode ser medido. As métricas de software podem ser de controle associado aos processos de gerenciamento do sistema, ou preditivas para fazer previsões, ou seja, prevendo características do software.

As métricas podem ser aplicadas ao longo de processo de software para medir o esforço em homens-mês; a métrica de custo que é fundamental e que deve ser acompanhada durante todo o projeto; a rotatividade de mão de obra, uma vez que o índice de turn-over seja alto, prejudica a equipe, tendo perda de tempo com integração do novo colaborador. (SCHACH, 2009).

De acordo com Pressman (1995), a medição e métricas ajudam a entender o processo técnico usado para desenvolver um produto, como também o próprio produto. Um processo é medido num esforço de melhorá-lo para aumentar sua qualidade.

O autor ainda cita a importância de coletar dados para uma medição:

> Quando é coletado um único ponto de dado (por exemplo, o número de erros descobertos em um componente de software), foi estabelecida uma medida. A medição ocorre como resultado da coleção de um ou mais pontos de dados (por exemplo, um conjunto de revisões de componente e testes de unidade é investigado para coletar medidas do número de erros para cada um). Uma métrica de software relaciona as medidas individuais de alguma maneira (por exemplo, o número médio de erros encontrados por revisão ou o número médio de erros encontrados por teste de unidade). Um engenheiro de software coleta medidas e desenvolve métricas para obter indicadores. (PRESSMAN, 2011, p. 539).

É de grande importância para empresa investir tempo e dinheiro em métricas de software, pois auxilia no gerenciamento do sistema. De acordo com Yourdon (1995), o argumento fundamental para a realização de métrica de software é aprender mais sobre o processo de desenvolvimento do sistema e como melhorá-la. Além disso, as métricas são necessárias para validar e ajustar modelos de produtividade, e investir em ferramentas, técnicas metodologias, e outras ferramentas que não podem ser justificados sem boas métricas.

## 7.2 PRINCÍPIOS DE MEDIÇÃO

Ao se realizar métricas de software é importante se entender os princípios da medição, de acordo com Roche (1994 citado por PRESSMAN, 2011) podem ser caracterizados por:

- Formulação: Criar ou utilizar medidas e métricas de software apropriadas para representar o software considerado.
- Coleção: O mecanismo para acumular e armazenar dados necessários para criar as métricas.
- Análise: Computar as métricas e aplicar as ferramentas matemáticas;
- Interpretação: Avaliar as métricas que resultam em informações sobre a qualidade da representação.
- Feedback: Transmitir os resultados originados da interpretação de métricas de produto para a equipe de trabalho.

O autor supracitado sugere que sempre se possível à coleta e análise de dados deveram ser automatizadas. Além disso, deverão ser aplicadas técnicas estáticas para

estabelecer relação entre atributos internos do produto e características de qualidade; e diretrizes e recomendações interpretáveis para cada métrica.

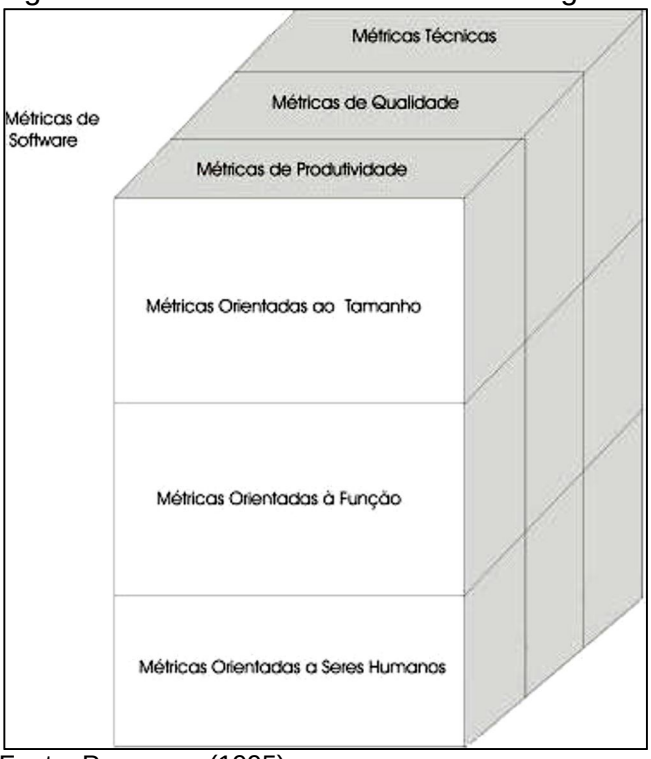
## 7.3 DIVISÕES DE MÉTRICAS EM CATEGORIAS

De acordo com Vazquez et al. (2013), o nível de abrangência da métrica de software é muito ampla e difícil de se acompanhar, por isso, a criação de categoria que agrupem os conjuntos de métricas é a solução para diminuir a complexidade envolvida, e permitir maior foco em assuntos mais importantes. Nessas categorias são incluídas informações sobre quantidade de esforço planejado e consumido nas atividades, quantidade e experiência da equipe alocada no projeto, assim como avaliar a adequação do esforço previsto, e analisar a situação do esforço consumido, sendo essenciais para avaliar a produtividade.

As medidas do mundo físico podem ser divididas em diretas e indiretas, da mesma forma ocorre em medições de sistemas. As medidas diretas do produto incluem linhas de código (LOC) produzidas, velocidade de execução, tamanho de memória e defeito registrados ao longo de certo espaço de tempo. Já as indiretas incluem funcionalidade, qualidade, complexidade, eficiência, confiabilidade e facilidade de manutenção. (PRESSMAN, 1995).

O autor ainda enfatiza que as métricas podem ser dividas em mais categorias como mostra a Figura 3. Essas divisões podem ser feitas em métricas de produtividade, que se concentram na saída do processo de engenharia de software; métricas de qualidade, as quais oferecem uma indicação do sistema conforme as exigências implícitas e explicitas do cliente; e métricas técnicas, que se focam nas características do software e não no processo do qual o sistema foi desenvolvido. E há outra divisão que se restringe em métricas orientadas ao tamanho, que são usadas para compilar as medições diretas da saída e da qualidade da engenharia de software. As métricas orientadas para a função oferecem medidas indiretas, e Métricas orientadas às pessoas que compilam informações sobre a maneira segundo a qual as pessoas desenvolvem sistema e as percepções humanas sobre a efetividade das ferramentas e métodos.

Figura 3 - Divisão das métricas em categorias.




Fonte: Pressman (1995).

Independentemente da categoria utilizada para implementar um programa de métricas, a coleta de dados para a medição é apenas um dos requisitos envolvidos. A combinação dessas medidas coletadas é o que gera visibilidade quanto a algum aspecto ou conceito relevante ao desenvolvimento e manutenção de sistemas. Esses indicadores, baseadas em métricas diferentes permitem o monitoramento de vários processos envolvido na construção do software. (VAZQUEZ et al., 2013).

## 7.4 MÉTRICAS DE PRODUTO

Métricas do produto são aquelas obtidas a partir de características do mesmo, ou de artefatos em qualquer estágio de desenvolvimento, como o código fonte ou uma especificação de requisitos. (PFLEEGER, 2004).

Para Pressman (2011), métricas de produto estão relacionadas com a qualidade e confiabilidade do software. Fenton (1994 citado por PRESSMAN, 2011, p. 534), complementa:

> Apesar das conexões intuitivas entre a estrutura interna dos artefatos [métricas de produto] e seus atributos externos de produto e processo, tem havido na realidade poucas tentativas científicas de estabelecer relações específicas. Há várias razões para isso, a mais citada é a impraticabilidade de executar experimentos relevantes.

As métricas de produto são feitas para previsão e usadas para medir atributos internos de um sistema. O tamanho e a complexibilidade são exemplos de métricas de produto. (SOMMERVILLE, 2003).

Ainda de acordo com o autor, as métricas de produto podem ser divididas em duas classes:

- Métricas dinâmicas: são coletadas por meio de medições efetuadas de um programa em execução, durante testes ou após o sistema estar em uso. Estes tipos de métricas ajudam na avaliação de eficiência e a confiabilidade do sistema.
- Métricas estáticas: são coletadas por meio de medições feitas de representações do sistema, como o projeto e a documentação. Estes tipos de métricas ajudam na avaliação de complexidade e a facilidade de manutenção do sistema;

As métricas de produto podem definir as características de qualidade desejáveis para um software.

## 7.5 MÉTRICAS DE QUALIDADE

Para Pressman (2011) existem muitas medidas de qualidade de software, mas a correção, manutenibilidade, integridade e usabilidade fornecem indicadores úteis para a equipe de projeto. Gilb (1988 citado por PRESSMAN, 2011) detalha cada uma delas:

- Correção: Um programa deve operar corretamente. A correção é o grau com o qual o software executa sua função. A medida mais comum é o número de defeitos por linhas de código, em que um defeito é definido como uma ocorrência de falta de conformidade com os requisitos. Para fins de avaliação de qualidade, os defeitos são contados durante um período de tempo;

- Manutenibilidade: É a facilidade com que um programa pode ser corrigido caso um problema for encontrado ou melhorando caso o cliente deseje alteração de um requisito. Não existe nenhuma forma de se medir a facilidade de manutenção diretamente, ou seja, devemos usar medidas indiretas.
- Integridade: Esse atributo mede a capacidade que um software tem de resistir a ataques (acidentais ou intencionais) à sua segurança. Para se medir a integridade, dois atributos devem ser definidos: ameaças e segurança. Ameaça é a probabilidade de que um ataque de um tipo específico ocorre em um determinado tempo. Segurança é a probabilidade de que o ataque de um tipo específico será repelido.
- Usabilidade: Se um programa for difícil de usar, seu destino pode ser um fracasso, mesmo ele tendo funções importantes. A usabilidade é uma tentativa de quantificar a facilidade do uso do software.

Pressman (1995) complementa que o uso de métricas de qualidade como componente principal da garantia estatística da qualidade do produto final. Ou seja, adicionando informações como defeitos descobertos e suas causas a essas métricas é possível oferecer um mecanismo para planejar ações de prevenção de processos que sejam foco de defeitos no sistema.

## 7.6 MÉTRICAS ORIENTADAS À FUNÇÃO

As métricas de software orientadas à função são medidas indiretas de um sistema e do processo ao qual é desenvolvido. A métrica orientada à função não conta linhas de código, e sim a “funcionalidade” ou ”utilidade” do programa. Ela foi proposta pela primeira vez por Albrecht (1979), que sugeriu uma abordagem à medição da produtividade chamada método do ponto por função. Esses pontos por função (FPs) são baseados em medidas de informações e complexidade do software. (PRESSMAN, 2011, grifo do autor).

Dekkers (2003 citado por HAZAN, 2008) complementa que a métrica de tamanho funcional de projetos de software, considera somente as funcionalidades

implementadas, sob o ponto de vista do usuário. Tamanho funcional é definido como “tamanho de software derivado pela quantificação dos requisitos funcionais do usuário”.

Schneider (2001) métrica orientada à função avalia o tamanho de um software. Tal métrica concentra-se na funcionalidade do software, e é calculada usando uma fórmula baseada em informações e complexidade do software.

## 8 ANÁLISES DE PONTOS DE FUNÇÃO

A técnica de APF surgiu na International Business Machines (IBM), início da década de 70, como alternativa às métricas baseadas em linhas de código. Na época, o encarregado de estudar a produtividade de software desenvolvido pela IBM foi Allan Albrecht, e o que o motivou foi a uma busca por uma medida que fosse independente da linguagem de programação utilizada, foi pelo fato de que aqueles projetos tinham sido desenvolvidos em linguagens de programação distintas, tornando-o inviável uma análise de produtividade utilizando métricas por linhas de código do sistema. (VAZQUEZ et al., 2013).

Após a apresentação da técnica a comunidade, final da década de 1970, e dos sucessivos trabalhos efetuados por Capers Jones, destacando sua importância, houve um alto crescimento em usuários utilizando APF, culminando em 1986, com a fundação da International Function Point Users Group (IFPUG), onde dois anos depois foi publicado um Manual de Práticas de Contagem (CPM – Counting Practices Manual), com o objetivo de padronização da técnica. (VAZQUEZ et al., 2013).

O Grupo IFPUG (2010) é uma organização sem fins lucrativos, e sua missão é ser reconhecida como líder na promoção e incentivo à gestão eficaz das atividades de desenvolvimento e manutenção de software, através do uso de APF e outras técnicas de medição de sistema. IFPUG subscreve APF como sua metodologia padrão para o dimensionamento de software, e também proporciona um fórum para a troca de informações que promove e incentiva o uso de sistemas e métricas de processo. Membros IFPUG podem se beneficiar de uma variedade de serviços, tais como: conferência anual, seminários e oficinas educacionais, certificação profissional, comissões de trabalho e grupos de tarefas.

No Brasil, existe a BFPUG (Brazilian Function Point Users Group) (2010), que é um grupo com o objetivo de estimular e divulgar a utilização de métricas no desenvolvimento de sistemas, em particular a APF. Destina-se aos profissionais interessados em aprender, praticar e divulgar o uso de métricas e de APF. O BFPUG é a representação brasileira oficial do IFPUG.

A IFPUG (2010 citado VAZQUEZ et al., 2013) define como objetivos do APF:

- Medir a funcionalidade que o usuário solicita e recebe.
- Medir o desenvolvimento e manutenção do software de forma independente da tecnologia utilizada para seu desenvolvimento.

Para Pressman (2011), a métrica da APF pode ser usada efetivamente como meio para medir a funcionalidade fornecida por um sistema. Através de dados históricos, a APF pode ser empregada para estimar o custo e o esforço necessário para projetar, codificar e testar o software. Além, de prever os números de erros que serão encontrados durante a fase de teste e prever os componentes e o número de linhas projetadas de código fonte no sistema projetado.

