Métricas de software: Comparação entre Pontos de Função e Cocomo

II Software metrics: Comparing Function Points and Cocomo II

Augusto Nogueira Zadra [1]
Erivelton Oliveira Carvalho [2]
Joécio Farley Santos Cardoso [3]

Resumo: Este artigo tem por objetivo fazer uma comparação entre as métricas pontos de função e Cocomo II, reunindo informações que podem auxiliar os gestores na escolha da melhor métrica a ser utilizada no desenvolvimento de software, baseado nas funcionalidades prazo e esforço no processo de desenvolvimento de software, trata-se de um tema pouco pesquisado, buscou-se reunir uma base conceitual e teórica sobre esta questão, quando realizou-se uma pesquisa do tipo exploratória.
Palavras chaves: Métricas, Pontos de função, Cocomo II

Abstract: *This article aims to make a comparison between the metric function points and Cocomo II, gathering information that can assist managers in the best metric choice to be used in software development, based on features time and effort in the software development process, it is a subject little researched, sought to put together a conceptual and theoretical basis on this issue, when there was a research of the exploratory type.*
Keywords: *Metrics, Function points, Cocomo II*

1 INTRODUÇÃO

Desde o surgimento da computação o mundo vem se tornando dependente de novas tecnologias, principalmente pelo software, esta dependência ficou conhecida como “a lei das consequências não pretendidas”(PRESSMAN,2010);

Este autor assinala que mesmo com software presentes em diversos segmentos como: transportes, educação, saúde, existindo muitas outras aplicações que ainda não foram previstas.

Com softwares cada vez maiores e mais complexos não existia um padrão definido para desenvolvimento e medição que propiciasse um maior entendimento

[1] Mestrando em Tecnologia da Informação Aplicada à Biologia Computacional. Professor na Faculdade Infórium de Tecnologia.
[2] Graduando em sistemas de informação na Faculdade Infórium de Tecnologia..
[3] Graduando em sistemas de informação na Faculdade Infórium de Tecnologia..

sobre seu processo de produção, assim o objetivo desse trabalho é conhecer alguns padrões de medições e sua contribuição para o aumento da produtividade nas empresas, como custos, tempos de desenvolvimento e produtividade de mão de obra. São objetivos específicos:

- Conhecer as principais técnicas para medição de prazo e esforço e seus benefícios para o processo de desenvolvimento de software.
- Comparar os pontos positivos e negativos de dois processos usados em medição de software.

Tendo como base o pensamento de Pressman (2010), a pergunta de pesquisa orientadora do estudo, é no sentido de verificar quais as principais vantagens e desvantagens das metodologias Cocomo e análise de ponto de função aplicada ao projeto de desenvolvimento de software.

A relevância deste estudo está em demonstrar os benefícios da utilização de metodologias e ferramentas adequadas para estimar a medição de software, possibilitando resultados satisfatórios no processo de desenvolvimento, bem como maior segurança para os gerentes de projeto no apoio a tomada de decisões, determinando parâmetros como quantidade de testes necessários e impacto das mudanças, formando uma linha básica para estimativas, reduzindo frustações e pressões de cronograma, antes mesmo do produto ser entregue ao cliente, facilitando a

estimativa pré-desenvolvimento diminuíndo a margem de erro garantindo a estimativa correta e mais próxima da realidade.

A crescente demanda pelo desenvolvimento de software torna imprescindível a medição e estimativa de fatores diretamente envolvidos nesta atividade. Fatores como: custo e tempo de desenvolvimento que geralmente são estimados utilizando-se apenas a experiência do profissional envolvido, são normalmente imprecisos e isso, muitas vezes, acarreta desperdícios de recursos, descumprimento de prazos, o que gera prejuízo para o desenvolvedor e atraso na entrega do produto ao cliente.

