

Use métricas adequadas Garanta a qualidade de projeto orientado a objeto

Jose Ricardo Correa Maia

Software é, atualmente, um dos produtos mais utilizados e valorizados no mercado. A preocupação com a qualidade tem se tornado requisito imprescindível e tem constituído a orientação básica para garantir a funcionalidade do *software* com o mínimo de erros, defeitos e maior satisfação das expectativas de qualidade. Apesar da dificuldade em se medir qualidade, sua ausência é facilmente percebida.

Assim, ela tem sido alvo de busca na maioria das atividades que envolvem produtos, e o *software* também deve ter uma garantia de qualidade objetiva e eficiente. Na tentativa de medir a qualidade de um *software* é possível identificar índices por meio de métricas, que têm sido apontadas eficazes como forma quantitativa de avaliar a qualidade, porque permitem comparações.

Este artigo apresenta um estudo de como melhorar a medição da qualidade do gerenciamento de projetos de desenvolvimento de *software* orientado a objeto por intermédio de métricas.

Introdução

O gerenciamento de projeto de desenvolvimento de *software* vem se constituindo peça fundamental na geração de produto com qualidade e ainda para que o projeto seja produtivo, ou seja, *software* que aumente a rentabilidade pela redução de custos em falhas internas e externas e com aumento da satisfação do cliente.

A utilização de métricas vem aumentando consideravelmente e a principal causa é a dificuldade de gerenciar projetos sem a dimensão do que se está gerenciando. Por meio da utilização de métricas consegue-se calcular, com mais precisão e confiabilidade, o tamanho do projeto do *software*.

A medição é a primeira etapa que leva ao controle. Se você não mede algo, não pode entender o processo. Se você não entende o processo, não o controla. Se você não o controla, não consegue aperfeiçoá-lo.

O objetivo deste artigo é recomendar a adoção de métricas para auxiliar a medição da qualidade de projetos de desenvolvimento de *software* orientado a objeto.

O artigo está dividido em tipos de métricas, apresentando as métricas mais adotadas e algumas conclusões.

Tipos de Métricas

Métrica pode ser definida como um atributo mensurável de uma entidade. No caso de um projeto, uma métrica pode ser, por exemplo, o seu tamanho, uma vez que o tamanho pode ser medido.

Uma métrica pode ser obtida da própria entidade, por leitura direta. Nesse caso, a métrica é chamada métrica primitiva. O atributo mensurável, que é obtido por cálculo a partir de métricas primitivas, é conhecido como métrica derivada.

O atributo mensurável que é obtido por cálculo a partir de métricas primitivas é conhecido como *métrica derivada*.

Normalmente, são coletadas métricas medidas diretamente, mas de pouco significado numa interpretação isolada. O número de classes de um modelo, analisado isoladamente, tem pouca relevância. Contudo, pode ser feita estimativa do tamanho do sistema se for combinado, por exemplo, o número de classes com o número de operações e atributos de cada classe. Isso reflete a necessidade de interpretação e indica as várias abordagens para a medição.

Segundo Pressman (1995, apud RIEDER, 2003), as métricas têm como princípios específicos:

- A** Coletar dados para avaliação de desempenho atribuindo essas responsabilidades a toda equipe envolvida no projeto.
- B** Reunir dados de desempenho relativos à complementação do *software*.
- C** Analisar os históricos de projetos anteriores para determinar o efeito desses fatores, prevenindo erros para projetos futuros.

De acordo com (PRESSMAN, 1995) apud (RIEDER, 2003), os principais benefícios das métricas são:

- A** Constatar a qualidade do produto.
- B** Avaliar a produtividade das pessoas.
- C** Ter uma base de estimativas (*baseline*).
- D** Justificar pedidos de recursos e treinamento adicional, entre outros.

As métricas podem ser divididas em: métricas de qualidade, métricas de produtividade, métricas orientadas à função e métricas orientadas ao objeto.

As métricas de qualidade e de produtividade são independentes do tipo de modelagem e análise aplicada no projeto de desenvolvimento do *software*. Todavia, as métricas orientadas à função e as orientadas ao objeto, normalmente são aplicadas de acordo com o tipo de modelagem e análise do projeto.

1 Métricas de Qualidade

Qualidade pode ser definida como conformidade com as especificações e atendimento das necessidades ou expectativas dos clientes.

Feigenbaum (1994, apud MAIA, 2001) define os custos da qualidade como aqueles associados à definição, criação e controle da qualidade, assim como avaliação e realimentação de conformidade com exigências em confiabilidade, segurança e também custos associados às conseqüências provenientes de falha em atender essas exigências, tanto na fábrica como no cliente.

