Análise e Melhorias no processo de estimativas de tempo no desenvolvimento de software

Luis Miguel Guellen [1], Alexandre Lazaretti Zanatta [2]

Ciência da Computação – Universidade de Passo Fundo (UPF)
Caixa Postal 611 – 99.052-900 – Passo Fundo – RS – Brasil

(1) 64349@upf.br; (2) zanatta@upf.br

Resumo. *O presente artigo trata da importância de medir o tempo gasto no desenvolvimento de um software. Serão objetos de estudo algumas técnicas para métricas de software e dentre as técnicas estudadas será proposta a aplicação da técnica de análise de pontos por função como uma ferramenta de auxílio para a estimativa de tempo para desenvolvimento de um novo código fonte. O processo de estimativas de tempo pode ser mais produtivo e confiável com a aplicação da metodologia apresentada no trabalho.*

1. Introdução

Atualmente, uma grande parte da população mundial depende de aplicações de software para realizar suas atividades diárias, contudo, cada vez mais as empresas de desenvolvimento de software devem se preocupar com o tempo gasto para desenvolvimento e também o custo que o mesmo irá representar para a organização. As questões de tempo e custo não são simples de serem resolvidas, pois alguns parâmetros são levados em consideração, como por exemplo, a experiência da equipe na tecnologia exigida, complexidade do software, metodologia de desenvolvimento, entre outros fatores. Na grande maioria dos projetos de software, acabam falhando nas estimativas de tempo, por isso, tornando-se necessário uma metodologia para medir o tempo gasto no desenvolvimento do software.

Pode-se estimar e medir software pelos seguintes motivos:

- ⇒ Indicar a qualidade do produto.
- ⇒ Avaliar a produtividade das pessoas.
- ⇒ Avaliar os benefícios derivados de novos métodos e ferramentas.
- ⇒ Formar uma linha básica para estimativas.
- ⇒ Ajudar a justificar o pedido de novas ferramentas e treinamentos.

O objetivo do trabalho é analisar e propor melhorias no processo de estimativas de tempo no desenvolvimento de um novo código fonte. Para que isso ocorra será apresentado um estudo sobre análise de pontos de função, que é uma técnica de métricas de software, esta técnica será uma ferramenta de auxílio no estudo proposto. Além do estudo da técnica será proposta uma tabela de conversão dos pontos de função em tempo na linguagem de programação Progress[1].

A linguagem de programação Progress é uma linguagem proprietária que é mantida hoje pela Progress Software. A linguagem Progress é uma linguagem com marca registrada.

O presente trabalho está organizado da seguinte forma. Na seção 2 é apresentado um estudo sobre as métricas de software, e suas classificações. Na mesma seção realiza-se um estudo das técnicas para métricas de software, que dá um panorama das técnicas mais utilizadas nos dias atuais e também de que forma são medidos os softwares. Na seção 3 o trabalho traz a metodologia adotada para o presente trabalho, os fatores que auxiliaram na escolha da técnica, o processo de contagem do tamanho do software, e ainda como se faz o cálculo para chegar no tempo que vai ser gasto para desenvolver um novo código fonte. Na seção 4 do trabalho são apresentados os resultados obtidos no trabalho em forma de gráficos comparativos entre as atividades sem o estudo implementado e com o estudo implementado. Por fim na seção 5 do presente trabalho é apresentada a conclusão, onde traz sugestões para novos trabalhos e também uma analise dos objetivos alcançados com o desenvolver do estudo.

2. Métricas de Software

As estimativas são realizadas com base em métricas. Métricas de tamanho, duração, produtividade e esforço são as mais utilizadas nos dias atuais (KAN, 1995). As estimativas podem ser feitas com base em duas estratégias (KAN, 1995):

- ⇒ Métricas com base no histórico da organização: Obtidas a partir de experiências da equipe.
- ⇒ Métricas estatísticas de diferentes organizações: Obtidas a partir de diferentes equipes e organizações.

De acordo com Tom Demarco (DEMARCO, 1991) as duas principais maneiras de estimar o tempo do desenvolvimento de um software são;

- ⇒ Por analogia – As estimativas de tamanho do projeto atual são baseadas em estimativas já realizadas em desenvolvimentos anteriores.
- ⇒ Realizando medições das características do produto e usando uma metodologia e algoritmo para converter a medição em uma estimativa de tempo.

Além de um controle detalhado do tempo para o desenvolvimento do produto as métricas também possibilitam avaliar a qualidade do produto, avaliar a produtividade da equipe, avaliar métodos e ferramentas utilizadas para o desenvolvimento do produto, realizar estimativas para futuros projetos.

Uma definição sobre medição é dada como sendo o “uso de uma métrica para se obter um valor (o qual pode ser numérico ou categoria), obtido por meio de uma escala, a um atributo de uma entidade” (NBR ISSO/IEC 9126-1).

Observa-se então que as métricas são cálculos matemáticos que podem ser implementados para se saber o tamanho ou até mesmo a complexidade do software. “Nenhuma investigação humana pode realmente ser chamada de ciência se não pode ser demonstrada matematicamente” (DA VINCI apud KOSCIANSKI; SOARES, 2006, p. 26). Além de cálculos matemáticos as métricas de software também tem objetivos de dar um suporte adequado aos desenvolvedores de software, permitindo a eles uma melhor administração dos projetos e também o ciclo de vida da engenharia do software.

“Uma métrica de software é qualquer tipo de medição que se refira a um sistema de software, processo ou documentação relacionada” (SOMMERVILLE, 2003, P. 468).

2.1 Classificações das Métricas de Software

A classificação das métricas de software é estabelecida por uma união dos conceitos de autores como: Koscianski e Soares, Pressman e Kan, Sommerville, visto que existem métricas para todo o ciclo de vida do software e assim tendo diversas classificações para as mesmas métricas. Dentre os conceitos em comum observam-se quatro métricas propostas:

- ⇒ Métricas de Software por caráter qualitativo.
- ⇒ Métricas de Software por caráter quantitativo.
- ⇒ Métricas de Software para processos.
- ⇒ Métricas de Software para produto.

As métricas de software por caráter qualitativo na visão de Sommervile (2003, p. 466), são configuradas pela segurança, proteção, confiabilidade, robustez entre outros atributos da qualidade de software. Koscianski; Soares (2006, p. 63) relatam que as características qualitativas estão associadas a uma noção de intensidade com muito ou pouco, e com adjetivo como agradável.

Para obter informações a respeito de algum dos aspectos que indiquem a qualidade dos softwares (tais como, facilidade de manutenção, confiabilidade, portabilidade, e facilidade de uso) é um dos objetivos ao se utilizar métricas de software. O uso de métricas qualitativas pode ser justificado pelo fato de existirem atributos com formato qualitativo que não podem ser representados por atributos internos do software e desta maneira é necessário à realização de comparações de diversas opiniões para se chegar a um consenso sobre o aspecto qualitativo.

