UNIDADE 02

Como Escolher Métricas de Software

META

Capacitar o estudante a compreender e aplicar os princípios fundamentais da mensuração em engenharia de software, reconhecendo os diferentes tipos de entidades, atributos e abordagens utilizadas para definir, aplicar e validar métricas de forma alinhada aos objetivos do projeto.

OBJETIVOS

Geral

Compreender os fundamentos teóricos e práticos da medição em software, incluindo a classificação das entidades mensuráveis, os tipos de atributos, a abordagem GQM para definição de métricas e os critérios de validação de sistemas de medição e predição.

Específicos

- Identificar e classificar as entidades de software;
- Diferenciar atributos internos e externos;
- Reconhecer as dificuldades na medição de atributos externos e os cuidados necessários ao utilizar métricas internas como aproximação;
- Aplicar a abordagem *Goal-Question-Metric* para estruturar a definição de métricas a partir de objetivos do projeto;
- Entender o conceito de validação de medições, distinguindo os critérios de validade em sentido estrito e em sentido amplo.

RESUMO

A unidade aborda os fundamentos da mensuração em engenharia de software, iniciando com a classificação das entidades de software que podem ser alvo de medições: processos, produtos e recursos. Cada uma dessas entidades representa diferentes elementos dentro do ciclo de vida

do software — os processos referem-se às atividades realizadas (como testes ou revisões), os produtos dizem respeito aos artefatos gerados (como código e documentação), enquanto os recursos envolvem os insumos utilizados (como tempo, equipe e ferramentas).

A unidade também diferencia dois tipos fundamentais de atributos: os atributos internos, que podem ser medidos diretamente a partir da entidade, como tamanho do código ou complexidade; e os atributos externos, que só podem ser avaliados por meio do comportamento da entidade em seu ambiente, como confiabilidade e usabilidade. Embora os atributos externos sejam muitas vezes mais relevantes para usuários e gerentes, sua medição é mais complexa e geralmente ocorre nas fases finais do desenvolvimento. Por isso, é comum usar atributos internos como aproximações — o que pode gerar imprecisões se não for feito com cautela.

Para guiar o processo de escolhas das métricas relevantes para a organização, a unidade apresenta a abordagem *Goal-Question-Metric*, que estrutura o processo de medição em três níveis: definição de um objetivo claro, formulação de perguntas baseadas nesse objetivo e identificação de métricas capazes de responder essas perguntas de maneira quantitativa. Essa abordagem garante que as métricas estejam sempre alinhadas com as metas organizacionais e evitem a coleta irrelevante de dados.

Por fim, a unidade discute a validação de medições e sistemas de predição. Uma métrica só é útil se for válida, isto é, se representar corretamente o atributo que se propõe a medir. A validação pode ocorrer em dois níveis: sentido estrito, em que se verifica a coerência do sistema de medição com a teoria da mensuração; e sentido amplo, que envolve também a avaliação da capacidade de predição baseada nessa métrica.

1. INTRODUÇÃO

Em um mundo cada vez mais orientado por dados, medir não é

apenas uma possibilidade, mas uma necessidade para quem busca qualidade, eficiência e controle no desenvolvimento de software. Porém, o simples ato de medir só se torna útil quando se sabe o que medir, por que medir e como interpretar os resultados. Essa é a proposta desta unidade: apresentar ao estudante os fundamentos conceituais e práticos que sustentam a atividade de mensuração em engenharia de software.

Você aprenderá a diferenciar os tipos de entidades que podem ser medidas (como processos, produtos e recursos), os atributos internos e externos associados a essas entidades, e compreenderá como a abordagem *Goal-Question-Metric* pode guiar o processo de definição de métricas com foco em metas reais e estratégicas. Além disso, serão discutidos aspectos fundamentais da validação das medições, ou seja, como garantir que as métricas utilizadas realmente representem aquilo que se deseja avaliar ou prever.

Ao final desta unidade, você será capaz de compreender não apenas o “como” medir, mas, sobretudo, o “porquê” e “com qual finalidade”. Essa base teórica é essencial para aplicar métricas com responsabilidade e precisão em projetos reais de software.

2. CLASSIFICAÇÃO DAS ENTIDADES DE SOFTWARE

A primeira tarefa da atividade de mensuração de software é identificar as entidades e atributos que se deseja medir. Entidades de software podem ser classificadas em:

- Processos;
- Produtos;
- Recursos.

