CAPÍTULO

16 GARANTIA DA QUALIDADE DE SOFTWARE

CONCEITOS-CHAVE

A abordagem de engenharia de software descrita neste livro tem um único objetivo: produzir software de alta qualidade em tempo condizente com as necessidades dos interessados. Mesmo assim, muitos leitores sentir-se-ão desafiados pela questão: "O que é qualidade de software?" Philip Crosby [Cro79], em seu livro histórico sobre qualidade, fornece uma resposta sarcástica a esta pergunta:

> O problema da gestão da qualidade não é o que as pessoas não sabem a seu respeito. Mas sim, o que eles pensam que sabem...
>
> Nesse aspecto, a qualidade tem muita semelhança com o sexo. Todo mundo está disposto a fazê-lo (sob certas condições, é claro). Todo mundo tem a impressão de que entende de sexo (muito embora não queiram fornecer explicações). Todo mundo pensa que sua execução é apenas uma questão de seguir instintos naturais (afinal de contas, de alguma forma consegue-se fazê-lo). E, obviamente, a maioria das pessoas acredita que os problemas nessas áreas são causados pelos outros (se ao menos eles se preocupassem em fazer as coisas certas...).

De fato, qualidade é um conceito desafiador — este conceito é apresentado em detalhes no Capítulo 14.[1]

Alguns desenvolvedores de software continuam a acreditar que a qualidade de software é algo sobre o qual começamos a nos preocupar depois que o código é gerado. Nada poderia estar tão distante da verdade! A *garantia da qualidade de software*, SQA (*software quality assurance*, muitas vezes denominada *gestão da qualidade*) é uma atividade universal (Capítulo 2) aplicada em toda a gestão de qualidade.

PANORAMA

O que é? Não basta dizer simplesmente que a qualidade de software é importante. É preciso: (1) definir explicitamente o seu significado, o que realmente se quer dizer com "qualidade de software", (2) criar um conjunto de atividades que ajude a garantir que todo artefato resultante da engenharia de software apresente alta qualidade, (3) realizar atividades de garantia e controle da qualidade de software em todos os projetos de software, (4) usar métricas para desenvolver estratégias para aperfeiçoar a gestão da qualidade e, consequentemente, a qualidade do produto final.

Quem realiza? Todos os envolvidos no processo de engenharia de software são responsáveis pela qualidade.

Por que é importante? Pode-se fazer certo da primeira vez ou então fazer tudo de novo. Se uma equipe de software buscar a qualidade em todas as atividades de engenharia de software, a quantidade de reformulações a ser feitas é reduzida. Isso resulta em custos menores e, mais importante, menor tempo para disponibilização do produto no mercado.

Quais são as etapas envolvidas? Antes das atividades de garantia da qualidade de software (SQA, *software quality assurance*) iniciarem, é importante definir *qualidade de software* em diferentes níveis de abstração. A partir do momento em que se entende o que é qualidade, a equipe de software deve identificar um conjunto de atividades de SQA que irão filtrar erros do artefato antes de ser passados adiante.

Qual é o artefato? É criado um Plano de Garantia da Qualidade de Software para definir a estratégia de SQA de uma equipe de software. Durante a modelagem e a codificação, o artefato principal da SQA é o resultado das revisões técnicas (Capítulo 15). Durante os testes (Capítulos 17 a 20), são produzidos procedimentos e planos de testes. Também podem ser gerados outros produtos associados ao aperfeiçoamento do processo.

Como garantir que o trabalho foi realizado corretamente? Encontrar erros antes de se tornarem defeitos! Ou seja, trabalhar para melhorar a eficiência de remoção dos defeitos (Capítulo 23) reduzindo, portanto, a quantidade de reformulações que a equipe de software terá de realizar.

1 Caso não tenha lido o Capítulo 14, você deve fazê-lo agora.

A garantia da qualidade de software (SQA) engloba: (1) um processo de SQA, (2) tarefas específicas de garantia da qualidade e controle da qualidade (inclusive revisões técnicas e uma estratégia de testes multiescalonados), (3) prática efetiva de engenharia de software (métodos e ferramentas), (4) controle de todos os artefatos de software e as mudanças feitas nesses produtos (Capítulo 22), (5) um procedimento para garantir a conformidade com os padrões de desenvolvimento de software (quando aplicáveis) e (6) mecanismos de medição e de relatórios.

Este capítulo concentra-se em problemas de gerenciamento e em atividades específicas de processos que permitem garantir a uma organização de software fazer "as coisas certas, no momento certo e da maneira certa".

16.1 Problemas de Background

A garantia e o controle da qualidade são atividades essenciais para qualquer empresa de produtos a ser usados por terceiros. Antes do século vinte, o controle de qualidade era responsabilidade exclusiva do artesão que construía um produto. À medida que o tempo foi passando e técnicas de produção em massa tornaram-se comuns, o controle de qualidade tornou-se uma atividade realizada por outras pessoas e não aquelas que constroem o produto.

"Você cometeu os piores erros."

Yogi Berra

A primeira função formal de garantia e controle da qualidade foi introduzida no Bell Labs em 1916 e difundiu-se rapidamente no mundo da manufatura. Durante a década de 1940, foram sugeridas abordagens mais formais para o controle de qualidade, fundamentadas em medições e aperfeiçoamento contínuo dos processos [Dem86] como elementos-chave da gestão da qualidade.

Hoje em dia, toda empresa possui mecanismos para garantir a qualidade de seus produtos. Na realidade, declarações explícitas da preocupação de uma empresa com a qualidade tornaram-se um estratagema de marketing nas últimas décadas.

A história da garantia da qualidade no desenvolvimento de software é análoga à história da qualidade na fabricação de hardware. Durante os primórdios da computação (décadas de 1950 e 1960), a qualidade era responsabilidade exclusiva do programador. Padrões para a garantia da qualidade foram introduzidos no desenvolvimento de software por parte de empresas terceirizadas pela indústria militar durante a década de 1970 e se difundiram rapidamente para o desenvolvimento de software no mundo comercial [IEE93]. Estendendo a definição apresentada anteriormente, a garantia da qualidade de software é um "padrão de ações planejado e sistematizado" [Sch98c], ações essas exigidas para garantir alta qualidade no software. O escopo da responsabilidade da garantia da qualidade poderia ser mais bem caracterizado parafraseando-se um famoso comercial de uma indústria automobilística: "A Qualidade é a Tarefa n.º 1". A implicação para a área de software é que os elementos distintos têm as suas responsabilidades sobre a garantia da qualidade de software — engenheiros de software, gerentes de projeto, clientes, vendedores e os indivíduos que trabalham em um grupo de SQA.

