Métricas de Software

João Carlos Testi Ferreira*

22-07-2016

RESUMO

Este texto explica alguns conceitos necessários para a compreensão da importância das métricas no desenvolvimento de software. O objetivo é apresentar o que é medir, porque medimos e as formas que podemos obter medidas. São comentados os modos de medição e suas características. Isso permite um introdução aos conceitos de métricas e tornará mais fácil o entendimento das várias formas de obter medidas com uma visão mais crítica do assunto.

Introdução

A construção de um software envolve a execução de muitas atividades que podem acontecer de forma sequencial ou paralela. Em sua construção podemos usar diferentes linguagens, tecnologoias ou métodos. Para podermos avaliar a melhor forma de fazer isso precisamos de medidas que nos auxiliem na distribuição destas atividades para o atendimento das necessidades envolvidas em um projeto, como por exemplo prazo e custo. Outro fato importante em que a medida é importante é na avaliação do que é feito. Martins (2007), p. 23, comenta que para podermos avaliar a eficácia de um projeto comparamos uma situação atual com dados de referência. Com isso podemos estabelecer nosso nível de qualidade. Por exemplo, encontrar cinco defeitos em algo não nos diz muito, mas se conseguirmos ter uma medida e definirmos a quantidade de erros em função de determinada medida poderemos ter critérios de comparação. A diferença dos sistemas, no entanto, dificultam muito a forma de medi-los, em especial se o que queremos é derivar informações como prazo e custo.

Sommerville (2011), p. 465-466, cita que a medição busca derivar um valor numérico ou perfil para um atributo de um componente de software, sistema ou processo. É a comparação entre estes valores que permitem que sejam tiradas conclusões a respeito da qualidade do software, a eficácia dos métodos, das ferramentas e dos processos de software.

Como medir e o que medir em um software é tarefa muito discutida na Engenharia de Software, outro problema significativo aqui é a limitação de como medir de forma confiável. Veremos um pouco sobre as formas de medir para uma compreensão adequada do problema e para identificar a importância de medição em um projeto.

*joao.c.ferreira@edu.sc.senai.br

A medição por si só é de pouco valor, e ela normalmente é acompanhada por estimativas. Prever o futuro ou identificar pontos de melhoria são os principais usos das medidas usadas nas estimativas. Dificilmente fazemos medidas apenas pelas medidas, mas sim porque precisamos fazer comparações ou alguma previsão de tendência que nos diga como algo estará no futuro. Muitas medidas são bastante subjetivas e se justificam pelo seu uso ou pela dinâmica que ela proporciona. BrooksJr (1995), p. 14, no entanto, nos esclarece que nossas técnicas de estimativas são pouco desenvolvidas, confundem esforço com progresso e o progresso é mal monitorado. Para melhorar isso precisamos de boas medidas e um bom acompanhamento da evolução da construção do software.

Fenton e Pleeger (1997), p. 14-15, relaciona métricas de software com várias atividades que permitem acompanhar a evolução do software. Ele apresenta uma figura que estrutura um modelo de produtividade.

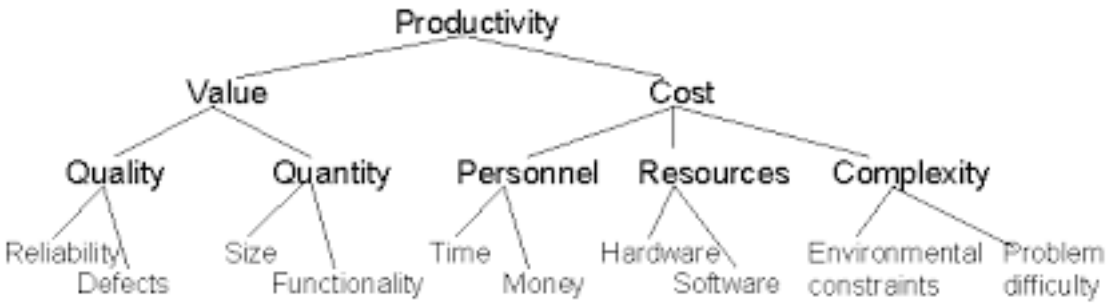

Figura 1 – Modelo de produtividade

Extraído de (FENTON; PLEEGER, 1997).