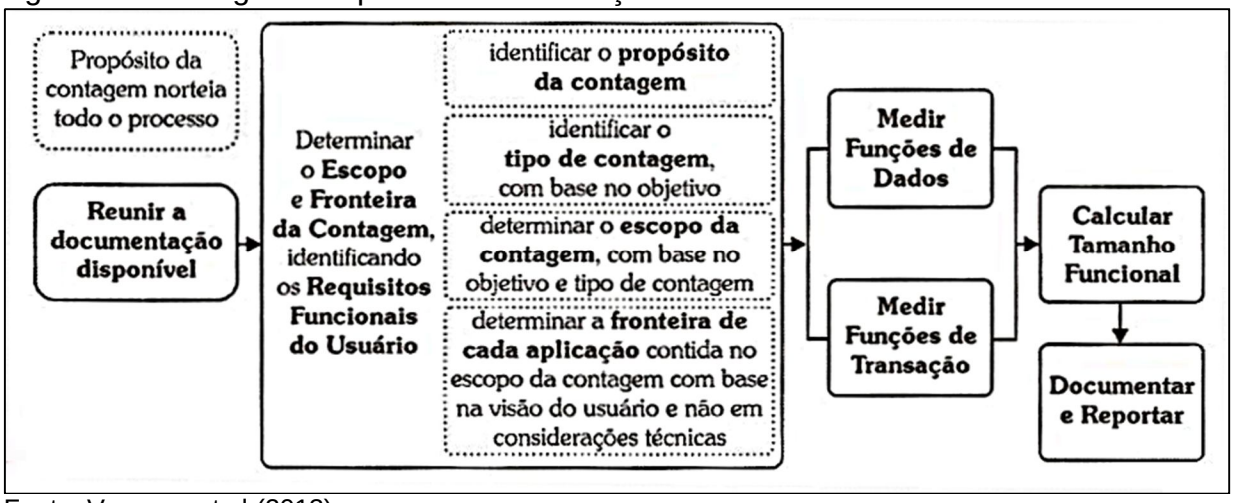
## 8.1 CONTAGEM DE PONTOS DE FUNÇÃO

A contagem de pontos da APF é baseada nas funções do sistema. Primeiramente devem-se contar apenas funcionalidades visíveis ao usuário. Por isso, não é todo requisito que conta, como algum cálculo interno, que não deve ser contado como função, embora possivelmente vá aparecer como parte de outra função. (WAZLAWICK, 2013).

Engholm (2010) descreve que a contagem de pontos de função, consiste em contagens de função apenas do ponto de vista funcional, que estão divididas em transações e dados. Para calcular o tamanho de uma aplicação, devem-se identificar as funcionalidades externas do sistema, como por exemplo, um cadastro ou uma consulta de uma aplicação. Ao aplicar as regras de contagem da APF, chega-se a um valor que determina o tamanho da funcionalidade, logo o tamanho do sistema.

A APF mede especificadamente os requisitos funcionais do usuário, e a essa dimensão é dada o nome de tamanho funcional. Essa medição permite expressar quantitativamente qualidades e funções do sistema. O diagrama da Figura 4 representa as etapas do processo de medição funcional, bem como as relações de interdependência entre seus passos. (VAZQUEZ et al., 2013).

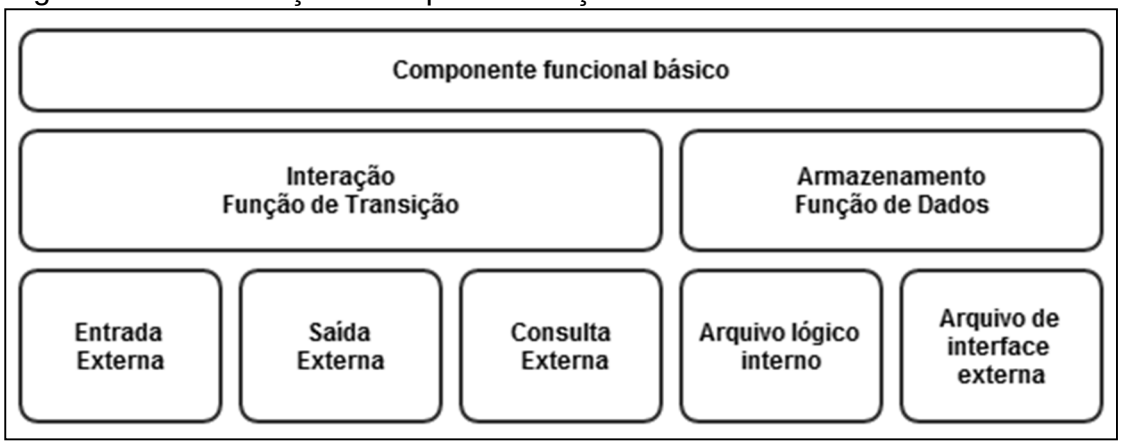
Figura 4 - Visão geral do processo de medição funcional do IFPUG.




Fonte: Vazquez et al (2013).

O autor ainda afirma que basicamente a medição consiste em decompor o projeto em elementos chamados componentes funcionais básicos ou funções. O método de medição conceitua abstrações, os tipos de função, nos quais os componentes básicos são classificados. Essa classificação é feita conforme sua função de armazenamento ou transação, conforme ilustra a Figura 5:

Figura 5 - Classificação dos tipos de função.

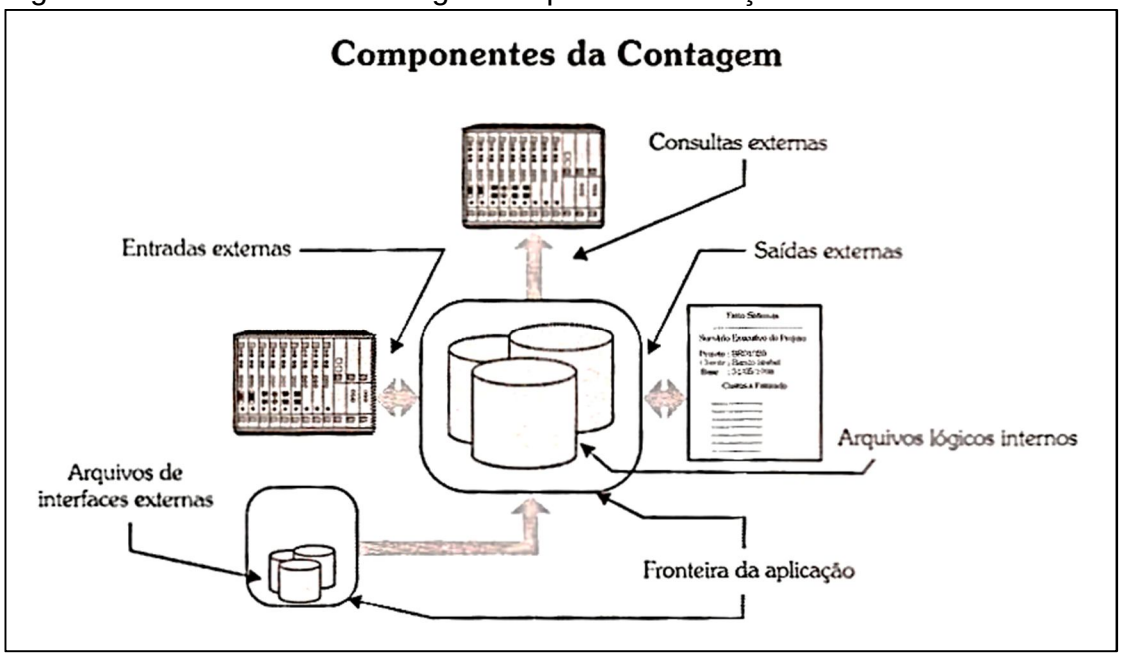



Fonte: Vazquez et al. (2013).

O método define referências para a identificação de cada componente funcional básico, classificação quanto ao tipo de função e à complexidade funcional, e a determinação de sua contribuição individual. Esses componentes funcionais básicos são denominados de função ou funcionalidades. O diagrama apresentado na Figura 6

define o que é externo e interno no sistema, bem como apresenta os tipos de função e a fronteira da aplicação.

Figura 6 - Elementos de contagem de pontos de função




Fonte: Vazquez et al. (2013).

Vazquez et al. (2013) concluíram que a contagem de pontos de função não é um processo final, mas tem como finalidade solucionar alguns problemas de negócios, como, por exemplo, estimar o custo e o esforço de um projeto de software.

## 8.2 DOCUMENTAÇÃO DISPONÍVEL

De acordo com Vazquez et al. (2013), o primeiro passo da medição de um sistema consiste em reunir a documentação disponível sobre o sistema que será medido.

O propósito da contagem ajuda a definir quais documentos são mais interessantes no processo de medição. A documentação ideal deve:

- Descrever a funcionalidade entregue pelo software.
- Descrever a funcionalidade que é de impacto pelo projeto do software medido.

Os documentos que podem ser utilizados na medição são os Diagramas de Classe, Diagramas de Fluxo de Dados, Casos de Uso, relatórios e manuais de usuários.

O autor ainda explica que não há um único documento que resolva todas as necessidades da medição, logo as métricas têm à sua disposição um conjunto de documentos já devidamente organizados do projeto, essa etapa do processo acontece quase de forma instantânea. Por outro lado, se não há documentação suficientemente disponível, é preciso buscar o acesso a especialistas no negócio para complementar essa lacuna. Isso implica em agendamento de reuniões, pesquisas e entrevistas, onde envolve um esforço adicional na medição.

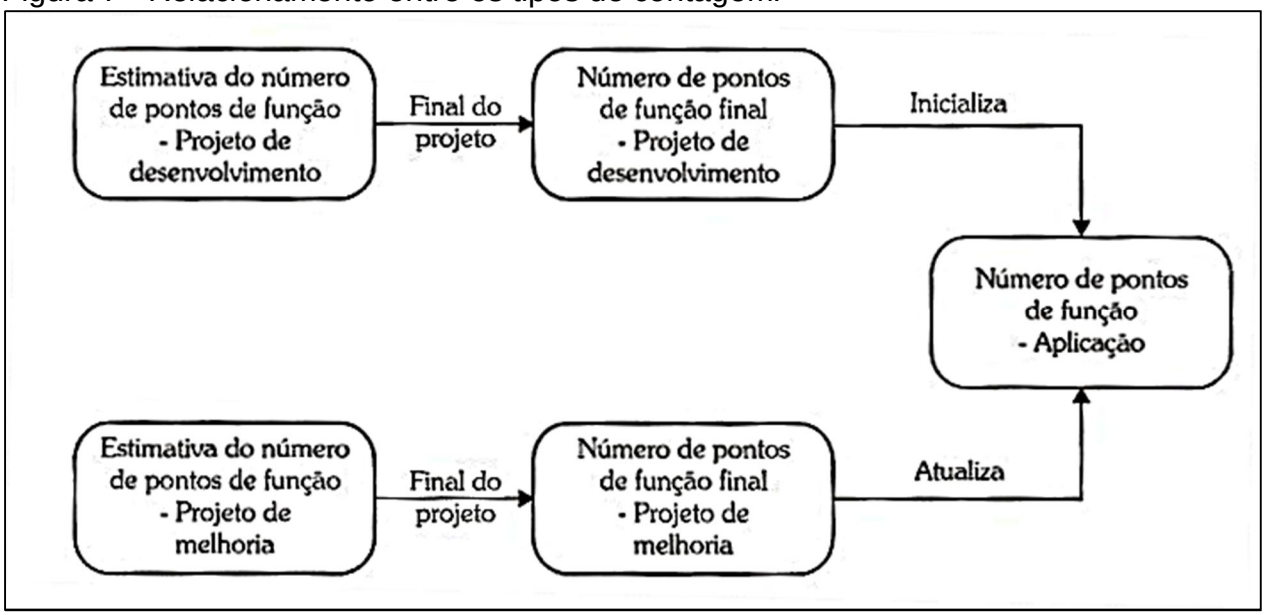
## 8.3 DETERMINAÇÃO DO TIPO DE CONTAGEM

Vazquez et al. (2013) afirmam que, ao determinar o tipo de contagem os responsáveis pela medição definem o tipo de contagem que será utilizada. Os três tipos de contagem de forma detalhada são:

- Projeto de desenvolvimento**:** Mede a funcionalidade fornecida aos usuários finais do software quando da sua primeira instalação, e abrange também as eventuais funções de conversão de dados referentes à implantação do sistema.
- Projeto de melhoria**:** Mede as funções adicionadas, modificadas ou excluídas do sistema, e também em eventuais conversões de dados.
- Aplicação: Mede a funcionalidade fornecida aos usuários por uma aplicação instalada. Conhecida também como baseline, ele é inicializado no final da contagem do número de pontos de função do projeto em desenvolvimento, sendo atualizado no término de todo projeto de melhoria que altera a funcionalidade da aplicação.

A Figura 7 demonstra o relacionamento entre os tipos de contagem mencionados acima.

Figura 7 - Relacionamento entre os tipos de contagem.




Fonte: Vazquez et al. (2013).

## 8.4 FRONTEIRA DA APLICAÇÃO E ESCOPO DA CONTAGEM

Para realizar uma contagem de pontos de função é necessário definir a fronteira da aplicação, assim como qual é o tipo de contagem e qual é o seu escopo. Este é uma das atividades que está dentro da segunda etapa dos procedimentos de contagem, e serve para identificar quais funcionalidades estão dentro da aplicação. (CAMPOS, 2010b).

Vazquez et al. (2013) definem a fronteira conceitual da aplicação como a interface que delimita o software que será medido e o mundo exterior, no caso, os usuários.

Os autores ainda afirmam que fazendo uma analogia com o mundo real, a fronteira da aplicação seria a cerca que delimita uma fazenda. Da mesma forma que não seria possível medir a área da fazenda se não houvesse a cerca estabelecendo seus limites, não é possível medir o tamanho funcional de um sistema sem que antes se tenha estabelecido claramente a sua fronteira. Ao identificar a fronteira é um passo essencial para a medição funcional. O IFPUG (2010 citado por VAZQUEZ et al.,2013) especifica as regras para determinar as fronteiras da aplicação:

- Sua determinação deve ser feita com base no ponto de vista do usuário.

- A fronteira entre aplicações deve ser baseada na separação das funções conforme estabelecidos pelos processos do negócio;
- Em projetos de melhoria, a fronteira estabelecida no início do projeto deve estar de acordo com a fronteira já estabelecida para a aplicação sendo modificada.