O grupo de SQA funciona como um serviço de defesa do cliente. Ou seja, as pessoas que realizam a SQA devem examinar o software sob o ponto de vista do cliente. O software atende adequadamente aos fatores de qualidade citados no Capítulo 14? O desenvolvimento de software foi conduzido de acordo com os padrões preestabelecidos? As disciplinas técnicas desempenharam apropriadamente seus papéis como parte da atividade de SQA? O grupo de SQA tenta responder a essas e outras perguntas para garantir que a qualidade de software seja mantida.

16.2 Elementos de Garantia da Qualidade de Software

A garantia da qualidade de software engloba um amplo espectro de preocupações e atividades que se concentram na gestão da qualidade de software e que podem ser sintetizadas da seguinte maneira [Hor03]:

Padrões. O IEEE, a ISO e outras organizações de padronizações produziram uma ampla gama de padrões para engenharia de software e os respectivos documentos. Os padrões

WebRef

Uma discussão aprofundada do SQA, incluindo uma ampla gama de definições, pode ser obtida em **www.swqual.com/newsletter/vol2/no1/vol2no1.html**.

podem ser adotados voluntariamente por uma organização de engenharia de software ou impostos pelo cliente ou outros interessados. O papel da SQA é garantir que padrões que tenham sido adotados sejam seguidos e que todos os produtos resultantes estejam em conformidade com eles.

Revisões e auditorias. As revisões técnicas são uma atividade de controle de qualidade realizada por engenheiros de software para engenheiros de software (Capítulo 15). Seu intuito é o de revelar erros. Auditorias são um tipo de revisão efetuado pelo pessoal de SQA com o intuito de assegurar-se de que as diretrizes de qualidade estejam sendo seguidas no trabalho de engenharia de software. Por exemplo, uma auditoria do processo de revisão pode ser realizada para garantir que as revisões estejam sendo realizadas de maneira que conduza à maior probabilidade de descoberta de erros.

Testes. Os testes de software (Capítulos 17 a 20) são uma função de controle de qualidade com um objetivo principal — descobrir erros. O papel da SQA é garantir que os testes sejam planejados apropriadamente e conduzidos eficientemente de modo que se tenha a maior probabilidade possível de alcançar seu objetivo primário.

Coleta e análise de erros/defeitos. A única forma de melhorar é medir o nosso desempenho. A SQA reúne e analisa dados de erros e defeitos para melhor compreender como os erros são introduzidos e quais atividades de engenharia de software melhor se adequam para sua eliminação.

Gerenciamento de mudanças. As mudanças são um dos aspectos mais negativos de qualquer projeto de software. Se não forem administradas apropriadamente, podem gerar confusão, e confusão quase sempre leva a uma qualidade inadequada. A SQA garante que práticas adequadas de gerenciamento de mudanças (Capítulo 22) tenham sido instituídas.

Educação. Toda organização de software quer melhorar suas práticas de engenharia de software. Um fator fundamental para o aperfeiçoamento é a educação dos engenheiros de software, seus gerentes e outros interessados. A organização de SQA assume a liderança no processo de aperfeiçoamento do software (Capítulo 30) e é um proponente fundamental e patrocinador de programas educacionais.

"Excelência é a capacidade ilimitada de melhorar a qualidade daquilo que se tem a oferecer."

Rick Petin

Gerência dos fornecedores. Adquirem-se três categorias de software de fornecedores externos de software — *pacotes prontos, comerciais* (por exemplo, Microsoft *Office, oferecidos ao usuário em embalagens*), um *shell personalizado* [Hor03] que fornece um esqueleto básico, personalizado de acordo com as necessidades do comprador e *software sob encomenda* que é projetado e construído de forma personalizada a partir de especificações fornecidas pela empresa-cliente. O papel do grupo de SQA é garantir software de alta qualidade por meio da sugestão de práticas específicas de garantia da qualidade que o fornecedor deve (sempre que possível) seguir, e incorporar exigências de qualidade como parte de qualquer contrato com um fornecedor externo.

Administração da segurança. Com o aumento dos crimes cibernéticos e novas regulamentações governamentais referentes à privacidade, toda organização de software deve instituir políticas que protejam os dados em todos os níveis, estabelecer proteção através de firewalls para as aplicações da Internet (WebApps) e garantir que o software não tenha sido alterado internamente, sem autorização. A SQA garante o emprego de processos e tecnologias apropriadas para ter a segurança de software desejada.

Proteção. O fato de o software ser quase sempre um componente fundamental de sistemas que envolvem vidas humanas (por exemplo, aplicações na indústria automotiva ou aeronáutica), o impacto de defeitos ocultos pode ser catastrófico. A SQA pode ser responsável por avaliar o impacto de falhas de software e por iniciar as etapas necessárias para redução de riscos.

Administração de riscos. Embora a análise e a redução de riscos (Capítulo 28) seja preocupação dos engenheiros de software, o grupo de SQA garante que as atividades de gestão de riscos sejam conduzidas apropriadamente e que planos de contingência relacionados a riscos tenham sido estabelecidos.

Além de cada uma dessas preocupações e atividades, a SQA trabalha para garantir que atividades de suporte ao software (por exemplo, manutenção, suporte on-line, documentação e manuais) sejam realizadas ou produzidas tendo a qualidade como preocupação dominante.

INFORMAÇÕES

Recursos para Gestão da Qualidade

Há dezenas de recursos para gestão da qualidade disponíveis na Internet, incluindo associações de profissionais, organizações de padrões e fontes de informação genéricas. Veja aqui os sites que são um bom ponto de partida:

American Society for Quality (ASQ) Software Division
www.asq.org/software
Association for Computer Machinery **www.acm.org**
Data and Analysis Center for Software (DACS)
www.dacs.dtic.mil/
International Organization for Standardization (ISO)
www.iso.ch
ISO SPICE
www.isospice.com
Malcolm Baldridge National Quality Award
www.quality.nist.gov
Software Engineering Institute
www.sei.cmu.edu/
Testes de Software e Engenharia da Qualidade
www.stickyminds.com
Recursos sobre o Seis Sigma
www.isixsigma.com/
www.asq.org/sixsigma/
TickIT International: Tópicos sobre certificação em qualidade
www.tickit.org/international.htm
Gestão da Qualidade Total (TQM, *Total Quality Management*)
Informações gerais:
www.gslis.utexas.edu/~rpollock/tqm.html
Artigos: **www.work911.com/tqmarticles.htm**
Glossário:
www.quality.org/TQM-MSI/TQM-glossary.html

16.3 Tarefas, Metas e Métricas da SQA

A garantia da qualidade de software é composta por uma série de tarefas associadas a dois elementos distintos — os engenheiros de software que realizam o trabalho técnico e um grupo de SQA que tem a responsabilidade pelo planejamento, supervisão, manutenção de registros, análise e relatórios referentes à garantia da qualidade.