No modelo podemos observar a avaliação de valor e custo. Cada medida obtida tem seu foco em alguma avaliação que se deseja fazer com relação ao projeto ou ao software. A produtividade é a taxa de saída versus entrada em determinado tempo.

$$\boxed{produtividade = \frac{entradas}{saidas}}$$

Para podermos usar esta fórmula para produtividade, a escala de entrada e saída deve ser a mesma.

1 Medição

1.1 Definição

Para definir medição alguns autores iniciam explicando medições físicas. Pressman (2011), p. 538, por exemplo, explica que as medidas na Engenharia de Software não são com na física. Na física podemos obter estas medidas diretamente. Alguns exemplos desse tipo de medida são a massa, velocidade ou temperatura. Como as medidas de software normalmente são feitas de forma indireta há muito debate sobre elas.

Sommerville (2011), p. 466, nos diz que uma métrica pode ser relativa a características do sistema, documentação ou mesmo do processo. Uma medida pode ser obtido de

forma objetiva. Alguns exemplos de métricas apresentados por ele são o número de linhas de código do produto ou o número de defeitos identificados em um produto entregue.

Pressman (2011), p. 539, diferencia medidas, métricas e indicadores. Essa diferenciação é importante porque muitas vezes estes termos são usados de forma intercambiável, tratadas como sinônimos, porém elas não dizem da mesma coisa, e isso pode causar alguma confusão. Pressman (2011), p. 539, explica que na Engenharia de Software usamos *medida* em relação a quantidade de extensão, qualidade, capacidade ou tamanho de um atributo e a *métrica* é usada para definir o grau que um sistema, componente ou processo possui determinado atributo. Para relacionar medida e métrica Pressman (2011) nos diz que:

> Quando é coletado um único ponto de dado (por exemplo, o número de erros descobertos em um componente de software), foi estabelecida uma medida. A medição ocorre como resultado da coleção de um ou mais pontos de dados (por exemplo, um conjunto de revisões de componente e testes de unidade são investigados para coletar medidas do número de erros para cada um). Uma métrica de software relaciona as medidas individuais de alguma maneira (por exemplo, o número médio de erros encontrados por revisão ou o número médio de erros encontrados por teste de unidade). (PRESSMAN, 2011), p. 539.

O termo *indicador* é uma métrica ou um conjunto delas que nos orienta sobre o projeto ou o produto com relação a algum objetivo. Usamos indicadores para saber se estamos indo no caminho certo ou se há algum desvio. Se temos como indicador, por exemplo, o número de erros encontrados a cada entrega, queremos que este indiocador diminua. Se este indicador aumentar temos que tomar alguma ação para melhorar o processo.

1.2 Medição Objetiva

Um problema comum é estabelecer a melhor forma de medir o software. Uma forma mais direta é contar o número de linhas de código. Vasquez, Simões e Albert (2007), p. 33, comenta que o que parece fácil pode nos induzir ao erro. O uso da contagem de linhas de código precisa ser bem definido. O autor cita algumas coisas que precisam ser definidas:

- A inclusão na contagem de linhas de comentários, linhas em branco ou comandos nulos;
- A inclusão na contagem de diretrizes de compilação;
- A contagem de múltiplos comandos ou declarações em uma única linha como várias linhas, um para cada comando ou declaração;
- A contagem de uma única linha nos casos em que o único comando ou declaração é expresso em múltiplas linhas;
- A inclusão na conta de delimitadores de blocos de comandos nos casos em que de fato haja mais de um comando;
- A desconsideração de delimitadores de blocos de comandos nos casos em que sua utilização seja opcional. (VASQUEZ; SIMõES; ALBERT, 2007), p. 34.

Outro fato citado pelo autor é que uma medição não deve ser somente feita, mas estimada. A estimativa de quantas linhas de código uma aplicação terá (previsão) pode ser muito difícil, mesmo por comparação com outros sistemas semelhantes.