Podem-se definir Custos da Qualidade como aqueles incorridos para garantir e assegurar a qualidade, bem como aqueles decorrentes das perdas, quando essa qualidade não é obtida. O custo da qualidade, segundo Ostrenga (1991, apud MAIA, 2001) constitui uma metodologia que aplica custos para atividades, e outros recursos consumidos em:

- **Conformidade com a especificação da qualidade e redução de variações.**
- **Custo da não conformidade.**

Os custos de prevenção e de avaliação são custos voluntários, e podem ser controlados, por decisão da empresa. Normalmente são incorridos durante a pesquisa e desenvolvimento, planejamento e desenho do produto. Esses custos são considerados custos de conformidade.

Os custos de falhas internas e externas, denominados custos involuntários, ocorrem na fase de produção e vendas.

Os custos da qualidade podem ser adotados como métricas da qualidade. A expectativa é pela redução de custos com eventuais não-conformidades (falhas internas e externas).

Pode-se definir retrabalho como o esforço consumido com mudanças, isso é, a carga de trabalho realizado para analisar, implementar e testar novamente as alterações no *software*. A expectativa é pela diminuição do retrabalho.

Se a tendência do retrabalho é aumentar, pode ser interpretado como um mau sinal sobre a manutenibilidade futura do *software*.

A primeira impressão que se tem ao se falar em defeitos é o número absoluto de defeitos que um *software* possui. Esses defeitos apresentam-se como falhas na operação ou são descobertos durante as atividades de testes.

Falha é o não cumprimento dos resultados esperados, tanto por comparação aos requisitos como por interrupção no funcionamento.

Defeito é a imperfeição na especificação técnica, no projeto de um componente ou na codificação de um *software*, que, em determinadas condições, causa a falha observada.

Assim, ao longo do tempo vão aparecendo falhas que constituem a curva de detecção de falhas. Por sua vez, os defeitos também podem ser analisados construindo curva de detecção de defeitos e curva de densidade de defeitos.

Como, geralmente, os defeitos mais fáceis de detectar são os primeiros a ser encontrados, o trabalho de testes fica mais difícil na medida em que avança. Novas interações ou manutenções freqüentemente introduzem novos defeitos.

Esse padrão de comportamento deve ser observado minuciosamente pelo gerente de projetos para verificar se há tendência em direção à qualidade.

Vale salientar que os valores absolutos são menos importantes do que a tendência, porque uma tendência a zero, mesmo partindo de patamares mais altos, significa um processo que está fazendo o seu trabalho de depuração.

Uma forma de medir a maturidade de um produto é determinar o tempo médio entre falhas (**MTBF** *Mean Time Between Failures*), que reflete a taxa de descoberta de defeitos quando certas condições de operação são satisfeitas. Portanto, essa métrica fornece uma maneira de estabelecer e acompanhar o nível de confiabilidade do *software*.

2 Métricas de Produtividade

Todo projeto deve ser produtivo. Para medir a produtividade de um projeto de software podem ser adotadas métricas, como por exemplo, o número de horas por homem, que aborda a produtividade da mão-de-obra aplicada, tendo também o objetivo de reduzir os custos do projeto.

Outra abordagem de produtividade pode ser considerada a métrica de número de horas por produto gerado (deliverables), que no caso da engenharia de software, é denominado de artefato. Portanto, pode-se medir o número de horas por artefato.

Com o apoio de uma base de informações históricas sobre projetos, podem ser estabelecidas metas para aumentar a produtividade, isto é, reduzir o número de horas por homem ou número de horas por artefato.

A fim de monitorar a produtividade pode ser adotada a métrica de custos, que visa comparar, ao longo do tempo do projeto, o custo planejado com o realizado, através do controle executado pelo gerente de projeto.

3 Métricas Orientadas a Função

Métricas orientadas à função constituem medidas indiretas do *software*.

O FPA (*Function Point Analysis*), ou análise de ponto por função, determina o tamanho e a complexidade de um *software*. Ao invés de contar as linhas de código, essa métrica concentra-se na quantificação da funcionalidade do *software*. Ponto por função baseia-se na visão do usuário e mede o que é um sistema, o seu tamanho funcional. Também pode medir a relação do sistema com usuários e com outros sistemas.

