Gerenciamento da Qualidade de Software

Capítulo 10
SWEBOK

Professor
Wilson Amaral

Sumário

Introdução

O que é qualidade de software e por que é tão importante que seja incluída em muitas áreas de conhecimento (Knoledge Areas - KA) do Guia SWEBOK?
Uma razão é que o termo qualidade de software vem sendo utilizado pra tudo. A qualidade do software pode se referir às características desejáveis de produtos de software, na medida em que um determinado produto de software possui essas características e a processos, ferramentas e técnicas usadas para atingir essas características.
Ao longo dos anos, autores e organizações definiram o termo qualidade de forma diferente. Para Phil Crosby, é "conformidade com os requisitos"[1]. Watts Humphrey refere-se à qualidade de software como "alcançar excelentes níveis de adequação ao uso"[2]. Enquanto isso, a IBM cunhou a frase "qualidade orientada para o mercado", onde o "cliente é o árbitro final"[3].

Para a norma ISO/IEC 25010, a qualidade do software é definida como a "capacidade do produto de software em satisfazer às necessidades explícitas e implícitas sob condições especificadas"[4] e também, como "o grau em que um produto de software atende com precisão aos requisitos estabelecidos, às necessidades, desejos e expectativas das partes interessadas"[5].

Todas as definições abordam a premissa de conformidade com os requisitos. Nenhum deles se refere a tipos de requisitos (funcionais ou não funcionais, como confiabilidade, desempenho, segurança ou qualquer outra característica). Significativamente, no entanto, essas definições enfatizam que a qualidade depende de requisitos.
Essas definições também ilustram outro motivo para a prevalência da qualidade do software em nosso curso, uma ambiguidade frequente de qualidade de software versus requisitos de qualidade de software. Os requisitos de qualidade de software são os atributos do que o software tem que fazer (Requisitos Funcionais) e as restrições que o software tem que atender (Requisitos Não Funcionais). Os requisitos de software também podem especificar o uso de recursos, um protocolo de comunicação ou muitas outras características.

A qualidade do software é obtida pela conformidade com todos os requisitos, independentemente de qual característica é especificada ou de como os requisitos são agrupados ou nomeados.
A qualidade do software é um parâmetro básico de um esforço de engenharia de software para todos os produtos projetados. O objetivo principal é fornecer o máximo de valor para as partes interessadas, equilibrando as restrições de custo e cronograma de desenvolvimento; às vezes, isso é caracterizado como "adequação para uso".

O valor das partes interessadas é expresso em requisitos. Para produtos de software, as partes interessadas poderiam avaliar o preço (o que pagam pelo produto), o prazo de entrega (a rapidez com que obtêm o produto) e a qualidade do software. A seguir teremos uma visão geral das práticas, ferramentas e técnicas para definir a qualidade do software e avaliar o estado da qualidade do software durante o desenvolvimento, a manutenção e a implantação.

As referências citadas (notas de rodapé) fornecem detalhes adicionais.
Boa leitura!

[1] P.B. Crosby, Quality Is Free, McGraw-Hill, 1979.
[2] W. Humphrey, Managing the Software Process, Addison-Wesley, 1989
[3] S.H. Kan, Metrics and Models in Software Quality Engineering, 2nd ed., Addison Wesley, 2002 pág. 08
[4] ISO/IEC 25010:2011 Systems and Software Engineering - Systems and Software Quality Models, ISO/IEC, 2011.
[5] IEEE P730™/D8 Draft Standard for Software Quality Assurance Processes, IEEE, 2012.

DISCRIMINAÇÃO DOS TÓPICOS DA QUALIDADE DO SOFTWARE

O detalhamento dos tópicos para a área de Qualidade de Software é apresentado na Figura abaixo:

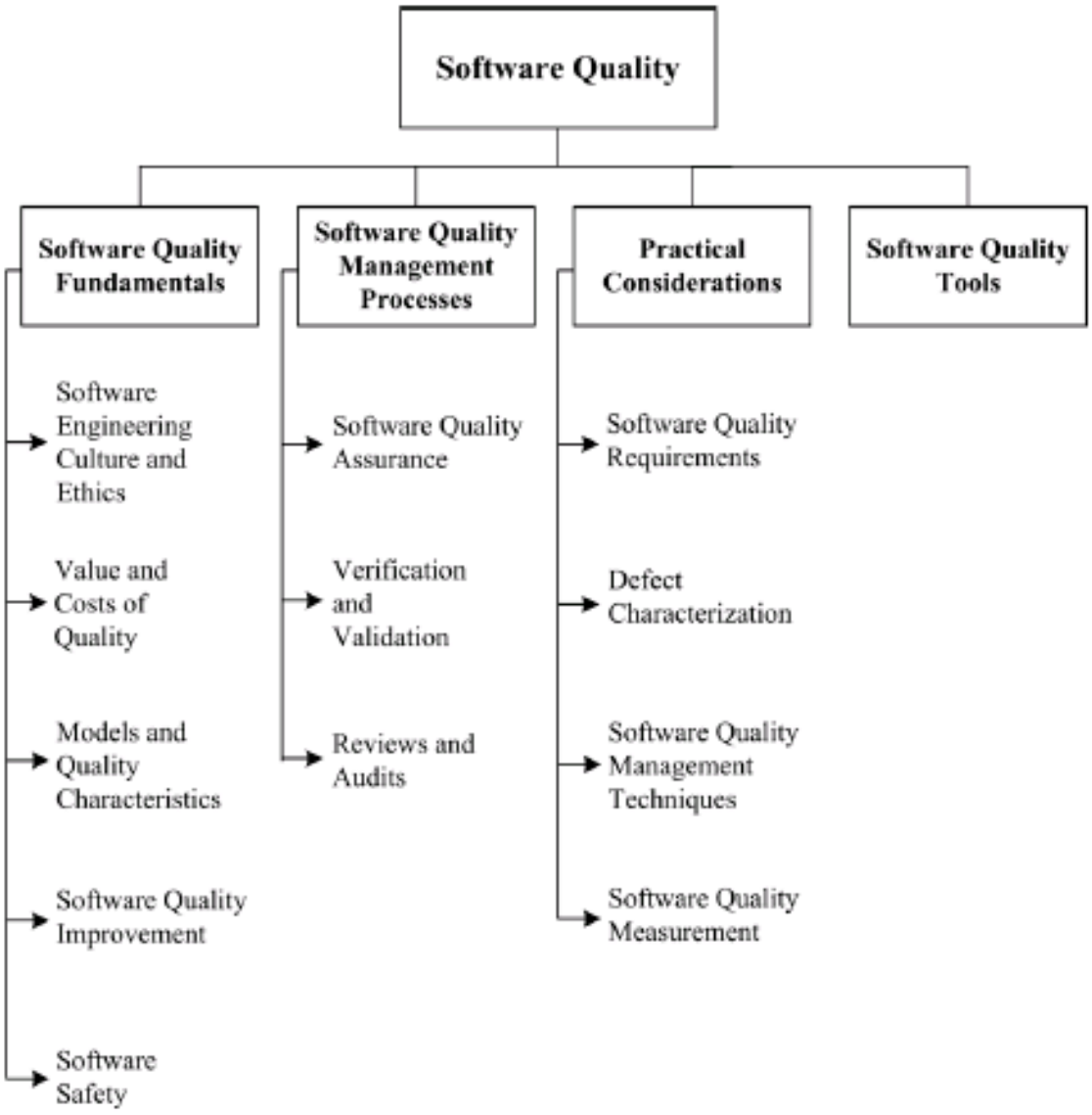


Figure 10.1. Breakdown of Topics for the Software Quality KA

1. Fundamentos de Qualidade de Software

Chegar a um acordo sobre o que constitui qualidade para todas as partes interessadas e comunicar claramente aos engenheiros de software exige que os vários aspectos da qualidade sejam formalmente definidos e discutidos. Um engenheiro de software deve entender conceitos de qualidade, características, valores e sua aplicação ao software em desenvolvimento ou manutenção. O conceito importante é que os requisitos de software definem os atributos de qualidade necessários do software e influenciam os métodos de medição e os critérios de aceitação para avaliar até que ponto o software e a documentação relacionada atingem os níveis de qualidade desejados.