Os engenheiros de software tratam da qualidade (e realizam atividades de controle de qualidade) por meio da aplicação de medidas e métodos técnicos consistentes, conduzindo as revisões técnicas e realizando os bem planejados testes de software.

16.3.1 Tarefas da SQA

A prerrogativa do grupo de SQA é ajudar a equipe de software a obter um produto final de alta qualidade. O SEI (Software Engineering Institute) recomenda um conjunto de ações de SQA que tratam do planejamento, da supervisão, da manutenção de registros, da análise e de relatórios relativos à garantia da qualidade. Essas ações são realizadas (ou facilitadas) por um grupo de SQA independente que:

Qual é o papel de um grupo de SQA?

Prepara um plano de SQA para um projeto. O plano é desenvolvido como parte do planejamento de projeto e é revisado por todos os interessados. As ações de garantia da qualidade realizadas pela equipe de engenharia de software e o grupo de SQA são governados pelo plano, que identifica as avaliações, auditorias e revisões a ser realizadas, padrões que são aplicáveis para o projeto, procedimentos para relatório e acompanhamento de erros,

produtos resultantes produzidos pelo grupo de SQA e o *feedback* que será fornecido à equipe de software.

Participa no desenvolvimento da descrição da gestão de qualidade do projeto. A equipe de software seleciona um processo para o trabalho a ser realizado. O grupo de SQA revisa a descrição de processos para conformidade com a política organizacional, padrões internos de software, padrões impostos externamente (por exemplo, ISO-9001) e outras partes do plano de projeto de software.

Revisa as atividades de engenharia de software para verificar sua conformidade com a gestão de qualidade definida. O grupo de SQA identifica, documenta e acompanha desvios do processo e verifica se as correções foram feitas.

Audita produtos de software resultantes designados para verificar sua conformidade com aqueles definidos como parte da gestão de qualidade. O grupo de SQA revisa produtos resultantes selecionados; identifica, documenta e acompanha os desvios; verifica se as correções foram feitas e, periodicamente, relata os resultados de seu trabalho para o gerente de projeto.

Garante que os desvios no trabalho de software e produtos resultantes sejam documentados e tratados de acordo com um procedimento documentado. Podem ser encontrados desvios no plano de projeto, na descrição do processo, padrões aplicáveis ou no artefato da engenharia de software.

Registra qualquer não aderência e relata ao gerenciamento superior. Itens com problemas (que não atendem às especificações) são acompanhados até que tais problemas sejam resolvidos.

Além dessas ações, o grupo de SQA coordena o controle e o gerenciamento de mudanças (Capítulo 22) e ajuda a coletar e analisar métricas de software.

16.3.2 Metas, atributos e métricas

"A qualidade jamais é por acidente; ela sempre é o resultado de intenção extraordinária, esforço sincero, direção inteligente e uma hábil execução; representa a escolha inteligente entre muitas alternativas."

William A. Foster

As ações de SQA descritas na seção anterior são realizadas para atingir um conjunto de metas pragmáticas:

Qualidade dos requisitos. A correção, a completude e a consistência do modelo de requisitos terão forte influência sobre a qualidade de todos os produtos seguintes. A SQA deve assegurar-se de que a equipe de software tenha revisto apropriadamente o modelo de requisitos para a obtenção de um alto nível de qualidade.

Qualidade do projeto. Todo elemento do modelo de projeto deve ser avaliado pela equipe de software para garantir que apresente alta qualidade e que o próprio projeto esteja de acordo com os requisitos. A SQA busca atributos do projeto que sejam indicadores de qualidade.

Qualidade do código. O código-fonte e os produtos relacionados (por exemplo, outras informações descritivas) devem estar em conformidade com os padrões locais de codificação e apresentar características que irão facilitar a manutenção. A SQA deve isolar esses atributos que permitem uma análise razoável da qualidade do código.

Eficácia do controle de qualidade. A equipe de software deve aplicar os recursos limitados de forma a obter a maior probabilidade possível de atingir um resultado de alta qualidade. A SQA analisa a alocação de recursos para revisões e realiza testes para verificar se eles estão ou não sendo alocados da maneira mais efetiva.

A Figura 16.1 (adaptada de [Hya96]) identifica os atributos indicadores da existência de qualidade para cada uma das metas discutidas. As métricas que podem ser utilizadas para indicar a força relativa de um atributo também são mostradas.

FIGURA 16.1

Metas, atributos e métricas para qualidade de software
Fonte: Adaptado de (Hya96)

Meta	Atributo	Métrica
Qualidade das necessidades	Ambiguidade	Número de modificadores ambíguos (por exemplo, muitos, grande, amigável)
	Completude	Número de TBA, TBD
	Compreensibilidade	Número de seções/subseções
	Volatilidade	Número de mudanças por requisito
		Tempo (por atividade) quando é solicitada a mudança
	Facilidade de atribuição	Número de requisitos não atribuíveis ao projeto/código
	Clareza do modelo	Número de modelos UML
		Número de páginas descritivas por modelo
		Número de erros UML
Qualidade do projeto	Integridade da arquitetura	Existência do modelo da arquitetura
	Completude dos componentes	Número de componentes que se atribui ao modelo da arquitetura
		Complexidade do projeto procedural
	Complexidade da interface	Número médio de cliques para chegar a uma função ou conteúdo típico
		Apropriabilidade do layout
	Padrões	Número de padrões usados
Qualidade do código	Complexidade	Complexidade ciclomática
	Facilidade de manutenção	Fatores de projeto (Capítulo 8)
	Compreensibilidade	Porcentagem de comentários internos
		Convenções de atribuição de variáveis
	Reusabilidade	Porcentagem de componentes reutilizados
	Documentação	Índice de legibilidade
Eficiência do controle de qualidade	Alocação de recursos	Porcentagem de horas de pessoal por atividade
	Taxa de completude	Tempo de finalização real *versus* previsto
	Eficácia da revisão	Ver métricas de revisão (Capítulo 14)
	Eficácia dos testes	Número de erros encontrados e criticalidade
		Esforço exigido para corrigir um erro
		Origem do erro