A contagem dos pontos por função se dá sobre cinco elementos funcionais básicos de um sistema modelado:

- **Arquivos Lógicos Internos (ALI).**
- **Arquivos de Interfaces Externas (AIE).**
- **Entradas Externas (EE).**
- **Saídas Externas (SE).**
- **Consultas Externas (CE).**

Use métricas adequadas Garanta a qualidade de projeto orientado a objeto

Jose Ricardo Correa Maia

Para essa contagem deve ser considerada a complexidade de cada elemento funcional. A complexidade dos Arquivos Lógicos Internos e dos Arquivos de Interfaces Externas leva em consideração o número de registros lógicos e o número de itens de dados referenciados.

A complexidade das Entradas Externas, Saídas Externas e Consultas Externas é baseada no número de arquivos referenciados e no número de itens de dados referenciados. No manual de práticas de contagem de pontos por função do *International Function Point Users Group (IFPUG)* são apresentadas tabelas e diretrizes para a determinação da complexidade dos elementos funcionais.

Os elementos funcionais são totalizados para a obtenção dos pontos por função não ajustados (PFNA).

O fator de ajuste de pontos por função (FAPF) é calculado a partir de **14 (quatorze) características gerais** que permitem uma avaliação do nível de influência das funcionalidades do *software*:

- | | |
|--|---------------------------------------|
| A Comunicação de dados. | H Processamento complexo. |
| B Processamento distribuído. | I Reutilização de código. |
| C Desempenho. | J Facilidade de implantação. |
| D Utilização do equipamento. | K Facilidade operacional. |
| E Volume de transações. | L Eficiência do usuário final. |
| F Entrada de dados on-line. | M Multiplicidade de locais. |
| G Atualização de dados on-line. | N Facilidade de mudanças. |

A cada característica atribui-se um peso de 0 (zero) a 5 (cinco), conforme o nível de influência no *software*. Somando-se o nível de influência de cada característica, obtem-se o nível de influência geral no *software* (NIGS).

4

Segundo (PINHEIRO, 1998) apud (MIRANDA, 2002) essas características influenciam em até 35% o tamanho do projeto de *software*, ou seja, o projeto pode variar de 0,65 até 1,35.

O fator de ajuste de pontos por função é obtido pela fórmula:

$$\text{FAPF} = \text{VTPS} + (\text{NIGS} * 0,01)$$

O total de pontos por função ou pontos por função ajustados (PFA) é o resultado da multiplicação do número de pontos por função não ajustados (PFNA) pelo fator de ajuste de pontos por função:

$$\text{PFA} = \text{PFNA} * \text{FAPF}$$

Assim, alguns softwares que a princípio parecem pequenos, quando em desenvolvimento, mostram-se muitas vezes maiores do que o inicialmente previsto, podendo afetar drasticamente o gerenciamento do projeto.

Onde
FAPF = Fator de Ajuste de Pontos por Função
VTPS = Variação do Tamanho do Projeto de *Software* (normalmente é adotado 0,65)
NIGS = Nível de Influência Geral no *Software*

Onde
FAPF = Fator de Ajuste de Pontos por Função
PFA = Pontos por Função Ajustados
PFNA = Pontos por Função Não Ajustados

4 Métricas Orientadas ao Objeto

Software orientado a objeto é baseado na composição e interação entre diversas unidades chamadas objetos.

Objeto é uma instância de uma classe. Um objeto é capaz de armazenar estados através de seus atributos e reagir a mensagens enviadas a ele, assim como se relacionar e enviar mensagens a outros objetos. Classe representa um conjunto de objetos com características afins. Uma classe define o comportamento dos objetos, através de métodos, e quais estados ele é capaz de manter, através de atributos.

Classe representa um conjunto de objetos com características afins. Uma classe define o comportamento dos objetos, por meio de métodos, e quais estados ele é capaz de manter, através de atributos.

Mensagem constitui uma chamada a um objeto para invocar um de seus métodos, ativando um comportamento descrito por sua classe, podendo mudar o valor de um ou mais atributos, alterando o estado de um objeto.

Método é uma rotina que é executada por um objeto ao receber uma mensagem. Ele pode ser público, quando é utilizado por vários aplicativos (*softwares*) ou privados, quando criado ou utilizado pelo aplicativo (*software*) do projeto.

Atributo significa o elemento que define a estrutura de uma classe. Os atributos também são conhecidos como variáveis de classe, e podem ser divididos em dois tipos básicos: atributos de instância e de classe. Os valores dos atributos de instância determinam o estado de cada objeto. Um atributo de classe possui um estado que é compartilhado por todos os objetos de uma classe.