Campos (2010a) define o escopo da contagem como um conjunto de requisitos funcionais de usuários que serão incluídas numa contagem de pontos de função. Pode ser que inclua todas as funcionalidades de um sistema ou uma parte dela, isso depende do propósito do tipo de contagem definida. Na medida em que o escopo define as funcionalidades que vão entrar na contagem, automaticamente está se definindo as funções que vão entrar na contagem da APF, para fornecer respostas relevantes ao propósito da contagem.

## 8.5 FUNÇÕES DE DADOS

Vazquez et al. (2013) explicam que as funções do tipo dado representam as funcionalidades fornecidas pelo sistema ao usuário, com o objetivo de atender as suas necessidades de dados internos e externos, ou seja, mostram os dados armazenados do software. São classificados como Arquivos Lógicos Internos (ALI) e Arquivos de Interface Externa (AIE).

Pressman (2011) detalha ALI ou internal logical files (ILFs), sendo cada arquivo lógico interno, que é um agrupamento lógico de dados que reside dentro das fronteiras do aplicativo e é mantido através de entradas externas.

IFPUG (1999 citado Macoratti, c2010) descreve exemplos de ALIs:

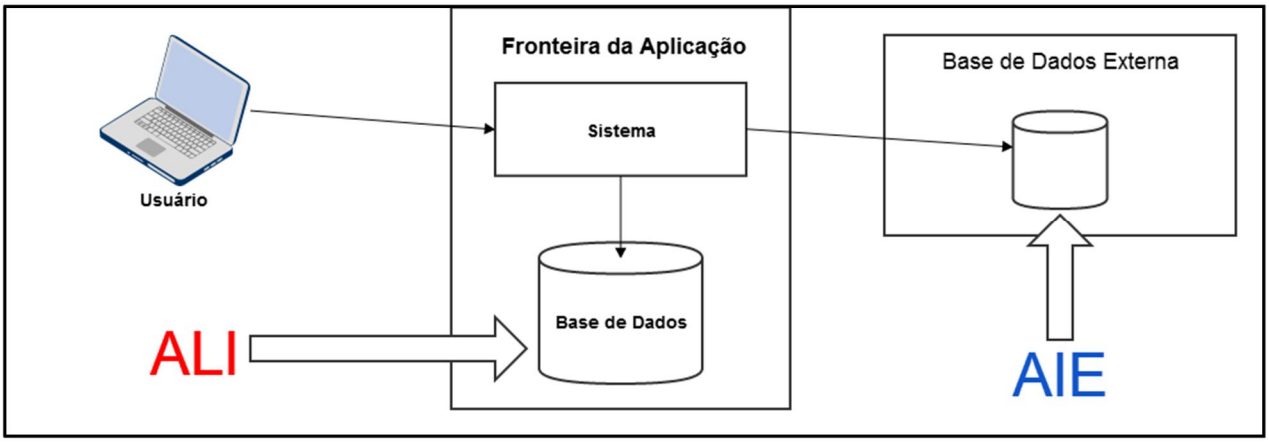
- Dados da aplicação (arquivos mestres como cadastro de clientes);
- Arquivos de dados de segurança da aplicação;
- Arquivos de dados de auditoria;
- Arquivos de mensagem de auxílio;
- Arquivos de mensagens de erro;
- Arquivo de cópia de segurança.
- Arquivo que sofra manutenção por mais de uma aplicação.

O principal objetivo de um ALI é armazenar dados adicionados, modificados e excluídos por meio de uma ou mais transações da aplicação sendo contada. (VAZQUEZ et al., 2013).

Pressman (2011) detalha AIE ou external interface files (EIFs), sendo cada arquivo de interface externo que é um agrupamento lógico de dados que reside fora da aplicação, mas fornece informações que podem ser usadas pelo sistema.

Para Vazquez et al. (2013), os AIE estão conceitualmente fora da fronteira da aplicação, e seu principal objetivo é utilizar dados armazenados fora da fronteira da aplicação.

Figura 8 - Representação da ALI e AIE




Fonte: Elaborada pelo autor.

De acordo com a representação da Figura 8, pode-se notar que AIE não é mantido pela aplicação. Ele está fora da fronteira da aplicação, enquanto o ALI está dentro.

Vazquez et al. (2013) afirmam que para calcular os pontos e função é preciso avaliar a complexidade funcional definida pelo número de Tipos de Dados (TD), e os Tipos de Registros (TR), e faz uma classificação com relação à complexidade fornecida pela tabela ilustrada na Figura 9.

Figura 9 - Complexidade funcional dos ALI e AIE.

| Tipos de Registros | Tipos de Dados | | | |
|---|---|---|---|---|
| | | **< 20** | **20 – 50** | **> 50** |
| | **1** | Baixa | Baixa | Média |
| | **2 – 5** | Baixa | Média | Alta |
| | **> 5** | Média | Alta | Alta |

Fonte: Vazquez et al. (2013).

8.5.1 Tipo de Dados e Tipos de Registros

Um TD é um campo único, reconhecido pelo usuário, não repetido. Em termos práticos, pode-se considerar um TD como um campo de arquivo, embora essa relação não seja perfeita. (VAZQUEZ et al., 2013).

Para realizar uma contagem de TD válida é necessário seguir determinadas regras:

- Conte um TD para cada campo único reconhecido pelo usuário e não repetido, mantido ou recuperado de um ALI e AIE por meio da execução de uma transação.
- Quando duas aplicações mantêm ou referenciam o mesmo ALI/ AIE, conte apenas os campos utilizados pela aplicação em análise.
- Conte um TD para cada campo solicitado pelo usuário para estabelecer um relacionamento com outros arquivos lógicos.

Um TR é um subgrupo de dados, reconhecido pelo usuário e componente dos ALIs e dos AIEs. (VAZQUEZ et al., 2013).

Os autores acima também explicam que há dois tipos de subgrupo:

- Opcionais: são aqueles em que o usuário tem a opção de não informar no processo elementar que inclui ou adiciona dados;
- Obrigatórios: são aqueles que o usuário requer que sejam sempre utilizados pelo processo elementar que inclui ou adiciona dados.

### 8.5.2 Determinação da Contribuição

Vazquez et al. (2013) afirmam que após a determinação da complexidade dos arquivos, deve-se calcular sua contribuição, utilizando a tabela ilustra da na Figura 10.

Figura 10 - Complexidade funcional dos ALI e AIE.

| Tipo de Função | Baixa | Média | Alta |
|---|---|---|---|
| Arquivo Lógico Interno | 7 PF | 10 PF | 15 PF |
| Arquivo de Interface Externa | 5 PF | 7 PF | 10 PF |

Fonte: Vazquez et al. (2013).

Como pode ser observado ma Figura 10, caso um ALI tenha uma complexidade alta, ela irá contribuir com 15 pontos de função. Já se um AIE tenha uma complexidade baixa, irá contribuir com seus 5 pontos de função.

## 8.6 FUNÇÕES DE TRANSAÇÃO

Vazquez et al. (2013) explicam que as funções de transação representam a funcionalidade fornecida ao usuário para atender às suas necessidades de processamento de dados pela aplicação. São classificados como Entrada Externa (EE), Saída Externa (SE) e Consulta Externa (CE).

Pressman (2011) detalha EE ou external inputs (EIs), sendo cada entrada originada de um usuário ou transmitida de outra aplicação, e fornece dados distintos orientados a aplicação ou informações de controle. As entradas são muitas vezes usadas para atualizar ALI.

Para Macoratti (c2010), um processo pode ser identificado como uma EE, se pelo menos uma das três opções a seguir for satisfeita:

- A lógica de processamento deve ser única e diferente das demais EE;
- O conjunto de dados elementares identificados é distinto dos conjuntos identificados por outras EE;
- Os ALIs mantidos e os AIEs referenciados são distintos dos utilizados por outras EE;

Em geral, EE são caracterizados por termos como incluir, alterar, excluir, importar, gravar e carregar. (VAZQUEZ et al., 2013).

Pressman (2011) detalha que SE ou external outputs (EOs) é formado por dados derivados da aplicação e fornece as informações para o usuário. Nesse caso, são exemplo de SE, os relatórios, telas de consulta, mensagens de erros etc.

Uma SE é uma consulta ou relatório no qual haja processamento de dados. Vazquez et al. (2013) destacam que consultas e relatórios que tenham totalizador ou com cálculo matemático, que apresentem gráficos, que atualizam arquivos, e que tenham dados derivados, ou que modificam o comportamento do sistema são considerados exemplos de SE.

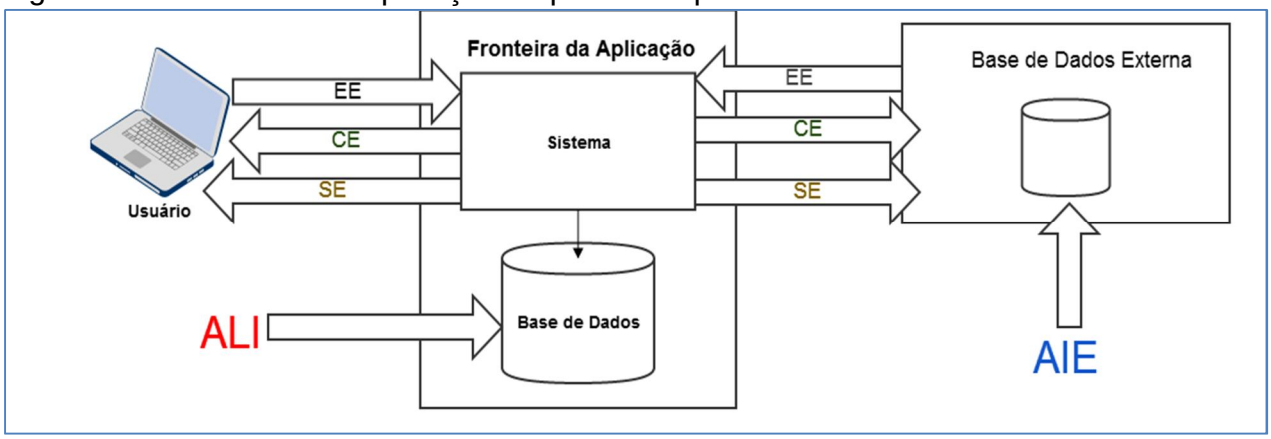
Pressman (2011) detalha CE ou External Inquiries (EQs), como uma entrada em tempo real que resulta na geração de alguma resposta imediata do sistema, na forma de uma saída online, onde muitas vezes é obtida de um ALI ou AIE.

São exemplos de CE, informações em formato gráfico, consulta e listagem de dados gerados por informações armazenadas em ALI e AIE, telas de login sem o uso de criptografia e menus gerados dinamicamente que sejam baseados na configuração da aplicação (VAZQUEZ et al., 2013).

Uma CE não há processamento de dados, como cálculos matemáticos, e nem criar dados derivados ou atualizar uma base de dados. Para Macoratti (c2010) não são consideradas CE:

- Menus que oferecem somente funcionalidade de seleção de telas;
- Dados derivados;
- Documentação On-Line;
- Sistema de Teste;
- Tutoriais do sistema;
- Relatórios e consultas que contenham cálculo ou gerem dados derivados.

Figura 11 - Fronteiras da aplicação e tipos de arquivos.




Fonte: Elaborada pelo autor.

A Figura 11 ilustra como as funções de dados e de transição influenciam o sistema. As funções de transição estão relacionadas com as entradas e saídas do sistema, e as funções de dados estão relacionados com o armazenamento interno e externo das informações do projeto.

Vazquez et al. (2013) definem que a EE, SE e CE tem como característica comum a de ser um processo elementar. Um processo elementar é a menor unidade de atividade que satisfaz todas as seguintes regras:

- Tem significado para o usuário.
- Constitui uma transação completa.
- É autocontido.
- Deixa o negócio da aplicação sendo contada em um estado consistente.

Os autores ainda descrevem que o conceito de processo elementar é o mais importante para a medição das funções de transação. Se por um lado ele evita que vários processos elementares sejam contados como um único, por outro lado impede que subprocessos componentes de processos elementares também sejam contados. De forma prática, uma tela de cadastro que permite incluir, alterar e excluir deve ser contada como três processos elementares.

Para cada tipo de função de transação, certos tipos de lógica de processamento podem ser executados. A lógica de processamento é definida como qualquer dos requisitos especificadamente solicitados pelo usuário para completar um processo

elementar. A tabela ilustrada na Figura 12 resume as lógicas de processamento que podem ser executadas por cada tipo de função de transação.

Figura 12 - Resumo das lógicas de processamento.

| **Formas de lógica de processamento** | **Tipos de Função de Transação** | | |
|---|---|---|---|
| | **EE** | **SE** | **CE** |
| 1. Validações efetuadas | Pode | Pode | Pode |
| 2. Fórmulas matemáticas e cálculos são efetuados | Pode | Deve* | Não |
| 3. Valores equivalentes são convertidos | Pode | Pode | Pode |
| 4. Dados são filtrados e selecionados por critérios específicos para comparar vários grupos de dados | Pode | Pode | Pode |
| 5. Condições são analisadas para determinas quais se aplicam | Pode | Pode | Pode |
| 6. Pelo menos um ALI é atualizado | Deve* | Deve* | Não |
| 7. Pelo menos um ALI ou AIE é referenciado | Pode | Pode | Deve |
| 8. Dados ou informações de controle são recuperados | Pode | Pode | Deve |
| 9. Dados derivados são criados | Pode | Deve* | Não |
| 10. O comportamento do sistema é alterado | Deve* | Deve* | Não |
| 11. Preparar e apresentar informações para fora da fronteira | Pode | Deve | Deve |
| 12. Capacidade de aceitar dados ou informações de controle que entram pela fronteira da aplicação | Deve | Pode | Pode |
| 13. Reclassificar ou reorganizar um grupo de dados | Pode | Pode | Pode |
| * Ao menos uma das lógicas de processamento deve estar presente. | | | |

Fonte: Vazquez et al. (2013) adaptado pelo autor.

De acordo com Vazquez et al. (2013), cada EE, SE e CE deve ser classificada com relação à sua complexidade, baseado no número de TD e de Arquivos referenciados (AR), conforme as tabelas de complexidade, ilustradas pelas Figuras 13 e 14.