As métricas de software por caráter quantitativo surgiram como alternativa a subjetividade das métricas qualitativas. A necessidade da determinação da qualidade de uma forma precisa tornou-se o objetivo de engenheiros e desenvolvedores de software, motivando o surgimento de diversas métricas para software não dependentes de subjetividade e julgamentos pessoais, mas sim baseados em dados obtidos através da aplicação de métricas.

Atributos internos de software são dados a respeito do software, provenientes do projeto e de documentos, ou do próprio software (codificação e funcionamento). O número de linhas de código de um software é o exemplo de uma métrica quantitativa do software. Existem inúmeras possibilidades de atributos internos, porém esses devem ser estudados para que representem informações úteis relacionadas a algum aspecto quantitativo do software.

Segundo Kan (1995, p. 83) o uso de métricas para processos de software podem ser usadas para auxiliar no desenvolvimento e manutenção dos processos de software. Segundo Pressman (2002, p. 506) as métricas são conduzidas durante os primeiros estágios do processo de software, onde fornecem aos engenheiros de software um mecanismo consistente e objetivo para avaliar a qualidade do software em desenvolvimento. Já para Sommerville (2003, p. 477) a melhoria de processo é uma

prática iterativa e de longo prazo e que deve ser apoiada por meio de métricas de processos que auxiliem os gerentes fornecendo dados quantitativos que demonstrem as variações existentes nas características de um processo para que o mesmo possa ser aperfeiçoado.

As métricas de software para processo têm por objetivo auxiliar no controle do gerenciamento e da evolução dos processos de software através de dados quantitativos possibilitando um nível de qualidade mensurável.

O uso de métricas de software para produto segundo Kan (1995, p. 83) são as que descrevem características dos produtos tais como, complexidade, características de design, performance e nível de qualidade. Todos estes dados estão totalmente relacionados com as características de qualidade do software, como por exemplo: densidade de defeitos, tempo médio para falhar entre outros. O objetivo das métricas de software para produto é avaliar o produto, podendo ser de caráter qualitativo ou quantitativo.

2.2 Técnicas para métricas de software

As técnicas para métricas de software são um conjunto de regras que servem para medir o tamanho do software, lembrando sempre que estas técnicas de medição independem da linguagem de programação utilizada pela organização.

As técnicas para métricas software abordadas no seguinte trabalho são:

- ⇒ *Lines of Code* (LOC).
- ⇒ Pontos de Função.
- ⇒ Complexidade Ciclomática.

2.2.1 Lines of Code (LOC)

É uma técnica baseada na contagem de linhas de código de um programa, podendo este ser a contagem de todos os programas de software ou então de um programa ou então apenas de um módulo de programas de software. A contagem das linhas de código representa o tamanho e a complexidade do programa e isso, segundo Kan (1995, p. 254) é um indício que pode influenciar na qualidade do software, pois, quanto mais linhas de código um programa possui mais defeitos são esperados. Koscianki e Soares (2006, p. 229) citam três motivos pelos quais as medidas de linhas de código são utilizadas:

- ⇒ "Está entre as métricas de aplicação mais simples" (KOSCIANSKI; SOARES, 2006, p. 229).
- ⇒ "É de fácil interpretação quando não houver muita precisão em jogo" (KOSCIANSKI; SOARES, 2006, p. 229).
- ⇒ "São de aplicação barata" (KOSCIANSKI; SOARES, 2006, p. 229).

A técnica, porém, tem um ponto negativo, trata-se da imprecisão na contagem das linhas, pois ela considera as linhas em branco e também os comentários que o programa apresenta.

As métricas de software baseadas na medição do número de linhas de código revelam certa imprecisão que pode ser atenuada com a utilização do SLOC (técnica onde os comentários e as linhas em branco não são considerados na contagem). Essas métricas são classificadas como métricas de caráter quantitativo para produtos de software, e podem ajudar a revelar dados qualitativos como complexidade do software e facilidade de compreensão, bem como, auxiliar na previsibilidade de defeitos no software através da relação KLOC (termo em inglês utilizado para representar milhares de linhas de código) e densidade de defeitos encontrados.

2.2.2 Pontos de Função

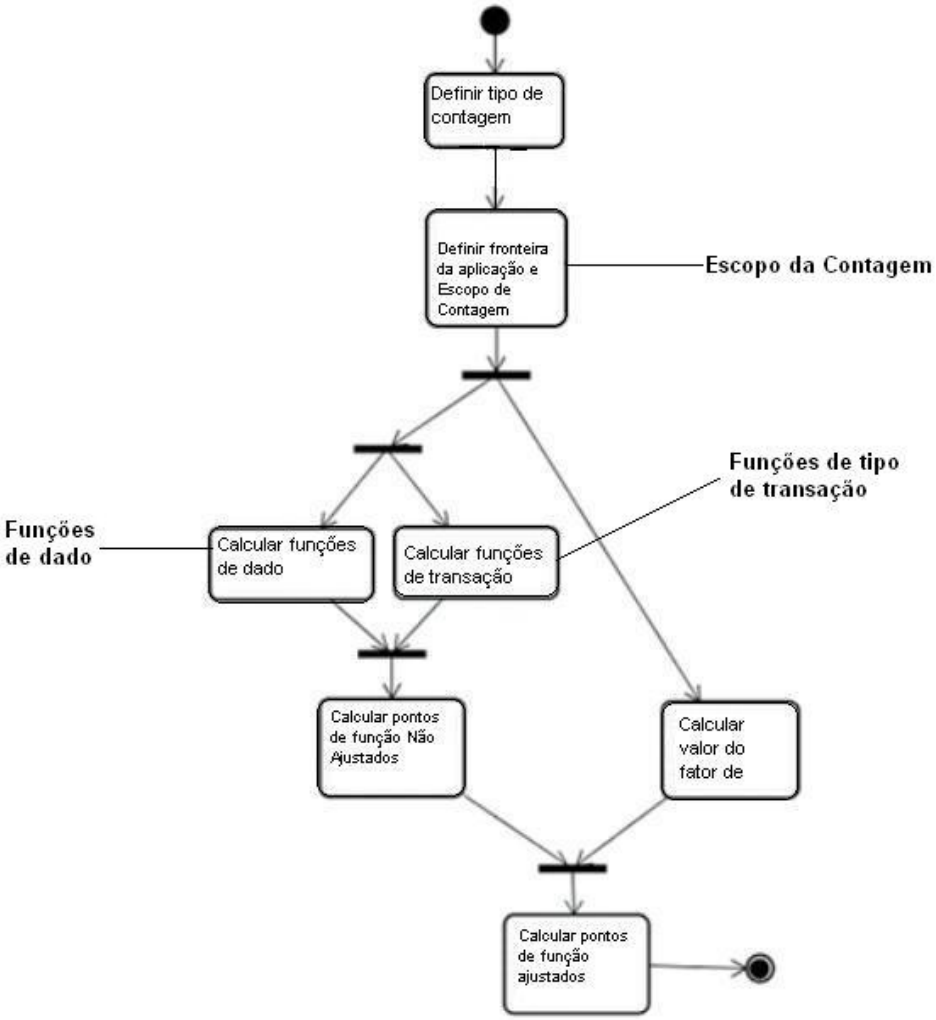
A técnica de APF (Análise de Pontos de Função) tem como objetivo obter a medida funcional de tamanho do software. A APF surgiu na IBM, no início da década de 1970, como uma alternativa para métricas baseadas em linhas de código. Os conceitos desta técnica foram introduzidos por Allan J. Albrecth, em uma conferência do *GUIDE* – Grupo de Usuários IBM, em 1979. A técnica foi refinada por Albert em 1984, e, a partir desta data houve um aumento considerável na sua utilização, segundo Vasquez (2007, p. 36), o que levou a necessidade de definir um padrão para aplicação da técnica. Com este objetivo foi criado em 1986 o *International Function Point Users Group* (IFPUG) que passou a ser responsável pela definição das regras de contagem, treinamento e certificação dos usuários da técnica. Em 1990 foi lançada a primeira versão do Manual de Práticas de Contagem ou *CPM – Counting Practices Manual* com o objetivo de padronizar a técnica. (VAZQUEZ,2005). O uso da técnica de analise de pontos de função do Brasil começou significativamente no início da década de 1990, com um forte apoio de uma grande empresa Unisys. Então em 1998 foi constituído o BFPUG (*Brazilian Function Point User Group*) que é o órgão certificados e APF no Brasil, além de ser o órgão certificador ele promove a troca de experiências entre os profissionais da área.