Conforme Gomes (2001), existem várias propostas para métricas orientadas ao objeto que consideram as características e interações do sistema. Tais métricas baseiam-se na análise detalhada do *design* do sistema. Para efetivação de uma métrica orientada a objetos, é essencial o acréscimo de um peso às características das classes, o qual produzirá um fator de ajuste de complexidade orientado a objeto (FACOO), assim como se faz com os parâmetros de métrica de ponto por função. Deve-se dar uma maior importância às características de reusabilidade de código produzido, assim como considerar a utilização de componentes pré-implementados, quando da definição das bases para o fator de ajuste de complexidade orientado a objeto. A maioria das medidas examina atributos em termos dos conceitos de Orientação a Objeto, como herança, polimorfismo e encapsulamento.

Herança constitui o mecanismo pelo qual uma classe pode estender outra classe, aproveitando seus comportamentos (métodos) e estados possíveis (atributos).

Polimorfismo é a ação que permite a um objeto se comportar de acordo com sua classe. Dessa forma, é possível se relacionar com objetos de classes diferentes, enviar mensagens iguais e deixar o objeto se comportar de acordo com sua classe.

Já o encapsulamento consiste na separação de aspectos internos e externos de um objeto. Esse mecanismo é amplamente utilizado para impedir o acesso direto ao estado de um objeto, disponibilizando externamente apenas os métodos que alteram esses estados.

O número de pontos por função orientados a objeto (PFOO) pode ser obtido aplicando-se a fórmula:

$$\text{PFOO} = \text{PFNA} \times (\text{VTPS} + \text{FACOO} \times 0,01)$$

Onde	FACOO = Fator de Ajuste de Complexidade Orientado a Objeto
	PFNA = Pontos por Função Não Ajustados
	PFOO = Pontos por Função Orientados a Objeto
	VTPS = Variação do Tamanho do Projeto de <i>Software</i> (normalmente é adotado 0,65)

Ressalta-se que é fundamental realizar medições em um número significativo de softwares já desenvolvidos, selecionando as classes, os métodos e os atributos desejáveis para a obtenção dos dados históricos necessários ao embasamento das estimativas dos próximos softwares a serem desenvolvidos. É possível que seja mais fácil colher os dados durante os projetos atuais de desenvolvimento, do que tentar obtê-los de softwares desenvolvidos anteriormente.

É interessante saber como o projeto está se comportando no tempo, com o objetivo de averiguar se o produto será concluído no prazo planejado. Para avaliar o progresso do projeto, deve-se medir quanto de trabalho é realizado em relação ao programado.

Para isso, podem ser aplicadas as seguintes métricas:

- A** Número de classes.
- B** Número de casos de uso.
- C** Pontos por casos de uso.
- D** Número de casos de teste.
- E** Número de métodos.
- F** Média de métodos por classe.
- G** Média de linhas de código por métodos.
- H** Relação existente entre métodos públicos e privados, entre outros.

Conclusões

As medições e as métricas permitem um melhor entendimento do processo utilizado para desenvolver um produto, assim como uma melhor avaliação do projeto e do próprio produto. As medições podem permitir melhorias no processo, aumentando a produtividade e comparações entre projetos de desenvolvimento.

Constata-se que a partir da utilização de um planejamento de garantia de qualidade que contenha o estabelecimento de métricas, adquire-se um histórico e, por meio dele, a qualidade e produtividade do projeto irá aumentar, diminuindo assim a margem de erros.

A princípio, é necessário determinar o que se quer medir, definindo como os dados serão coletados. Essas decisões devem ser compatíveis com o projeto de desenvolvimento.

Um modelo de métrica não vai solucionar de imediato os problemas do desenvolvimento, no entanto, as métricas devem ser utilizadas para tornar possível o entendimento do processo, para facilitar a previsão de suas fases e mostrar como controlá-las.

Verifica-se, então, que não existem modelos consistentes de métricas para aplicação orientada a objeto. Uma prática comum é aplicar as métricas de tamanho e complexidade que foram desenvolvidas para *softwares* procedimentais, tratando métodos da mesma forma que procedimentos, adequando a utilização à orientação a objeto.

Recomenda-se não adotar apenas métricas primitivas, mas estabelecer métricas derivadas com a composição de várias métricas obtendo, assim, uma abordagem mais específica, de acordo com a necessidade do projeto, com vistas à criação de uma base histórica para poder realizarem-se comparações entre os projetos.

