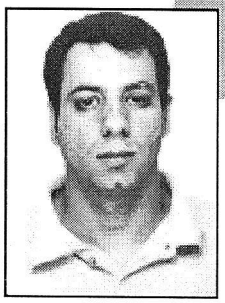
MÉTRICAS DE SOFTWARE – VISÕES E IMPORTÂNCIAS

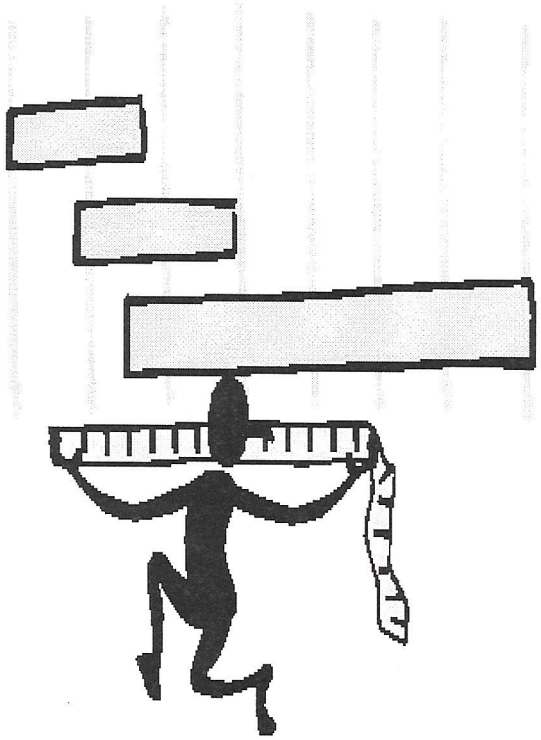


Scharles Martins Baliza
Acadêmico do curso de Ciência da Computação e Monitor da disciplina Linguagens de Programação na UNAMA

MÉTRICAS DE SOFTWARE – VISÕES E IMPORTÂNCIAS

RESUMO:

Visando, sobretudo, à relevância de se estimar, num processo de desenvolvimento de software, argumentos como tempo, gastos, custos, eficiência, entre outros. Este trabalho faz um apanhado de técnicas mais utilizadas nas métricas de software descrevendo suas principais características e criticando ponto de vistas de diversos autores. Classifica tais métricas sob o foco da aplicação e medição, destacando ainda, comparações entre determinados sistemas. Esclarece sobre métricas orientada à função e ao tamanho, desmitificando estimativa de tempo e esforço.



I- INTRODUÇÃO

Efeitos da globalização tornaram o mercado cada vez mais exigente e competitivo, e elegeu o software um dos maiores componentes do orçamento de muitas organizações. Elevando, com isso, a importância de controlar gastos com seu desenvolvimento e analisar sua qualidade. Entende-se por qualidade, não só aquele aplicativo que supre as necessidades do usuário, mas que também é implementado a tempo e de acordo com o orçamento. Neste caso não resta dúvidas da necessidade de se medir o software, com intuito de criar padrões e otimizar esforços no seu desenvolvimento.

E a partir de medições torna-se possível realizar uma das atividades mais fundamentais do processo de gerenciamento de projetos que é o planejamento. A partir deste planejamento, passamos identificar a qualidade de esforços, custos e as atividades que serão necessárias para a realização do projeto. (PRESSMAN, 1995)

O termo métrica de software refere-se à mensuração dos indicadores quantitativos do tamanho e complexidade de um sistema. Estes indicadores são, por sua vez, utilizados para co-relatar contra os desempenhos observados no passado a fim de derivar previsões de desempenho futuro.

A métrica de software tem como princípios especificar as funções de coleta de dados de avaliação e desempenho, atribuir essas responsabi-

lidades a toda a equipe envolvida no projeto, reunir dados de desempenho pertencentes à complementação do software, analisar os históricos dos projetos anteriores para determinar o efeito desses fatores e utilizar esses efeitos para pesar as previsões futuras. Estes princípios nos permite prever o resto do processo, avaliar o progresso e reduzir a complexidade.

II - SISTEMAS MÉTRICOS – HISTÓRICO

Por anos utilizou-se apenas da experiência da equipe técnica envolvida no projeto como base de estimativas na realização de novos projetos, onde se deparava com a grande dificuldade de se estabelecer semelhanças de funcionalidade e tamanho entre os casos vividos e o novo caso. Como conseqüência da falta de margens, obter-se-ia um produto final com deficiências funcionais, prazos atropelados ou atividades não realizadas e custos além do previsto, etc.

"Existem algumas formas de medir sistemas que são aceitáveis e consistentes. Algumas técnicas são implementadas visando mensurar os sistemas e identificar a produtividade na área da informática." (IFPUG,1991) [2].