Figura 13 - Complexidade para entradas externas (EEs).

| Arquivos Referenciados | Tipos de Dados | | | |
|---|---|---|---|---|
| | | **< 5** | **5 – 15** | **> 15** |
| | **< 2** | Baixa | Baixa | Média |
| | **2** | Baixa | Média | Alta |
| | **> 2** | Média | Alta | Alta |

Fonte: Vazquez et al. (2013).

Figura 14 - Complexidade para saídas externas (SEs) e consultas externas (CEs).

| Arquivos Referenciados | Tipos de Dados | | | |
|---|---|---|---|---|
| | | **< 6** | **6 – 19** | **> 19** |
| | **< 2** | Baixa | Baixa | Média |
| | **2** | Baixa | Média | Alta |
| | **> 2** | Média | Alta | Alta |

Fonte: Vazquez et al. (2013).

Os autores ainda afirmam que das tabelas de complexidade percebe-se que EE e SE, podem não ter arquivos referenciados, no entanto uma CE, pela própria definição, deve referenciar ao menos um ALI ou AIE.

### 8.6.1 Arquivo Referenciado e Tipo de Dado

IFPUG (1999 citado por Macoratti, c2010) afirma que um arquivo referenciado é qualquer ALI que foi consultado ou atualizado pelo processo ou qualquer AIE, sendo que o número de ARs é a soma dos ALI e AIE atualizados ou consultados.

Vazquez et al. (2013) definem que as seguintes regras são válidas para a contagem de um AR. As duas primeiras tratam de atualização de arquivos, e não são aplicáveis para CE.

- Conte um AR para cada ALI mantido.
- Conte apenas um AR para cada ALI que seja tanto mantido como lido.
- Conte um AR para cada ALI ou AIE lido durante o processamento.

Vazquez et al. (2013), ainda definem as regras válidas para contagem de TD de uma função de transação:

- Conte um único TD para cada atributo que ultrapasse a fronteira da aplicação, sendo de entrada ou saída, reconhecida pelo cliente, que não seja repetido.
- Conte um único TD para a capacidade de envio ou emissão de mensagem de resposta do sistema para fora da fronteira da aplicação, indicando um erro ou confirmando a execução de um processamento.
- Conte um único TD para a capacidade de especificar uma ação a ser tomada. Mesmo que haja múltiplos meios para ativar o processo, deve ser contado apenas um TD.

### 8.6.2 Determinação da Contribuição

Vazquez et al. (2013) afirmam que após a determinação da complexidade das funções de transação, deve-se calcular sua contribuição, utilizando a tabela ilustrada na Figura 15.

Figura 15 - Contribuição dos pontos de função das funções do tipo transação.

| Tipo de Função | Baixa | Média | Alta |
|---|---|---|---|
| Entrada Externa | 3 PF | 4 PF | 6 PF |
| Saída Externa | 4 PF | 5 PF | 7 PF |
| Consulta Externa | 3 PF | 4 PF | 6 PF |

Fonte: Vazquez et al. (2013).

Conforme apresentada na Figura 15, caso um EE tenha uma complexidade alta, ela irá contribuir com 6 pontos de função. Já se um SE tenha uma complexidade média, irá contribuir com seus 5 pontos de função. E se uma CE apresente complexidade baixa, irá contribuir com seus 3 pontos de função

## 8.7 FATOR DO AJUSTE

O fator de ajuste, segundo o IFPUG é opcional na aplicação da técnica de APF. O propósito do fator de ajuste é medir requisitos gerais da aplicação (não funcionais),

ele ajusta os pontos de função em mais ou menos 35% de acordo com a influência de 14 características gerais. (VAZQUEZ et al., 2013).

Macoratti (c2010) afirma que a técnica de APF considera que outros fatores afetam o tamanho funcional de um sistema, e estão relacionados com características da aplicação. No cálculo dos pontos de função brutos não é levada em conta a tecnologia usada nem os requisitos não funcionais. Por este motivo é calculado o valor do fator de ajuste (VFA) que é baseado em 14 características gerais de sistema, listada na Figura 16.

Figura 16 - Características gerais do sistema.

| **Características gerais do sistema** |
|---|
| 01 - Comunicação de dados |
| 02 - Processamento distribuído |
| 03 – Performance |
| 04 - Utilização de Equipamento |
| 05 - Volume de transações |
| 06 - Entrada de dados on-line |
| 07 - Eficiência do Usuário Final |
| 08 - Atualização On-Line |
| 09 - Processamento complexo |
| 10 - Reutilização de código |
| 11 - Facilidade de Implantação |
| 12 - Facilidade Operacional |
| 13 - Múltiplos Locais |
| 14 - Facilidade de mudanças |

Fonte: Vazquez et al. (2013) adaptado pelo autor.

De acordo Vazquez et al. (2013), as funções de tipo de dados definem requisitos específicos de armazenamento, e as funções de transação refletem requisitos de processamento, já as características gerais refletem funções que afetam a aplicação de maneira geral. Essas características possuem nível de influência em uma escala que varia de 0 a 5, conforme a tabela ilustrada na Figura 17.

Figura 17 - Níveis de influência das Características gerais do sistema.

| Níveis de Influência | Valor |
|---|---|
| Nenhuma Influência | 0 |
| Influência Mínima | 1 |
| Influência Moderada | 2 |
| Influência Média | 3 |
| Influência Significativa | 4 |
| Grande Influência | 5 |

Fonte: Vazquez et al. (2013) adaptado pelo autor.

Após apurar-se o nível de influência de todas as Características gerais do sistema, o VFA é baseado na equação:

VFA = 0,65 + (Soma das Características Gerais do Sistema x 0,01).

Se o VFA é igual a 1, a influência total das características gerais do sistema é neutra. Nesta situação, a contagem dos pontos de função ajustados equivale à contagem de pontos de função não ajustados. (MACORATTI, c2010).

## 8.8 CÁLCULO DO TAMANHO FUNCIONAL

Após a identificação e classificação de todas as funções de dados, e de transação na contagem, o número de pontos de função será simplesmente a soma do peso de cada uma dessas funções. (VAZQUEZ et al., 2013).

Para IFPUG (1999 citado por VAZQUEZ, 2005), após definir a fronteira da aplicação, o tipo de contagem, e reconhecer as funções de dados e de transação pode se calcular os pontos de função não ajustados, ou brutos, multiplicando-se o total de ALI, AIE, EE, SE, e CE pela respectiva complexidade conforme a tabela da Figura 18.

Figura 18 - Complexidade dos tipos funcionais.

| Descrição do | Complexidade | | |
|---|---|---|---|
| Tipo Funcional | Simples | Média | Complexa |
| Arquivo Lógico Interno (ALI) | 7 PFs | 10 PFs | 15 PFs |
| Arquivo de Interface Externa (AIE) | 5 PFs | 7 PFs | 10 PFs |
| Entrada Externa (EE) | 3 PFs | 4 PFs | 6 PFs |
| Saída Externa (SE) | 4 PFs | 5 PFs | 7 PFs |
| Consulta Externa (CE) | 3 PFs | 4 PFs | **6 PFs** |

Fonte: Hazan (2008).

Vazquez et al. (2013) descreve que o passo de contagem de pontos de função envolve o cálculo final para os três tipos de contagem, sendo projeto desenvolvimento, melhoria e aplicação.

### 8.8.1 Cálculo de Pontos de Função não Ajustados ou Brutos

Para calcular os pontos de função não ajustados ou brutos, é a soma dos pesos das funções de tipo de dado e de transação, que é determinado pela complexidade da função, podendo ser baixa, média ou alta.

### 9.8.2 Projeto de Desenvolvimento

Vazquez et al. (2013) descrevem os componentes de pontos de função de um projeto de desenvolvimento:

- Funcionalidade da aplicação requisitada pelo usuário para o projeto.
- Funcionalidade de conversão requisitada pelo usuário para o projeto.

A fórmula para o cálculo do projeto de desenvolvimento é:

DFP = ADD + CFP.

Sendo que:

- DFP: tamanho do projeto de desenvolvimento.
- ADD: tamanho das funções entregues.

- CFP: tamanho das funções de conversão.

Para exemplificar a fórmula, os autores apresentam a tabela ilustrada na Figura 19, onde é possível visualizar as funções de um sistema de controle de ponto bem simplificado.

Figura 19 - Planilha de contagem de função do sistema de controle

| Nome da Função | Tipo | TD | AR/ TR | Complexidade | Contribuição |
|---|---|---|---|---|---|
| Pessoa (do controle de segurança) | AIE | 3 | 1 | Baixa | 5 |
| Login | CE | 4 | 1 | Baixa | 3 |
| Apontamento | ALI | 4 | 1 | Baixa | 7 |
| Apontamento – Importação | EE | 4 | 1 | Baixa | 3 |
| Registro de Ponto | EE | 4 | 2 | Baixa | 3 |
| Justificativa | ALI | 3 | 1 | Baixa | 7 |
| Justificativa – Importação | EE | 3 | 1 | Baixa | 3 |
| Justificativa – Seleção | CE | 6 | 2 | Média | 4 |
| Justificativa – Inclusão | EE | 5 | 2 | Média | 4 |
| Justificativa – Alteração | EE | 5 | 2 | Média | 4 |
| Justificativa – Exclusão | EE | 3 | 1 | Baixa | 3 |
| Justificativa – Consulta | CE | 5 | 2 | Baixa | 3 |
| Horário Padrão | ALI | 3 | 1 | Baixa | 7 |
| Horário Padrão – Importação | EE | 3 | 1 | Baixa | 3 |
| Horário Padrão – Inclusão | EE | 4 | 2 | Baixa | 3 |
| Horário Padrão – Alteração | EE | 4 | 2 | Baixa | 3 |
| Horário Padrão – Consulta | CE | 4 | 2 | Baixa | 3 |
| Relatório de Horas | SE | 10 | 4 | Alta | 7 |
| Relatório de Justificativas | SE | 8 | 4 | Alta | 7 |

Fonte: Vazquez et al (2013) adaptada pelo autor.

Aplicando a fórmula, tem-se:

DFP = ADD + CFP

DFP = 73 + 9

DFP = 82 pontos de função.

Caso seja utilizado o valor de fator ajustado (VFA), multiplica-se o resultado de DFP com o valor resultante de VFA.

### 8.8.3 Projeto de Melhoria

Vazquez et al. (2005, citado por Macoratti, c2010) citam que de acordo com o IFPUG, o conceito de projeto de melhoria envolve apenas manutenções evolutivas na aplicação, ou seja, alterações feitas no sistema para atender aos novos requisitos de negócio do usuário. Não são levadas em conta manutenções corretivas e preventivas.

Observe que não é necessário saber o número de pontos de função da aplicação para determinar o tamanho do projeto de melhoria. Neste caso, o intuito é medir apenas funções que serão afetadas pela manutenção. Caso a contagem da aplicação já esteja disponível, a medição do projeto de melhoria irá ficar mais fácil, pois para funções alteradas, basta apenas trazê-las da contagem do sistema, ajustado uma eventual mudança. Para as funções de exclusões fica mais fácil ainda, onde do mesmo jeito que foram contadas na aplicação, serão contadas na melhoria. (VAZQUEZ et al., 2013).

Os autores também definem a fórmula para o cálculo:

EFP = ADD + CHGA + CEP + DEL.

Em que:

- EFP: número de pontos de função do projeto de melhoria.
- ADD: tamanho das funções incluídas pelo projeto de melhoria.
- CHGA: tamanho das funções modificadas. Reflete as funções depois das modificações.
- CFP: tamanho das funções de conversão.
- DEL: tamanho das funções excluídas pelo projeto de melhoria.

Por meio da Figura 19, Vazquez et al. (2013) mostram um exemplo com o cálculo de projeto de melhoria através da Figura 20

Figura 20 - Exemplo de contagem de pontos de função do projeto de melhoria

| Descrição de função | Tipo | Depois do Projeto de Melhoria | | | | Antes do Projeto de Melhoria | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | TD | AR/TR | Complexidade | FP | TD | AR/TR | Complexidade | FP |
| Senha | AIE | 2 | 1 | Baixa | 5 | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Relatório de Ponto | SE | 12 | 4 | Alta | 7 | | | | |
| Login | CE | 4 | 1 | Baixa | 3 | 4 | 2 | Baixa | 3 |
| Relatório de Justificativas | SE | | | | | 8 | 4 | Alta | 7 |
| Relatório de Horas | SE | | | | | 10 | 4 | Alta | 7 |

Fonte: Vazquez et al (2013) adaptada pelo autor.

Aplicando a fórmula para determinação do tamanho do projeto de melhoria, têm-se:

EFP = (ADD + CHGA + CFP + DEL)

EFP = (12 + 3 + 0 + 14)

EFP = 29 pontos de função.

A fórmula para o cálculo com o VFA. Segundo macoratti (c2010) seria:

EFP = [(ADD + CHGA + CFP) * VAFA] + (DEL * VAFB).

Onde:

- EFP: Número de pontos de função do projeto de melhoria.
- ADD: Número de pontos de função não ajustados das funções incluídas pelo projeto de melhoria.
- CHGA: Número de pontos de função não ajustados das funções modificadas depois das modificações.
- CFP: Número de pontos de função não ajustados adicionados pela conversão.
- VAFA: Fatores de ajuste de valor da aplicação depois do projeto de melhoria.
- DEL: Número de pontos de função não ajustados das funções excluídas pelo projeto de melhoria;
- VAFB: Fatores de ajuste de valor da aplicação antes do projeto de melhoria.

8.8.4 Aplicação

Para Vazquez et al. (2013), existem duas fórmulas para calcular o número de pontos de função da aplicação, uma para contagem inicial, e outra para recalcular o projeto de melhoria.

A fórmula da contagem inicial da aplicação é:

AFP = ADD

Onde:

- AFP: tamanho da aplicação;
- ADD: tamanho das funções entregue;

Caso contar o valor do fator de ajuste:

AFP = ADD * VAF

Utilizando o mesmo exemplo da tabela ilustrada pela Figura 20, resulta em:

AFP = ADD = 73 pontos de função

Quando o projeto de melhoria é concluído, o número de pontos de função da aplicação deve ser atualizado para refletir tais modificações na aplicação. (VAZQUEZ et al., 2013).

