Garantia da Qualidade de Software

Herysson R. Figueiredo
herysson.figueiredo@ufn.edu.br

Sumário

- O que é SQA
- Os elementos da SQA
- Tarefas, Metas e Métricas da SQA
- Estatística da Garantia da Qualidade de Software
- Seis sigma
- Confiabilidade de Software
- Proteção do software
- ISO 9000

Garantia da Qualidade - *Software Quality Assurance*

A garantia da qualidade estabelece a infraestrutura que suporta métodos sólidos de engenharia de software, gerenciamento racional de projeto e ações de controle de qualidade fundamentais para a construção de software de alta qualidade.

Garantia da Qualidade - *Software Quality Assurance*

A garantia da qualidade de software (SQA) engloba:

- um processo de SQA,
- tarefas específicas de garantia da qualidade e controle da qualidade (inclusive revisões técnicas e uma estratégia de testes multi escalonados),
- prática efetiva de engenharia de software (métodos e ferramentas),
- controle de todos os artefatos de software e as mudanças feitas nesses produtos,
- um procedimento para garantir a conformidade com os padrões de desenvolvimento de software (quando aplicáveis)
- mecanismos de medição e de relatórios.

Garantia da Qualidade - *Software Quality Assurance*

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 como elementos-chave da gestão da qualidade.

Garantia da Qualidade - *Software Quality Assurance*

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.

Garantia da Qualidade - *Software Quality Assurance*

A garantia e o controle da qualidade são atividades essenciais para qualquer empresa de produtos a ser usados por terceiros.

A definição da garantia da qualidade de software é um:

> “padrão de ações planejado e sistematizado, ações essas exigidas para garantir alta qualidade no software.”

Verificação e Validação (V&V)

O Papel da Verificação e Validação (V&V) na SQA

Verificação (**Estou construindo o produto da forma correta?**): A verificação foca em garantir que os produtos de uma determinada fase do desenvolvimento estejam em conformidade com as condições e padrões estabelecidos no início daquela fase.

- Revisões
- Padrões
- Auditorias
- Qualidade do Projeto e do Código

O Papel da Verificação e Validação (V&V) na SQA

Validação (Estou construindo o produto certo?): A validação é o processo de avaliar o software ao final do desenvolvimento para garantir que ele atende às necessidades e requisitos do cliente.

- Testes de Software

Elementos da SQA

Elementos da SQA

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:

- Padrões
- Revisões e auditorias
- Testes
- Coleta e análise de erros/defeitos.
- Gerenciamento de mudanças

Elementos da SQA

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:

- Educação
- Gerência dos fornecedores
- Administração da segurança
- Proteção
- Administração de riscos

Padrões

Os padrões (ISO, IEEE) 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. 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.

Testes

Os **testes** de software são uma função de controle de qualidade com um objetivo principal de 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 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 e é um proponente fundamental e patrocinador de programas educacionais.

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** : que fornece um esqueleto básico, personalizado de acordo com as necessidades do comprador;
- **software sob encomenda** que é projetado e construído de forma personalizada a partir de especificações fornecidas pela empresa-cliente.

Gerência dos fornecedores

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.

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 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.

SQA

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.

Tarefas, Metas e Métricas da SQA

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.

Tarefas, Metas e Métricas da SQA

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.

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.

Tarefas da SQA

Essas ações são realizadas (ou facilitadas) por um grupo de SQA independente que:

- Prepara um plano de SQA para um projeto.
- Participa no desenvolvimento da descrição da gestão de qualidade do projeto.
- Revisa as atividades de engenharia de software para verificar sua conformidade com a gestão de qualidade definida.

Tarefas da SQA

Essas ações são realizadas (ou facilitadas) por um grupo de SQA independente que:

- Audita produtos de software resultantes designados para verificar sua conformidade com aqueles definidos como parte da gestão de qualidade.
- Garante que os desvios no trabalho de software e produtos resultantes sejam documentados e tratados de acordo com um procedimento documentado.
- Registra qualquer não aderência e relata ao gerenciamento superior.