Embora métricas de produtos para programas de computadores sejam imperfeitas, podem proporcionar uma maneira sistemática de avaliar a qualidade com base em um conjunto de regras claramente definidas, proporcionando uma visão objetiva que vai direto ao ponto e não após o fato. Isso permite descobrir e corrigir problemas potenciais antes que se tornem defeitos catastróficos. (PRESSMAN, 2010).

Para compreensão deste tema dividiu-se este artigo em 8 seções. A seção 1, está a introdução, e indicativa do estudo; a seção 2 apresenta a abordagem teórica sobre métricas de software; a seção 3 aborda a medição na metodologia Cocomo II; a seção 4 aborda a medição na metodologia de análise de pontos de função; a seção 5 apresenta o comparativo entre as metodologias; a seção 6 elabora a conclusão do estudo; a seção 7 apresenta as referências bibliográficas.

2 ABORDAGEM TEÓRICA SOBRE MÉTRICAS DE SOFTWARE

A falta de padrões de desenvolvimento de software tornou necessária a criação da engenharia de software e consequentemente de seus processos de desenvolvimento e medição de tamanho, custo e tempo de construção.

Para tal diversas metodologias foram desenvolvidas ao longo dos anos, cada uma buscando uma abordagem diferente para a resolução do problema de medição de software.

Para Sommervile (2003) a engenharia de software é uma disciplina da engenharia que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de

especificação do sistema até a manutenção do sistema, depois que ele entrou em operação.

Jalote (2005, p.566**)** assinala que um processo de software é:

> *Um conjunto de atividades, ligadas por padrões de relacionamento entre ela, pelas quais se as atividades operarem corretamente e de acordo com os padrões requeridos, o resultado desejado é produzido. O resultado desejado é um software de alta qualidade e baixo custo. Obviamente, um processo que não aumenta a produção (não suporta projetos de software grandes) ou não pode produzir software com boa qualidade não é um processo adequado.*

As métricas de software proporcionam uma maneira quantitativa de avaliar a qualidade dos atributos internos do produto, possibilitando avaliar a qualidade antes que ele seja criado. As métricas proporcionam as informações necessárias para criar requisitos eficazes e modelos de projeto, código sólido e testes completos (PRESSMAN, 2010).

As métricas de software permitem avaliar o projeto a ser desenvolvido de uma forma mais completa, por isso tal procedimento é tão importante no cenário atual do desenvolvimento.

Para Pressman (2010, pag. 583), métricas são[...] medidas quantitativas que permitem que você tenha discernimento sobre a eficácia do processo de software e os projetos que são executados usando o processo como estrutura.

As medições foram criadas principalmente para garantir que indicativos pudessem ser obtidos e assim houvesse uma otimização dos custos de produção já que na década de 90, bilhões de dólares eram gastos em softwares que não atendiam as empresas da época (PRESSMAN, 2010).

As métricas de software podem ser divididas em dois tipos, as medidas diretas e as medidas indiretas.

As medidas diretas incluem fatores como o custo de produção, o trabalho

aplicado para a produção do produto de software, tamanho do software em linhas de código e outras mais, este tipo de medida pode ser estimado antes mesmo da produção

do produto de software enquanto as medidas indiretas baseiam em fatores como funcionalidades, qualidade e complexidade e estas características não podem ser medidas de uma forma simples por isso são consideradas medidas indiretas (PRESSMAN, 2010).

Quando um software é desenvolvido, é certo que um produto perfeito é impossível de ser obtido a partir do desenvolvimento primário e por isso cada vez mais se faz necessária medição de diversos fatores a fim de se alcançar um software com alto nível de qualidade.

Segundo (PRESSMAN, 2010) a medição de software torna possível que gerentes e profissionais entendam melhor o processo de engenharia de software e o produto (software) que ele produz. Usando medidas diretas e indiretas, as métricas de produtividade e qualidade podem ser definidas.

a) Métricas orientadas a tamanho

As métricas orientadas a tamanho levam em consideração a quantidade de linhas de código escritas no decorrer do desenvolvimento do software e podem ser representadas até mesmo por um por uma tabela (PRESSMAN, 2010) e assim ao armazenar dados de um projeto desenvolvido pode se ter base para aplicar o mesmo em futuros projetos.