Esta técnica mede as funcionalidades fornecidas por um software do ponto de vista de seu usuário. Ponto de função é a unidade de medida desta técnica que tem por objetivo tornar a medição independente da tecnologia utilizada para a construção do software. A Análise de pontos de função busca medir o que o software faz, e não como ele foi construído. O processo de medição, ou a contagem de pontos de função são baseados em uma avaliação padronizada dos requisitos lógicos do usuário. Os requisitos lógicos do usuário, na técnica de análise de pontos de função, referem-se aos requisitos ‘conceituais’ ou ‘funcionais’ dos usuários, excluindo a implementação física ou os requisitos de projeto. Os requisitos lógicos dos usuários são aqueles que um usuário experiente no assunto identificaria como requisitos do software. Empiricamente as principais técnicas de estimativa de projetos de desenvolvimento de software assumem que o tamanho de um software é um vetor importante para a determinação do esforço para a construção. Portanto, saber o seu tamanho é um dos primeiros passos do processo de estimativa de esforço, prazo e custo. É importante destacar que pontos de função não medem diretamente esforço, produtividade ou custo, porém a partir da contagem dos pontos de função em conjunto com outras variáveis é que poderá ser usado para derivar o tempo que será gasto para desenvolver um novo código fonte.

Na técnica de APF é possível estabelecer uma estimativa da quantidade de pontos de função que um programa de software pode ter e baseando-se nos requisitos

levantados para o programa de software é possível estabelecer um processo de contagem.

Primeiramente no processo de APF é preciso estabelecer o processo de contagem, a figura 1 demonstra o fluxo deste processo.



Fonte: (VAZQUEZ; SIMÕES; ALBERT, 2007).

Figura 1 – O processo de contagem

A figura 1 mostra o fluxo da contagem dos pontos de função, e cada etapa do fluxograma será detalhada a seguir:

⇒ Escopo da Contagem: É a fronteira da aplicação. Define as funções que serão incluídas em uma determinada contagem de pontos de função.

⇒ Funções de Dado: Consiste na contagem dos tipos de dados utilizados nas funcionalidades a serem desenvolvidas. Esses tipos de dados são agrupados em um arquivo lógico interno (ALI) que representa grupos de dados relacionados e reconhecidos pelo usuário, dentro da fronteira da aplicação. Um exemplo são as tabelas do banco de dados utilizadas pela aplicação; ou um Arquivo de Interface Externa (AIE) que representa dados referenciados pela aplicação, mas mantidos

dentro da fronteira de outra aplicação. Um exemplo pode ser as tabelas de banco de dados lidas pela aplicação, mas atualizadas por outra aplicação.

Na figura 2 é demonstrando a complexidade funcional dos ALI e AIE em forma de uma tabela, onde o número de registros de tipos de dados e tipo de registros forma a complexidade da tabela.

Tipos de Registro(TR)	Tipos de dados (TD)		
	Abaixo de 20	20 a 50	Acima de 50
1	Baixa	Baixa	Baixa
2 a 5	Baixa	Média	Alta
Acima de 5	Média	Alta	Alta

Fonte: (VAZQUEZ; SIMÕES; ALBERT, 2007).

Figura 2 – Tabela de complexidade dos ALI e AIE.

As funções de tipo de transação representam os requisitos de processamento fornecidos pelo sistema ao usuário. Segundo Koscianski e Soares são as seguintes:

⇒ Entradas externas (EE): um processo lógico em que dados são introduzidos no sistema. Os dados podem ser informações de controle ou de negócios. As entradas externas representam o fluxo de informações que adentra o sistema. Exemplos de EE é a exclusão, alteração e inclusão de registros.

⇒ Saídas externas (EE): um processo lógico em que dados são enviados ao exterior das fronteiras do sistema – por exemplo, na emissão de relatórios ou mesmo na apresentação de dados na tela. Tais dados são computados a partir de arquivos lógicos internos e arquivos de interface externos.

⇒ Consulta externa (CE): um processo lógico que envolve um par consulta-resposta. Os dados são recuperados exclusivamente de arquivos internos e interfaces externas (nada é computado). Nenhum Arquivo Lógico Interno é alterado nesse processo. São exemplos de CE as consultas ao cadastro de clientes. (KOSCIANSKI; SOARES, 2006 p. 231).

O manual de APF que é estabelecido pela IFPUG conceitua como deve ser a contagem de pontos por função e a complexidade dos números dos pontos por função. Nas figuras 3 e 4, estão demonstradas a complexidade de entrada externa e a complexidade das saídas externas e consultas externas.

Arquivos Referenciados (AR)	Tipos de Dados (TD)		
	Abaixo de 5	5 a 15	Acima de 15
0 ou 1	Baixa	Baixa	Baixa
2	Baixa	Média	Alta
Acima de 2	Média	Alta	Alta

Fonte: (VAZQUEZ; SIMÕES; ALBERT, 2007).

Figura 3 – Tabela de complexidade de Entrada externa (EE)

Na figura 3 são apresentadas as entradas externas no processo de contagem dos pontos de função, ela representa o processamento dos dados ou informações de controle recebido de fora da fronteira da aplicação. Um exemplo de entradas externas são as transações que recebem dados externos utilizados na manutenção de arquivos lógicos internos.

