

Métricas: tipos e uso em um processo de desenvolvimento de software

O OBJETIVO DESTA AULA É APRESENTAR OS VÁRIOS TIPOS DE MÉTRICAS E SEU USO NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE.

Introdução

Existe um grande número de métricas que você pode utilizar, mas você deve notar que apenas algumas delas têm sido amplamente utilizadas ou aceitas.

Apesar de tudo, se você construir uma aplicação apoiada por algumas das métricas e modelos disponíveis, escolhidas de forma cuidadosa, elas poderão produzir resultados úteis se forem utilizadas de acordo com o ambiente especificado.

Apresentaremos a seguir alguns tipos de métricas.

Linhas de Código (LOC – *Lines of Code*)

Podemos dizer que, ainda hoje, essa é uma das medidas mais utilizadas para medir o tamanho de um programa.

Ela tem a vantagem de ser de fácil definição e precisão, apesar de que algumas dúvidas podem surgir na contagem do que se considera linhas de código.

Por exemplo, podem surgir dúvidas quanto a considerar ou não linhas em branco, comentários, trechos não executáveis, múltiplas declarações por linha e várias linhas por declaração, assim como a questão das linhas de código repetidas ou reusadas.

O mais usual é contar qualquer linha que não seja em branco ou comentário, independentemente do número de declarações por linha.

Métricas calculadas por este método poderão somente ser comparadas se referirem a mesma linguagem de programação e se o estilo de programação estiver normalizado.

Pontos por Função (PF ou FP- *Function Points*)

É uma medida baseada nos modelos e ciclos de vida tradicionais de desenvolvimento de software, proposta para medir o tamanho estimado do software no início do desenvolvimento.


No modelo proposto para o cálculo de PF, você tem que seguir três passos, conforme descrito a seguir.

No primeiro, você terá que preencher o Quadro 1, apresentado logo mais. Note que nessa tabela há parâmetros ligados a funcionalidades do software a ser desenvolvido, bem como fatores de ponderação relacionados a cada um desses parâmetros.

Tanto os parâmetros como os fatores de ponderação constantes na tabela são propostos pelo modelo e a princípio fixos.

Então, nesse passo, você terá que estimar a quantidade de cada parâmetro previsto para o software a ser desenvolvido e atribuir um fator de ponderação para cada um deles.

Uma grande questão que surge é como minimizar a subjetividade na atribuição desses fatores de ponderação, porque isso pode variar de pessoa para pessoa. O IFPUG (*International Function Points Users Group*) disponibiliza uma série de recomendações visando minimizar essa subjetividade.

		Fator de Ponderação			
Parâmetro	Contagem	Simples	Médio	Complexo	Contagem X Fator de Ponderação
Nº de entradas do Usuário		3	4	6	
Nº de saídas do Usuário		4	5	7	
Nº de consultas do Usuário		3	4	6	
Nº de Arquivos		7	10	15	
Nº de Interfaces Externas		5	7	10	
Contagem Total 					

Legenda: QUADRO 1 ? OBTENÇÃO DA CONTAGEM TOTAL FONTE: [PRESSMAN, 2010]

Uma vez preenchido o Quadro 1, você terá o valor de contagem total, o qual será utilizado no terceiro passo do método de cálculo.

No segundo passo de cálculo dos pontos por função, você terá que avaliar 14 questões (Quadro 2) também relacionadas a funcionalidades do software a ser elaborado.

Para cada uma dessas perguntas (Quadro 2), você terá que atribuir um valor de influência, obedecendo a uma escala proposta pelo modelo (Quadro 3).

1. O sistema exige backup e recuperação confiáveis?	8. Os arquivos são atualizados on-line?
2. É requerida comunicação de dados?	9. Entradas, saídas, arquivos e consultas são complexos?
3. Existem funções de processamento distribuído?	10. O processamento interno é complexo?
4. O desempenho é crítico?	11. O código é projetado para ser reusável?
5. O sistema funcionará num sistema operacional existente e intensamente utilizado?	12. A conversão e a instalação estão incluídas no projeto?
6. São requeridas entrada de dados on-line?	13. O sistema é projetado para múltiplas instalações em diferentes organizações?
7. As entradas on-line requerem que as transações de entrada sejam construídas com várias telas e operações?	14. A aplicação é projetada de forma a facilitar mudanças e o uso pelo usuário?