16.4 ABORDAGENS FORMAIS DA SQA

Nas seções anteriores, argumentamos que a qualidade de software é tarefa de todos e que pode ser atingida por meio da prática competente de engenharia de software, bem como da aplicação de revisões técnicas, uma estratégia de testes multiescalonados, melhor controle do artefato de software resultante e as mudanças nele feitas e a aplicação dos padrões aceitos de engenharia de software. Além disso, a qualidade pode ser definida em termos de uma ampla gama de atributos de qualidade, e medida (indiretamente) usando uma variedade de índices e métricas.

WebRef

Informações úteis sobre SQA e métodos formais da qualidade podem ser encontradas em **www.gslis.utexas.edu/~rpollock/tqm.html**.

Ao longo das últimas três décadas, um segmento pequeno, mas eloquente, da comunidade de engenharia de software sustentava que era necessária uma abordagem mais formal para garantir a qualidade de software. Pode-se argumentar que um programa de computador é um objeto matemático. Pode-se definir uma sintaxe e semântica rigorosas para todas as linguagens de programação e está disponível uma rigorosa abordagem da especificação dos requisitos de software (Capítulo 21). Se o modelo de requisitos (especificação) e a linguagem de programação podem ser representados de maneira rigorosa, deve ser possível aplicar uma prova matemática da correção para demonstrar a adequação exata de um programa às suas especificações.

Tentativas de se provar que os programas são corretos não são novas. Dijkstra [Dij76a] e Linger, Mills e Witt [Lin79], entre outros, defenderam provas da correção de programas e associaram-nas ao uso dos conceitos de programação estruturada (Capítulo 10).

16.5 Estatística da Garantia da Qualidade de Software

A estatística da garantia de qualidade reflete uma tendência crescente em toda a indústria de software para tornar mais quantitativa a análise da qualidade. Para software, a estatística da garantia da qualidade implica as seguintes etapas:

? Quais são as etapas necessárias para realizar estatística de SQA?

1. Informações sobre erros e defeitos de software são coletadas e classificadas.
2. É feita uma tentativa de associar cada erro e defeito a sua causa subjacente (por exemplo, a não adequação às especificações, erros de projeto, violação de padrões, comunicação inadequada com o cliente).
3. Usando o princípio de Pareto (80% dos defeitos podem ser associados a 20% de todas as possíveis causas), são isoladas os 20% (as *poucas causas vitais*).
4. Assim que as poucas causas vitais tiverem sido identificadas, prossegue-se para a correção dos problemas que provocaram os erros e defeitos.

Esse conceito relativamente simples representa um importante passo para a criação de uma gestão de qualidade adaptativa em que mudanças são feitas para melhorar aqueles elementos do processo que introduzem erros.

16.5.1 Um exemplo genérico

> "Uma análise estatística, conduzida apropriadamente, é uma dissecação delicada de incertezas, uma cirurgia de suposições."
>
> **M. J. Moroney**

Para ilustrar o uso de métodos estatísticos para a engenharia de software, suponhamos que uma organização de engenharia de software reúna informações sobre erros e defeitos por um período de um ano. Alguns desses erros são revelados à medida que o software é desenvolvido. São encontrados outros (defeitos) após o software ter sido liberado para os usuários finais. Embora centenas de problemas diferentes sejam encontrados, todos podem ser associados a uma (ou mais) das seguintes causas:

- Especificações incompletas ou errôneas (IES, incomplete or erroneous especifications)
- Má interpretação da comunicação do cliente (MCC, misinterpretation of customer communication)
- Desvio intencional das especificações (IDS, intentional deviation from especifications)
- Violação dos padrões de programação (VPS, violation of programming standards)
- Erro na representação de dados (EDR, error in data representation)
- Interface inconsistente de componentes (ICI, inconsistent component interface)
- Erro na lógica de projeto (EDL, error in design logic)
- Testes incompletos ou errôneos (IET, incomplete or erroneous testing)
- Documentação imprecisa ou incompleta (IID, inaccurate or incomplete documentation)
- Erro na tradução do projeto para linguagem de programação (PLT, error in programming language translation of design)
- Interface homem-máquina ambígua ou inconsistente (HCI, ambiguous or inconsistent human/computer interface)
- Outros (MIS, miscellaneous)

"20% do código contém 80% dos erros. Encontre-os, corrija-os!"

Lowell Arthur

Para aplicar a estatística da SQA, é construída a tabela da Figura 16.2. A tabela indica que IES, MCC e EDR são as poucas causas vitais responsáveis por 53% de todos os erros. Deve-se notar, entretanto, que IES, EDR, PLT e EDL seriam escolhidas como as poucas causas vitais se forem considerados apenas os erros graves. Uma vez que forem determinadas as poucas causas vitais, a organização de engenharia de software pode começar a ação corretiva. Por exemplo, para corrigir a MCC, talvez fosse preciso implementar as técnicas de reunião de requisitos (Capítulo 5) para melhorar a qualidade da comunicação do cliente e das especificações. Para melhorar o EDR, pode-se adquirir ferramentas para modelagem de dados e realizar revisões mais rigorosas do projeto de dados.

É importante notar que a ação corretiva se concentra basicamente nas poucas causas vitais. À medida que as poucas causas vitais forem corrigidas, novos candidatos vão para o topo da lista.

As técnicas de estatística da garantia da qualidade para software demonstraram fornecer um aperfeiçoamento substancial da qualidade [Art97]. Em alguns casos, as organizações de software atingiram uma redução de 50% por ano nos defeitos após a aplicação dessas técnicas.