Arquivos Referenciados (AR)	Tipos de dados (TD)		
	Abaixo de 6	6 a 19	Acima de 19
0 ou 1	Baixa	Baixa	Média
2 ou 3	Baixa	Média	Alta
Acima de 3	Média	Alta	Alta

Fonte: (VAZQUEZ; SIMÕES; ALBERT, 2007).

Figura 4 – Complexidade das Saídas externas (SE) e Consulta Externa (CE).

Na figura 4 são apresentadas as saídas e consultas externas no processo de contagem dos pontos de função, ambas representam o envio de dados ou informações de controle para fora da fronteira da aplicação. Um exemplo de entrada externa são relatórios com a totalização de dados.

"Com base na complexidade e em tabelas específicas, a cada componente é atribuído uma quantidade de pontos de função uma quantidade de pontos de função, denominada contribuição do componente de contagem" (BFPUG, 2008). As contribuições dos componentes somadas resultam em uma quantidade de pontos de função não ajustados. Neste artigo não serão tratados os pontos por função ajustados.

Após ajuste do cálculo, é possível ter o resultado final da contagem de pontos de função. De posse destas informações se obtém a complexidade funcional do software, podendo auxiliar nas previsões dos projetos de software e até mesmo o gerenciamento dos mesmos.

Com todos os argumentos apresentados é possível classificar a métrica de pontos de função como uma métrica de software de caráter quantitativo para produtos e processos de software uma vez que essa métrica busca medir características internas do software, bem como ser utilizada para o melhoramento de processos.

2.2.3 Complexidade Ciclomática

É uma técnica que segundo Kan (1995, p.257) indica a testabilidade e a facilidade de entendimento do código. Koscianski e Soares apresentam a definição sobre a métrica da seguinte forma: A premissa básica é que o nível de aninhamento de laços e comandos de decisão no código tem relação com sua complexidade de execução e com a complexidade psicológica, ou seja, o esforço necessário para compreendê-lo. Exemplificando, um comando if em um programa abre duas possibilidades de execução diferentes, o que pode significar o dobro de esforço de teste e verificação.

Esta métrica atua no código de software por meio da criação de uma representação dos caminhos que a execução pode tomar baseado em comandos condicionais. Dessa maneira estabelece um nível de complexidade que pode servir de auxilio para calcular o esforço necessário para os testes de software. Essa métrica

funciona também como um indício da complexidade do programa e, com base nessa informação, pode-se prever o esforço para implementá-lo ou modificá-lo.

A aplicação da métrica complexidade ciclomática é realizada a partir de um grafo que representa o fluxo de controle do programa que pode ser o código de um modulo do software ou de todo o software. Cada nó representa um trecho de código ou um comando.

Essa métrica é classificada de caráter quantitativo para produtos de software que auxilia nos teste de software e também na avaliação da sua complexidade para os módulos do software em desenvolvimento.

2.3 Considerações finais sobre as métricas de software

Nesta seção foi apresentado um resumo sobre métricas de software, onde as mesmas foram classificadas como de caráter qualitativo, de caráter quantitativo, métricas de processo e métricas de produto. Essa classificação foi uma mescla de conceitos de autores como: Sommerville, Pressman, Koscianski e Soares e Kan.

Foram abordadas também três técnicas de métricas de software. A primeira técnica denominada *Lines of Code* (LOC) demonstrou ser útil em determinados aspectos como, por exemplo, para estabelecer uma relação entre o tamanho do software em linhas de código e a densidade de defeitos que o mesmo possa vir a apresentar. A segunda técnica foi a Análise de Pontos de Função (APF), é uma técnica que mede o tamanho funcional do software baseadas em transações externas e internas do software. Por fim a terceira e última técnica apresentada é a técnica de complexidade ciclomática, que é baseada na contagem de comandos condicionais e incondicionais dos programas, estabelecendo um valor para a complexidade de cada programa de software.

Para o estudo proposto, será escolhida uma técnica de métricas de software que vai auxiliar no processo de contagem do tamanho do software, posteriormente será feito uma relação do tamanho do software com o tempo estimado para desenvolver um novo código fonte. Para o presente trabalho vale ressaltar que o estudo é apenas para novos códigos fontes de um sistema que já existe na empresa, ou seja, para novos projetos de software será necessária uma avaliação mais aprofundada para determinar se é possível ou não à utilização das técnicas no projeto.

3. Estudo de Caso

Após a apresentação do estudo das técnicas de métricas de software é preciso estabelecer uma metodologia para ser aplicada no presente trabalho. Esta seção vai identificar os processos realizados atualmente na empresa em que será realizado o trabalho, como vão ficar os processos a partir do estudo implementado e ainda como será o processo de cálculo para as estimativas de tempo no desenvolvimento de um novo código fonte.

A empresa aonde será implementado o trabalho atua no mercado de desenvolvimento de software a mais de 25 anos, conta com 25 colaboradores e está situada na região de Passo Fundo, atualmente possui clientes espalhados por vários estados do país.

3.1 Fluxos do processo com e sem o estudo

A seguir serão apresentados os fluxos dos processos utilizados na empresa. Na figura 5 é ilustrado o processo atual, já na figura 6 é apresentado o conjunto de atividades baseadas no estudo proposto pelo presente trabalho.

O processo atual inicia-se com a necessidade da construção de um novo código fonte, para que seja atendida uma determinada situação. Esta situação pode ser gerada por um cliente externo ou interno, então é lançada uma tarefa solicitando que seja atendida a solicitação, após o lançamento da tarefa a mesma vai para o analista de sistemas da empresa que colocará o tempo previsto para um desenvolvedor executar tal tarefa. Depois do tempo incluído a tarefa é colocada em um banco de tarefas onde o gerente de operações distribui as tarefas para os desenvolvedores. De posse dos desenvolvedores a tarefa então é executada e o mesmo coloca o tempo que gastou para realizá-la, então após a conclusão da mesma é realizado a distribuição do código fonte criado ou alterado.

A figura 5 apresenta o fluxo das atividades que ocorrem hoje na empresa, a qual esta sendo desenvolvido o presente trabalho.