Legenda: QUADRO 2: 14 PERGUNTAS FONTE: [PRESSMAN, 2010]

Influência	0	1	2	3	4	5
	Nenhuma	Pouca	Moderada	Média	Significante	Essencial

Legenda: QUADRO 3: ESCALA DE INFLUÊNCIA FONTE: [PRESSMAN, 2010]

Para evitar o problema da subjetividade ao avaliar a influência de cada funcionalidade, o IFPUG desenvolveu recomendações a serem seguidas.

No final desse segundo passo, você terá a soma das influências de cada uma das 14 perguntas. O valor dessa soma, obviamente, variará de um mínimo de 0 (todas as 14 perguntas tendo valor 0) a um valor máximo de 70 (todas as perguntas tendo valor igual a 5).

No terceiro e último passo, faz-se uso de uma expressão empírica proposta pelo modelo para obtenção dos pontos por função:

$$PF = Contagem\ Total * \left(0,65 + 0,01 * \sum_{i=1}^{14} F_i \right)$$

De posse do valor de PF, você poderá calcular métricas baseadas em pontos por função (por exemplo, produtividade da equipe, custo por PF, etc.) e fazer estimativas de custo e prazo para entrega de um software, dentre outras utilizações.

Métricas de qualidade de software

Qualidade é uma característica que, nos modelos tradicionais de desenvolvimento de software, pode ser medida em cada fase do ciclo de desenvolvimento de software. Métricas para a qualidade na fase de projeto, por exemplo, foram propostas há décadas.

Os primeiros estudos sobre métricas mostram mais relatos e experiências com métricas de dimensão e complexidade; posteriormente, algumas áreas que tratam as características de qualidade de software foram investigadas.

Corretude, eficiência, portabilidade, confiabilidade, usabilidade, modularidade, grau de reutilização e facilidade de manutenção (manutenibilidade) são características ligadas à qualidade de software.

Métricas de orientação a objetos

Existem métricas voltadas especificamente aos aspectos relevantes no contexto da programação orientada a objetos: métricas de classe, métricas de métodos, métricas de herança, métricas de acoplamento e métricas do sistema (características gerais do software orientado a objetos) são exemplos de características que podem gerar métricas orientadas a objetos.

Medidas aplicadas à metodologia ágil

Caro aluno, a seguir estão listadas e definidas algumas medidas para auxiliar uma equipe na escolha das melhores métricas que possam ser utilizadas nas metodologias ágeis.

Entre as métricas apresentadas, estão exemplos daquelas que podem ser coletadas de forma automatizada, por serem compostas de medidas quantitativas e objetivas.

Destas, algumas já foram citadas anteriormente, como: linhas de código, métodos ponderados por classe, falta de coesão dos métodos, profundidade da árvore de herança, número de filhos, acoplamento aferente, acoplamento eferente.

Além das métricas já apresentadas, podemos citar:

- Total de Linhas de Código de Teste (TLCT): representa o número total de pontos de teste do sistema. Um ponto de teste é considerado como um passo do cenário de um teste de aceitação, automatizado ou como uma linha de código de teste de unidade automatizado, descartando linhas em branco e comentários.
- Número de Linhas Alteradas: representa o número de linhas (não apenas código-fonte) adicionadas, removidas e atualizadas no repositório de código unificado.
- Número de *Commits*: representa o número de *commits* efetuados no repositório de código unificado.
- Estimativas Originais: representa o total de pontos (ou horas) originalmente estimados pela equipe para todas as histórias da iteração.
- Estimativas Finais: representa o total de horas (ou pontos) efetivamente reportadas como gastas para implementar as histórias da iteração.