Metas, atributos e métricas

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.

Metas, atributos e métricas

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.

Metas, atributos e métricas

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.

Metas, atributos e métricas

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.

Metas, atributos e métricas

Nas próximas imagens identificamos 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.

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

Meta	Atributo	Métrica
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 ciclométrica
	Facilidade de manutenção	Fatores de projeto
	Compreensibilidade	Porcentagem de comentários internos
		Convenções de atribuição de variáveis
	Reusabilidade	Porcentagem de componentes reutilizados
	Documentação	Índice de legibilidade

Metas, atributos e métricas

Meta	Atributo	Métrica
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
	Eficácia dos testes	Número de erros encontrados e criticalidade
		Esforço exigido para corrigir um erro
		Origem do erro

Estatística da Garantia da Qualidade de Software

Para software, a estatística da garantia da qualidade implica as seguintes etapas:

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
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.

Um exemplo genérico

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*)

Um exemplo genérico

Embora centenas de problemas diferentes sejam encontrados, todos podem ser associados a uma (ou mais) das seguintes causas:

- 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*)

Um exemplo genérico

Embora centenas de problemas diferentes sejam encontrados, todos podem ser associados a uma (ou mais) das seguintes causas:

- Interface homem-máquina ambígua ou inconsistente (HCI, *ambiguous or inconsistent human/computer interface*)
- Outros (MIS, *miscellaneous*)

	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%

Six Sigma (Seis Sigma)


CONTROL
DEFINE
SIX
SIGMA
MEASURE
IMPROVE
ANALYZE

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"

Seis sigma para engenharia de software

A metodologia **Seis Sigma** define três etapas essenciais:

- **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.

Seis sigma para engenharia de software

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).

Seis sigma para engenharia de software

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

- **Projetar** o processo para:
  - evitar as causas fundamentais dos defeitos
  - 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).

Confiabilidade de Software

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.

Confiabilidade de Software

A confiabilidade de software, diferentemente de outros fatores de qualidade, pode ser medida diretamente e estimada usando-se dados históricos e de desenvolvimento.

Confiabilidade de Software

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"*

Medidas de confiabilidade e disponibilidade

Todas as falhas de software podem ser associadas a problemas de projeto ou de implementação; o desgaste (ao contrário do hardware) não entra em questão.

Medidas de confiabilidade e disponibilidade

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

$$MTBF = MTTF + MTTR$$

Medidas de confiabilidade e disponibilidade

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.

Medidas de confiabilidade e disponibilidade

Entretanto, o MTBF (*mean-time-between-failure*) pode ser problemático por duas razões:

- ele projeta um período de tempo entre falhas, mas não fornece uma projeção da taxa de falhas
- o MTBF pode ser mal interpretado como sendo tempo de vida médio, muito embora não seja esse o significado.

Medidas de confiabilidade e disponibilidade

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.

Medidas de confiabilidade e disponibilidade

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:

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

Proteção do software

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.

Proteção do 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.

Proteção do software

Por exemplo problemas associados a um controle computadorizado de um automóvel podem:

- provocar uma aceleração descontrolada que não pode ser interrompida,
- não responder ao acionamento do pedal do breque (através de uma desativação),
- não operar quando a chave é ativada e
- perder ou ganhar velocidade lentamente.

Proteção do software

Uma vez identificados esses perigos no nível de sistema, técnicas de análise são utilizadas para atribuir gravidade e probabilidade de ocorrência.

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.

Bibliografia

PRESSMAN, Roger S. Engenharia de software. 7. ed. Rio de Janeiro: McGraw Hill, 20011.

KOSCIANSKI, André; SOARES, Michel dos Santos. Qualidade de software: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software. São Paulo: Novatec, 2ª ed., 2007.

SOMMERVILLE, Ian. Engenharia de Software. 8. ed. São Paulo: Addison Wesley, 2007