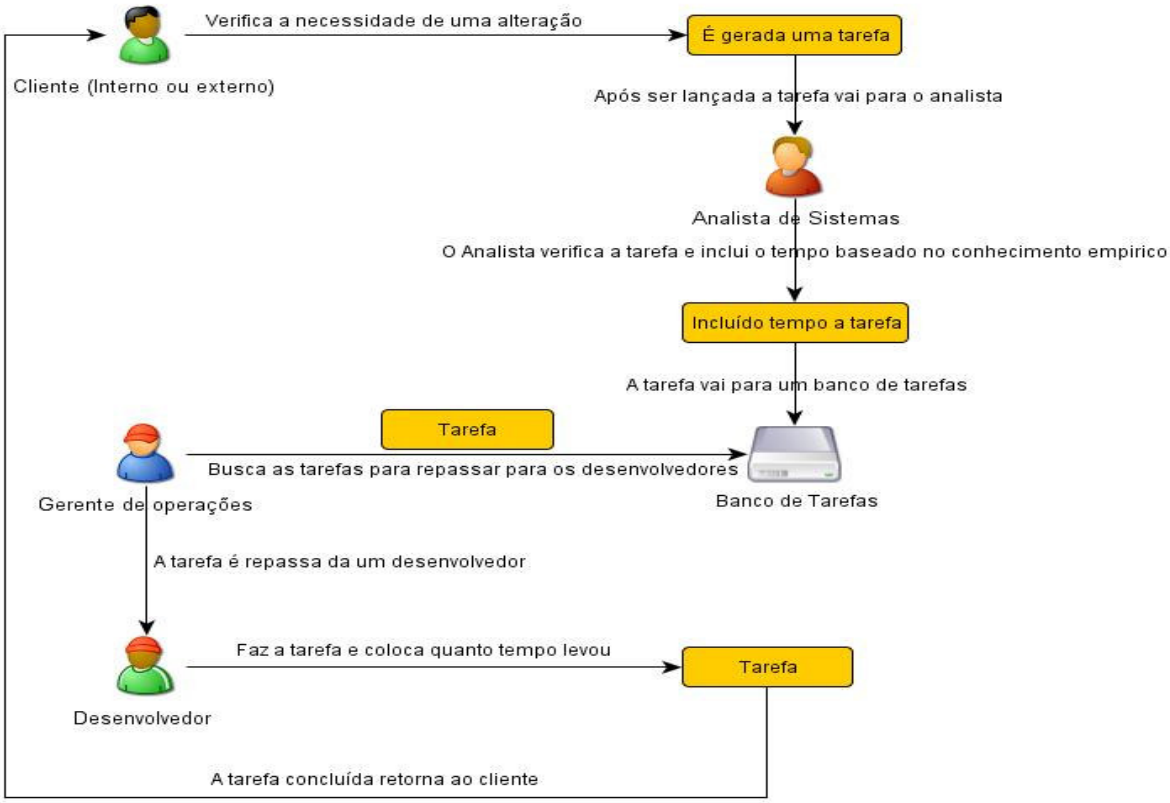


Figura 5 – Fluxo atual das atividades da empresa

O fluxo de atividades proposto pelo trabalho tem início da mesma forma que o Fluxo antigo da empresa, porém quando a tarefa chega ao analista de sistemas o mesmo não colocará mais o tempo estimado para realizar a tarefa, mas sim a quantidade de pontos de função que aquele código fonte irá possuir. O gerente de operações continua fazendo a distribuição das tarefas, porém com uma mudança. Esta mudança diz respeito

à colocação do tempo na tarefa, pois o mesmo quando distribuí-la irá verificar o nível do desenvolvedor que irá executar a mesma, só assim será atribuído à tarefa o tempo estimado para sua realização. Após a tarefa chegar ao desenvolvedor o fluxo das atividades segue como no processo antigo, o desenvolvedor executando e colocando o tempo gasto para a realização da tarefa.

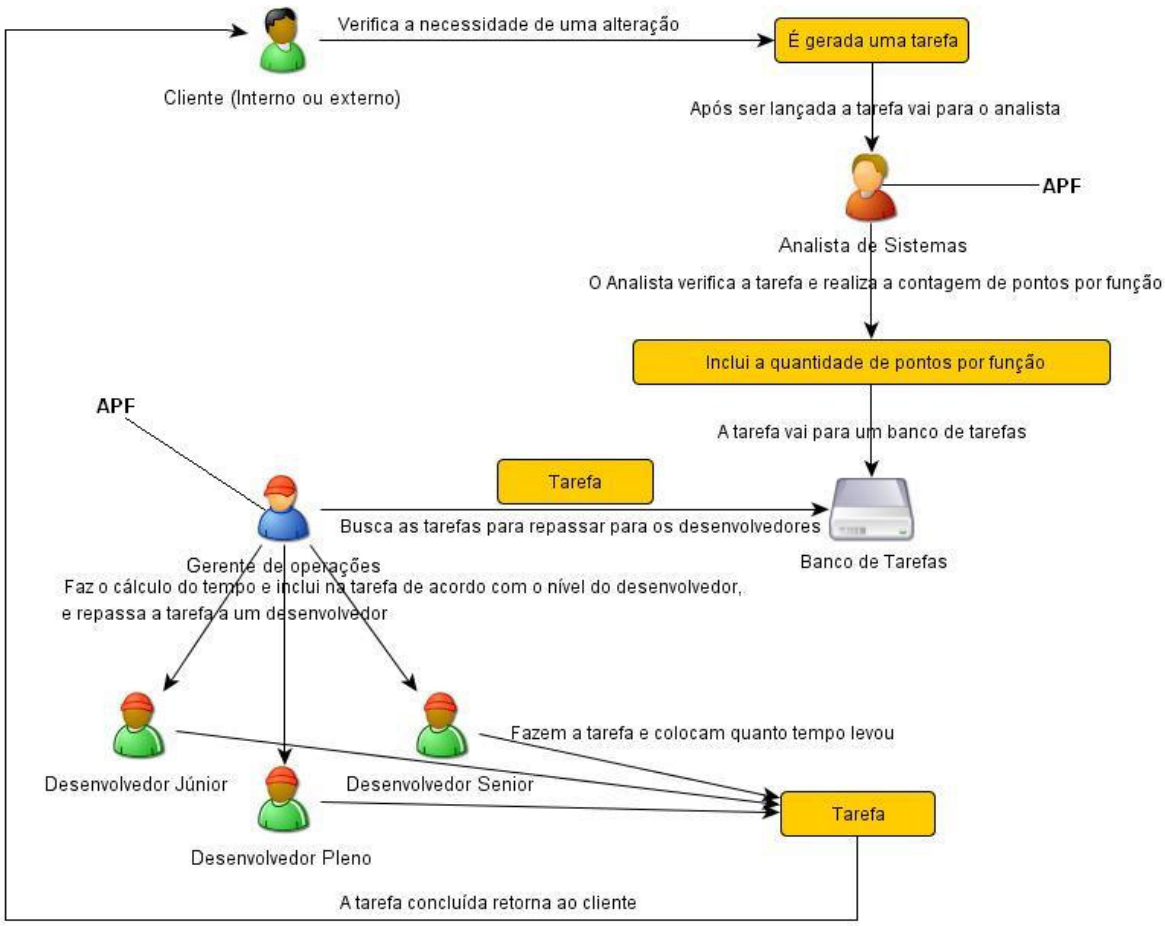


Figura 6 – Fluxo das atividades propostas

3.2 Técnica de métrica de software escolhida

No presente trabalho foram apresentadas três técnicas de métricas de software que são citadas por autores como Kan, Koscianski e Sommerville. A escolha se deu em virtude de um estudo detalhado das técnicas, e assim, foi escolhida a técnica que mais se adaptou ao contexto da empresa de desenvolvimento de software onde será aplicado o estudo realizado.