A. CONTAGENS DE LINHAS DE CÓDIGOS FONTE

"A primeira forma encontrada para medir o tamanho de um sistema foi à contagem de suas linha de código." (IFPUG,1991)

Segundo (CONTE, 1985), esta técnica tem como vantagem à simplicidade, em contrapartida está muito presa à linguagem utilizada, tendo pouco significado para o usuário final. E também afirma que esta técnica só é aplicável após a codificação dos programas, não permitindo estimativas em projetos .

B - SISTEMA MÉTRICO DE HALSTEAD

Em 1972, Maurice Halstead, da Universidade de Purdue iniciou estudos sobre algoritmos tentando testar empiricamente a hipótese de que os operadores (comandos e palavras reservadas) e os operandos (itens de dados) em um programa deviam se relacionar com a quantidade de erros nos algoritmos (HALTEAD,1977).

Dado o sucesso do estudo, a pesquisa continuou em 1977, dando origem a um sistema que consiste em registrar para cada programa desenvolvido o número de operadores e operandos utilizados, permitindo o cálculo do tamanho do programa e o esforço de programação. Essa medida é independente da linguagem de programação, mas está baseada na sintaxe dos programas e não considera seu conteúdo. Outra desvantagem é que seu processo é incompreensível ao usuário final. E assim como na contagem de linha de código, esta técnica só é aplicável após a codificação dos programas, não permitindo estimativas em projetos.

C. ANÁLISE DE PONTOS POR FUNÇÃO

No início da década de 70, pesquisadores do Serviço de Processamento de Dados da IBM, a pedido do grupo de usuários (GUIDE), começaram a analisar centenas de programas para isolar as variáveis críticas, que determinam a produtividade da programação (BRAGA1996) (IFPUG,1991).

[2] INTERNATIONAL FUNCTION POINT USER GROUP (Baseado na Release 3.4 do Manual de Práticas de Contagem do IFPUG).

Em 1979 Allan J. Albrecht (IBM White Plains), prosseguindo essas pesquisas, introduziu uma técnica de avaliação conhecida como FPA - Function Point Analysis. A técnica está baseada na visão externa do usuário, sendo, portanto, independente da linguagem utilizada, permitindo calcular o esforço de programação e auxiliando o usuário final a melhorar o exame e avaliação de projetos.

Em 1986, foi formado o Grupo Internacional de Usuários de FPA (IFPUG - International Function Point User Group) destinado a divulgar informações e novas implementações da técnica a todos os seus associados.

D. COMPARAÇÃO DE TÉCNICAS

Pode-se observar, na ***Tabela 1*** a seguir, um resumo das principais características das técnicas apresentadas nesta seção.

Característica	Linhas de código	Sistema Halstead	Pontos por função
Independência de tecnologia	Não	Sim	Sim
Produção de resultados consistentes	Sim	Sim	Sim
Avaliação por usuário sem conhecimento de PD	Não	Não	Sim
Significância para o usuário final	Não	Não	Sim
Utilizado em estimativas	Não	Não	Sim

Tabela 1 **–Quadro comparativo**

III – SOFTWARE E SUAS MEDIDAS

A palavra métrica subentende um conjunto de medidas e o pessoal de ciências exatas considera medidas como sendo obtidas a partir da comparação com um padrão. Infelizmente, a engenharia de software está longe de ter uma medição padrão amplamente aceita e com resultados sem nenhum fator subjetivo. O mais comum é ter-se dificuldade em concordar sobre o que medir e como avaliar o resultado das medições obtidas.

As métricas de software, do ponto de vista de medição, podem ser divididas em categorias: medidas diretas e medidas indiretas.

Podemos considerar como medidas diretas do processo de engenharia de software o custo e o esforço aplicados no desenvolvimento e manutenção do software e do produto, a quantidade de linhas de código produzidas e o total de defeitos registrados durante um determinado período de tempo. Porém, a qualidade e a funcionalidade do software ou a sua capacidade de manutenção são mais difíceis de serem avaliadas e só podem ser medidas de forma indireta.

Também podemos dividir as métricas de software, sob o ponto de vista de aplicações, em duas categorias: métricas de produtividade e de qualidade. As métricas de produtividade se concentram na saída do processo de engenharia de software e métricas de qualidade indicam o quanto o software atende aos requisitos definidos pelo usuário.

A. MÉTRICAS ORIENTÁDAS AO TAMANHO

Métricas orientadas ao tamanho são medidas diretas do software e do processo por meio do qual ele é desenvolvido.