Como forma de resolver isso, Allan Albrecht definiu conceitos que serviram de base para uma medição funcional: a análise de pontos de função. Esta é uma técnica que oferece

meios de medir as funcionalidades de um software do ponto de vista do usuário. Com esta abordagem tornamos a medição independente da tecnologia empregada em sua construção.

Há outras abordagens de medição funcional, como o COSMIC (Common Software Measurement International Consortium), que possui uma visão mais moderna de medição, separando as partes que serão medidas e não o considerando como um todo, e os Pontos por Caso de Uso.

Conforme o que foi apresentado, podemos medir usando métricas que se relacionam com o tamanho do código (linhas de código), mas que são fortemente dependentes de linguagem e métricas associaas a funções, que independem da linguagem.

1.3 Estimativa

Conforme comentado, uma das razões mais comuns para medirmos é para auxiliar no processo de estimar. As estimativas nos orientam sobre nosso estado atual e nos auxiliam a planejar com mais informação, tendo mais confiança em definir resultados futuros.

Alexander (2011), p. 11-12, cita três objetivos para as métricas: auxiliar no controle e compreensão das situações ocorridas, melhorar a comunicação dos fatos ocorridos entre as pessoas e identificar pontos de melhoria. Usamos as métricas, portanto, para estimar (prever) o futuro, obtendo controle e identificando formas de trabalhar melhor, com um resultado mais eficiente.

Sommerville (2011), p. 442, declara que para fazer estimativas podemos usar dois tipos de técnicas: baseadas em experiência e modelagem algorítmica de custos. Quando adotamos técnicas baseadas em experiência usamos nosso conhecimento prévio com outros projetos. Na modelagem algorítmica de custos é necessário obter estimativas de atributos do produto para que ela possa ser realizada.

Fairley (2009), p. 209-213, apresenta três princípios da produção de estimativas:

- A estimativa do projeto é uma projeção das experiências passadas para o futuro, ajustando-se para abranger as diferenças entre o passado e o futuro.
- Todas as estimativas baseiam-se em pressupostos que devem ser realizados e um conjunto de restrições que devem ser satisfeitas.
- Os projetos devem ser reavaliados periodicamente, conforme cresce o seu entendimento e na medida em que seus parâmetros mudam.

A estimativa possui precisão bastante limitada, assim, ela poderá ser tão precisa quanto possível ou desejado. O que vai nos orientar sobre o quão precisa ser a estimativa é a razão pela qual precisamos dela. A precisão não depende somente da importância da estimativa, mas de sua viabilidade e de seu custo. Estimativas devem ser baratas e simples. Devemos preferir dados objetivos para derivar nossas estimativas.

Usamos estimativas para vários fins, por exemplo, para previsão de custos, prazos ou esforço. Normalmente previsões de custos são derivadas de prazo e esforço. Previsões de prazo estão associadas a cronogramas, distribuição de tarefas entre as pessoas em determinado calendário, e previsão de esforço associados ao tamanho do software. Assim, para estabelecer estas estimativas partimos de medidas de tamanho do software. Cabe destacar que tamanho não é capaz de relacionar-se diretamente com esforço.

Para poder relacionar tamanho e esforço e definir estimativas frequentemente se usa dados históricos. Com isso podemos *estimar* o esforço necessário para construir software por unidade de medida. Para isso ou usamos a comparação com projetos anteriores ou alguma técnica que defina um algoritmo para estimar. Um exemplo de algoritmo para estimar é o COCOMO. Outra forma comum é por meio empírico, baseado em opinião de um especialista. Uma técnica deste tipo é a Delphi.

2 Ágil

2.1 Métricas

No ágil as formas mais comuns de se fazer estimativas envolvem *pontos de história* e o *planning poker*. Os pontos de história são definidos pelos desenvolvedores, que estimam estes pontos com base na complexidade de cada história. Observe que há total subjetividade nessa definição, e a quantidade de pontos para cada história vai depender de desenvolvedor para desenvolvedor. Para reduzir esta variabilidade costuma-se trabalhar com a média dos pontos obtidos pela história dos integrantes da equipe.