A aplicação de estatística de SQA e o princípio de Pareto podem ser sintetizados em uma única sentença: *Invista seu tempo concentrando-se em coisas que realmente importam, mas, primeiramente, certifique-se de ter entendido aquilo que realmente importa!*

16.5.2 Seis sigma para engenharia de software

Seis Sigma é a estratégia para a estatística da garantia da qualidade mais utilizada na indústria atual. Originalmente popularizada pela Motorola na década de 1980, a estratégia *Seis Sigma* "é uma metodologia rigorosa e disciplinada que usa análise estatística e de dados para medir e melhorar o desempenho operacional de uma empresa através da identificação e da eliminação de defeitos em processos de fabricação e relacionados a serviços" [ISI08]. O termo *Seis Sigma* é derivado de seis desvios-padrão — 3,4 ocorrências (defeitos) por milhão — implicando em um padrão de qualidade extremamente elevado. A metodologia *Seis Sigma* define três etapas essenciais:

Quais são as etapas essenciais da metodologia Seis Sigma?

- *Definir* as necessidades do cliente e os artefatos passíveis de entrega, bem como as metas de projeto através de métodos bem definidos da comunicação com o cliente.
- *Medir* o processo existente e seu resultado para determinar o desempenho da qualidade atual (reunir métricas para defeitos).
- *Analisar* as métricas para defeitos e determinar as poucas causas vitais.

FIGURA 16.2

Coleta de dados para estatística da SQA

	Total		Graves		Moderados		Secundários	
Erro	**No.**	**%**	**No.**	**%**	**No.**	**%**	**No.**	**%**
IES	205	22%	34	27%	68	18%	103	24%
MCC	156	17%	12	9%	68	18%	76	17%
IDS	48	5%	1	1%	24	6%	23	5%
VPS	25	3%	0	0%	15	4%	10	2%
EDR	130	14%	26	20%	68	18%	36	8%
ICI	58	6%	9	7%	18	5%	31	7%
EDL	45	5%	14	11%	12	3%	19	4%
IET	95	10%	12	9%	35	9%	48	11%
IID	36	4%	2	2%	20	5%	14	3%
PLT	60	6%	15	12%	19	5%	26	6%
HCI	28	3%	3	2%	17	4%	8	2%
MIS	56	6%	0	0%	15	4%	41	9%
Total	942	100%	128	100%	379	100%	435	100%

Se já existir uma gestão de qualidade, e for necessário um aperfeiçoamento, a estratégia Seis Sigma sugere duas etapas adicionais:

- *Melhorar* o processo por meio da eliminação das causas fundamentais dos defeitos.
- *Controlar* o processo para garantir que trabalhos futuros não reintroduzam as causas dos defeitos.

Essas etapas essenciais e adicionais são, algumas vezes, conhecidas como método DMAIC (**d**efinir, **m**edir, **a**nalisar, aperfe**i**çoar e **c**ontrolar).

Se uma organização estiver desenvolvendo uma gestão de qualidade (e não aperfeiçoando um já existente), nas etapas essenciais são incluídas:

- *Projetar* o processo para: (1) evitar as causas fundamentais dos defeitos e (2) atender as necessidades do cliente.
- *Verificar* se o modelo de processos irá, de fato, evitar defeitos e atender as necessidades do cliente.

Essa variação é algumas vezes denominada método DMADV (**d**efinir, **m**edir, **a**nalisar, projetar [**d**esign] e **v**erificar).

É melhor deixarmos uma discussão completa sobre Seis Sigma para fontes dedicadas ao assunto. Caso tenha maior interesse, veja [ISI08], [Pyz03] e [Sne03].

16.6 Confiabilidade de Software

"O preço inevitável da confiabilidade é a simplicidade."

C. A. R. Hoare

Não há nenhuma dúvida de que a confiabilidade de um programa de computador é um elemento importante de sua qualidade global. Se um programa falhar frequentemente e repetidas vezes, pouco importa se outros fatores de qualidade de software sejam aceitáveis.

A confiabilidade de software, diferentemente de outros fatores de qualidade, pode ser medida diretamente e estimada usando-se dados históricos e de desenvolvimento. A *confiabilidade de software* é definida em termos estatísticos como "a probabilidade de operação sem falhas de um programa de computador em um dado ambiente por um determinado tempo" [Mus87]. Para ilustrarmos esse conceito, estima-se que o programa *X* tenha uma confiabilidade de 0,999 depois de decorridas oito horas de processamento. Em outras palavras, se o programa *X* tiver de que ser executado 1.000 vezes e precisar de um total de oito horas de tempo de processamento (tempo de execução), é provável que ele opere corretamente (sem falhas) 999 vezes.

Toda vez que discutimos confiabilidade de software, surge uma questão fundamental: Qual o significado do termo *falha*? No contexto de qualquer discussão sobre qualidade de software e confiabilidade, falha é a falta de conformidade com os requisitos de software. Mesmo dentro dessa definição existem variações. As falhas podem ser apenas problemáticas ou catastróficas. Uma determinada falha pode ser corrigida em segundos, enquanto outras necessitarão de semanas ou até mesmo meses para serem corrigidas. Para complicar ainda mais o problema, a correção de uma falha pode, na realidade, resultar na introdução de outros erros que resultarão em outras falhas.

16.6.1 Medidas de confiabilidade e disponibilidade

PONTO-CHAVE

Os problemas de confiabilidade de software podem quase sempre ser associados a defeitos de projeto ou de implementação.

Os primeiros trabalhos sobre confiabilidade de software tentaram extrapolar a matemática da teoria da confiabilidade de hardware para a previsão da confiabilidade de software. A maioria dos modelos de confiabilidade relacionada com hardware tem como base falhas devido a desgaste e não as falhas devido a defeitos de projeto. Em hardware, as falhas devido a desgaste físico (por exemplo, os efeitos decorrentes de temperatura, corrosão, choque) são mais prováveis do que uma falha relacionada ao projeto. Infelizmente, o contrário é verdadeiro para software. Na realidade, todas as falhas de software podem ser associadas a problemas de projeto ou de implementação; o desgaste (ver Capítulo 1) não entra em questão.

Tem havido um debate contínuo sobre a relação entre conceitos-chave na confiabilidade de hardware e sua aplicabilidade ao software. Embora ainda seja preciso estabelecer uma associação

irrefutável, consideram-se alguns conceitos simples que se aplicam aos elementos de ambos os sistemas.

PONTO-CHAVE

É importante notar que o MTBF e medidas relacionadas se baseiam em tempo de CPU, e não em tempo de relógio tradicional.