Para recalcular o projeto de melhoria é utilizada a formula:

AFPA = (AFPB + ADD + CHGA) – (CHGB + DEL)

Caso for usado o valor de fator de ajuste:

AFP = [(UFPB + ADD + CHGA) – (CHGB + DEL)] * VAFA

Sendo que:

- AFPA: tamanho da aplicação após a melhoria.
- AFPB: 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 do seu término.

- CHGB: tamanho das funções alteradas pelo projeto de melhoria antes do seu término.
- DEL: tamanho das funções excluídas pelo projeto de melhoria.

Aplicando a fórmula à aplicação alterada pelo projeto de melhoria usado a Figura 20, têm-se:

AFPA = (AFPB + ADD + CHGA) – (CHGB + DEL)

AFPA = (73 + 12 + 3) – (3 + 14)

AFPA = 71 pontos de função

## 8.9 DOCUMENTAR E REPORTAR

Vazquez et al. (2013) afirmam que documentar e reportar os resultados é a etapa final do processo de medição. O nível de detalhamento da documentação deve estar de acordo com o propósito da contagem.

Os autores ainda detalham que o nível de detalhamento da documentação deve estar previamente acordado entre as partes interessadas na medição, ponderando-se os custos e benefícios envolvidos. Um nível de documentação alto implica mais tempo no processo e custo do projeto, porém, os autores citam algumas facilidades de uma documentação bem detalhada, tais como: Auditoria da medição, rastrear as funções identificadas até os artefatos do projeto usados na medição, usar os resultados da medição e manter e evoluir a medição.

# 9 TRABALHOS CORRELATOS

Neste capitulo serão descritos alguns trabalhos que descrevem e justificam o uso da APF para estimativa do software.

Mecenas (2013) projetou a ferramenta desktop APFplus, que foi desenvolvida para profissionais interessados em utilizar a metodologia de contagem de pontos de função. A ferramenta atualmente está na versão 4.3.1.

HAZAN e STAA (2004) desenvolveram uma pesquisa que têm como objetivo analisar um processo de estimativas do Serviço Federal de Processamento de Dados (SERPRO), e implantar melhorias através de uma metodologia de estimativas de tamanho de projetos de desenvolvimento, e de manutenção evolutiva de software, inserindo os conceitos de APF.

Justulin Junior (2013) desenvolveu um projeto que consiste na criação de um protótipo de um software, a ser desenvolvido na linguagem JAVA, que calcula o tamanho de um projeto em Pontos de Função, para que a partir deste resultado, apresente as estimativas necessárias que o engenheiro de software precisa: estimativa de tamanho, esforço, prazo e custo.

Dessa forma, a intenção deste trabalho é desenvolver um sistema web que calcule pontos por função de forma online, e que facilite o trabalho do engenheiro de software.

# 10 METODOLOGIA

O desenvolvimento desse projeto inicialmente foi uma pesquisa exploratória, com a seleção de material bibliográfico adequado, entre eles, livros, artigos e trabalhos relacionados.

Em seguida, foram realizados estudos sobre engenharia de software, qualidade de sistemas, planejamento e estimativas de projetos e métricas de software, No estudo realizado, foi dado o maior destaque na medição por APF, incluindo seu conceito, características e descrição da mesma

A pesquisa realizada teve como objetivo a construção de um protótipo de um sistema para estimativa utilizando a técnica de ponto por função.

## 10.1 MODELAGEM UML

Em um seguinte momento foi realizada a modelagem do sistema utilizando a linguagem UML (Unified Modeling Language). A UML é uma linguagem gráfica padrão para a elaboração da estrutura de projetos complexos de software, e pode ser empregada para visualizar, especificar, construir e documentar os artefatos de um sistema.

Foram modelados os Diagramas de Classes e de Casos de Uso com suas devidas descrições.

O Diagrama de Classe representa a estrutura e relações das classes que serão utilizadas no projeto, sendo possível detalha-las com seus relacionamentos, definindo também os atributos e as operações, e é a base para a construção dos diagramas de comunicação, sequência e estados.

O Diagrama de Casos de Uso descreve a funcionalidade proposta do sistema que será projetado, e é uma excelente ferramenta para o levantamento dos requisitos funcionais do sistema.

A ferramenta utilizada para a modelagem do software foi o JUDE, pois além de ser gratuito (open source), suporta sistemas orientados a objeto.

## 10.2 BANCO DE DADOS

O Sistema de Gerenciamento do Banco de Dados (SGBD) escolhido para armazenar as informações do projeto foi o MySQL, pois é um servidor rápido, de multiprocessamento e multiusuários em SQL (Structured Query Language), muito confiável.

O MySQL é o banco de dados de código-fonte aberto mais popular do mundo e gratuito, tendo mais de 70 milhões de instalações no mundo todo e é utilizado por diversas empresas.

A ferramenta que foi utilizada para o gerenciamento dos dados foi o MySQL Workbench. Além de ser gratuita, é uma ferramenta visual unificada para arquitetos de banco de dados e desenvolvedores. MySQL Workbench fornece modelagem de dados, desenvolvimento de SQL e ferramentas de administração abrangentes para configuração do servidor, administração de usuários, backup e outro recurso que possibilita o uso de engenharia reversa do banco de dados, facilitando muito o trabalho de modelagem do banco de dados.

## 10.3 LINGUAGENS DE PROGRAMAÇÃO UTILIZADAS

O protótipo deste trabalho foi destinado para web e foi desenvolvido na linguagem de marcação HTML e a linguagem de programação PHP.

A HTML consiste em uma linguagem de marcação utilizada para produção de páginas web, que permite a criação de documentos que podem ser lidos em praticamente qualquer tipo de dispositivo com acesso à internet. Os códigos HTML, conhecidos como tags, servem para indicar a função de cada elemento da página web. As tags funcionam como comandos de formatação de textos, formulários, links, imagens, tabelas, e outros.

PHP é uma linguagem orientada a objeto e gratuita que conta com o apoio de uma grande comunidade de usuários e desenvolvedores. Para Deital J. e Deital M. (2008), PHP é uma ferramenta valiosa. Por que o poder da Web não está em apenas servir conteúdo aos usuários, mas também em responder às solicitações deste e gerar

páginas web com conteúdo dinâmico. A interatividade entre o usuário e servidor tornou-se parte fundamental da funcionalidade da web.

## 10.4 RESULTADOS ESPERADOS DO SOFTWARE

O protótipo deve calcular os pontos de função, e apresentar as estimativas para o usuário utilizá-las ao longo do projeto.

O sistema tem que apresentar interface clara e que obtenha resultados precisos, devido a dificuldades em determinar a exatidão do custo e do esforço no desenvolvimento do projeto e a data de sua conclusão. Antes disso, o que pode ser feito são estimativas que auxiliam a aproximação, e servem como indicador para próximos projetos.

Vazquez et al. (2013) afirma que o processo de estimativa de um projeto envolve basicamente quatros atividades:

- Estimar o tamanho do projeto a ser gerado.
- Estimar o esforço empregado na execução do sistema.
- Estimar a duração do projeto.
- Estimar o custo do software.

Hazan (2008) afirma que para totalizar o tamanho em pontos de função, deve-se proceder multiplicando-se o total de ALI, AIE, EE, SE e CE pela respectiva complexidade de cada uma, fazendo uma somatória total. Após obter este resultado, multiplica-se com o resultado do valor do fator de ajuste (VFA).

Depois de calcular a estimativa de tamanho, procede-se com a geração da estimativa de esforço. Deve-se obter, primeiramente, um índice de produtividade por meio de análise do banco de dados de histórico de projetos antigos, observando-se os atributos do projeto em questão e o esforço realizado em projetos similares. O cálculo nada mais é do que o produto entre o valor obtido da estimativa de tamanho com o número de horas para produzir um ponto de função.

Vazquez et al. (2013) afirmam que para determinar o cronograma do projeto a partir do esforço estimado. Esta estimativa é uma das atividades das quais exigem mais experiências e conhecimento dos aspectos envolvidos em sua execução.

A fórmula para a estimativa de prazo consiste em:

Prazo (em dias) = Esforço (horas) / Tamanho da equipe * 6 horas.

Caso o engenheiro de software não tenha ideia de quantos funcionários ele alocará no projeto, existe uma fórmula onde se calcula a estimativa ideal de prazo para um projeto.

Para esta estimativa, foi aplicada a fórmula de Caper Jones (1998 citada por HAZAN, 2008), onde consiste em:

Td (meses) = V ^ t

Onde:

- Td é o tempo ótimo de desenvolvimento, em meses.
- V é o volume em Pontos de Função
- t é um expoente que depende do ambiente computacional considerado, conforme mostrado na Figura 21.

Figura 21 - Índice de aproximação de estimativa de prazo de Caper Jones.

| **Ambiente** | **Expoente t** |
|---|---|
| Sistema Comum | 0,32-0,35 |
| Sistema Orientado a Objeto | 0,36 |
| Sistema Cliente/Servidor | 0,37 |
| Sistema Terceirizado | 0,38 |
| Sistema de Informações Gerenciais | 0,39 |
| Programa Produto Comercial | 0,40 |
| Programa de Sistema Operacional | 0,41 |
| Software Militar | 0,43-0,45 |

Fonte: Aguiar (2000).

Para Vazquez et al. (2013), a estimativa de custo deve considerar o valor do salário da equipe alocada ao projeto, bem como outros custos envolvidos no projeto, como ferramentas, deslocamentos, consultoria, etc. O ideal é ter dados históricos de custo por PF de projetos concluídos, possibilitando a derivação direta da estimativa de custo a partir da estimativa de volume em Pontos de Função.

Ainda de acordo com os autores, após ter vários projetos similares passados em base de dados, estima-se o tamanho do novo projeto com base em percentual do tamanho do software que está sendo medido.

# 11 RESULTADOS

As modelagens de classe e de caso de uso encontram-se no Apêndice A e B, elas buscam representar as classes do sistema e como ele se interage com o usuário de forma de fácil entendimento para o leitor, utilizando de variáveis claras e que possibilitam extrair os requisitos iniciais do software.

A modelagem de dados foi bem representada e mostra as tabelas e suas ligações, utilizando variáveis claras, conforme segue no Apêndice C.

## 11.1 RESULTADOS DO SOFTWARE

No sistema, o usuário encontrará um ambiente bem dinâmico e de fácil aprendizagem, no qual o usuário precise de uma menor quantidade de esforço para acessar determinado resultado.

Por ser um sistema web, é de grande importância que contenha confidencialidade, integridade e disponibilidade das informações nele cadastradas. O sistema estará disponível online, basta que o usuário tenha acesso a internet, tornado o acessível em qualquer dispositivo com acesso à web como computadores, smartfones e tablets. Para que o software mantenha seus dados confiáveis e intrigo, o sistema contém uma página de login e senha conforme ilustra a Figura 22. Sendo que a senha é criptografada com o algoritmo MD5 para manter a segurança dos dados.

De acordo com Abreu (2014), o algoritmo de criptografia MD5 é o mais utilizado atualmente, sendo nativo em várias linguagens de programação. A criptografia MD5 é unidirecional, não sendo possível descriptografar. O resultado da função é um retorno de um hash string binária de 32 dígitos hexadecimais.

Figura 22 - Tela de login


Fonte: Elaborada pelo autor.

No Menu Principal no topo do sistema, o usuário encontrará o link para todos os módulos do sistema, permitindo que o mesmo obtenha o resultado de forma rápida e com um menor número de cliques conforme a Figura 23.

Figura 23 - Menu principal


Fonte: Elaborada pelo autor.

### 11.1.1 Testes no Software

Para testar o software foram simuladas as estimativas com o mesmo *software* desenvolvido neste trabalho. Primeiramente foram cadastradas informações sobre a aplicação e o projeto conforme mostra as Figuras 24 e 25.

Figura 24 - Cadastro de Aplicações


Fonte: Elaborado pelo Autor.

Conforme ilustrada na Figura 25, para cadastrar um projeto, é necessário informar a categoria, tamanho da equipe, e qual o preço médio da equipe que está envolvida no projeto.

Figura 25 - Cadastro de Projetos


Fonte: Elaborado pelo Autor.

De posse dessas informações, já se pode iniciar os cálculos das métricas. O próximo passo seria obter o valor do Fator de Ajuste, na qual se devem responder as 14 perguntas referentes ao sistema com pesos de 0 a 5.

Conforme a Figura 26, o resultado obtido no teste do *software* é um fator de ajuste de valor 1,23.

Figura 26 – Cadastro de Fator de Ajuste


Fonte: Elaborada pelo autor.

Depois de realizar o fator de ajuste é possível é possível cadastrar informações para realizar o tamanho do *software* em pontos de função não ajustados, ou seja, ainda sem influência do Fator de Ajuste.

A tela de cadastro de módulo é apresentada na Figura 27 é onde são cadastradas as funcionalidades do sistema. Cada modulo pode ser classificado por cincos tipos de funções: Entradas externas, saídas externas, consultas externas, arquivos lógicos internos e arquivos de interface externas.

Após clicar no botão de inclusão, o sistema fará o cálculo de complexidade e contribuição de pontos de função, a somatória das contribuições de pontos de função de todos os módulos do sistema irá resultar nos pontos de função bruto.

Figura 27 – Cadastro de Módulos


Fonte: Ebolarado pelo Autor (2014).

Analisando as funções, é possível determinar o tipo de contagem, sua complexidade e o total de pontos de função, o total de pontos de função bruto das funções do sistema é 155.

Para este exemplo, como será o primeiro projeto, e não há bases históricas, será utilizada a produtividade de 5 horas para produzir um ponto de função conforme mostra a Figura 28.

Figura 28 – Produtividade em horas por pontos de função.


Fonte: Souza (2011).

Por fim, os pontos de função ajustados são calculados com a multiplicação do VFA com os pontos de função brutos que tem como resultado 190, 65 conforme a tela ilustrada na Figura 29.

Figura 29 - Tela de Resultado


Fonte: Elaborado pelo Autor.

Para Hazan (2008), o cálculo do esforço se dá ao produto entre o valor obtido da estimativa de tamanho com o número de horas para produzir um PF. O resultado será o esforço em HH (homens_hora).

O cálculo do custo depende de muitos fatores. Neste projeto, foi utilizado apenas o custo médio da equipe que foi cadastrado na tela de projetos conforme mostra a Figura 26, multiplicado pelo esforço utilizado no desenvolvimento do projeto.

