10. Medições e Métricas de Software

A medição de software se preocupa com a derivação de um valor numérico ou o perfil para o atributo de um componente de software, sistema ou processo. Comparando esses valores entre si e com os padrões que se aplicam a toda a organização, você pode ser capaz e tirar conclusões sobre a qualidade do software ou avaliar a eficácia dos métodos, das ferramentas e dos processos de software.

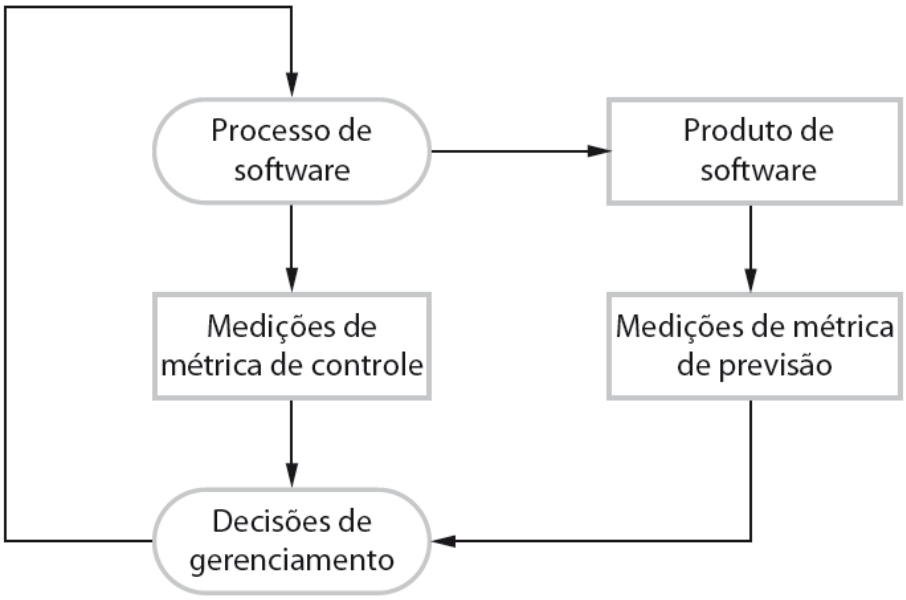
Por exemplo, digamos que uma organização tem a intenção de introduzir uma nova ferramenta de teste de software. Antes de introduzir a ferramenta, em um determinado momento você registra o número de defeitos de software descobertos. Essa é uma boa baseline de avaliação da eficácia da ferramenta. Depois de algum tempo usando a ferramenta, esse processo é repetido. Se mais defeitos forem encontrados durante o tempo usando a ferramenta, esse processo é repetido. Se mais defeitos forem encontrados durante o mesmo período, depois que a ferramenta foi introduzida, você pode decidir se ela fornece suporte útil para o processo de validação de software.

O objetivo a longo prazo de medição de software é usá-la no lugar de revisões para fazer julgamentos sobre a qualidade de software. Usando a medição de software, um sistema poderia, idealmente, ser avaliado usando uma variedade de métricas e, a partir dessa medição, deduzir um valor para a qualidade do sistema. Se o software atingir o limiar de qualidade requerido, então ele poderia ser aprovado sem revisão. Quando apropriado, as ferramentas de medição também podem realçar áreas do software que poderiam ser melhoradas. No entanto, estamos ainda longe dessa situação ideal e não há sinal de que as avaliações automatizadas de qualidade venham a se tornar realidade em um futuro próximo.

Uma métrica de software é uma característica de um sistema de software, documentação de sistema ou processo de desenvolvimento que pode ser objetivamente medido. Exemplos de métricas incluem: o tamanho de um produto em linhas de código; o índice Fog (GUNNING, 1962), que é uma medida da legibilidade de uma passagem de texto escrito, o número de defeitos relatados em um produto de software entregue, e o número de pessoas/dia requerido para desenvolver um componente de sistema.