Se considerarmos um sistema computacional, uma medida de confiabilidade simples é o *tempo médio entre falhas* (MTBF, *mean-time-between-failure*):

$$\text{MTBF} = \text{MTTF} + \text{MTTR}$$

em que os acrônimos MTTF e MTTR são, respectivamente, *tempo médio para falhar* (*mean-time-to-failure*) e *tempo médio para reparar* (*mean-time-to-repair*)[2].

Muitos pesquisadores defendem que o MTBF é uma medida mais útil do que quaisquer outras métricas de software relacionadas com a garantia da qualidade discutidas no Capítulo 23. De maneira simples, um usuário final se preocupa com falhas e não com o número total de defeitos. Como cada defeito contido em um programa não tem a mesma taxa de falhas, o número total de defeitos fornece pouca indicação da confiabilidade de um sistema. Por exemplo, considere um programa que esteve em operação por 3.000 horas de processamento sem nenhuma falha. Vários defeitos neste programa podem não ser detectados por dezenas de milhares de horas antes de serem descobertos. O MTBF com esses erros obscuros poderia ser de 30.000 ou até mesmo 60.000 horas de processador. Outros defeitos, embora ainda não descobertos, poderiam ter uma taxa de falhas de 4.000 ou 5.000 horas. Mesmo se cada um dos erros da primeira categoria (aqueles com MTBF longo) fosse eliminado, o impacto sobre a confiabilidade de software seria desprezível.

AVISO

Alguns aspectos da disponibilidade (não discutidos aqui) não têm nada a ver com falha. Por exemplo, programar a parada do sistema (para funções de suporte) faz com que o software fique indisponível.

Entretanto, o MTBF pode ser problemático por duas razões: (1) ele projeta um período de tempo entre falhas, mas não fornece uma projeção da taxa de falhas e (2) o MTBF pode ser mal interpretado como sendo tempo de vida médio, muito embora *não* seja esse o significado.

Uma medida alternativa de confiabilidade é *falha ao longo do tempo* (FIT, *failures-in-time*) — uma medida estatística de quantas falhas um componente terá ao longo de um bilhão de horas de operação. Consequentemente, 1 FIT equivale a uma falha a cada bilhão de horas de operação.

Além de uma medida da confiabilidade, deve-se também desenvolver uma medida de disponibilidade. *Disponibilidade de software* é a probabilidade de que um programa esteja operando de acordo com os requisitos em um dado instante e é definida da seguinte forma:

$$\text{Disponibilidade} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} \times 100\%$$

A medida de confiabilidade MTBF é igualmente sensível ao MTTF e ao MTTR. A medida de disponibilidade é ligeiramente mais sensível ao MTTR, uma medida indireta da manutenibilidade do software.

16.6.2 Proteção do software

"A segurança das pessoas deve ser a maior das leis."

Cícero

Proteção do software é uma atividade da garantia da qualidade de software que se concentra na identificação e na avaliação de potenciais problemas que podem afetar negativamente um software e provocar falha em todo o sistema. Se os problemas podem ser identificados precocemente na gestão de qualidade, as características para eliminar ou controlar estes problemas podem ser especificadas no projeto de software.

Um processo de modelagem e análise é efetuado como parte de proteção do software. Inicialmente, os problemas são identificados e classificados por criticalidade e risco. Por exemplo, problemas associados a um controle computadorizado de um automóvel podem: (1) provocar uma aceleração descontrolada que não pode ser interrompida, (2) não responder ao acionamento do pedal do breque (através de uma desativação), (3) não operar quando a chave é ativada e (4) perder ou ganhar velocidade lentamente. Uma vez identificados esses perigos no nível de sistema,

2 Embora a depuração (e correções relacionadas) possam ser necessárias como consequência de uma falha, em muitos casos o software funcionará apropriadamente depois de um reinício sem nenhuma outra mudança.

"Não consigo imaginar nenhuma condição que faria com que esse navio viesse a afundar. A construção naval moderna ultrapassou essa condição."

E. J. Smith, capitão do *Titanic*

técnicas de análise são utilizadas para atribuir gravidade e probabilidade de ocorrência.[3] Para ser efetivo, o software deve ser analisado no contexto de todo o sistema. Por exemplo, um erro sutil cometido pelo usuário na entrada de dados (as pessoas são componentes do sistema) talvez seja ampliado por uma falha de software e produza dados de controle que posicione de forma inapropriada um dispositivo mecânico. Se e somente se um conjunto de condições ambientais externas for atendido, a posição imprópria do dispositivo mecânico provocará uma falha desastrosa. Análises técnicas [Eri05] como a análise da árvore de falhas, lógica em tempo real e modelos em redes de Petri podem ser usadas para prever a cadeia de eventos que podem causar problemas e a probabilidade que cada um dos eventos irá ocorrer para criar a cadeia.

Uma vez que os problemas são identificados e analisados, os requisitos relacionados com a proteção podem ser especificados para o software. Ou seja, a especificação pode conter uma lista de eventos indesejáveis e as respostas desejadas pelo sistema para esses eventos. O papel do software em administrar eventos indesejáveis é então indicado.

WebRef

Uma proveitosa coleção de artigos sobre proteção de software pode ser encontrada em **www.safewareeng.com/**.

Embora a confiabilidade de software e a proteção do software estejam intimamente relacionadas entre si, é importante entender a diferença sutil entre elas. A confiabilidade de software usa análise estatística para determinar a probabilidade de ocorrência de uma falha de software. Entretanto, a ocorrência de uma falha não resulta, necessariamente, em um problema ou contratempo. A proteção de software examina as maneiras em que falhas resultam em condições que podem levar a um contratempo. Ou seja, as falhas não são consideradas isoladamente, mas sim avaliadas no contexto de todo o sistema computacional e seu ambiente.

Uma discussão completa sobre proteção de software vai além do escopo deste livro. Caso tenha maior interesse no tema proteção de software e questões relacionadas, veja [Smi05], [Dun02] e [Lev95].

16.7 Os Padrões[4] de Qualidade ISO 9000

Um *sistema de garantia da qualidade* pode ser definido como a estrutura organizacional com responsabilidades, procedimentos, processos e recursos para implementação da gestão da qualidade [ANS87]. Os sistemas de garantia da qualidade são criados para ajudar as organizações a garantir que seus produtos e serviços satisfaçam às expectativas do cliente por meio do atendimento às suas especificações. Tais sistemas cobrem uma grande variedade de atividades englobando todo o ciclo de vida de um produto incluindo planejamento, controle, medições, testes e geração de relatórios e melhorando os níveis de qualidade ao longo de todo o processo de desenvolvimento e fabricação. A ISO 9000 descreve elementos de garantia da qualidade em termos gerais que podem ser aplicados a qualquer empresa, independentemente do tipo de produtos ou serviços.