Para se chegar ao prazo, utilizou-se a seguinte formula: Esforço (horas) / Tamanho da equipe * 6 horas. (HAZAN, 2002). Essa constante de 6 horas refere-se à produtividade média diária no Brasil (JONES 1997 citado por HAZAN, 2008). O resultado é dado em dias úteis. Para transformar em meses, basta apenas dividir o resultado por 30.

E, finalmente, para se chegar ao prazo ideal, utiliza-se fórmula de Caper Jones (1998 citada por HAZAN, 2008), onde consiste na formula (Td (meses) = V ^ t).

Onde:

- Td é o tempo ótimo de desenvolvimento, em meses.
- V é o volume em Pontos de Função
- t é um expoente que depende do ambiente computacional considerado, conforme mostrado na Figura 22.

No caso deste projeto, o ambiente escolhido foi o “sistema cliente-servidor”, na qual condiz com a arquitetura web do software.

Esse resultado é armazenado na base de dados para que o engenheiro de software tenha uma base para cadastrar novos projetos.

## 12 CONSIDERAÇÕES FINAIS

O sistema cumpriu com que foi proposto inicialmente, conforme demonstrado nos resultados do software. Além de realizar as estimativas do projeto de forma clara e objetiva, a ferramenta realizou os cálculos de maneira rápida, segura e por ser um sistema web, possibilitou ao usuário acessá-lo de qualquer lugar, desde que tenha um ponto de acesso à internet.

O prazo estimado no software foi de 5 meses, esse resultado ficou muito próximo a realidade, pois o início do projeto começou em torno de junho de 2014 até o termino no final de novembro de 2014.

Como recomendações para trabalhos futuros, sugere-se a análise de métodos de aprendizado de máquina com técnicas em inteligência artificial para o tratamento dos dados históricos de produtividade para as estimativas de custo e de esforço.

Outro fator que pode ser utilizado em trabalhos futuros é possibilitar a ferramenta o cálculo de outras métricas, como por exemplo a métrica de caso de uso para que o usuário tenha acesso a mais de um tipo de métrica para que sirva como comparativo.

Embora a ferramenta não consiga acertar com exatidão o tamanho do projeto. Ele é de grande importância, pois, possibilita o engenheiro de software tenha uma estimativa do tamanho, prazo e custo do sistema no início do projeto possibilitando o usuário um maior controle do sistema a ser desenvolvido e uma justificativa para contratação de novos membros para a equipe, compra de novas tecnologias de desenvolvimento e treinamento dos seus profissionais  Além disso o projeto contém um do histórico de produtividade e experiências passadas, que são de suma importância para avaliar projetos futuros.

# REFERÊNCIAS

ABREU, Fabiano. **Como Criptografar senha no MySQL.** 2014. Disponível em: <http://paposql.blogspot.com.br/2014/06/como-criptografar-senha-no-mysql.html>. Acesso em: 12 Nov. 2014.

AGUIAR, M. Uso de métricas na melhoria do processo de software. **BFPUG – Brazilian Function Point Users Group,** 2000. Disponível em: <http://www.bfpug.com.br/Artigo/SPIN-SP-23-10-2000.htm>. Acesso em: 12 Nov. 2014.

BENYON, David. **Interação humano-computador**. Tradução de Heloísa Coimbra de Souza. 2. ed. São Paulo: Pearson Prentice Hall, 2011.

Brazilian Function Point Users Group. O que é o BFPUG - Brazilian Function Point Users Group. **BFPUG**. Disponível em: <http://www.bfpug.com.br/>. Acessado em: 27 abr. 2014.

CAMPO, Carlos. **Escopo da contagem.** 2010a. Disponível em: <http://carloscamposinfo.com/mundoapf/?p=192>. Acesso em: 26 abr. 2014.

_____. **Fronteira da aplicação**. 2010b. Disponível em: <http://carloscamposinfo.com/cjec/?p=194>. Acesso em: 26 abr. 2014.

DEITEL, Paul J; DEITEL, Harvey M. **Ajax, rich internet applications e desenvolvimento web para programadores**. São Paulo: Pearson Prentice Hall, 2008.

ENGHOLM, Jr. H. **Engenharia de Software na Prática.** São Paulo: Novatec, 2010.

HAZAN, C. Análise de Pontos de Função: Uma Aplicação nas Estimativas de Tamanho de Projetos de Software. **Engenharia de Software Magazine**, 2. ed., 2008. Disponível em: <http://www.devmedia.com.br/revista-engenharia-de-software-2/9138>. Acesso em: 25 abr. 2014.

_____. Implementação de um Processo de Medições de Software. **Governo de Pernambuco,** 2002. Diponível em: < http://www.portaisgoverno.pe.gov.br/c/document_library/get_file?uuid=396bec67-5e04-4779-9614-10268e68dc12&groupId=335215>. Acesso em: 12 Nov. 2014.

HAZAN, Claudia; VON STAA, Arndt. **Análise e Melhoria de um Processo de Estimativas de Tamanho de Projetos de Software**. 2005. 31 f. Monografia (Especialização) - Curso de Ciência da Computação, Departamento de Informática, Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2005.

INTERNATIONAL FUNCTION POINT USERS GROUP. **About IFPUG**. 2010. Disponível em: <http://www.ifpug.org/?page_id=6>. Acessado em: 26 abr. 2014.

JUSTULIN JUNIOR, Paulo Roberto. **SOFTWARE PARA ESTIMATIVAS DE PROJETOS BASEADO EM ANÁLISE DE PONTOS DE FUNÇÃO**. 2013. 86 f. TCC (Graduação) - Curso de Ciência da Computação, Departamento de Informática, Universidade Sagrado CoraÇÃo, Bauru, 2013.

KOSCIANSKI, André; SOARES, Michel dos Santos. **Qualidade de software**: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software. 2. ed. São Paulo: Novatec, 2007.

MECENAS, Ivan. **APFplus-Análise de Pontos de Função**. 2013. Disponível em: <http://www.ivanmecenas.ecn.br/apf.htm>. Acesso em: 14 nov. 2014.

MARQUES, Daniela. **Métricas de software**. 2011. Disponível em: <https://www.ibm.com/developerworks/community/blogs/tlcbr/entry/metricas_de_software?lang=en>. Acesso em: 05 abr. 2014.

MACORATTI, J.C. Análise de pontos por função - o processo de contagem. **Macoratti.net**, c2010. Disponível em: <http://www.macoratti.net/apf_pcta.htm>. Acesso em: 01 maio 2014.

NOBREGA FILHO, José Martins da. **Projeto arquitetural**, 2009. Disponível em: <https://code.google.com/p/jogos-lotericos/wiki/projeto_arquitetural>. Acesso em: 22 abr. 2014.

PFLEEGER, S. L. **Engenharia de software**: Teoria e Prática. 2. ed. São Paulo: Prentice Hall, 2004

PRESSMAN, Roger S. **Engenharia de software**. Tradução de José Carlos Barbosa Dos Santos. 3. ed. São Paulo: Makron Books, 1995.

_____. **Engenharia de software**: Uma Abordagem Profissional. Tradução de Ariovaldo Griesi. 7. ed. Porto Alegre: AMGH, 2011.

_____. **Engenharia web**. Tradução de David Lowe. Rio de Janeiro: LTC, 2009.

SCHACH, Stephen R. **Engenharia de software**: Os Paradigmas Clássico Orientado a Objeto. Tradução de Ariovaldo Griese. 7. ed. São Paulo: Mcgraw Hill, 2009.

SCHNEIDER, Ricardo. **Fundamentos da engenharia de software**: Pesquisa sobre Métricas de Pontos de função, 2001. Disponível em: <http://www.dcc.ufrj.br/~schneide/es/2001/1/g04/FES.htm>. Acesso em: 26 abr. 2014.

SOMMERVILLE, Ian. **Engenharia de software**. Tradução de André Maurício de Andrade Ribeiro. 6. ed. São Paulo: Addison Wesley, 2003.

SOUZA, W. Produtividade das linguagens em pontos por função (APF). **Blog CMMI**, 2011. Disponível em: <http://www.blogcmmi.com.br/engenharia/produtividade-das-linguagens-em-pontos-por-funcao-apf>. Acesso em: 10 nov.2014.

STAA, Arndt Von. **Engenharia de programas**. Rio de Janeiro: LTC, 1983.

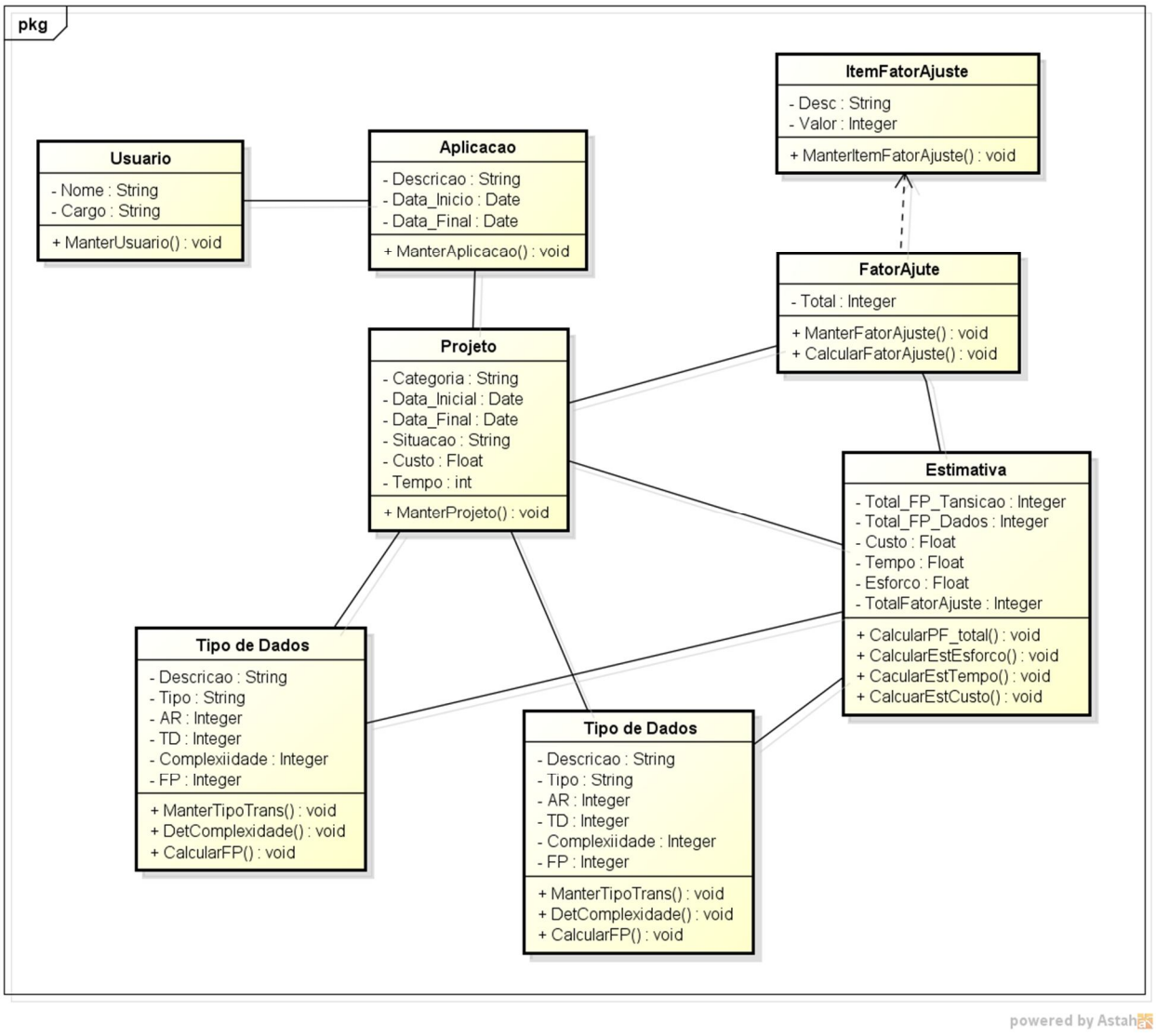
TONSIG, Sérgio Luiz. **Engenharia de software**: Análise e Projeto de Sistemas. 2. ed. Rio de Janeiro: Ciência Moderna Ltda, 2008.

VAZQUEZ, Carlos Eduardo; SIMÕES, Guilherme Siqueira; ALBERT, Renato Machado. **Análise de pontos de função**: Medição, Estimativa e Gerenciamento de Projetos de Software. 13. ed. São Paulo: Érica, 2013.