As métricas de software podem ser métricas de controle ou métricas de previsão. Como os nomes sugerem, as primeiras suportam os processos de gerenciamento e as outras o ajudam a prever as características do software. As métricas de controle são geralmente associadas com os processos de software. Exemplos de métricas de controle ou de processos são o esforço médio e o tempo necessário para reparar os defeitos relatados. Métricas de previsão são associadas com o software em si e, por vezes, são conhecidas como “métricas de produto. São exemplos de métricas de previsão: a complexidade ciclomática de um módulo (mede a quantidade de caminhos de execução independentes a partir de um código fonte), o comprimento médio dos identificadores em um programa e o número de atributos e operações associadas com as classes de objeto em um projeto.

As métricas de controle e de previsão podem influenciar a tomada de decisão de gerenciamento, como mostrado na Figura abaixo. Os gerentes usam métricas de processo para decidir se devem ser feitas alterações no processo; as métricas de previsão são usadas para ajudar a estimar o esforço necessário para fazer as alterações no software. Vamos nos deter principalmente, as métricas de previsão, cujos valores são avaliados, analisando o código de um sistema de software.



Medições de previsão e de controle

Existem duas maneiras para o uso das medições de um software de sistema:

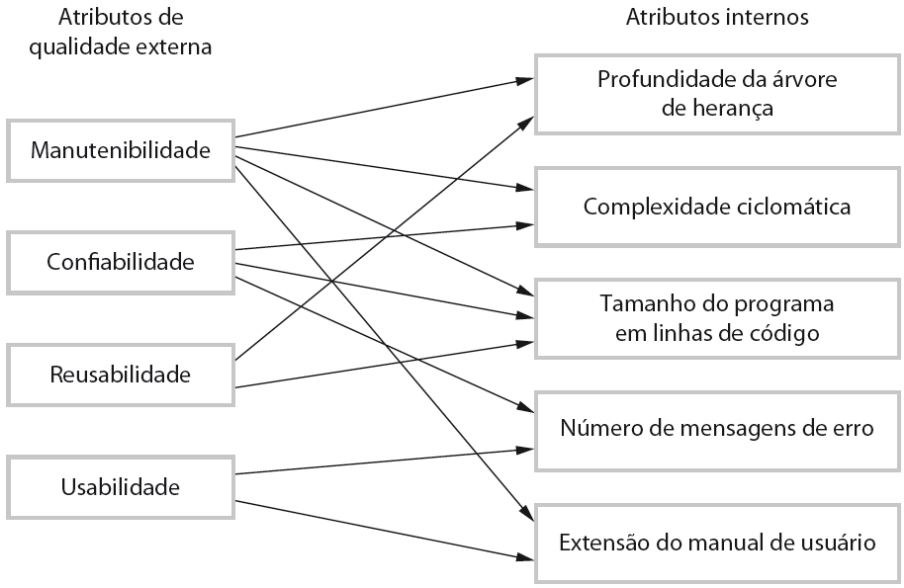
1. ***Para atribuir um valor aos atributos de qualidade de sistema.*** Ao medir as características dos componentes de sistema, bem como sua complexidade ciclomática e, em seguida, agregar essas medições, você pode avaliar os atributos de qualidade do sistema, como a manutenibilidade.
2. ***Para identificar componentes de sistema cuja qualidade não atingiu o padrão.*** As medições podem identificar componentes individuais com características que se desviem da norma. Por exemplo, você pode medir componentes para descobrir aqueles com a mais alta complexidade. Esses são mais passíveis de conter bugs porque a complexidade os torna mais difíceis de entender.

Infelizmente, é difícil fazer medições diretas de muitos dos atributos de qualidade de software mostrados na Tabela a seguir. Os atributos de qualidade como manutenibilidade, compreensibilidade e usabilidade são atributos externos relacionados com os desenvolvedores e usuários que experimentam o software. Eles são afetados por fatores subjetivos, como a experiência e a educação do usuário e, portanto, não podem ser medidos objetivamente. Para fazer um julgamento sobre esses atributos, você deve medir alguns atributos internos do software (como tamanho, complexidade etc.) e assumir que estão relacionados com as características de qualidade com as quais você se preocupa.