No início dos estudos a primeira técnica a ser descartada foi à técnica de *Lines of Code* (LOC), pois a mesma é baseada no número de linhas de código de um programa de software, contudo, o objetivo do estudo é estabelecer quanto tempo irá ser necessário para desenvolver um novo código fonte. Neste caso não seria possível aplicar a técnica para um novo código fonte, pois como o código fonte ainda esta em desenvolvimento não se tem como contar quantas linhas de código possui tal programa. Outro empecilho da técnica foi que ela também considera o número de linhas de código com comentário

dentro de um programa de software, e a tecnologia de desenvolvimento utilizada na empresa coloca muitos comentários internamente em um código fonte.

Após o descarte da primeira técnica permaneceram em estudo as técnicas de Análise de Pontos de Função (APF) e Complexidade Ciclomática. Então a segunda técnica a ser descartada foi a de Complexidade Ciclomática, está técnica leva em consideração a contagem do número de comandos condicionais e incondicionais dos códigos fonte, e neste caso mais uma vez a tecnologia de desenvolvimento da empresa em que será realizado o estudo foi decisiva. A tecnologia de desenvolvimento que a empresa utiliza existe inúmeros comandos incondicionais, onde apenas para atribuir um valor para uma variável é necessário um comando além do próprio comando de atribuição.

Por fim a técnica que mais de adaptou a realidade da empresa foi a APF, pois é uma técnica que pode ser implementada independente da linguagem de programação utilizada na empresa.

3.3 Processos de contagem dos pontos de função

O processo de contagem dos pontos de função será feito pelos Analistas de Sistema da empresa, que, no entanto irão se basear nas fórmulas e tabelas do manual de APF estabelecido pela IFPUG. Em um primeiro momento do estudo o Analista responsável pelo sistema sugeriu que as contagens dos pontos de função dos programas seriam feitos apenas para novos códigos fonte, cuja categorização conforme a empresa denomina-se construção de novos códigos fontes. Para auxiliar no processo de contagem dos pontos de função será utilizado a ferramenta APFplus, que foi construída para dar suporte à contagem de pontos de função de aplicações (sistemas). A ferramenta está em conformidade com a metodologia de contagem prevista no Manual de Práticas de Contagens, versão 4.2.1, do IFPUG (International Function Point Users Group). A ferramenta incorpora os métodos de Contagem Indicativa e de Contagem Estimada de Pontos de Função, conforme proposto pela NESMA (Netherlands Software Metrics Users Association). A NESMA, originalmente chamado de NEFPUG (Netherlands Function Point Users Group), foi fundada em 1989, sendo o primeiro grupo a ser criado após o IFPUG.

Atualmente, na empresa existem vários sistemas que estão no mercado, porém o analista chefe da empresa escolheu o sistema de Administração de Recursos Humanos para que seja aplicado este estudo. Sua escolha se baseou no planejamento estratégico da empresa, pois o mesmo diz que o sistema escolhido é o sistema com maior perspectiva de vendas nos próximos 5 anos.

Na empresa, dentre seus procedimentos e normas está estabelecido como um item da avaliação de desempenho da equipe, o índice de acertos das estimativas de tempo que será gasto para desenvolvimento de um novo código fonte. De acordo com este documento é necessário que as estimativas de tempo projetadas fiquem com um percentual igual ou maior a 92% de acertos. Para cada tarefa onde o tempo é estimado há índice de 10% de erro para mais ou para menos, ou seja, se o tempo estimado para uma solicitação é de 60 minutos, a tarefa pode ser realizada entre 54 a 66 minutos que estará dentro do percentual de acertos.

3.4 Transformações dos pontos de função em tempo (Hora/Minuto)

O processo seguinte após definir a contagem dos pontos de função dos programas de software, é incluir o tempo (Hora/Minuto) estimado para o desenvolvimento da tarefa, este processo será feito pelo gerente de operações.

Para a transformação dos pontos de função em tempo foi necessário que o gerente de operações enquadrasse todos os desenvolvedores de software em um nível dentro da empresa. Este nível é a referência da experiência de desenvolvimento de software na tecnologia que a empresa trabalha. Dentro da estrutura organizacional da empresa os níveis já estão estabelecidos e cada nível tem uma atribuição diferente. De acordo com a empresa os níveis são:

- ⇒ Júnior: É o desenvolvedor de software ainda com pouca experiência na tecnologia que a empresa trabalha, é considerado desenvolvedor de software júnior o desenvolvedor que tenha de 1 a 6 anos de empresa.
- ⇒ Pleno: É o desenvolvedor de software com um determinado nível de experiência na tecnologia que a empresa trabalha, é considerado desenvolvedor de software pleno o desenvolvedor que tenha de 7 a 10 anos de empresa.
- ⇒ Sênior: É o desenvolvedor de software com experiência na tecnologia que a empresa trabalha, é considerado desenvolvedor de software sênior o desenvolvedor que tenha mais de 10 anos de empresa.

Os níveis apresentados servem de base para o cálculo do tempo de uma atividade de desenvolvimento de um novo código fonte na empresa. O cálculo proposto será de vital importância para tal estudo. Primeiramente, antes de fazer o cálculo para determinar o tempo que aquele código fonte irá levar para ser desenvolvido é preciso descobrir quanto tempo equivale um ponto de função na linguagem de desenvolvimento Progress, que é a linguagem atual que a empresa utiliza no desenvolvimento de suas aplicações. No passado a empresa SPR - Software Productivity Research - disponibilizava gratuitamente uma tabela de linguagens de programação na internet. Essa tabela atribuía a cada linguagem um nível, sendo fornecidos intervalos de produtividade estimados para cada nível de linguagem. Além disso, a chamada "Tabela da SPR" fornecia estimativas para a razão Linhas de Código Fonte / Ponto de Função, para cada linguagem (também chamada Fator de Backfiring). Essa tabela, embora contivesse dados estatísticos interessantes para pesquisas, foi muitas vezes indevidamente utilizada como base em relacionamentos comerciais. Então a própria SPR optou por primeiro retirar a tabela do ar e depois trazê-la de volta, desta vez como um serviço pago.