Rasmusson (2010), p. 126, conta um caso retirado de um livro sobre a sabedoria das multidões. Um experimento foi realizado em uma feira de uma cidade na Inglaterra, em que havia um touro esquartejado e as pessoas deveriam estimar seu peso. O esperado era que especialistas (açougueiros) fossem mais precisos na estimativa, mas o que se observou foi que a estimativa do público em geral foi a melhor, com precisão de libras. Essa é o caso que o autor usa para explicar o planning poker. Ele comenta que o planning poker usa o conhecimento coletivo para dar adequação em suas medidas. As histórias de usuário são apresentadas e cada desenvolvedor usa uma carta para definir seu tamanho. Cartas de planning poker possuem valores 0, 1/2, 1, 2, 3, 5, 8, 13, 20, 40 e 100. Depois que os desenvolvedores definiram o valor para a história, caso seja o mesmo (ou muito parecido) o valor é assumido, senão eles fazem rodadas de discussão para chegarem a um consenso do tamanho. O autor cita que cartas de valor 8 ou maiores não são normalmente usadas.

O uso de pontos de história é muito usado no ágil para determinar a velocidade da equipe e planejar o que cabe em cada iteração, dado que normalmente as iterações possuem tamanho (tempo) definido. Aqui o mais importante é conhecer a equipe, medidas obtidas da própria equipe fornecem segurança e comprometimento. Essas medidas, no entanto, não são úteis para outras equipes — cada equipe possui suas características que fazem com que suas medidas sejam diferentes.

Considerações finais

Medimos porque precisamos entender o processo em execução. As medidas nos auxiliam na comunicação, oferece mais consistência. Precisamos de valores para poder saber o que vai bem e o que não vai bem. Boas medidas podem nos orientar na localização de problemas, facilitando a busca de soluções.

Algumas medidas são mais precisas que outras e a escolha da forma de medir vai depender da razão pela qual precisamos da medida. Em alguns casos uma medida pode ser melhor do que outra simplesmente pelo fator de agregação que ela proporciona para a equipe.

As medidas devem ser fáceis de se obter e seu significado deve ser claro. A necessidade de precisão da medida pode exigir mais complexidade na sua obtenção, mas ainda dentro de um custo justificável.

REFERÊNCIAS

MARTINS, J. C. C. *Técnicas para Gerenciamento de Projetos de Software*. Rio de Janeiro: Brasport, 2007. ISBN 9788574523088. Citado na página 1.

SOMMERVILLE, I. *Engenharia de Software*: Uma abordagem profissional. 9. ed. São Paulo: Pearson Prentice Hall, 2011. ISBN 9788579361081. Citado 3 vezes nas páginas 1, 2 e 4.

BROOKSJR, F. P. *The Mythical Man-Month*: Essays on software engineering. Anniversary edition. Boston: Addison Wesley Longman, 1995. ISBN 0201835959. Citado na página 2.

FENTON, N. E.; PLEEGER, S. L. *Software Metrics*: A rigorous and practical approach. 2. ed. Boston: PWS Publishing Company, 1997. ISBN 0534954251. Citado na página 2.

PRESSMAN, R. S. *Engenharia de Software*: Uma abordagem profissional. 7. ed. Porto Alegre: AMGH, 2011. ISBN 9788580550443. Citado 2 vezes nas páginas 2 e 3.

VASQUEZ, C. E.; SIMõES, G. S.; ALBERT, R. M. *Análise de Pontos de Função*: Medição, estimativas e gerenciamento de projetos de software. 6. ed. São Paulo: Editora Érica, 2007. ISBN 9788571948990. Citado na página 3.

ALEXANDER, J. *Codermetrics*: Analytics for improving software teams. Sebastopol: O'Reilly Media, 2011. ISBN 9781449305154. Citado na página 4.

FAIRLEY, R. E. *Managing and Leading Software Projects*. New Jersey: John Wiley and Sons, 2009. ISBN 9780470294550. Citado na página 4.

RASMUSSON, J. *The Agile Samurai*: How agile masters deliver great software. Dallas: Pragmatic Bookshelf, 2010. ISBN 9781934356586. Citado na página 5.