A Figura a seguir mostra alguns atributos externos de qualidade dos softwares e atributos internos que poderiam, intuitivamente, ser relacionados a eles. O diagrama sugere que pode haver relacionamentos entre atributos internos e externos, mas não diz como esses atributos estão relacionados. Se a medida do atributo interno deve ser uma previsão útil de características externas de software, três condições devem ser mantidas (KITCHENHAM, 1990):

1. O atributo interno deve ser medido com precisão. Isso nem sempre é simples e pode exigir ferramentas especiais para fazer as medições.
2. Deve existir um relacionamento entre o atributo que pode ser medido e o atributo de qualidade externa de interesse. Ou seja, o valor do atributo de qualidade deve ser relacionado, de alguma forma, com o valor do atributo que pode ser medido.

3. Esse relacionamento entre os atributos internos e externos deve ser compreendido, validado e expresso em termos de uma fórmula ou modelo. A formulação de modelo envolve a identificação da forma funcional do modelo (linear, exponencial etc.) pela análise de dados coletados, identificando os parâmetros que devem ser incluídos no modelo e calibrando esses parâmetros com o uso dos dados existentes.



Relacionamentos entre os atributos internos e externos de software

Os atributos internos de software, como a complexidade ciclomática de um componente, são medidos com o uso de ferramentas de software que analisam o código-fonte do software. As ferramentas open source disponíveis podem ser usadas para fazer essas medições. Embora a intuição sugira que poderia haver um relacionamento entre a complexidade de um componente de software e o número de falhas observadas em uso, é difícil demonstrar objetivamente que é este o caso. Para testar essa hipótese, você precisa de dados de falha de um grande número de componentes e de acesso ao código-fonte do componente para análise. Poucas empresas fizeram um compromisso a longo prazo para a coleta de dados sobre seu software, portanto, dados de falha raramente são disponibilizados para análise.

Na década de 1990, grandes empresas, como a Hewlett Packard (GRADY, 1993), a AT&T (BARNARD e PRICE, 1994) e a Nokia (KILPI, 2001) introduziram programas de métricas. Elas fizeram medições de seus produtos e processos e usaram-nas em seus processos de gerenciamento de qualidade. Elas se centraram em coletar métricas de defeitos de programa, bem como em processos de verificação e validação. Offen e Jeffrey (1997) e Hall e Fenton (1997) discutem a introdução de programas de métricas na indústria com mais detalhes.

Existem poucas informações publicamente disponíveis sobre o uso atual da medição sistemática de software na indústria. Muitas empresas coletam informações sobre seu software, como o número de solicitações de mudança de requisitos ou o número de defeitos descobertos nos testes. No entanto, não está claro se, em seguida, usam essas medições sistematicamente para comparar produtos e processos de software ou avaliar o impacto das mudanças nos processos e ferramentas de software. Existem várias razões pelas quais isso é difícil:

1. É impossível quantificar o retorno sobre o investimento da introdução de um programa de métricas em organizações. Durante os últimos anos, houve melhorias significativas na qualidade de software sem o uso de métricas.

Por isso é difícil justificar os custos iniciais de introdução de medição e avaliação sistemática de software.
2. Não existe um padrão para as métricas de software ou processos padronizados para medição e análise. Muitas empresas relutam em introduzir programas de medição até que estes e as ferramentas de suporte estejam disponíveis.
3. Em muitas empresas, os processos de software não são padronizados e são mal definidos e mal controlados. Além disso, existe muita variabilidade em processos da mesma empresa para que as medições sejam usadas de forma significativa.
4. Grande parte da pesquisa sobre medição e métricas de software centra-se nas métricas baseadas em códigos e processos de desenvolvimento dirigidos a planos. No entanto, cada vez mais o software é desenvolvido configurando sistemas ERP ou COTS ou usando os métodos ágeis. Não sabemos, portanto, se a pesquisa anterior é aplicável a essas técnicas de desenvolvimento de software.
5. A introdução da medição acrescenta overhead aos processos. Estes contradizem os objetivos dos métodos ágeis, os quais recomendam a eliminação das atividades de processos que não estão diretamente relacionadas ao desenvolvimento de programa. Portanto, as empresas que adotaram os métodos ágeis geralmente não adotam um programa de métricas.