Então por não ter esta tabela disponível, foi preciso descobrir o tempo que equivale um ponto por função na linguagem de programação Progress. Um conjunto de atividades foram distribuídas para todos os níveis de desenvolvedores da empresa que trabalham no sistema de Administração de Recursos Humanos. Todos os desenvolvedores da equipe realizaram o mesmo conjunto de atividades e só assim foi estabelecido quanto tempo equivale um ponto de função nos três níveis de desenvolvedores da empresa. Este conjunto de atividades foi realizado por 4 desenvolvedores e os mesmos passaram pelas mesmas atividades no mesmo período de

tempo estabelecido para o estudo. Cada um desenvolveu a mesma tarefa e pontuou em minutos quanto tempo cada um levou para terminar o desenvolvimento no novo código fonte estabelecido. A figura 7 mostra o demonstrativo das atividades no período de 26/04/2010 até 07/05/2010 desempenhado pela equipe de desenvolvedores.

Dados coletados entre 26/04/2010 a 07/05/2010			
Código Fonte	**adm050v41 CNAE do Empregado**	**adm050v43 Médias do Empregado**	**adm050v45 Provisões do Empregado**
	7,15 pontos por função	**7,13 pontos por função**	**7,80 pontos por função**
Desenvolvedor	Tempo em minutos	Tempo em minutos	Tempo em minutos
CSC (Sênior)	104	93	116
CLM (Pleno)	115	126	132
LMG (Júnior)	123	132	142
MVB (Júnior)	130	148	148

Categoria	**Média dos PF**	**Média dos PF(arredondamento)**
Júnior	18,63677536	19 minutos
Pleno	16,89311594	17 minutos
Sênior	14,17572464	14 minutos

Fonte: Dados Coletados na empresa entre 26/04/2010 até 07/05/2010.

Figura 7 – Distribuição das tarefas para encontrar o tempo de um ponto por função na linguagem de programação Progress.

Para confirmar as informações coletadas anteriormente foi proposto a mesma seqüência de atividades entre o período de 12/05/2010 e 18/05/2010 para a mesma equipe de desenvolvedores que fizeram parte do estudo. Os tempos foram muito semelhantes aos tempos coletados no estudo anterior. A figura 8 apresenta o demonstrativo dos tempos gastos para o desenvolvimento das atividades na segunda coleta de informações.

Dados coletados entre 12/05/2010 a 18/05/2010			
Código Fonte	**adm002v15 CNAE da Filial**	**adm002v17 INSS e Fap da Filial**	**adm236 Parâmetro Média**
	7,14 pontos por função	**7,02 pontos por função**	**7,85 pontos por função**
Desenvolvedor	Tempo em minutos	Tempo em minutos	Tempo em minutos
CSC (Sênior)	96	84	118
CLM (Pleno)	113	98	142
LMG (Júnior)	121	105	158
MVB (Júnior)	119	122	172

Categoria	**Média dos PF**	**Média dos PF(arredondamento)**
Júnior	18,10540663	19 minutos
Pleno	16,03816447	17 minutos
Sênior	13,53930032	14 minutos

Fonte: Dados Coletados na empresa entre 12/05/2010 até 18/05/2010.

Figura 8 – Distribuição das tarefas para encontrar o tempo de um ponto por função na linguagem de programação Progress na segunda coleta de dados.

De posse das informações coletadas foi possível identificar o tempo que cada categoria de desenvolvedor levaria para construir um ponto de função no Progress. A categoria de desenvolvedor Júnior leva em média 19 minutos para implementação de um ponto de função. A categoria Pleno leva em média cerca de 17 minutos para o desenvolvimento de um ponto por função. Já a categoria Sênior leva em torno de 14 minutos para implementar um ponto de função. Nota-se que estes tempos foram medidos para uma categoria de programas, que de acordo com a empresa a categoria se denomina construção de novos códigos fontes.

De posse do tempo que equivale um ponto de função em cada nível de desenvolvedores da empresa, será preciso apenas fazer o cálculo matemático para descobrir quanto tempo um programa de software irá levar para ser desenvolvido. No novo código fonte já foi contado o número de pontos de função, o tempo estimado para cada nível de desenvolvedor já está estabelecido, então o que precisa ser feito é um cálculo de regra de três simples. Se um ponto de função equivale a "X" tempo, quanto equivale "n" pontos de função em tempo. É com este propósito que será estabelecido quanto tempo um novo código fonte levará para ser construído.

3.5 Implementação do estudo nas tarefas do dia a dia

Atualmente, na empresa, o tempo estimado para o desenvolvimento de um programa de software é totalmente empírico, ou seja, é baseada totalmente na experiência do analista que esta verificando a tarefa, contudo, muitas vezes os tempos previstos ficam distorcidos dos tempos realmente gastos para concluir tal tarefa. Sempre que se tem a necessidade da criação de um novo código fonte é gerada uma tarefa. Esta tarefa é solicitada por um cliente externo ou até mesmo por um cliente interno. Após o lançamento da tarefa é incluído o tempo estimado que o novo código fonte vá levar para ser desenvolvido.

Após colocar o tempo na tarefa, a mesma é encaminhada para o gerente de operações, que é o responsável por encaminhá-la a um desenvolvedor. Este verifica as atividades de cada desenvolvedor e encaminha para o qual se encontra disponível, caso não tenha nenhum desenvolvedor disponível ou então com tempo para realizar a tarefa, a mesma fica em um banco de tarefas.

O objetivo do trabalho entraria no momento que se deve colocar o tempo na tarefa, pois o fluxo das atividades seria o mesmo, a diferença no processo é a inclusão da contagem dos pontos de função, e no momento em que o gerente de operações fosse colocar o tempo para tarefa o mesmo teria que fazer o cálculo do tempo em cima do nível do desenvolvedor que irá trabalhar na tarefa.

4. Comparativo entre o tempo estimado e realizado com e sem o estudo

Nesta seção será apresentado um comparativo entre o tempo estimado e o tempo realizado de uma tarefa. Nos comparativos serão considerados os tempos estimados com o estudo realizado e também comparativos sem o estudo realizado.

Para analisar os tempos primeiramente é preciso identificar uma tarefa e quantificar o tempo que a mesma levará para ser desenvolvida. Esta tarefa é o desenvolvimento de um código fonte novo.

De posse da tarefa de análise são verificados quantos pontos de função este programa terá, e a partir deste momento será calculado o tempo previsto para desenvolvimento do código fonte. A quantidade de pontos de função deste código fonte é de 7,15 pontos por função. Na figura 9 é demonstrado um gráfico onde não o existe o estudo desenvolvido, então se pode observar que foi estimado um tempo de 120 minutos para a realização do no novo código fonte, no entanto este novo código fonte foi desenvolvido em menos tempo, desta forma ficando claro uma disparidade entre o tempo estimado e o tempo gasto para desenvolvimento do código fonte.