As entidades de processo representam as atividades e procedimentos realizados durante o ciclo de vida do software. Usualmente, um processo está associado com alguma escala de tempo, visto que atividades de processo tem uma duração: ocorrem por um

período; existe alguma ordenação entre as atividades que depende do tempo – por exemplo, uma atividade deve ter finalizada antes de iniciar outra. São exemplos de entidades de processo revisão de código, teste de unidade, integração contínua.

As entidades de produto são os artefatos concretos gerados ao longo do desenvolvimento por uma atividade de processo. Código-fonte, documentações, diagramas UML e requisitos especificados são bons exemplos de entidades de produtos. As entidades de recursos são os insumos ou recursos utilizados nas atividades de processos para gerar os produtos, por exemplo o número de desenvolvedores que trabalharam numa determinada atividade, o valor monetário para realizar a atividade ou ferramentas utilizadas.

Nota-se que recursos e produtos estão associados com um processo. Atividades de processo usam recursos e produtos como entradas e produzem produtos. Um produto de um processo pode ser usado como entrada para outro. Por exemplo, o documento de requisitos é o produto da atividade de requisitos que, por sua vez, será usado na atividade de codificação.

Os atributos de cada entidade de software são classificados em atributos ***internos*** ou ***externos***. Atributos internos são aqueles que podem ser mensurados em termos da própria entidade, sem a necessidade de observar seu comportamento ou como se relaciona com o ambiente externo. Por outro lado, os atributos externos somente podem ser medidos a respeito como a entendida (produto, processo ou recurso) se relaciona com o ambiente. Nesse caso, o comportamento do produto, processo ou recurso, é mais importante que a entidade em si.

Para melhor entender a diferença entre atributos internos e externos, considere um conjunto de módulos de software. Sem a necessidade de executar o código, podemos medir diversos atributos: tamanho, “complexidade” e dependência entre os módulos. Esses são atributos internos, pois podem ser mensurados analisando a entidade em si sem a necessidade de observar o comportamento em um ambiente de execução. No entanto, há atributos que dependem da execução do código

para serem medidos. Esse é o caso da quantidade de falhas que um usuário experimenta ao usar o programa, da dificuldade que um usuário tem ao navegar nas telas ou do tempo de resposta para realizar uma tarefa. Todos são atributos que dependem do comportamento do programa e, portanto, só podem ser mensurados durante a sua execução, sendo exemplos de atributos externos.

Gerentes de projetos frequentemente desejam medir e prever atributos externos de um sistema de software. Do lado dos usuários, os atributos externos também são de grande interesse, pois afetam diretamente a experiência com o sistema. No entanto, medir atributos externos são, usualmente, mais difíceis de medir que os internos e só podem ser medidos nas fases finais do desenvolvimento, quando o sistema já está pronto para ser testado ou em produção. Por exemplo, não é possível medir a confiabilidade de um sistema enquanto ele ainda está em desenvolvimento — é preciso que o sistema esteja finalizado e em uso para que se possam coletar dados reais sobre falhas, disponibilidade, dentre outros atributos.

Além disso, há o desafio de definir esses atributos de maneira clara e mensurável. Por exemplo, todos desejam construir ou adquirir sistemas de alta qualidade, mas nem sempre há consenso sobre o que realmente significa qualidade. Por isso, é comum que esses atributos de alto nível (como qualidade) sejam definidos em termos de outros atributos mais concretos, bem-definidos e mensuráveis.

No entanto, em muitos casos, desenvolvedores e usuários concentram seus esforços em uma única faceta de um atributo amplo. Por exemplo, alguns medem a qualidade, um atributo externo de produto, como o número de falhas encontradas no teste formal, um atributo interno de processo. O uso de atributos internos para fazer julgamentos de atributos externos pode levar a conclusões inválidas. Porém, existe uma necessidade de usar atributos internos para tomar decisões embasadas acerca de atributos externos. Nesse sentido, um dos objetivos da pesquisa em métricas de software é identificar o relacionamento entre atributos internos e externos, assim como encontrar novos métodos para mensurar diretamente o atributo de interesse.

3. DETERMINANDO O QUE MEDIR: O PARADIGMA GQM

Infelizmente, os desenvolvedores não têm tempo para medir, analisar e monitorar todos os atributos de software. Logo, é crucial focar em medir as áreas que necessitam de mais atenção, entendimento e melhorias. Determinar os atributos a serem medidos dependerá dos seus objetivos, de modo a selecionar a medição apropriada com base nas informações necessárias para satisfazer as suas metas.