É conveniente realizar a coleta das métricas ao longo do desenvolvimento do projeto. Então, deve-se eleger a abrangência da coleta, isto é, estabelecer se as métricas serão coletadas por semana, por mês, por iteração, por módulo do sistema, por equipe etc. É indicada a adoção da coleta por iteração para o monitoramento do projeto e a coleta por módulo ou por classe para avaliação da qualidade do produto.

Vale salientar que quando se adota um processo iterativo, a coleta e a interpretação das métricas não deve ser feita da maneira usada num processo cascata para não gerar problemas e preocupações desnecessárias. Num processo cascata, para se medir o progresso do projeto, a disciplina de requisitos deve estar completa quando o projeto estiver com 25% de seu tempo. Essa realidade não é aplicada em um processo iterativo, uma vez que a disciplina de requisitos fica completa somente na última iteração. Fato que normalmente não ocorre na marca de 25% do tempo do projeto.

É proposta a utilização da medição dos custos da qualidade, número de não conformidades do processo por FPA, número de horas por classe, número de horas de retrabalho por FPA ou por método, tempo médio entre falhas, número de defeitos por classe, método ou FPA, número de solicitações de mudança por FPA ou classe (para medir as alterações de escopo), número de classes privadas por classes públicas e percentual de reutilização de métodos/componentes, entre outros.

Conclui-se que existe a preocupação com a qualidade dos projetos de desenvolvimento de *software* orientado a objeto e que as métricas são ferramentas que o gerente de projeto deve utilizar para diagnosticar e corrigir o rumo do projeto. Portanto, deve-se por em prática a adoção de métricas orientadas a objeto, combinadas com os demais tipos de métricas.

Por se tratar de mudança de paradigma, essas métricas ainda não estão consistentes, entretanto, o exercício constante das medições formará uma base para os ajustes necessários no modelo adotado.

Referências Bibliográficas

FEIGENBAUM, Armand V. **Controle da Qualidade Total: Gestão e Sistemas**. Vol 1. São Paulo, Makron Books, 1994.

GOMES, Alvaro Eduardo. **Métricas e estimativas de software o início de um rally de regularidades**. (2001). Available: www.apinfo.com/artigo44.htm. [abr. 2001].

MAIA, José Ricardo Correa (2001). **A gestão dos custos da qualidade otimizando o processo de garantia da qualidade**. VIII Congresso Brasileiro de Custos - UNISINOS - Rio Grande do Sul. [out. 2001].

MELLO FILHO, Moacyr Cardoso (2002). **Aplicando as sete métricas de controle de projeto**. *Rational Software White Paper*. Revisão 1.1. Available: www.sit.com.br/textos/7metricas.pdf. [jan. 2002].

MIRANDA, Everton Alves (2002). **Uma análise de métricas de software orientadas a função e sua aplicação ao desenvolvimento orientado a objetos**. *Vértices*. Ano 4 nr. 1. Available: www.cefetcampos.br/publicacoes/vertices/v4n1/07-%20uma_analise_de_metricas.pdf. [jan. 2002].

OSTRENGA, Michael R. **Return on Investment through the Cost of Quality**. *Journal of Cost Management for the Manufacturing*, v.5, n.2, p.37- 44, sum., 1991.

PINHEIRO, C. A. R. (1998). **Análise de pontos por função como métrica de desenvolvimento**. *Developers Magazine*. Rio de Janeiro. [nov. 1998].

PRESSMAN, R. S. **Engenharia de Software**. São Paulo: Makron Books, 1995.

RIEDER, Clarice (2003). **Estudo sobre a métrica pontos por função**. SIRC/RS'2003 - II Simpósio de Informática da Região Centro do RS (UNIFRA). Santa Maria Rio Grande do Sul. Available: www.inf.unifra.br/tfg2002.sis_info/tfg.2002.35.pdf. [ago.2003].



José Ricardo Correa Maia é profissional com 16 anos de experiência em desenvolvimento e qualidade de *software*. Atuou como analista de sistemas na MCI Informática, analista de negócios e analista de testes na Datasul S.A., coordenador da qualidade na Softplan/Poligraph e atualmente é especialista de processos na Euax Gestão de Projetos, atuando no gerenciamento da qualidade de projetos e processos. Graduado em processamento de dados (UDESC) e pós-graduado em administração industrial (UNIVILLE). Autor dos artigos **Reduzindo custos de falhas com a implantação do processo de garantia de qualidade** (III Simpósio Internacional de Melhoria de Processo de Software SIMPROS - 2001) e **A gestão dos custos da qualidade otimizando o processo de garantia da qualidade** (VIII Congresso Brasileiro de Custos - 2001).