As métricas e medições de software são as bases da engenharia de software empírica (ENDRES e ROMBACH, 2003). Essa é uma área de pesquisa em que as experiências em sistemas de software e a coleta de dados sobre projetos reais foram usadas para formar e validar hipóteses sobre os métodos e as técnicas de engenharia de software. Os pesquisadores que trabalham nessa área argumentam que podemos confiar no valor dos métodos e das técnicas de engenharia de software apenas se pudermos fornecer evidências concretas de que eles realmente oferecem os benefícios que seus inventores sugerem.

Infelizmente. mesmo quando é possível fazer medições objetivas e tirar conclusões a partir delas, isso pode não convencer os tomadores de decisões necessariamente. Em vez disso, as tomadas de decisões muitas vezes são influenciadas por fatores subjetivos, como a novidade ou a extensão em que as técnicas são de interesse para os profissionais. Portanto, penso que ainda se passarão muitos anos antes que a engenharia de software empírica tenha efeitos significativos nas práticas de engenharia de software.

10.1 Métricas do produto

As métricas de produto são métricas de previsão usadas para medir atributos internos de um sistema de software. O tamanho de sistema, medido em linhas de código, ou o número de métodos associados a cada classe de objeto são exemplos de métricas de produto. Contudo, como já foi explicado, as características de software que podem ser facilmente medidas, como tamanho e complexidade ciclomática, não têm uma relação clara e consistente com atributos de qualidade tais como capacidade de compreensão e manutenibilidade. Os relacionamentos variam de acordo com os processos de desenvolvimento e tecnologia usada, bem como o tipo de sistema que está sendo desenvolvido.

As métricas de produto se dividem em duas classes:

1. Métricas **dinâmicas**, as quais são coletadas por meio de medições efetuadas de um programa em execução. Essas métricas podem ser coletadas durante o teste de sistema ou após o sistema estar em uso.

Um exemplo pode ser o número de relatórios de bugs ou o tempo necessário para concluir uma computação.

2. Métricas **estáticas**, que são coletadas por meio de medições feitas de representações do sistema, como o projeto, o programa ou a documentação. Exemplos de métricas estáticas são o tamanho de código e o comprimento médio de identificadores usados.

Esses tipos de métrica estão relacionados com diferentes atributos de qualidade. Métricas dinâmicas ajudam a avaliar a eficiência e a confiabilidade de um programa. Métricas estáticas ajudam a avaliar a complexidade, a compreensibilidade e a manutenibilidade de um sistema ou componentes de um sistema de software.

Geralmente, existe um relacionamento claro entre as métricas dinâmicas e as características de qualidade de software. É bastante fácil medir o tempo de execução necessário para funções particulares e avaliar o tempo necessário para iniciar um sistema. Eles se relacionam diretamente com a eficiência do sistema. Da mesma forma, o número de falhas de sistema e o tipo de falha podem ser registrados e relacionados diretamente com a confiabilidade do software.

Como já discutido, métricas estáticas, tais como as mostradas na Tabela, têm um relacionamento indireto com atributos de qualidade. Um grande número de métricas diferentes foi proposto e muitos experimentos tentaram derivar e validar os relacionamentos entre essas métricas e atributos, como a complexidade de sistema e manutenibilidade. Nenhum desses experimentos foi conclusivo, mas o tamanho de programa e a complexidade de controle parecem ser os mais confiáveis mecanismos de previsão de compreensibilidade, complexidade e manutenibilidade de sistema.

As métricas na Tabela são aplicáveis a qualquer programa, mas métricas orientadas a objetos (OO) mais específicas também foram propostas. A Tabela seguinte resume o conjunto de Chidamber e Kemerer (1994), às vezes, chamado suite CK, de seis métricas orientadas a objetos. Embora elas tenham sido originalmente propostas na década de 1990, ainda são as métricas OO mais amplamente usadas. Algumas ferramentas de projeto de UML coletam valores automaticamente para essas métricas quando são criados diagramas UML.

El-Amam (2001), em uma excelente revisão de métricas orientadas a objetos, discute as métricas CK e outras métricas OO e conclui que nós ainda não temos provas suficientes para compreender como essas e outras métricas orientadas a objetos se relacionam com as qualidades externas de software. Essa situação não mudou muito desde sua análise em 2001. Ainda não sabemos como usar as medições de programas orientados a objeto para tirar conclusões confiáveis sobre sua qualidade.