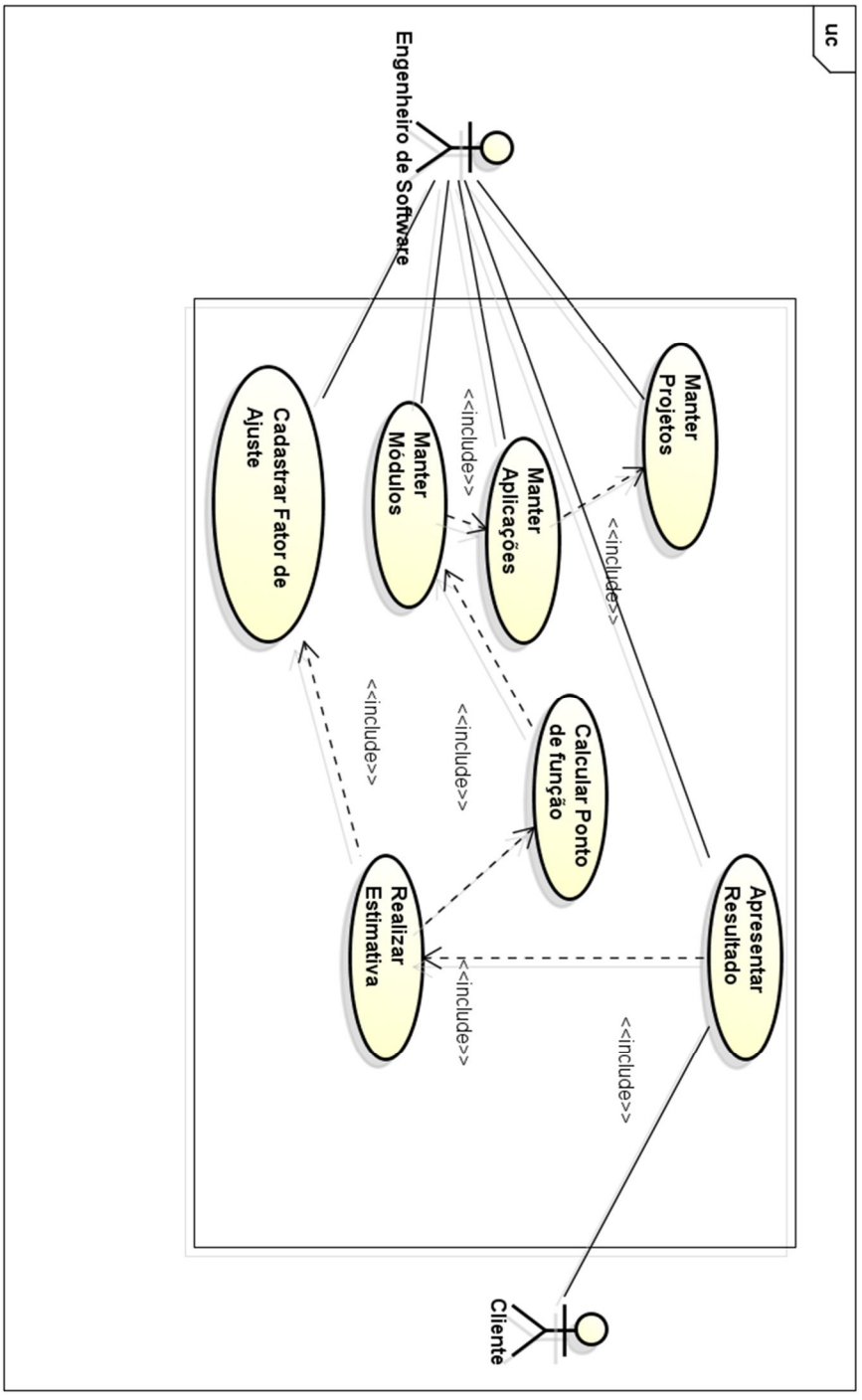
YOURDON, Edward. **Declínio e queda dos analistas e dos programadores**. Tradução de José Carlos Barbosa dos Santos. São Paulo: Makron Books, 1995.

WAZLAWICK, Raul Sidnei. **Engenharia de software**: Conceitos e Práticas. Rio de Janeiro: Elsevier, 2013.

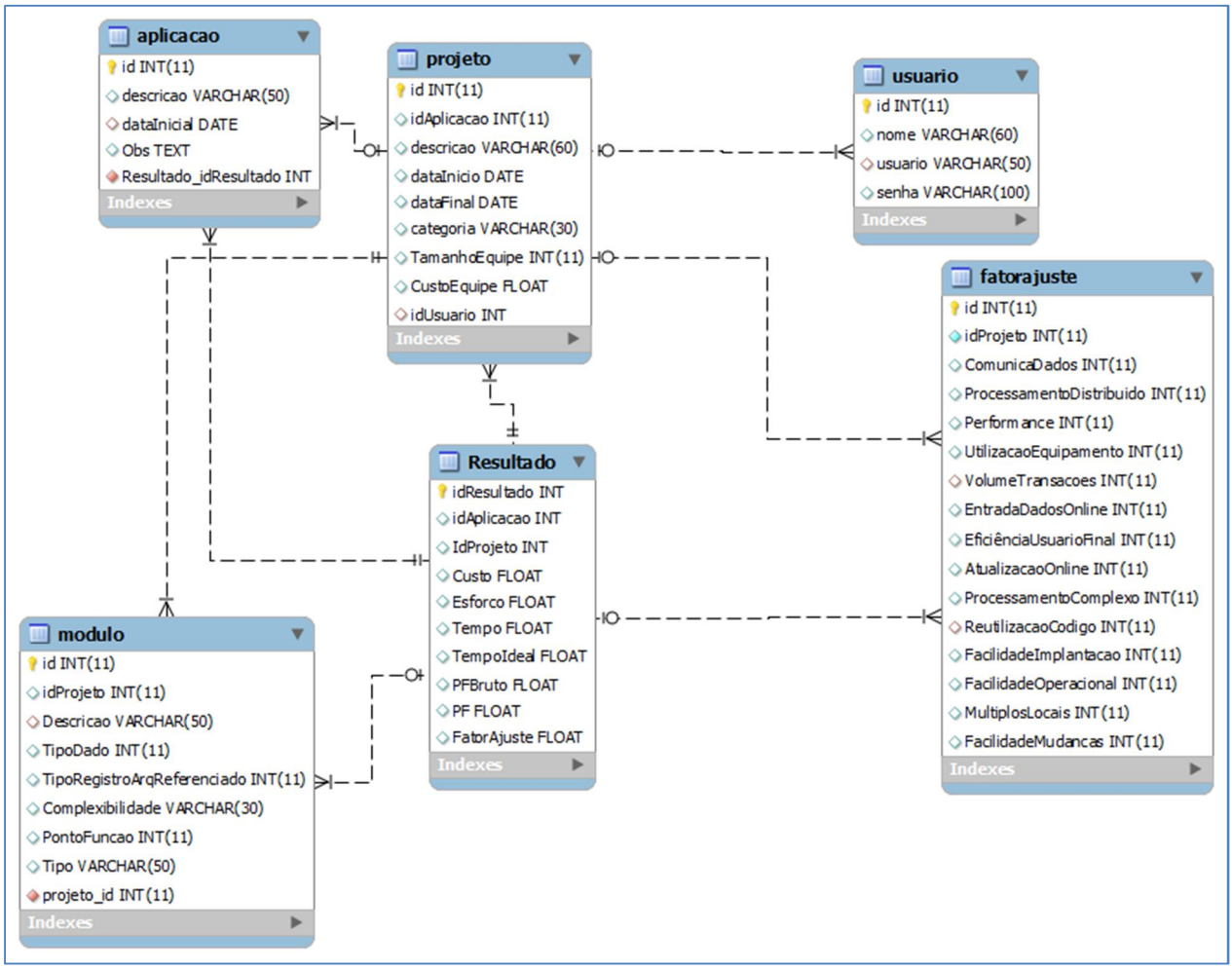
# APÊNDICE A – DIAGRAMA DE CLASSES



# APÊNDICE B – DIAGRAMA DE CASOS DE USO



# APÊNDICE C – DIAGRAMA ENTIDADE RELACIONAMENTO



# SOFTWARE WEB PARA ESTIMATIVAS DE PROJETOS BASEADAS EM ANÁLISE DE PONTOS DE FUNÇÃO

**Maicon Andre Vieira Carvalho.[1] , Elvio Gilberto da Silva.[1], Patrick Pedreira Silva.[1], Alex Setolin Beirigo [1].**

[1]Centro de Ciências Exatas e Sociais – Universidade do Sagrado Coração (USC) 17011-160 – Bauru – SP – Brasil.

maicon_carvalho@yahoo.com.br, egsilva@usc.br, patrick.silva@usc.br, setolin@gmail.com.

***Abstract.*** *The article is to create a prototype of a web metrics software for PHP and for estimates of system design, based on Function Point Analysis, we calculate the size of a project in Function Points, and present the necessary estimates that software engineer needs: size estimate, effort, schedule and cost . The prototype features a simple interface that enables users to reach your goal faster , saving projections of manual calculations . It also stores a history of all estimates from past projects , so that from these , has an indicator for future software becoming increasingly accurate estimate.*

***Resumo.*** *O Artigo consiste na criação de um protótipo de um software web em PHP para métricas e estimativas em projeto de sistema, baseado em Análise de Pontos de Função, que calcule o tamanho de um projeto em Pontos de Função, e apresente as estimativas necessárias que o engenheiro de software precisa: estimativa de tamanho, esforço, prazo e custo. O protótipo apresenta uma interface simples e que possibilite os usuários alcance seu objetivo mais rápido, poupando-se de projeções de cálculos manuais. Também armazena todo um histórico de estimativas de projetos passados, para que a partir destes, tenha-se um indicador para futuros softwares, tornando-se a estimativa cada vez mais exata.*

## 1. Introdução

O uso de sistemas de informação é essencial para todas as pessoas e empresas nos dias de hoje. Por isso, o planejamento e a gerencia do software é de grande importância para desenvolver um sistema com qualidade e com menos falhas.

Um sistema bem planejado e gerenciado tem mais qualidade e por isso atrai mais clientes. Para Koscianski e Soares (2007) a qualidade de um produto tem o propósito de satisfazer o cliente. Por isso a qualidade não é uma entidade abstrata e sim um objetivo concreto e por isso que cliente e desenvolvedores tem que sempre procurar a qualidade no seu produto.

Questões como preço e tempo de entrega do produto são duas variáveis que tem grande interferência na qualidade de um software, por isso o planejamento, estimativa de esforço de desenvolvimento e a métrica de software são fundamentais em um sistema.

Ao avaliar os recursos e o custo no desenvolvimento de um sistema é possível se estimar a quantidade de esforço e a quantidade e experiência da equipe alocada no projeto. Também pode ser monitorado o tamanho e estabilidade do projeto. (VAZQUEZ et al., 2013).

Existem várias maneiras de se medir um software, de acordo com Marques (2011), as métricas de software podem ser classificadas em medidas diretas e medidas indiretas. As medidas diretas são aquelas que representam atributos observáveis, tais como custo, esforço, número de linhas de código, tempo de execução e número de defeitos. Já as medidas indiretas são aquelas relacionadas com a funcionalidade, qualidade, complexidade e facilidade de manutenção do sistema.

APF é independente da linguagem de programação usada, tornando-se ideal para aplicações que usam linguagens convencionais e não procedimentais e se baseia em dados que são mais conhecidos no começo do projeto, tornando essa técnica mais atraente no momento da estimativa. (PRESSMAN, 1995).

De acordo com Vazquez et al. (2013) o escopo de ponto de função define quais funções serão incluídas na contagem, se ele abrangerá parte do sistema ou sistema inteiro. A APF pode medir todas as funcionalidades do sistema, apenas as funcionalidades usadas pelo cliente ou apenas algumas funções específicas. A APF mede o número entradas externas (EE), saídas externas (SE), consultas externas (CE), arquivos lógicos internos (ALI) e arquivos de interface externas (AIE).

Por isso, o cálculo do ponto de função exige um software de interface simples e que possibilite os usuários alcance seu objetivo mais rápido. Além disso, o sistema irá guarda informações de projetos passados para estimar o esforço de uma equipe no desenvolvimento do software futuros.

## 2. Software Web

No início do World Wide Web (WWW) os ‘websites’, consistiam em um conjunto de arquivos e hipertexto que apresentam informações de texto, imagens e gráficos bem limitados. Com passar do tempo, o Hypertext Markup Language (HTML) foi evoluindo com o uso de ferramentas e tecnologia de desenvolvimento como o Entensible Markup Language (XML), Java, Personal Home Page (PHP) e outras, que permitem a interação do computador do cliente com o servidor. As aplicações web se tornaram ferramentas sofisticadas que se integram com bancos de dados e outros sistemas de grande porte (PRESSMAN, 2009).

De acordo com Deitel, P. e Deitel, H. (2008) software web provê vários benefícios como redução de demandas no departamento interno, aumenta a acessibilidade do sistema fora do escritório e facilita a manutenção do software em grande escala. Em vez de ser instalado na máquina local, o sistema é instalado no servidor Web e é acessado aos clientes como um serviço de internet e as atualizações são diretos no servidor e manter atualizações uniforme em toda a organização.

Sistemas na web são ferramentas complexas e inovadoras, não precisam de instalação nem de arquivos de configuração, e não necessitam de espaço em disco. Além disso, reduz o custo da infraestrutura, pois é necessária somente internet para acessar o software e o aplicativo fica disponível em qualquer hora e local, desde que o usuário tenha conexão com a internet.

## 3. Engenharia de Software

A engenharia de software é uma área que tem como objetivo do desenvolvimento de um sistema sem falhas, entregue no prazo, dentro do orçamento previsto e que atenda às necessidades do cliente. Para atingir esses objetivos são necessárias técnicas apropriadas ao analisar o projeto, desenvolvimento e manutenção do sistema (SCHACH, 2008).

Para Pressman (1995) engenharia de software é um conjunto de etapas que envolvem métodos, ferramentas e procedimento conhecidos como paradigmas da engenharia de software. Esses paradigmas são escolhidos de acordo com a natureza do projeto. De forma detalhada os paradigmas da engenharia de software são os métodos, ferramentas e procedimentos.

## 4. Planejamento e Estimativa de Software

Para obter um sistema com qualidade, entrega na data prevista e com custos baixos é importante um bom planejamento. Identificar os processos, gerenciar prazos e custo, antecipar problemas e buscar soluções fazem parte de um planejamento de software adequado.

Estimativas são realizadas com base em métricas de software. As métricas têm como principal função reunir dados de desempenho do software e analisar os históricos dos projetos anteriores para utilizar essas informações nas previsões de projetos futuros.

## 5. Métricas

Segundo Fenton (1991 citado por PRESSMAN, 2011), medição e o processo pelo qual números e símbolos são atribuídos a entidades no mundo real para defini-los de acordo com regras claramente estabelecidas. Em várias áreas da ciência e da tecnologia é possível medir atributos consideráveis incomensuráveis e é claro que essas medidas não são tão refinadas como muitas feitas nas ciências físicas, mas “medir o incomensurável” é uma ferramenta para melhorar a compreensão de entidades particulares e de extrema importância na engenharia de software.

É de grande importância para empresa investir tempo e dinheiro em métricas de software, pois auxilia no gerenciamento do sistema. De acordo com Yourdon (1995), o argumento fundamental para a realização de métrica de software é aprender mais sobre o processo de desenvolvimento do sistema e como melhorá-la. Além disso, as métricas são necessárias para validar e ajustar modelos de produtividade e investir em ferramentas, técnicas metodologias e outras ferramentas que não podem ser justificados sem boas métricas.

## 6. Análises de Pontos de Função

Para Pressman (2011), a métrica da APF pode ser usada efetivamente como meio para medir a funcionalidade fornecida por um sistema. Através de dados históricos, a APF pode ser empregada para estimar o custo e o esforço necessário para projetar, codificar e testar o software. Além, de prever os números de erros que serão encontrados durante a fase de teste e prever os componentes e o número de linhas projetadas de código fonte no sistema projetado.