Para obter a certificação em um dos programas de garantia da qualidade contidos na ISO 9000, as operações e o sistema de qualidade de uma empresa são examinados por auditores independentes, para verificação de sua conformidade ao padrão e operação efetiva. Após aprovação, um organismo representado pelos auditores emite um certificado para a empresa. Auditorias de inspeção semestrais garantem conformidade contínua ao padrão.

WebRef

Podemos encontrar uma longa lista de links para recursos relacionados à ISO 9000/9001 em **www.tantara.ab.ca/info.htm**.

As necessidades delineadas pelos tópicos da ISO 9001:2000 são: responsabilidade administrativa, um sistema de qualidade, revisão contratada, controle de projeto, controle de dados e documentos, identificação e rastreabilidade de produtos, controle de processos, inspeções e testes, ações preventivas e corretivas, registros de controle de qualidade, auditorias de qualidade internas, treinamento, manutenção e técnicas estatísticas. Para que uma organização de software seja certificada com a ISO 9001:2000, tem de estabelecer políticas e procedimentos

3 Essa abordagem é similar aos métodos de análise de riscos descritos no Capítulo 28. A diferença fundamental é a ênfase em problemas de tecnologia em vez de tópicos relacionados a projeto.

4 Essa seção, escrita por Michael Stovsky, foi adaptada de *Fundamentals of ISO 9000*, um livro desenvolvido para *Essential Software Engineering*, um programa de estudos em vídeo desenvolvido pela R. S. Pressman & Associates, Inc. Reimpresso com permissão.

para atender a cada uma das necessidades que acabamos de citar (e outras também) e depois ser capaz de demonstrar que tais políticas e procedimentos estão sendo seguidos. Caso deseje maiores informações sobre a ISO 9001:2000, veja [Ant06], [Mut03] ou [Dob04].

INFORMAÇÕES

O Padrão ISO 9001:2000

A descrição a seguir define os elementos básicos do padrão ISO 9001:2000.

Informações completas sobre o padrão podem ser obtidas da International Organization for Standardization (**www.iso.ch**) e outras fontes na Internet (por exemplo, **www.praxiom.com**).

Estabelecer os elementos de um sistema de gestão da qualidade.
- Desenvolver, implementar e aperfeiçoar o sistema.
- Definir uma política que enfatize a importância do sistema.

Documentar o sistema de qualidade.
- Descrever o processo.
- Produzir um manual operacional.
- Desenvolver métodos para controlar (atualizar) documentos.
- Estabelecer métodos para manutenção de registros.

Dar suporte ao controle e à garantia da qualidade.
- Promover a importância da qualidade entre todos os interessados.
- Focar na satisfação do cliente.
- Definir um plano de qualidade que atenda aos objetivos, às responsabilidades e à autoridade.
- Definir mecanismos de comunicação entre os interessados.

Estabelecer mecanismos de revisão para um sistema de gestão da qualidade.
- Identificar métodos de revisão e mecanismos de *feedback*.
- Definir procedimentos de acompanhamento.
- Identificar recursos de qualidade, incluindo elementos de pessoal, treinamento e infraestrutura.
- Estabelecer mecanismos de controle.
- Para planejamento.
- Para as necessidades do cliente.
- Para as atividades técnicas (por exemplo, análise, projeto, testes).
- Para monitoramento e gerenciamento de projetos.

Definir métodos para reparação.
- Avaliar dados e métricas de qualidade.
- Definir a abordagem para processos e aperfeiçoamento contínuos da qualidade.

16.8 O PLANO DE SQA

O *plano de SQA* fornece um roteiro para instituir a garantia da qualidade de software. Desenvolvido pelo grupo de SQA (ou pela equipe de software, caso não exista um grupo de SQA), o plano serve como um gabarito para atividades de SQA que são instituídas para cada projeto de software.

FERRAMENTAS DO SOFTWARE

Gestão da qualidade de software

Objetivo: o objetivo das ferramentas de SQA é ajudar uma equipe de projeto a avaliar e aperfeiçoar a qualidade do artefato de software resultante.

Mecânica: a mecânica das ferramentas varia. Em geral, o intuito é avaliar a qualidade de um artefato específico. Nota: Normalmente são incluídas várias ferramentas para teste de software (ver Capítulos 17 a 20) dentro da categoria de ferramentas para SQA.

Ferramentas representativas:[1]

ARM, desenvolvida pela NASA (**satc.gsfc.nasa.gov/tools/index.html**), fornece medidas que podem ser utilizadas para avaliar a qualidade de um documento de requisitos de software.

QPR ProcessGuide and Scorecard, desenvolvida pela QPR Software (**www.qpronline.com**), oferece suporte para Seis Sigma e outras abordagens da gestão de qualidade.

Quality Tools and Templates, desenvolvida pela iSixSigma (**www.isixsigma.com/tt/**), descreve uma ampla gama de ferramentas e métodos úteis para gestão da qualidade.

NASA Quality Resources, desenvolvida pelo Goddard Space Flight Center (**sw-assurance.gsfc.nasa.gov/index.php**) fornece formulários proveitosos, gabaritos, listas de verificação e ferramentas para SQA.

5 As ferramentas aqui apresentadas não significam um aval, mas sim uma amostra de ferramentas nessa categoria. Na maioria dos casos, os nomes das ferramentas são marcas registradas por seus respectivos desenvolvedores.

Foi publicado pela IEEE [IEE93] um padrão para planos de SQA. O padrão recomenda uma estrutura que identifique: (1) o propósito e o escopo do plano, (2) uma descrição de todos os artefatos resultantes de engenharia de software (por exemplo, modelos, documentos, código-fonte) que caem dentro do âmbito da SQA, (3) todos os padrões e práticas que são aplicados durante a gestão de qualidade, (4) as ações e tarefas da SQA (incluindo revisões e auditorias) e sua aplicação na gestão de qualidade, (5) as ferramentas e os métodos que dão suporte às ações e tarefas da SQA, (6) procedimentos para administração de configurações de software (Capítulo 22), (7) métodos para montagem, salvaguarda e manutenção de todos os registros relativos à SQA e (8) papéis e responsabilidades dentro da organização relacionados com a qualidade do produto.