[Tabela] Métricas estáticas de produto de software

Métrica de software	Descrição
Fan-in/Fan-out	Fan-in é a medida do número de funções ou métodos que chamam outra função ou método (digamos X). Fan-out é o número de funções que são chamadas pela função de X. Um valor alto para fan-in significa que X está fortemente acoplado ao resto do projeto e alterações em X terão repercussões extensas. Um valor alto para fan-out sugere que a complexidade geral do X pode ser alta por causa da complexidade da lógica de controle necessário para coordenar os componentes chamados.

Comprimento de código	Essa é uma medida do tamanho de um programa. Geralmente, quanto maior o tamanho do código de um componente, mais complexo e sujeito a erros o componente é. O comprimento de código tem mostrado ser uma das métricas mais confiáveis para prever a propensão a erros em componentes.
Complexidade cidomática	Essa é uma medida da complexidade de controle de um programa. Essa complexidade de controle pode estar relacionada à compreensibilidade de programa.
Comprimento de identificadores	Essa é uma medida do comprimento médio dos identificadores (nomes de variáveis classes, métodos etc) em um programa. Quanto mais longos os identificadores, mais provável que sejam significativos e, portanto, mais compreensível o programa.
Profundidade de aninhamento condicional	Essa é uma medida da profundidade de aninhamento de declarações if em um programa. Declarações if profundamente aninhadas são difíceis de entender e potencialmente sujeitas a erros
Índice Fog	Essa é uma medida do compnmento médio de palavras e sentenças em documentos. Quanto maior o valor de um índice Fog de um documento, mais difícil a sua compreensão.

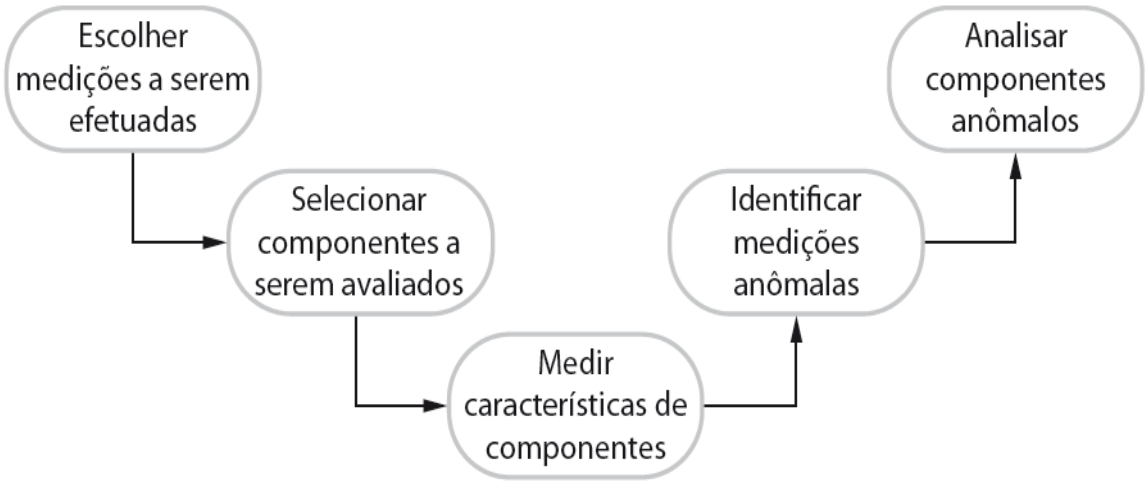
[Tabela] Conjunto de Métricas de CK orientadas a objetos