Basili et al. (Basili e Weiss, 1984; Basili e Rombach, 1988; Mandić et al., 2009) propuseram a abordagem ***Goal-Question-Metric (GQM)*** para orientar o processo de definição e uso de métricas, com foco no alinhamento entre os objetivos da organização e os dados medidos. A GQM baseia-se em uma estrutura hierárquica de três níveis — objetivo, pergunta e métrica — que permite construir um sistema de medição coerente, partindo da compreensão clara do que se deseja alcançar com a medição.

No primeiro nível da abordagem (*Goal*), define-se o objetivo da medição, considerando o que se deseja entender, controlar, avaliar ou melhorar. Esse objetivo deve especificar o objetivo de análise (seja ele um produto, processo ou recurso), o propósito da medição, o ponto de vista envolvido (desenvolvedor, gerente, cliente, etc.) e o contexto organizacional. Por exemplo, um objetivo pode ser “avaliar a manutenibilidade do código-fonte de um sistema legado, do ponto de vista da equipe de manutenção”.

Com base no objetivo definido, o segundo nível da abordagem (*Question*) propõe a formulação de uma ou mais perguntas que visem esclarecer se o objetivo está sendo cumprido. Essas perguntas devem explorar os diversos aspectos do objetivo da medição e guiar a análise dos dados. Elas servem como ponte entre os objetivos e as métricas concretas. A partir do objetivo do exemplo anterior, poderíamos formular as seguintes perguntas:

- O código apresenta muitos trechos duplicados?

MULTIMÍDIA

The Goal Question Metric Approach

CURIOSIDADE

Você sabia que a abordagem GQM foi desenvolvida inicialmente pela NASA? Ela surgiu para atender à necessidade de melhorar a qualidade do software embarcado em projetos espaciais, nos quais falhas eram inaceitáveis.

- Qual a complexidade média das funções / métodos?
- Os módulos são coesos e bem encapsulados?
- Qual o tempo médio para fazer uma correção no sistema?

Finalmente, no terceiro nível (*Metric*), para cada pergunta formulada, são definidas as medições que permitirão quantificar as respostas de forma objetiva. Essas podem ser diretas (como o número de linhas de código ou complexidade ciclomática) ou indiretas (como o índice de manutenibilidade ou densidade de defeitos). Segundo o exemplo anterior, as medições associadas poderiam ser:

- Percentual de duplicação de código;
- Complexidade ciclomática por função;
- Índice de coesão por módulo;
- Duração média entre a identificação de um defeito e a sua correção.

A principal vantagem da abordagem GQM é que ela evita a coleta indiscriminada de dados. Ao partir de objetivos concretos, garante que apenas as informações relevantes para o contexto do projeto sejam coletadas e analisadas. Isso torna o uso de métricas mais eficiente, econômico e direcionado à tomada de decisão informada. Além disso, por estabelecer uma conexão explícita entre estratégia organizacional e as medições, a GQM facilita a comunicação entre diferentes níveis da equipe e promove a melhoria contínua do processo de desenvolvimento de software.

SAIBA-MAIS

Para aqueles que desejam se aprofundar no método GQM, recomendo o livro The Goal / Question / Metric Method: A Practical Guide for Quality Improvement of Software Development (van Solingen e Berghout, 1999)

4. VALIDAÇÃO DE MEDIDAS DE SOFWATRE

Uma medição pode ser usada para avaliar / compreender uma entidade, conforme discutido na Unidade 1, ou como entrada para um sistema de predição. Assim, distinguimos a medição nos tipos:

1. Medição ou *sistema de medição*, o qual é usado para mapear atributos das entidades num sistema numérico;
2. *Sistema de predição*, o qual é usado para atributos de entidades futuras. Envolve um modelo matemático de predição associado.

Na Unidade 1, estudamos sistema de medição e, portanto, a validade decorre da teoria representacional da medida. Assim, a validação de um sistema de medição é o processo de verificar se o mapeamento satisfaz a condição da teoria representacional, ou seja, se é um homomorfismo da relação empírica observada.

No entanto, a validação de um sistema de predição envolve verificar se o sistema fornece predições precisas. Ou seja, deve-se verificar a performance da previsão, comparando-se previsão com o dado real. A validação de um sistema de predição necessita de experimentação e teste de hipóteses.

Por exemplo, suponhamos que desejamos medir o tamanho (atributo) de um programa (entidade) de maneira válida. A medida a ser escolhida deve preservar as noções intuitivas do que seja tamanho de programas:

- se um programa P1 é maior que P2, então a medida de P1, m(P1), deve ser maior que a medida de P2, m(P2). Isto é:

$$m(P1) > m(P2)$$

- o tamanho de dois programas, P1 e P2, concatenados deve ser igual a junção do tamanho dos programas individuais, ou seja:

$$m(P1.P2) = m(P1) + m(P2)$$

MULTIMÍDIA

Validating software metrics: a spectrum of philosophies

Qualquer medição que mantenha essas propriedades será um sistema de medição válido, como o número de linha de códigos ou complexidade ciclomática.