Este tipo de métrica provoca controvérsias e não são universalmente aceitas como a melhor maneira de se medir o processo de desenvolvimento de software. A maior parte da discussão gira em torno do uso das linhas de código (LOC)

como uma medida-chave. Embora esta métrica possa parecer simples, existe discordância sobre o que constitui uma linha de código. Para a maioria dos pesquisadores, a medida de linhas de código não deveria contar linhas de comentários e linhas em branco, uma vez que estas servem para documentação interna do programa e não afetam a sua funcionalidade. Outros fatores que denigrem o uso desta métrica seria a idéia de que as medidas LOC são dependentes da linguagem de programação utilizada na codificação do projeto, e que elas penalizam programas bem projetados, portanto mais curtos, que elas não podem acomodar facilmente linguagens não-procedurais e que seu uso em estimativas requer um nível de detalhes que podem ser difícil de conseguir, isto é, o planejador deve estimar as linhas de código a ser produzidas muito antes que a análise e o projeto tenham sido construídos.

Por outro lado, os adeptos a este método afirmam que as mesmas são "artefatos" de todos os projetos de desenvolvimento de software, que podem ser facilmente contados, que muitos modelos existentes usam LOC ou KLOC (milhares de linhas de códigos) como entrada-chave e que já existe um grande volume de literatura e dados baseados nas linhas de código.

"A medida de software mais familiar é a contagem de linhas de código".

(CONTE,1985)

Na verdade este tipo de medida é mais utilizado para a obtenção de informações de realização do projeto, sendo muito difícil o seu uso em estimativas. A ***Tabela 2*** mostra um conjunto de métricas de qualidade e produtividade que podem ser desenvolvido com esta técnica :

PRODUTIVIDADE	KLOC/pessoa mês
QUALIDADE	Defeitos/KLOC
CUSTO	$/LOC
DOCUMENTAÇÃO	Paginas/KLOC

Tabela 2 **– Métricas orientadas ao tamanho**

B. MÉTRICAS ORIENTÁ-LAS FUNÇÃO

Em vez de contar as linhas de código, métrica orientada à função concentra-se na funcionalidade do software.

Proposta na década de 70 por pesquisadores da IBM, cujo trabalho era identificar as variáveis críticas que determinam a produtividade da programação.

"Descobriram que poderiam basear a avaliação de um software medindo o valor das funções executadas pelos programas, em vez de utilizar como base o volume ou a complexidade do código dos programas." (BRAGA,1996)

Na indústria há atualmente várias maneiras para se medir tamanho funcional, a mais antiga das quais são os pontos de função. Outras medidas que alegam também medir o tamanho funcional do software incluem Pontos de Função Mark II, Pontos de Função 3D da Boeing e Pontos de Característica. Com finalidade ilustrativa, este artigo focaliza Pontos de Função.

Um dos princípios da análise de pontos por função focaliza-se na perspectiva de como os usuários "enxergam" os resultados que um sistema produz. A análise considera as várias formas com que os usuários interagem com o sistema.

As métricas orientadas à função apresentam vários beneficio, dentre eles podemos citar:

1. Uma ferramenta para dimensionar aplicações;
2. Um veículo para quantificar custos, esforço e tempo;
3. Um veículo para calcular produtividade e qualidade;
4. Um fator de normalização para comparar software.

Tal métrica parece ser útil e funcional para o desenvolvimento tradicional, mas apresenta algumas falhas com o modelo orientado a objeto

(OO), pois alguns atributos do design em OO invalidam o cálculo de alguns pontos por função. As características fundamentais de OO têm efeito de reduzir a validade da contagem de funções para a avaliação de esforço e recurso necessário para a execução de um projeto.

A métrica de pontos por função (FP), assim como as linhas de código (LOC), é controversa. Este sistema é independente da linguagem de programação e se baseia em dados que são conhecidos logo no começo da evolução de um projeto, tornando-se mais atraente como abordagem de estimativa. Porém, a contagem se baseia parcialmente em dados subjetivos, implicando à organização estabelecer um plano de implantação da sistemática de medição, definindo padrões para a contagem, antes do início efetivo da utilização histórica coletadas e armazenadas. Mais adequadamente para a prática de estimativas de software, esta técnica chamou a atenção das organizações de desenvolvimento de software e da academia, sendo formado, em 1986, um grupo internacional de usuários da técnica de pontos por função, chamado IFPUG (International Function Point User Group) destinado a implementar melhorias e disseminar informações da técnica.

Oponentes acham que o método requer alguma prestidigitação em que a computação é baseada em dados subjetivos, que a contagem das informações de domínio (e outras dimensões) podem ser difíceis de coletar depois de terminado o projeto, e que FP não tem significado físico direto. É só um número, assim como metros quadrados pra engenharia civil.

C. MÉTRICAS VOLTADAS PARA ORIENTAÇÃO A OBJETO

Muitas métricas já foram desenvolvidas para gerações passadas de tecnologia e em muitos casos, são usadas até para desenvolvimento orientada a objeto (OO), porém não são muito coerentes, pois a diferença entre sistemas tradicionais e sistemas OO são muito grandes.