Métrica orientada a objeto	Descrição
Métodos ponderados por classe (WMC, do inglês Weighted Methods per Class)	É o número de métodos em cada classe, ponderados pela complexidade de cada método. Portanto, um método simples pode ter uma complexidade de 1 e um método grande e complexo pode ter um valor muito superior. Quanto maior o valor para essa métrica, mais complexa a classe de objeto. Geralmente, os objetos complexos são mais difíceis de compreender. Eles podem não ser logicamente coesos, portanto, não podem ser reusados efetivamente como superclasses em uma árvore de herança.
Árvore de profundidade de herança (DIT, do inglês Depth of Inheritance Tree)	Representa o número de níveis discretos na árvore de herança em que as subclasses herdam atributos e operações (métodos) de superclasses. Quanto mais profunda a árvore de herança, mais complexo o projeto. Muitas classes de objeto podem precisar ser compreendidas para que as classes de objeto nas folhas da árvore sejam entendidas.
Número de filhos (NOC do inglês Number of Children)	É uma medida do número de subclasses imediatas em uma classe. Ele mede a largura de uma hierarquia de classe, considerando que DIT mede sua profundidade. Um valor alto para NOC pode indicar um maior reuso. Isso pode significar que mais esforço deve ser dispendido na validação de classes de base por causa do número de subclasses que dependem delas.

Acoplamento entre classes de objeto (CBO, do inglês Coupling Between Object classes)	Classes são acopladas quando métodos em uma classe usam métodos ou variáveis de instância definidas em uma classe diferente. CBO é uma medida de quanto acoplamento existe. Um valor alto para o CBO significa que as classes são altamente dependentes e, portanto, é mais provável que a mudança em uma classe afete outras classes do programa.
Resposta para uma classe (RFC, do inglês Response For a Class)	RFC é a medida do número de métodos que poderiam ser executados em resposta a uma mensagem recebida por um objeto dessa classe. Mais uma vez, RFC está relacionada com a complexidade. Quanto maior o valor de RFC, mais complexa é a classe e, portanto, mais provável que inclua erros.
Falta de coesão em métodos (LCOM, do inglês Lack Of Cohesion in Methods)	LCOM é calculada considerando os pares de métodos em uma classe. LCOM é a diferença entre o número de pares de métodos sem atributos compartilhados e o número de pares de métodos com atributos compartilhados. O valor dessa métrica tem sido amplamente discutido, e ele existe em diversas variações. Não está claro se realmente adiciona qualquer informação útil além das que já são fornecidas por outras métricas.

10.2 Análise de componentes de software

Um processo de medição que pode ser parte de um processo de avaliação de qualidade de software é mostrado na Figura a seguir. Cada componente de sistema pode ser analisado separadamente, usando uma variedade de métricas. Os valores dessas métricas podem ser comparados com diferentes componentes e, talvez, com dados históricos de medição obtidos em projetos anteriores. Medições anômalas, que diferem significativamente da norma, podem significar que existem problemas com a qualidade desses componentes.



O processo de medição de produto

Os estágios principais nesse processo de medição de componente são:

1. **Escolher medições o serem efetuadas**. As questões que a medição está destinada a responder devem ser formuladas e as medições necessárias para responder a essas questões devem ser definidas. As medições que não são diretamente relevantes para essas questões não necessitam ser coletadas.
2. **Selecionar componentes a serem avaliados**. Você pode não precisar avaliar valores de métricas para todos os componentes de um sistema de software. Às vezes, pode selecionar um conjunto representativo de componentes para medição, que lhe permita fazer uma avaliação global da qualidade de sistema. Outras vezes, você pode se centrar nos componentes principais do sistema que estão em uso quase constante A qualidade desses componentes é mais importante do que a qualidade de componentes raramente usados.
3. **Medir características de componentes**. Os componentes selecionados são medidos e os valores e as métricas associados, computados. Normalmente, isso envolve o processamento da representação de componente (projeto, código etc.) usando uma ferramenta automatizada de coleta de dados. Essa ferramenta pode ser especialmente escrita ou pode ser um recurso de ferramentas de projeto que já esteja em uso.
4. **Identificar medições anômalas**. Após terem sido realizadas as medições de componentes, você deve compará-los uns com os outros e com as medições anteriores que foram registradas em um banco de dados de medições, Você deve procurar valores anormalmente altos ou baixos para cada métrica, pois estes sugerem que pode haver problemas com o componente que exibe tais valores.
5. **Analisar componentes anômalos**. Ao identificar os componentes que tem valores anômalos para sua métrica escolhida, você deve examiná-los para decidir se esses valores de métricas anômalos significam que a qualidade do componente está comprometida. Um valor anômalo de métrica de complexidade (por exemplo) não significa, necessariamente, um componente de má qualidade. Pode haver alguma outra razão para o alto valor, por isso pode não significar que existam problemas de qualidade de componente.