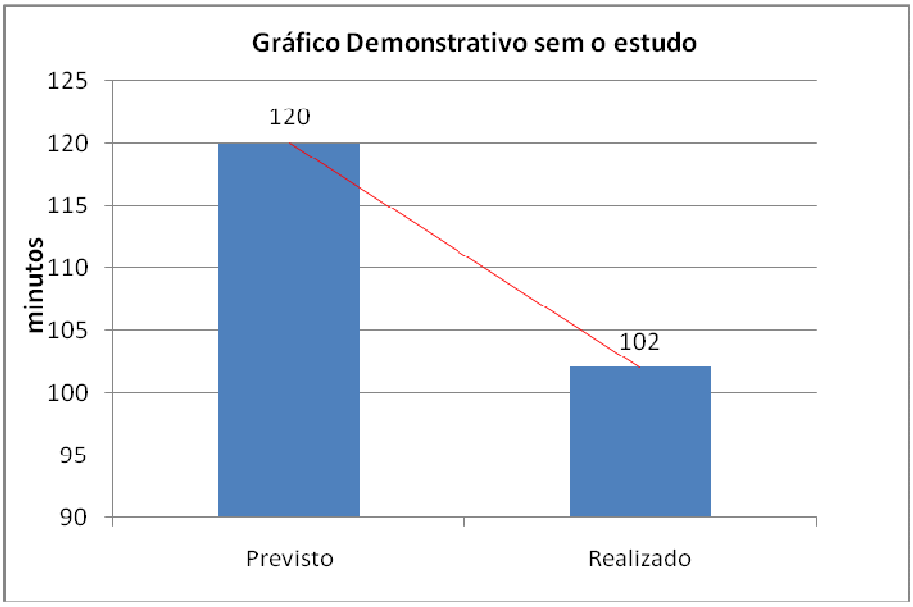


Figura 9 – Demonstrativo do tempo sem o estudo realizado.

Na figura 10 é demonstrado um gráfico onde as atividades seguiram os procedimentos propostos no presente trabalho. Todos os passos foram seguidos conforme o fluxo demonstrado na figura 6. Nota-se que na figura 10 há uma certa coerência entre o tempo estimado e o tempo que realmente foi gasto para desenvolver a tarefa. Vale ressaltar que o código utilizado para realizar esta figura é o mesmo código que foi utilizado na figura anterior.

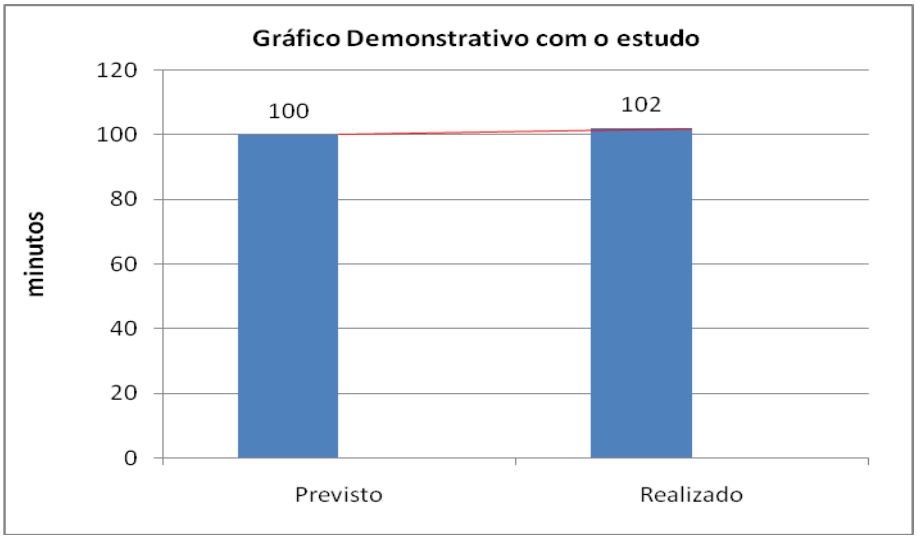


Figura 10 – Demonstrativo do tempo sem o estudo realizado.

Após a análise, pode-se observar que na figura 10 os tempos estão mais equiparados, ou seja, o tempo previsto para a construção do novo código fonte ficou muito próximo do tempo que foi gasto para realizar a construção no código. Nesta análise o tempo estimado na figura 9 ficou acima do tempo realizado, mas da mesma forma que foi estimado mais do que realizado também pode ser utilizado mais tempo do que o previsto para a construção de um novo código fonte.

5. Conclusões e Trabalhos futuros

Este trabalho apresentou um estudo sobre métricas de software e alguns tipos de técnicas de métricas de software. Também apresentou o desenvolvimento de uma metodologia para estimar tempo gasto no desenvolvimento de um ponto por função na linguagem de programação Progress.

A importância de medir o tempo de desenvolvimento de um novo código fonte ficou clara, pois foi possível perceber que o objetivo de estimar um tempo mais perto do real foi alcançado, estando dentro da margem de erro estabelecido nos documentos da própria empresa onde se desenvolveu o trabalho. Foi ainda desenvolvida uma metodologia para a criação de uma tabela com o tempo estimado para desenvolver um ponto de função na linguagem de programação Progress, para cada categoria de desenvolvedor existente na empresa.

Para trabalhos futuros fica a possibilidade de avaliar se é possível estabelecer o tempo estimado para outras categorias de desenvolvimento de software que existem na empresa, como por exemplo, para manutenções adaptativas e corretivas. Ainda é possível identificar se a produtividade da equipe esta dentro da esperada ou não, estabelecendo maneiras e métodos de treinamento na tecnologia utilizada no desenvolvimento dos códigos fonte da empresa. Ainda, para efeitos de confirmação dos pontos por função com a linguagem Progress pode ser aplicado o estudo de caso mais vezes nos procedimentos da empresa. Ainda como sugestão de trabalhos futuros pode ser verificada a possibilidade de estimar os custos gastos para o desenvolvimento de novos códigos fontes ou até mesmo os custos que serão gastos para desenvolver um novo projeto de software utilizando a mesma metodologia aplicada.

6. Referências

PRESSMAN, Roger S.;. *Engenharia de Software.* 6 ed. São Paulo: MCGRAW-HILL,2006.

SOMMERVILLE, Ian.;. *Engenharia de Software.* São Paulo: PEARSON BRASIL, 2003.

DEMARCO, TOM. Controle de Projetos de Software. 9.ed. Rio de Janeiro: Editora Campus, 1991.

PFLEEGER, Shari Lawrence.;. *Engenharia de Software: Teoria e Prática.* São Paulo:Pearson Brasil, 2004.

VAZQUEZ, Carlos Eduardo; SIMÕES Guilherme Siqueira; ALBERT Renato Machado. *Análise de Pontos Por Função: Medição, Estimativas e Gerenciamento de Projetos de Software.* São Paulo:Editora Érica Ltda, 2007.

KAN, Stephen H.. Metrics and models in software quality engineering. Boston: Addison Wesley, 1995.

KOSCIANSKI, André; SOARES, Michel dos Santos. *Qualidade de Software: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software.* São Paulo: Novatec Editora LTDA, 2006.