A APF mede especificadamente os requisitos funcionais do usuário, e essa dimensão é dada o nome de tamanho funcional. Essa medição permite expressar quantitativamente

qualidades e funções do sistema. O diagrama da Figura 4 representa as etapas do processo de medição funcional, bem como as relações de interdependência entre seus passos (VAZQUEZ et al., 2013).

Basicamente a medição consiste em decompor o projeto em elementos chamados componentes funcionais básicos ou funções. O método de medição conceitua abstrações, os tipos de função, nos quais os componentes básicos são classificados. Essa classificação é feita conforme sua função de dados ou transação.

Vazquez et al. (2013) explicam que as funções do tipo dado representam as funcionalidades fornecidas pelo sistema ao usuário com o objetivo de atender as suas necessidades de dados internos e externos, ou seja, mostram os dados armazenados do software. São classificados como Arquivos Lógicos Internos (ALI) e Arquivos de Interface Externa (AIE).

Os autores ainda afirmam para calcular os pontos e função é preciso avaliar a complexidade funcional definida pelo número de Tipos de Dados (TD) e os Tipos de Registros (TR) e faz uma classificação com relação à complexidade fornecida pela tabela ilustrada na Figura 1.

| Tipos de Registros | Tipos de Dados | | | |
|---|---|---|---|---|
| | | < 20 | 20 – 50 | > 50 |
| | 1 | Baixa | Baixa | Média |
| | 2 – 5 | Baixa | Média | Alta |
| | > 5 | Média | Alta | Alta |

**Figura 1 -** Complexidade funcional dos ALI e AIE.
Fonte: Vazquez et al. (2013).

As funções de transação representam a lógica em que os processamentos podem ser executados. A lógica de processamento é definida como qualquer dos seguintes requisitos especificadamente solicitados pelo usuário para completar um processo elementar.

De acordo com Vazquez et al. (2013) cada EE, SE e CE deve ser classificado com relação à sua complexidade baseado no número de TD e de Arquivos referenciados (AR), conforme as tabelas de complexidade, ilustrada pelas Figuras 2 e 3.

| Arquivos Referenciados | Tipos de Dados | | | |
|---|---|---|---|---|
| | | < 5 | 5 – 15 | > 15 |
| | < 2 | Baixa | Baixa | Média |
| | 2 | Baixa | Média | Alta |
| | > 2 | Média | Alta | Alta |

**Figura 2** - Complexidade para entradas externas (EEs).
Fonte: Vazquez et al. (2013).

| Arquivos Referenciados | Tipos de Dados | | | |
|---|---|---|---|---|
| | | < 6 | 6 – 19 | > 19 |
| | < 2 | Baixa | Baixa | Média |
| | 2 | Baixa | Média | Alta |
| | > 2 | Média | Alta | Alta |

**Figura 3** - Complexidade para saídas externas (SEs) e consultas externas (CEs).
Fonte: Vazquez et al. (2013).

Após a identificação e classificação de todas as funções de dados e de transação na contagem, o número de pontos de função será simplesmente a soma do peso de cada uma dessas funções. (VAZQUEZ et al., 2013).

Para IFPUG (1999 citado por VAZQUEZ, 2005), após definir a fronteira da aplicação, o tipo de contagem e reconhecer as funções de dados e de transação podem se calcular os pontos de função não ajustados, ou brutos, multiplicando-se o total de ALI, AIE, EE, SE, e CE pela respectiva complexidade conforme a tabela da Figura 4

| Descrição do | Complexidade | | |
|---|---|---|---|
| Tipo Funcional | Simples | Média | Complexa |
| Arquivo Lógico Interno (ALI) | 7 PFs | 10 PFs | 15 PFs |
| Arquivo de Interface Externa (AIE) | 5 PFs | 7 PFs | 10 PFs |
| Entrada Externa (EE) | 3 PFs | 4 PFs | 6 PFs |
| Saída Externa (SE) | 4 PFs | 5 PFs | 7 PFs |
| Consulta Externa (CE) | 3 PFs | 4 PFs | **6 PFs** |

**Figura 4** - Complexidade para saídas externas (SEs) e consultas externas (CEs).
Fonte: Hazan (2008).

O fator de ajuste, segundo o IFPUG é opcional na aplicação da técnica de APF. O propósito do fator de ajuste é medir requisitos gerais da aplicação (não funcionais), ele ajusta os pontos de função em mais ou menos 35% de acordo com a influência de 14 características gerais. (VAZQUEZ et al., 2013):

- 01 - Comunicação de dados
- 02 - Processamento distribuído
- 03 – Performance
- 04 - Utilização de Equipamento
- 05 - Volume de transações
- 06 - Entrada de dados on-line
- 07 - Eficiência do Usuário Final
- 08 - Atualização On-Line

- 09 - Processamento complexo
- 10 - Reutilização de código
- 11 - Facilidade de Implantação
- 12 - Facilidade Operacional.
- 13 - Múltiplos Locais.
- 14 - Facilidade de mudanças.

De acordo Vazquez et al. (2013) as funções de tipo de dados definem requisitos específicos de armazenamento e as funções de transação refletem requisitos de processamento, já as características gerais refletem funções que afetam a aplicação de maneira geral. Essas características possuem nível de influência em uma escala que varia de 0 a 5.

Após apurar-se o nível de influência de todas as Características gerais do sistema, o VFA é baseado na equação: VFA = 0,65 + (Soma das Características Gerais do Sistema x 0,01).

## 7 Metodologia

O desenvolvimento desse projeto inicialmente é uma pesquisa exploratória, com a seleção de material bibliográfico adequado, entre eles, livros, artigos e trabalhos relacionados.

Em seguida, foram realizados estudos sobre engenharia de software, qualidade de sistemas, planejamento e estimativas de projetos e métricas de software, No estudo realizado, foi dado o maior destaque na medição por APF, incluindo seu conceito, características e descrição da mesma

A pesquisa realizada tem como objetivo a construção de um protótipo de um sistema para estimativa utilizando a técnica de ponto por função.

No primeiro passo do projeto será realizado a modelagem do sistema utilizando a linguagem UML (Unified Modeling Language). A UML é uma linguagem gráfica padrão para a elaboração da estrutura de projetos complexos de software e pode ser empregada para visualizar, especificar, construir e documentar os artefatos de um sistema.

O Sistema de Gerenciamento do Banco de Dados (SGBD) escolhido para armazenar as informações do projeto será o MySQL, pois é um servidor rápido, de multiprocessamento e multiusuários em SQL (Structured Query Language), muito confiável.

O protótipo deste trabalho é destinado para web e será desenvolvido na linguagem de marcação HTML e a linguagem de programação PHP.

## 8 Resultados

Primeiramente foi desenvolvida as modelagens de classe e de caso de uso, elas buscam representar as classes do sistema e como ele se interage com o usuário de forma de fácil entendimento para o leitor, utilizando de variáveis claras e que possibilitam extrair os requisitos iniciais do software.

No sistema, o usuário se encontrará num ambiente bem dinâmico e de fácil aprendizagem, no qual o usuário precise de uma menor quantidade de esforço para acessar determinado resultado.

Para testar o software foram simuladas as estimativas com o mesmo software desenvolvido neste trabalho. Primeiramente foram cadastradas informações sobre a aplicação e o projeto.

De posse dessas informações, já se pode iniciar os cálculos das métricas. O próximo passo seria obter o valor do Fator de Ajuste, na qual se devem responder as 14 perguntas referentes ao sistema com pesos de 0 a 5.

A tela de cadastro de modulo é onde são cadastradas as funcionalidades do sistema. Cada modulo pode ser classificado por cincos tipos de funções: Entradas externas, saídas externas, consultas externas, arquivos lógicos internos e arquivos de interface externas.

Após clicar no botão de inclusão, o sistema fará o cálculo de complexidade e contribuição de pontos de função, a somatória das contribuições de pontos de função de todos os módulos do sistema irá resultar nos pontos de função bruto.

Analisando as funções, é possível determinar o tipo de contagem, sua complexidade e o total de pontos de função, o total de pontos de função bruto das funções do sistema é 155.

Como será o primeiro projeto e não há bases históricas, será utilizada a produtividade de 5 horas para produzir um ponto de função.

Portanto, os pontos de função ajustados são calculados com a multiplicação do VFA com os pontos de função brutos que tem como resultado 190, 65 conforme a tela ilustrada na Figura 5.


**Figura 5** - Tela de Resultado
Fonte: Desenvolvido pelo autor (2014).

Para Hazan (2008), o cálculo do esforço se dá ao produto entre o valor obtido da estimativa de tamanho com o número de horas para produzir um PF. O resultado será o esforço em HH (homens_hora).

O cálculo do custo depende de muitos fatores. Neste projeto, foi utilizado apenas o custo médio da equipe que foi cadastrado na tela de projetos conforme Figura 26 multiplicado pelo esforço utilizado no desenvolvimento do projeto.

Para se chegar ao prazo, utilizou-se a seguinte formula: Esforço (horas) / Tamanho da equipe * 6 horas. (HAZAN, 2002). Essa constante de 6 horas refere-se à produtividade média diária no Brasil (JONES 1997 citado por HAZAN, 2008). O resultado é dado em dias úteis. Para transformar em meses, basta apenas dividir o resultado por 30.

E, finalmente, para se chegar ao prazo ideal, utiliza-se fórmula de Caper Jones (1998 citada por HAZAN, 2008), onde consiste na formula (Td (meses) = V ^ t).

Onde:

- Td é o tempo ótimo de desenvolvimento, em meses.
- V é o volume em Pontos de Função
- t é um expoente que depende do ambiente computacional considerado.

No caso deste projeto, o ambiente escolhido foi o "sistema cliente-servidor", na qual condiz com a arquitetura web do software.

Esse resultado é armazenado na base de dados para que o engenheiro de software tenha uma base para cadastrar novos projetos.

## 9 Considerações Finais

O sistema cumpriu com tudo que foi proposto inicialmente, conforme demonstrado nos resultados do software. Além de realizar as estimativas do projeto de forma clara e objetiva, a ferramenta realizou os cálculos de maneira rápida, segura e por ser um sistema web, possibilita ao usuário acessá-lo de qualquer lugar, desde que tenha um ponto de acesso à internet.

O prazo estimado no software foi de 5 meses, esse resultado ficou muito próximo a realidade, pois o início do projeto começou em torno de junho de 2014 até o termino no final de novembro de 2014.

Como recomendações para trabalhos futuros, sugere-se a análise de métodos de aprendizado de máquina com técnicas em inteligência artificial para o tratamento dos dados históricos de produtividade para as estimativas de custo e de esforço.

Outro fator que pode ser utilizado em trabalhos futuros é possibilitar a ferramenta o cálculo de outras métricas, como por exemplo a métrica de caso de uso para que o usuário tenha acesso a mais de um tipo de métrica para que sirva como comparativo.

Embora a ferramenta não consiga acertar com exatidão o tamanho do projeto. Ele é de grande importância, pois, possibilita o engenheiro de software tenha uma estimativa do tamanho, prazo e custo do sistema no início do projeto possibilitando o usuário um maior controle do sistema a ser desenvolvido e uma justificativa para contratação de novos membros para a equipe, compra de novas tecnologias de desenvolvimento e treinamento dos seus profissionais  Além

disso o projeto contém um do histórico de produtividade e experiências passadas, que são de suma importância para avaliar projetos futuros.

## Referências


Brazilian Function Point Users Group. O que é o BFPUG - Brazilian Function Point Users Group. **BFPUG**. Disponível em: <http://www.bfpug.com.br/>. Acessado em: 27 abr. 2014.

DEITEL, Paul J; DEITEL, Harvey M. **Ajax, rich internet applications e desenvolvimento web para programadores**. São Paulo: Pearson Prentice Hall, 2008.

HAZAN, C. Análise de Pontos de Função: Uma Aplicação nas Estimativas de Tamanho de Projetos de Software. **Engenharia de Software Magazine**, 2. ed., 2008. Disponível em: <http://www.devmedia.com.br/revista-engenharia-de-software-2/9138>. Acesso em: 25 abr. 2014.

HAZAN, C. Implementação de um Processo de Medições de Software. **Governo de Pernambuco,** 2002. Diponível em: < http://www.portaisgoverno.pe.gov.br/c/document_library/get_file?uuid=396bec67-5e04-4779-9614-10268e68dc12&groupId=335215>. Acesso em: 12 Nov. 2014.

KOSCIANSKI, André; SOARES, Michel dos Santos. **Qualidade de software**: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software. 2. ed. São Paulo: Novatec, 2007.

MARQUES, Daniela. **Métricas de software**. 2011. Disponível em: <https://www.ibm.com/developerworks/community/blogs/tlcbr/entry/metricas_de_software?lang=en>. Acesso em: 05 abr. 2014.

MACORATTI, J.C. Análise de pontos por função - o processo de contagem. **Macoratti.net**, c2010. Disponível em: <http://www.macoratti.net/apf_pcta.htm>. Acesso em: 01 maio 2014.

PRESSMAN, Roger S. **Engenharia de software**. Tradução de José Carlos Barbosa Dos Santos. 3. ed. São Paulo: Makron Books, 1995.

_____. **Engenharia de software**: Uma Abordagem Profissional. Tradução de Ariovaldo Griesi. 7. ed. Porto Alegre: AMGH, 2011.

_____. **Engenharia web**. Tradução de David Lowe. Rio de Janeiro: LTC, 2009.

SCHACH, Stephen R. **Engenharia de software**: Os Paradigmas Clássico Orientado a Objeto. Tradução de Ariovaldo Griese. 7. ed. São Paulo: Mcgraw Hill, 2009.

VAZQUEZ, Carlos Eduardo; SIMÕES, Guilherme Siqueira; ALBERT, Renato Machado. **Análise de pontos de função**: Medição, Estimativa e Gerenciamento de Projetos de Software. 13. ed. São Paulo: Érica, 2013.

YOURDON, Edward. **Declínio e queda dos analistas e dos programadores**. Tradução de José Carlos Barbosa dos Santos. São Paulo: Makron Books, 1995.