Este tipo de métrica pode ser usado para medições mais detalhadas ao invés de medir apenas a quantidade de linhas de código escritas como, por exemplo:

- Quantidade de erro por KLOC.

- Defeitos por KLOC.

- Quantidade de documentação por KLOC.

Apesar de se mostrar um modelo muito interessante para a medição de software as

métricas orientadas a tamanho não são aceitas universalmente e um dos principais motivos se deve ao fato de haver diferença entre a quantidade de linhas de código

necessárias e diferentes tipos de linguagens de programação (PRESSMAN, 2010) por isso outros modelos ganharam espaço no mercado utilizando de outras abordagens.

b) Métricas orientadas a função

As métricas orientadas a função para medir a quantidade de funções de um software são estimadas através de dados históricos, os custos e o esforço para o projeto codificação, teste de novos produtos de software.

Pontos de função são derivados, por meio de uma relação empírica baseada em medidas calculáveis (diretas) do domínio de informações do software e avaliações qualitativas de complexidade do software (PRESSMAN, 2010). Definidos por:

- Número de entradas externas (EEs).
- Número de saídas externas (EOs).
- Número de consultas externas (EQs).
- Número de arquivos lógicos internos (ILFs).
- Número de arquivos de interface externos (EIFs).

Coletam-se esses dados, juntamente com as respostas de um questionário de complexidade, elabora-se uma fórmula que servirá de base para se calcular o ponto de função (FP).

3 MEDIÇÃO NA METODOLOGIA COCOMO II

Como uma das principais abordagens para medir a construção de um produto software pode-se destacar o modelo COCOMO II que é derivado da versão anterior conhecida como COCOMO ou COCOMO 81, segundo (AGUIAR,2010) o modelo de construção de custo (COCOMO II) é capaz de estimar o esforço, prazo e custo para desenvolvimento do produto.

Conforme descrito por seu criador (BOEHM, 1981) a metodologia COCOMO foi desenvolvida com base na analise de 63 projetos de software para diversas áreas entre elas ciência, negócios e suporte.

Ainda conforme o modelo de Boemh pode se classificar os projetos em 3 tipos:

Orgânico: Projetos pequenos e de baixa complexidade.

Semidestacado: Tamanho e complexidade intermediários.

Embutido: Requer um cenário rígido e restritivo no que tange hardware e software

Para que possa ser realizado o calculo os parâmetros que devem ser incluídos nas equações padrão da metodologia conforme as tabelas demonstradas por (Júnior e Sanches,2000) cada tabela contem os valores para os parâmetros ai e bi de esforço e prazo.

Tabela 1 – Parâmetros Para Estimativa de Esforço - COCOMO

Modo	a_i	b_i
Orgânico	2,4	1,05
Semidestacado	3,0	1,12
Embutido	3,6	1,20

Fonte: (JUNIOR;SANCHES,2000,p.5)

Tabela 2 – Parâmetros Para Estimativa de Prazo - COCOMO

Modo	a_i	b_i
Orgânico	2,5	0,38
Semidestacado	2,5	0,35
Embutido	2,5	0,32

Fonte: (JUNIOR;SANCHES,2000,p.5)

A medida usada para estimar o tamanho de software através da metodologia COCOMO II é o KLOC que representa mil linhas de código escritas, seguindo essa abordagem pode se afirmar que um software com 50 mil linha pode ser considerado um software pequeno e a partir de 300 mil linhas começa-se a considerar um software grande.

No entanto como afirmam (Júnior e Sanches,2000) uma das desvantagens da metodologia que utiliza linhas de código como forma de medir esbarra na falta de padronização encontrada entre as diferentes linguagens de programação presentes no mercado atual e isto pode gerar uma falha na estimativa realizada.