Você sempre deve manter os dados coletados como um recurso organizacional e manter registros históricos de todos os projetos, mesmo quando dados não tenham sido usados em um determinado projeto. Uma vez estabelecido um banco de dados de medições suficientemente grande, você pode comparar a qualidade do software entre projetos e validar as relações entre os atributos de componentes internos e as características de qualidade.

10.3 Ambiguidade de medições

Quando você coleta dados quantitativos sobre software e processos de software, precisa analisar esses dados para entender seu significado. É fácil interpretar dados erroneamente e fazer inferências incorretas. Você não pode simplesmente olhar os dados por si – você também deve considerar o contexto em que eles são coletados.

Para ilustrar como os dados coletados podem ser interpretados de maneiras diferentes, considere o cenário adiante, relacionado com o número de solicitações de mudança feitas pelos usuários de um sistema:

> *Um gerente decide monitorar o número de solicitações de mudança enviadas pelos clientes com base em uma pré-suposição de que há um relacionamento entre essas solicitações de mudança e a usabilidade e adequabilidade do produto. Ele parte do principio de que quanto maior for o número de*

solicitações, menos o software atenderá às necessidades do cliente

Tratar as solicitações de mudança de software é caro. Portanto, a organização decide modificar seu processo com o objetivo de aumentar a satisfação do cliente e, ao mesmo tempo, reduzir os custos de mudanças. A intenção é que as mudanças de processo resultarão em melhores produtos e menos solicitações de mudança.

As mudanças de processo são iniciadas para aumentar o envolvimento do cliente no processo de projeto de software. São introduzidos testes Beta de todos os produtos e as solicitações dos clientes por modificações são incorporadas no produto entregue. Novas versões de produtos, desenvolvidas com esse processo modificado, são entregues. Em alguns casos, o número de solicitações de mudança é reduzido. Em outros, ele aumenta. O gerente fica desconcertado e considera impossível avaliar os efeitos das mudanças nos processos sobre a qualidade do produto.

Para entender por que esse tipo de ambiguidade pode ocorrer, você deve compreender as razões pelas quais os usuários podem fazer solicitações de mudança:

1. O software não é bom o suficiente e não faz o que os clientes querem que ele faça. Portanto, eles solicitam mudanças que ofereçam a funcionalidade exigida.

2. Alternativamente, o software pode ser muito bom e por isso é ampla e fortemente usado. As solicitações de mudança podem ser geradas porque existem muitos usuários de software que pensam criativamente em novas coisas que poderiam ser feitas com o software.

Portanto, aumentar o envolvimento do cliente no processo pode reduzir o número de solicitações de mudança de produtos com os quais os clientes estavam descontentes. As mudanças de processo têm sido eficazes e tornaram o software mais usável e adequado. No entanto, as mudanças de processo podem não ter funcionado e os clientes podem decidir buscar um sistema alternativo. O número de solicitações de mudança pode diminuir porque o produto perdeu parte do mercado para um produto rival e, consequentemente, existem menos usuários.

Por outro lado, as mudanças de processo poderiam trazer muitos clientes novos, felizes, que desejem participar no processo de desenvolvimento de produto. Portanto, eles geram mais solicitações de mudança. As mudanças no processo de tratamento de solicitações de mudança podem contribuir com esse aumento. Se a empresa é mais sensível aos clientes, estes podem gerar um número maior de solicitações de mudança, pois sabem que essas solicitações serão levadas a sério. Eles acreditam que suas sugestões provavelmente serão incorporadas em versões posteriores do software. Então, o número de solicitações de mudança pode ter aumentado porque os sites de beta-teste não eram típicos do maior uso do programa.