Existem várias propostas para métricas OO que levam em consideração as características básicas e interações do sistema como: número de classe, linhas de código por método, profundidade máxima da hierarquia de classe, a relação existente entre métodos públicos e privados, entre outros.

Tais métricas baseiam-se na análise detalhada do design do sistema. Como na técnica de prontos por função, faz sentido adicionar um peso às métricas das classes para produzir uma medida de complexidade do sistema. A maioria das medidas examina atributos em termos dos conceitos de OO, como herança, polimorfismo e encapsulamento. Para tanto, seria necessário coletar um número significativo de contagens, ou seja, seria necessário tomar valores de vários projetos e dimensioná-los selecionando as classes, os métodos e os atributos desejáveis para medir o tamanho e a complexidade de um novo software, o que nos tomaria um longo tempo.

IV – ESTIMATIVA DE TEMPO

Após desenvolver uma estimativa do volume de trabalho a ser realizado, pode-se pensar ser fácil estimar a extensão do tempo a ser exigido pelo projeto. De modo geral a preocupação fica a cargo da relação tempo/pessoal. "A priori" imagina-se, no caso de um projeto estimado em dez pessoas-mês e com uma equipe de cinco pessoas deva gastar dois meses para conclusão, então na possibilidade de disponibilizar-se de somente duas pessoas para o mesmo projeto, gastar-se-ia cinco meses. Contudo anos de dolorosa experiência ensinaram que tal relação não é tão trivial. Duplicar o número de pessoas em um projeto não reduz necessariamente a duração de projeto pela

metade. Muito pelo contrário, se for introduzir mais pessoas num projeto em andamento isso apenas retardará ainda mais o processo, uma vez que estas pessoas deverão receber treinamento adequado e "aprender" todo o projeto desde seu início até a fase atual, e isso consome muito tempo.

A estimativa do esforço é a técnica mais comum para se levantar os custos de qualquer projeto de desenvolvimento de engenharia. Um número de pessoas-dia, pessoas-mês ou pessoas-ano é aplicado à solução de cada tarefa do projeto. Um custo em dólares é associado a cada unidade de esforço e um custo estimado será derivado. Como a técnica LOC (linhas de código) ou FP (pontos-por-função), a estimativa de esforço inicia-se com um delineamento das funções do software obtidas a partir do escopo do projeto. Uma série de tarefas de engenharia de software, análise de requisitos, projeto, codificação e teste, devem ser executada para cada função.

V – CONCLUSÃO

Métricas de software é como uma virtude: todos são a favor mas não fica claro se alguém a pratica. Não existe um mecanismo genérico e preciso para estimativa de custos de software, pois não são apenas fatores técnicos e palpáveis que fazem parte de um projeto, mas também existem pessoas, sentimentos, políticas, crenças, ambiente e outros mais que não se consegem medir e são absolutamente variáveis. Contudo, se um produtor de software não possui um processo para estimativas apropriadas, está com seu processo de desenvolvimento do sistema comprometido. Pois como poderá confiar em um planejamento para construção de um produto baseado em "achômetro"?

Por mais capaz que seja a equipe técnica envolvida no projeto, a dificuldade de se estabelecer tamanho de um projeto de software com base na experiência passada é muito grande, pelo fato da dificuldade de se estabelecer as similaridades e as diferenças entre a funcionalidade dos softwares. Esta dificuldade é que tem levado as organizações a investimentos elevados nesta área, e muita evolução neste assunto é esperada para os próximos anos.

Não só a conclusão do projeto deve ser avaliada, mas todo o seu processo de desenvolvimento, principalmente no ato de coletar dados e requisitos para o desenvolvimento, onde deve-se ter muita coerência e estabelecer a princípio o que se deseja medir. Porque estimar é necessário, mas estimar não é adivinhar.

VI - REFERÊNCIAS

PESSMAM, Roger S. **Software engineering**: a practitioner´s approach, 3 ed. New York: McGraw-Hill, 1995.

PESSMAM, Roger S. **Engenharia de Software**; *MKROON BOOKS;* 4 ed. São Paulo, 2001.

BRAGA, Antônio. **Análise por pontos de função**. Rio de Janeiro : Infobook, 1996.

CONTE, S.D.; SHEN V.Y. **Software Engineering Metrics and Models**. Menlo Park : [s.n.], 1985.

HALTEAD, M.H. **Elements of Software Science.** New York : Elsevier Computer Science Library, 1977.

INTERNATIONAL FUNCTION POINT USER GROUP. **Análise por pontos de função**. [S.l.] : IFPUG, 1991. (Baseado na Release 3.4 do Manual de Práticas de Contagem do IFPUG).