Para calcular os fatores de esforço e prazo utiliza-se as equações pré-definidas da metodologia e os valores padrão dos parâmetros ai e bi (tabela 1 e tabela 2) conforme (Júnior e Sanches, 2000 apud Boemh) demonstram.

A equação para calcular o esforço pode ser vista abaixo onde E representa o esforço a ser calculado, S representa o tamanho do software expresso em milhares de linhas de código (KLOC) :

Equação 1 – Estimativa de Esforço de Desenvolvimento

$$E = ai\ S^{bi}$$

Para realizar o calculo de prazo de desenvolvimento utiliza-se a equação 2 onde T representa o tempo de desenvolvimento e E o esforço já calculado:

Equação 2 – Estimativa de Tempo de Desenvolvimento

$$T = ai\ E^{bi}$$

4 MEDIÇÃO NA METODOLOGIA DE ANÁLISE DE PONTOS DE FUNÇÃO

A análise de pontos de função é uma técnica usada para medir software baseando-se nas funcionalidades e no ponto de vista do usuário. O ponto de função é a unidade de medida usada pela metodologia e representa uma funcionalidade (VASQUEZ; SIMÕES; ALBERT, 2010).

Entretanto através da analise de pontos de função só será possível estimar o tamanho do software, outros fatores como esforço, prazo e custo do mesmo dependerá de outras variáveis tais como dispomos nas linhas que se sucedem.

Na década de 70, Allan Albrecht então funcionário da IBM definiu os processos de medição que vieram a se tornar a analise de pontos de função, com o crescimento do numero de usuários da metodologia foi criado então em 1986 o Internacional Function Point User Group (IFPUG) que é a organização responsável pela manutenção e normatização da metodologia de APF (IFPUG, 2010).

O IFPUG também é responsável pelo lançamento e manutenção do Manual de Praticas de Contagem de Pontos de Função (CPM).

O processo de contagem de pontos de função passa por algumas etapas até chagar ao resultado final para dar inicio a contagem é necessário classificar as funções,

segundo o IFPUG (2013) as funções podem ser classificadas em:

a) Funções de Dados

São funções referentes ao armazenamento ou referencia de dados. As funções de dados podem ser divididas em arquivo logico interno (ALI) ou arquivo logico externo (ALE).

Arquivo Logico Interno	É um conjunto de dados mantido pela aplicação, reconhecido pelo usuário e mantido dentro da fronteira da aplicação. Exemplo: Cadastro de cliente.
Arquivo de Interface Externa	É um conjunto de dados mantido pela aplicação, reconhecido pelo usuário e mantido dentro da fronteira de outra aplicação, o que significa que um AIE deve ser um ALI em outra aplicação, no entanto não podem ser considerados arquivos que adicionem, alterem ou excluam dados de um ALI. Exemplo: Arquivo de mensagens de erro.

Fonte: IFPUG, 2013

b) Funções de Transações

É uma função que provê funcionalidade de processamento de dados, podem ser divididas em:

Entrada Externa	É um processo realizado em dados ou informação de controle que são recebidos de fora da fronteira da aplicação. O objetivo do mesmo é manter um ou mais ALIs e/ou alterar o comportamento do sistema.
Saída Externa	É um processo que enviam dados ou informações de controle para fora da fronteira da aplicação e inclui processamento adicional além

	daquele existente em uma consulta externa. O objetivo de uma saída externa é apresentar dados ao usuário através de lógica de processamento que não seja apenas recuperação de dados ou informação de controle. A lógica de processamento deve contar ao menos uma fórmula matemática ou cálculo, e/ou criar dados, e/ou manter um ou mais ALIs, e/ou alterar o comportamento do sistema.
Consulta Externa É um processo	que enviam dados ou informações de controle para fora da fronteira da aplicação. A intenção primária de uma consulta externa é apresentar dados ao usuário através de recuperação de dados ou informação de controle. A lógica de processamento não contém fórmula matemática, nem cálculo, nem cria dados derivados. Nenhum ALI é mantido durante o processamento, nem o comportamento do sistema é alterado.