1.1. Cultura e Ética em Engenharia de Software

Espera-se que os engenheiros de software compartilhem um compromisso com a qualidade do software como parte de sua cultura. Uma cultura saudável de engenharia de software inclui muitas características, incluindo o entendimento de que as compensações entre custo, cronograma e qualidade são fatores básicos da engenharia de qualquer produto. Uma forte ética de engenharia de software pressupõe que os engenheiros relatam com precisão informações, condições e resultados relacionados à qualidade. A ética também desempenha um papel significativo na qualidade do software, na cultura e nas atitudes dos engenheiros de software. A IEEE Computer Society e a ACM desenvolveram um código de ética e prática profissional para atender a esses requisitos de qualidade (ver capítulo 11 SWEBOK).

1.2. Valor e Custos da Qualidade

Definir e alcançar a qualidade de software não é simples. As características de qualidade podem ou não ser requeridas, ou podem ser exigidas em maior ou menor grau, e compensações podem ser feitas entre elas. Para ajudar a determinar o nível de qualidade do software, ou seja, alcançar o valor das partes interessadas, esta seção apresenta o custo da qualidade do software, definindo como um conjunto de medições derivadas da avaliação econômica dos processos de desenvolvimento e manutenção da qualidade do software. As medidas do custo da qualidade do software são exemplos de medições de processo que podem ser usadas para inferir características de um produto. A premissa subjacente ao custo da qualidade do software é que o nível de qualidade de um produto de software pode ser inferido do custo das atividades relacionadas a lidar com as consequências da má qualidade.

Má qualidade significa que o produto de software não "satisfaz plenamente as necessidades declaradas e implícitas" ou "requisitos estabelecidos".

Existem quatro categorias de custo de qualidade:

Categoria	Descrição
Prevenção	Incluem investimentos em esforços de melhoria de processos de software, infraestrutura de qualidade, ferramentas de qualidade, treinamento, auditorias e revisões de gerenciamento. Esses custos geralmente não são específicos de um projeto; eles abrangem a organização.
Avaliação	Os custos de avaliação surgem de atividades de projeto que encontram defeitos. Essas atividades de avaliação podem ser categorizadas em custos de revisões (design) e custos de testes (teste de unidade de software, integração de software, teste de nível de sistema, teste de aceitação). Estes custos são estendidos aos fornecedores de software subcontratados.
Falha interna	Custos de falhas internas são aqueles incorridos para corrigir defeitos encontrados durante as atividades de avaliação e descobertos antes da entrega do produto de software ao cliente.
Falha externa	Os custos de falha externa incluem atividades para responder a problemas de software descobertos após a entrega ao cliente.

Os engenheiros de software devem ser capazes de usar os métodos de custo para verificar os níveis de qualidade de software e também devem apresentar alternativas de qualidade e seus custos para que as compensações entre custo, cronograma e entrega do valor das partes interessadas possam ser feitas. Os custos de um projeto de software devem ser medidos com base na literatura apresentada no capítulo 7 do PMBOK.

1.3. Modelos e características de qualidade

A terminologia para características de qualidade de software difere de uma taxonomia (ou modelo de qualidade de software) para outra, podendo cada modelo ter um número diferente de níveis hierárquicos e um número total diferente de características. Vários autores produziram modelos de características ou atributos de qualidade de software que podem ser úteis para discutir, planejar e classificar a qualidade de produtos de software. A ISO / IEC 25010: 2011[6] define a qualidade e a qualidade do produto em uso como dois modelos de qualidade relacionados.

1.3.1. Qualidade do Processo de Software

O gerenciamento da qualidade de software e a qualidade do processo de engenharia de software têm um impacto direto na qualidade do produto de software. Os modelos e critérios que avaliam os recursos das organizações de software são principalmente considerações de organização e gerenciamento de projetos e, como tal, são abordados nos capítulos de Gerenciamento de Engenharia de Software (Capítulo 7 SWEBOK) e de Processos da Engenharia de Software (Capítulo 8 SWEBOK). Não é possível distinguir completamente a qualidade do processo da qualidade do produto porque os resultados do processo incluem produtos. Determinar se um processo tem a capacidade de produzir consistentemente produtos de qualidade desejada não é simples.

O processo de engenharia de software é discutido no capítulo 08 do SWEBOK, Processo de Engenharia de Software e influencia as características de qualidade dos produtos de software, que por sua vez, afetam a qualidade como percebida pelos stakeholders.