No entanto, se o tamanho do programa, em LOC por exemplo, for usado para estimar o custo de manutenção do sistema, a validade dessa predição deve ser verificada prevendo-se o custo de manutenção de sistema e comparando com o valor observado na prática.

Apesar de negligenciada no passado, a comunidade de engenharia de software sempre soube da necessidade de um processo rigoroso de validação de medições. A medida que novas medições são propostas, é importante perguntar se ela captura o atributo a qual pretende descrever. Ciente disso, alguns pesquisadores da área argumentam que um sistema

de medição para ser considerado válido também deve fazer parte de um sistema de predição válido. Assim, distinguimos o conceito de validação de um sistema de medição em dois níveis: sentido estrito e sentido amplo.

Um sistema de medição válido, conforme discutido anteriormente, é dito ser válido em sentido estrito. Por sua vez, o sistema de medição é válido em sentido amplo se é:

- válido em sentido estrito; e
- compõe de um sistema de predição válido.

Mostrar que um sistema de medição é válido em sentido amplo envolve realização de experimentos e teste de hipóteses, ou, alternativamente, demonstrar um relacionamento e correlação estatística com uma outra medida válida. As ferramentas para realizar experimentos e estudos estatístico para validar sistemas de predições não são abordadas neste curso. Para o aluno que desejar se aprofundar no assunto, Fenton e Bieman (2014) abordam os detalhes de como proceder nos capítulos 4 a 7.

EXPLICA ATIVO

Uma métrica pode ser matematicamente válida (isto é, mapeia corretamente um atributo a números), mas não ser útil na prática. Isso acontece quando ela não oferece informações relevantes para a tomada de decisão, ou quando não é compreendida pelos stakeholders

5. CONSIDERAÇÕES FINAIS

Nesta unidade, discutimos a classificação das entidades de software — processos, produtos e recursos — e a distinção entre atributos internos e externos. Vimos que os internos podem ser medidos diretamente nas entidades, enquanto os externos exigem a observação de seu comportamento em execução. Compreendemos também as limitações e desafios na medição de atributos externos, além dos riscos de se fazer inferências incorretas quando se usa métricas internas como aproximação.

Em seguida, apresentamos a abordagem Goal-Question-Metric (GQM). A GQM auxilia na definição de métricas alinhadas aos objetivos do projeto, por meio da formulação de perguntas específicas e da escolha de métricas apropriadas. Essa abordagem evita medições irrelevantes e promove decisões mais embasadas e alinhadas com as necessidades reais da equipe ou da organização.

Por fim, tratamos da validação das medições, um aspecto fundamental para garantir que as métricas utilizadas sejam confiáveis e úteis tanto para avaliação quanto para previsão. Distinguimos a validação em sentido estrito, baseada na teoria da mensuração, e a validação em sentido amplo, que envolve a eficácia preditiva das métricas em contextos práticos.

Em resumo, esta unidade forneceu os conceitos e ferramentas necessários para realizar medições significativas, alinhadas com objetivos claros e fundamentadas em critérios de validade, preparando o estudante para aplicar métricas de forma crítica e eficaz no desenvolvimento de software.

REFERÊNCIAS

BASILI, Victor R.; WEISS, David M. A method for collecting valid software engineering data. *IEEE Transactions on Software Engineering*, v. 10, n. 6, p. 728–738, nov. 1984.

BASILI, Victor R.; ROMBACH, Hans Dieter. The TAME project: towards improvement-oriented software environments. *IEEE Transactions on Software Engineering*, v. 14, n. 6, p. 758–773, 1988.

FENTON, Norman E.; BIEMAN, James. Software Metrics: a Rigorous and Practical Approach. 3. ed. Boca Raton: CRC Press, 2014.

MANDIĆ, Vladimir; BASILI, Victor; HARJUMAA, Lasse; OIVO, Markku; MARKKULA, Jouni. Utilizing GQM+Strategies for business value analysis: an approach for evaluating business goals. In: *PROCEEDINGS OF THE 2010 ACM-IEEE INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT (ESEM'10)*, 2010, New York. New York: Association for Computing Machinery, 2010. Article 20, p. 1–10.

VAN SOLINGEN, Rini; BERGHOUT, Erik. *The Goal/Question/Metric method: a practical guide for quality improvement of software development*. London: McGraw-Hill, 1999.