- Número de Estórias Entregues: representa o número total de estórias implementadas pela equipe e aceitas pelo cliente.
- Número de Pontos Entregues: representa o número total de pontos implementados pela equipe e aceitos pelo cliente.
- Tempo: utilizado no cálculo de diversas métricas, pode ser utilizado como duração (em meses, semanas, dias, etc.) ou como número de iterações.
- Tamanho da Equipe: geralmente utilizado no cálculo de métricas, representa a quantidade de pessoas na equipe.
- Esforço: uma medida que leva em conta o tamanho da equipe durante um certo intervalo de tempo, geralmente medido em pessoas-mês ou pessoas-ano.

Métricas orientadas a caso de uso

A especificação dos requisitos por meio da escrita de casos de uso (*Use Case*) é comum nos projetos de desenvolvimento de software.

Sendo assim, é desejável a utilização de uma métrica de software que leve em conta essa realidade.

Foi provavelmente este desejo e esta necessidade que levaram *Gustav Karner*, da empresa *Objectory* (posteriormente *Rational® Software*) a criar, em 1993, a técnica de Pontos por Caso de Uso (UCP- *Use Case Points*).

Os UCPs são uma adaptação da Análise de Pontos por Função (FPA) para a utilização com casos de uso.

A maioria das características da FPA está presente no UCP, como a contagem de pontos não ajustados e a aplicação de fatores de ajuste.

Assim que os primeiros casos de uso do sistema ficam prontos, a análise já pode ser iniciada.

O formato e a granularidade dos casos de uso podem influenciar muito na contagem dos UCPs.

É neste momento que se faz necessário apontar uma das ressalvas em relação a essa métrica: casos de uso não são produzidos em tamanhos padrões. Por causa disso, alguns desenvolvedores alegam que o seu uso como uma medida de normalização deve ser feito com parcimônia.

Métricas para aplicações web

Assim como os desenvolvedores de outras áreas, os desenvolvedores da área de aplicações Web (*WebApps*) devem fazer uso de métricas para aperfeiçoar os processos e produtos desenvolvidos.

Segundo Pressman (2010), no contexto da engenharia web, as métricas têm três objetivos principais. São eles:

- Fornecer uma indicação da qualidade da aplicação de um ponto de vista técnico.
- Fornecer uma base para estimativa de esforço.
- Fornecer uma indicação da aplicação do ponto de vista do negócio.

As métricas elencadas a seguir são apenas alguns exemplos das várias que podem ser derivadas e utilizadas no desenvolvimento de aplicações web.

- Número de páginas web estáticas (Nsp).
 - Número de páginas web dinâmicas (Ndp).
 - Índice de customização: $C = Nsp / (Ndp + Nsp)$.
 - Número de links de páginas internas.
 - Número de objetos de dados persistentes.
 - Número de sistemas externos interfaceados.
 - Número de objetos de conteúdo estático.
 - Número de objetos de conteúdo dinâmico.
 - Número de funções executáveis.
-
- Páginas vistas (*Pageviews*) – as páginas só devem ser consideradas como vistas se estas forem totalmente mostradas aos visitantes. Essa é uma das razões para se incluir as *tags* das ferramentas de análise no final da página.
 - Visita/Sessão (*Visits/Sessions*) – é contabilizada uma visita sempre que um visitante chega ao seu site. Essa visita se mantém até que ele finalize o browser, ou saia do site, ou ainda se ficar mais de 30 minutos sem nenhuma atividade.

Vimos nessas duas últimas aulas a definição, os princípios e alguns exemplos de aplicações das métricas.

Na próxima aula, falaremos um pouco sobre modelagem de processos.

Agora que você já estudou esta aula, resolva os exercícios e verifique seu conhecimento. Caso fique alguma dúvida, leve a questão ao Fórum e divida com seus colegas e professor.

EXERCÍCIOS (https://ead.uninove.br/ead/disciplinas/impressos/_g/pdsoft80_100/a17ex01_pdsoft80_100.pdf)

REFERÊNCIA

PRESSMAN, R. S. *Engenharia de Software*. 7.ed. São Paulo: McGraw-Hill, 2010.

SOMMERVILLE, Ian. *Engenharia de Software*. São Paulo: Addison-Wesley, 2007.