Fonte: IFPUG, 2013.

c) Definição da Complexidade e Contagem

Para determinar o tamanho e a complexidade funcional de dados é necessário utilizar as tabelas de contribuição padrão da metodologia (tabelas 3 e 4).

Tabela 3 - Valores padrão para atribuição da complexidade funcional para funções de dados:

Complexidade Funcional	ALI	AIE
Baixo	7 PF	5 PF
Médio	10 PF	7 PF

Alto	15 PF	10 PF

Fonte: IFPUG, 2013 p.46

Tabela 4 – Valores padrão para atribuição da complexidade funcional das funções de transação:

Complexidade Funcional	CE	SE	EE
Baixo	3 PF	4 PF	3 PF
Médio	4 PF	5 PF	4 PF
Alto	6 PF	7 PF	6 PF

Fonte: IFPUG, 2013 p.47

d) Equações

Para realização dos cálculos das estimativas de esforço, prazo e custo são necessários utilizarem as equações abaixo conforme demonstrado por (VASQUEZ; SIMÕES; ALBERT, 2010):

Equação 1 – Cálculo do Total de Pontos de Função

$$PF = (ALI \times TF) + (AIE \times TF) + (SE \times TF) + (CE \times TF) + (EE \times TF)$$

Onde PF representa o total de pontos de função a se obter através da soma das funções de dados e transação multiplicadas por seus respectivos tamanhos funcionais que podem ser encontrados nas tabelas 3 e 4.

Equação 2 – Cálculo de Esforço

$$E = T / PF$$

Onde (E) representa o esforço que é calculado através da divisão do tempo necessário por ponto de função (T) pela quantidade de pontos de função calculada previamente (PF).

Equação 3 – Cálculo de Prazo

$$P = E / (QTP * JT)$$

Onde (P) representa o prazo a ser calculado através da divisão do esforço obtido anteriormente pela produtividade ou seja quantidade de pessoas na equipe (QTP) multiplicada pela jornada de trabalho diária (JT).

Equação 4 – Cálculo de Custo

$$C = E \times CPF$$

Onde (C) representa o custo a ser calculado através da multiplicação do esforço obtido anteriormente pelo custo por ponto de função (CPF).

5 COMPARATIVO ENTRE AS METODOLOGIAS

Baseando no estudo realizado sobre as metodologias COCOMO e APF apresenta-se o comparativo entre as duas apontando-se as principais vantagens e desvantagens encontradas na utilização das mesmas visando descobrir qual é a mais vantajosa para determinada situação, para demonstrar os cálculos serão utilizados o método A e o método B e as estimativas realizadas nas duas metodologias.

a) Projeto

O a ser estimado é referente a um sistema de gestão de contratos que deverá conter as funcionalidades de cadastro (inclusão, alteração e exclusão) de contratos e relatórios que poderão ser visualizados em tela ou gerados em um arquivo de texto.

b) Estimativa na metodologia COCOMO

Para o projeto em questão obtive-se a seguinte estimativa descrita no quadro abaixo:

Método A: COCOMO	
Equações:	$E = ai\ S^{bi}$ $T = ai\ E^{bi}$
Calculo: **Esforço (E)**	$E = 2.4 * (0.5)^{1.05}$ $E = 2.4 * 0.48$ **E = 1.1 pessoas/mês**
Tempo (T)	$T = 2.5 * (1.1)^{0.38}$

	T = 2.5 * 1.04 **T = 2,6 meses**

Vantagens:	Métrica objetiva. Fácil de medir.
Desvantagens:	Dependência dalinguagemde programação. Penaliza softwares pequenos.

c) Estimativa na metodologia de Pontos de Função

Para o projeto em questão obtive-se a seguinte estimativa descrita no quadro abaixo:

Método B: Pontos de Função	
Equações:	PF = (ALI x TF)+(AIE X TF)+(SE x TF)+(CE x TF)+(EE x TF) E = T * PF P = E / (QTP * JT) C = E x CPF
Calculo:	PF = (1 x 7)+(0 X 5)+(1 x 3)+(1 x 4)+(4 x 3) PF = 7+0+3+4+12 **PF = 26 pontos de função** **E = 8 * 26 = 208 horas** **P = 208 / 8 = 26 dias**

	C = 208 x 25 = R$ 5200
Vantagens:	Independe de Linguagem de Programação. Fácil de medir.
Desvantagens:	Difícil de automatizar Melhores resultados em estimativas de softwares de grande porte.

6 CONCLUSÂO

Para a compreensão do tema de estudo a comparação entre métricas de software, pontos de função e Cocomo II, buscou-se inicialmente rever a base conceitual e teórica sobre esta questão.

Assim, a fundamentação teórica permitiu verificar que nos modelos acima citados podemos perceber que em um projeto com cronograma bem especificado e um escopo bem definido o modelo ponto de função melhor se destaca, pois independe da linguagem de programação e apresenta facilidade na medição, demonstrando um melhor resultado em softwares de grande porte, em contrapartida o Cocomo II é uma métrica mais objetiva e sua estimativa depende da linguagem de programação e penaliza softwares de pequeno porte.

Autores como (PRESSMAN, 2010) enfatizam que a métrica ponto de função pode ser usada efetivamente como meio para medir a funcionalidade fornecida pelo sistema, isto permite afirmar que este instrumento proporciona um enorme benefício ás organizações no processo de desenvolvimento de software.

Desta forma, pode-se concluir que o melhor método para gerenciar riscos da estimativa de tamanho em um projeto de software durante o desenvolvimento é a

análise de ponto de função, pois primeiro, o cliente (usuário / especificador) pode aceitar mais facilmente o risco para um dado tamanho de projeto de software (em pontos de função). Segundo, o desenvolvedor pode aceitar mais facilmente o risco para o custo de produção (o custo por ponto de função). A adesão a uma forma consistente de contagem de pontos de função otimiza este relacionamento e facilita o desenvolvimento dentro do prazo.

Diante do exposto, pode-se afirmar que a pergunta de pesquisa do estudo foi respondida e os objetivos alcançados.

REFERÊNCIAS

Livros:

Boehm, Barry W. (1981) “Software Engineering Economics”. Prentice Hall.

IFPUG. Manual de Praticas de Contagem de Pontos de Função. Versão 4.3, 2010.

JALOTE, P. An Integrated Approach to Software Engineering. 3.ed. New York: Springer, 2005, 566p.

PRESSMAN, Roger S. **Engenharia de software: uma abordagem profissional.** 7. ed. NY: AMGH ,2011.

SOMMERVILLE, I. Engenharia de Software. 6ª ed. São Paulo: Addison Wesley, 2003, 592p.

VASQUEZ, Carlos Eduardo; SIMÕES, Guilherme Siqueira; ALBERT, Renato Machado; **Análise de ponto de função: Medição, Estimativas e Gerenciamento de Projetos de Software.** 13. ed. SP: Érica, 2013.

Material da internet:

Aguiar, Maurício (2004) “Estimando os Projetos com COCOMO II”. Disponível em: <http://www.univasf.edu.br/~ricardo.aramos/disciplinas/ES_I_2010_2/COCOMO_II.pdf>,

Acessado em: 13 set. 2014.

JUNIOR, Wayne Teixeira; SANCHES Rosely. SÃO CARLOS. (São Paulo). Modelo de Estimativa de Custo de Software: Cocomo & Cocomo II. São Carlos, 2000. Nº 106.

Disponivel em :
<http://www.icmc.usp.br/CMS/Arquivos/Arquivos_Enviados/BIBLIOTECAS_113_RT_106.pdf > acessado em: 25 mar. 2014.