Para analisar os dados de solicitação de mudança, não basta saber o número de solicitações de mudança. Você precisa saber quem fez a solicitação, como essas pessoas usam o software e por que a solicitação foi feita. Também precisa de informações sobre fatores externos, como modificações nos procedimentos de solicitação de mudança ou alterações de mercado que podem ter efeito. Com essas informações, é possível descobrir se as mudanças de processo foram eficazes no aumento da qualidade de produto.

Isso ilustra as dificuldades de compreensão dos efeitos das mudanças e a abordagem 'científica' para esse problema é reduzir o número de fatores que possam afetar as medições feitas. No entanto, processos e produtos que estão sendo medidos não são isolados de seu ambiente. O ambiente de negócio está

mudando constantemente e é impossível evitar mudanças nas práticas de trabalho só porque eles podem fazer comparações de dados inválidos. Assim, dados quantitativos sobre as atividades humanas não podem sempre ser tomados pelo valor de face. Muitas vezes, as razões pelas quais um valor medido se altera são ambíguas. Essas razões devem ser pesquisadas em detalhes antes de se tirarem conclusões a respeito de quaisquer medições efetuadas anteriormente.

Pontos Chave

- O gerenciamento de qualidade de software visa assegurar que o software tenha um pequeno número de defeitos e que se adeque aos padrões de manutenibilidade, confiabilidade, portabilidade etc. em vigor. Inclui a definição de padrões para produtos e processos e o estabelecimento de processos para verificar se tais padrões foram seguidos.
- Os padrões de software são importantes para a garantia de qualidade, pois representam uma identificação das 'melhores práticas'. Durante o desenvolvimento de software, os padrões fornecem uma base sólida para a construção de software de boa qualidade.
- Você deve documentar um conjunto de procedimentos de garantia de qualidade em um manual de qualidade organizacional. Este pode ser baseado no modelo genérico para um manual de qualidade sugerido na norma 150 9001.
- Revisões dos entregáveis de processos de software envolvem uma equipe de pessoas que verifica se os padrões de qualidade estão sendo seguidos. Revisões são técnicas largamente usadas para avaliação de qualidade.
- Em uma inspeção de programa ou revisão em pares, uma pequena equipe verifica o código sistematicamente. Ela lê o código em detalhes procurando por possíveis erros e omissões. Em seguida, os problemas detectados são discutidos em uma reunião de revisão de código.
- A medição de software pode ser usada para coletar dados quantitativos sobre o software e o processo de software. Você pode ser capaz de usar os valores das métricas de software que são coletadas para fazer inferências sobre a qualidade de produto e de processo.
- Métricas de qualidade de produto são particularmente úteis para realçar componentes anômalos que podem ter problemas de qualidade. Em seguida, esses componentes devem ser analisados em mais detalhes.

Exercícios

1) Fale sobre a Norma ISO 9001.

2) Discuta a avaliação de qualidade de software de acordo com os atributos de qualidade mostrados na Primeira Tabela. Você deve considerar cada atributo e explicar como ele pode ser avaliado.

3) Projete um formulário eletrônico que possa ser usado para registrar comentários de revisão e que poderá ser usado para enviar por e-mail os comentários para os revisores.

4) Descreva rapidamente possíveis padrões que possam ser usados para:

a. O uso de construções de controle em C, C# ou Java;
b. Relatórios que possam ser submetidos a um projeto de formatura em uma universidade;
c. O processo de fazer e aprovar mudanças de programa;
d. O processo de comprar e instalar um novo computador.

Suponha que você trabalhe para uma organização que desenvolve produtos de

banco de dados para indivíduos e empresas de pequeno porte. Essa organização está interessada na quantificação de seu desenvolvimento de software. Escreva um relatório sugerindo métricas adequadas e sugira como estas podem ser coletadas.

5) Explique por que as métricas de projeto são, por si só, um método inadequado de previsão de qualidade de projeto.

6) Pesquise sobre Complexidade Ciclomática

7) Explique por que é difícil validar os relacionamentos entre os atributos internos de produto, como complexidade ciclomática e atributos externos, como a manutenibilidade.

8) Um colega, que é um excelente programador, produz softwares com poucos defeitos, mas constantemente ignora os padrões de qualidade da organização. Como seus gerentes deveriam reagir a esse comportamento?