16.9 Resumo

A garantia da qualidade de software é uma atividade universal da engenharia de software que é aplicada a cada etapa da gestão de qualidade. A SQA abrange procedimentos para a aplicação efetiva de métodos e ferramentas, a supervisão de atividades de controle de qualidade como revisões técnicas e testes de software, procedimentos para o gerenciamento de mudanças, procedimentos para garantir a conformidade a padrões, bem como mecanismos para medição e geração de relatórios.

Para realizar a garantia da qualidade de software de forma apropriada, devem ser reunidos, avaliados e disseminados os dados sobre o processo de engenharia de software. A estatística da SQA ajuda a melhorar a qualidade do produto e da própria gestão de qualidade. Os modelos de confiabilidade de software estendem as medidas obtidas, possibilitando que dados coletados de defeitos sejam extrapolados para projeção de taxas de falhas e previsões de confiabilidade.

Em suma, devemos observar as palavras de Dunn e Ullman [Dun82]: "A garantia da qualidade de software é uma tradução dos preceitos administrativos e das disciplinas de projeto da garantia da qualidade para a área gerencial e tecnológica da engenharia de software". A capacidade de garantir a qualidade é a medida de uma engenharia disciplinada e madura. Quando esse mapeamento é realizado com sucesso, o resultado é uma engenharia de software com maturidade.

Problemas e Pontos a Ponderar

16.1. Algumas pessoas dizem que "o controle das variações é o cerne do controle de qualidade". Como todo programa que é criado é diferente de qualquer outro programa, quais são as variações que buscamos e como controlá-las?

16.2. É possível avaliar a qualidade de software se o cliente ficar alterando continuamente aquilo que supostamente deveria estar pronto?

16.3. Qualidade e confiabilidade são conceitos relacionados mas são, fundamentalmente, diferentes em uma série de aspectos. Discuta as diferenças.

16.4. Um programa pode ser correto e ainda assim não ser confiável? Explique.

16.5. Um programa pode ser correto e ainda assim não apresentar boa qualidade? Explique.

16.6. Por que normalmente existe tensão entre um grupo de engenharia de software e um grupo de garantia da qualidade de software independente? Isso é salutar?

16.7. Dada a responsabilidade para melhorar a qualidade de software na organização. Qual é a primeira coisa a ser feita? E a seguinte?

16.8. Além da contagem de erros e defeitos, existem outras características contáveis de software que impliquem na qualidade? Quais são, e elas podem ser medidas diretamente?

16.9. O conceito de MTBF para software está sujeito a críticas. Explique o motivo.

16.10. Considere dois sistemas críticos de proteção que são controlados por computador. Enumere pelo menos três perigos para cada um deles que podem ser associados diretamente a falhas de software.

16.11. Adquira uma cópia da ISO 9001:2000 e da ISO 9000-3. Prepare uma apresentação que discuta três necessidades da ISO 9001 e como elas se aplicam no contexto de software.

Leituras e Fontes de Informação Complementares

Livros como os de Hoyle (*Quality Management Fundamentals,* Butterworth-Heinemann, 2007), Tian (*Software Quality Engineering,*Wiley-IEEE Computer Society Press, 2005), El Emam (*The ROI from Software Quality,* Auerbach, 2005), Horch (*Practical Guide to Software Quality Management,* Artech House, 2003) e Nance e Arthur (*Managing Software Quality,* Springer, 2002) são excelentes apresentações em termos gerenciais dos benefícios dos programas formais para garantia da qualidade de software. Livros como os de Deming [Dem86], Juran (*Juran on Quality by Design,* Free Press, 1992) e Crosby [Cro79] e *Quality Is Still Free,* McGraw-Hill, 1995) não se concentram em software, mas são leituras obrigatórias para gerentes seniores com responsabilidade pelo desenvolvimento de software. Gluckman e Roome (*Everyday Heroes of Quality Movement,* Dorset House, 1993) humaniza as questões da qualidade contando a história dos participantes em um processo de qualidade. Kan (*Metrics and Models in Software Quality Engineering,* Addison-Wesley, 1995) apresenta uma visão quantitativa da qualidade de software.

Livros como os de Evans (*Total Quality: Management, Organization and Strategy,* 4. ed., South-Western College Publishing, 2004), Bru (*Six Sigma for Managers,* McGraw-Hill, 2005) e Dobb (*ISO 9001:2000 Quality Registration Step-by-Step,* 3. ed., Butterworth-Heinemann, 2004) são representativos dos muitos livros escritos sobre, respectivamente, TQM, Seis Sigma e ISO 9001:2000.

Pham (*System Software Reliability*, Springer, 2006), Musa (*Software Reliability Engineering: More Reliable Software, Faster Development and Testing,* 2. ed., McGraw-Hill, 2004) e Peled (*Software Reliability Methods*, Springer, 2001) escreveram guias práticos que descrevem métodos para medir e analisar a confiabilidade de software.

Vincoli (*Basic Guide to System Safety,* Wiley, 2006), Dhillon (*Engineering Safety,* World Scientific Publishing Co., Inc., 2003), Hermann (*Software Safety and Reliability,* Wiley-IEEE Computer Society Press, 2000), Storey (*Safety-Critical Computer Systems,* Addison-Wesley, 1996) e Leveson [Lev95] são as discussões mais abrangentes sobre segurança de software e sistemas publicadas até hoje. Além desses, van der Meulen (*Definitions for Hardware and Software Safety Engineers,* Springer-Verlag, 2000) oferece um compêndio completo de importantes conceitos e termos de confiabilidade e proteção; Gartner (*Testing Safety-Related Software,* Springer-Verlag, 1999) é um guia especializado para testar sistemas de proteção críticos; Friedman e Voas (*Software Assessment: Reliability Safety and Testability,* Wiley, 1995) fornecem modelos úteis para avaliar a confiabilidade e a proteção. Ericson (*Hazard Analysis Techniques for System Safety,* Wiley, 2005) tratam da área cada vez mais importante da análise de riscos.

Uma ampla gama de fontes de informação sobre garantia da qualidade de software encontra-se à disposição na Internet. Uma lista atualizada de referências que são relevantes para a SQA pode ser encontrada no site **www.mhhe.com/engcs/compsci/pressman/professional/olc/ser.htm**.