1.3.2. Qualidade do produto de software

O engenheiro de software, em primeiro lugar, deve determinar o verdadeiro propósito do software. A esse respeito, os requisitos das partes interessadas são primordiais e incluem requisitos de qualidade além dos requisitos funcionais. Assim, os engenheiros de software têm a responsabilidade de elicitar requisitos de qualidade que podem não estar explícitos no início e entender sua importância, bem como o nível de dificuldade em alcançá-los. Todos os processos de desenvolvimento de software (por exemplo, elicitar requisitos, projetar, construir, verificar e melhorar a qualidade) são projetados tendo em mente esses requisitos de qualidade e podem acarretar custos adicionais de desenvolvimento se atributos como segurança e confiabilidade forem importantes. O termo produto de trabalho significa qualquer artefato resultante de um processo usado para criar o produto de software final. Exemplos de um produto de trabalho incluem uma especificação de sistema / subsistema, uma especificação de requisitos de software para um componente de software de um sistema, uma descrição de design de software, código-fonte, documentação de teste de software ou relatórios. Embora alguns tratamentos de qualidade sejam descritos em termos de software final e desempenho do sistema, a boa prática de engenharia exige que os produtos de trabalho intermediários relevantes para a qualidade sejam avaliados em todo o processo de engenharia de software.

1.4. Melhoria da Qualidade de Software

A qualidade dos produtos de software pode ser melhorada através de processos preventivos ou de um processo iterativo de melhoria contínua, o que requer controle de gerenciamento, coordenação e feedback de muitos processos concorrentes:

1) os processos do ciclo de vida do software,
2) o processo de falha / detecção, remoção e prevenção de defeitos,
3) o processo de melhoria da qualidade.

[6] ISO/IEC 25010:2011 Systems and Software Engineering - Systems and Software Quality Models, ISO/IEC, 2011.

A teoria e os conceitos por trás da melhoria da qualidade - como a criação de qualidade por meio da prevenção e detecção precoce de defeitos, melhoria contínua e foco das partes interessadas - são pertinentes à engenharia de software. Esses conceitos são baseados no trabalho de especialistas em qualidade que afirmaram que a qualidade de um produto está diretamente ligada à qualidade do processo usado para criá-lo. Abordagens como o ciclo de melhoria de Deming do Planejar-Fazer-Verificar-Agir (PDCA), entrega evolutiva, kaizen e implantação de função de qualidade (QFD) oferecem técnicas para especificar objetivos de qualidade e determinar se eles são atendidos. O IDEAL do Software Engineering Institute é outro método[7]. A gestão da qualidade é agora reconhecida pelo Guia SWEBOK como uma disciplina importante. O patrocínio da gerência apóia avaliações de processos e produtos e seus resultados. Em seguida, um programa de melhoria é desenvolvido, identificando ações detalhadas e projetos de melhoria a serem abordados em um prazo viável. O suporte de gerenciamento implica que cada projeto de melhoria possui recursos suficientes para atingir a meta definida para ele. O patrocínio de gerenciamento é solicitado com frequência pela implementação de atividades de comunicação pró-ativas.

1.5. Segurança de Software

Sistemas críticos são aqueles em que uma falha pode prejudicar a vida humana, outros seres vivos, estruturas físicas ou o ambiente. A segurança nesses sistemas é crítica para o software. Há um número crescente de aplicativos de software críticos em um número crescente de indústrias. Exemplos de sistemas com software críticos incluem sistemas de transporte de massa, fábricas de produtos químicos e dispositivos médicos. A falha do software nesses sistemas pode ter efeitos catastróficos. Existem padrões da indústria, como o DO-178C[8], e processos emergentes, ferramentas e técnicas para o desenvolvimento de software crítico.

A intenção desses padrões, ferramentas e técnicas é reduzir o risco de injetar falhas no software e, assim, melhorar a confiabilidade do software. O software crítico pode ser classificado como direto ou indireto. Direto é aquele software incorporado em um sistema de segurança crítica, como o computador de controle de vôo de uma aeronave. Indireto inclui aplicativos de software usados para desenvolver softwares críticos. O software indireto está incluído em ambientes de engenharia de software e ambientes de teste de software. Três técnicas complementares para reduzir o risco de falha são evitação, detecção e remoção, e limitação de danos. Essas técnicas afetam os requisitos funcionais do software, os requisitos de desempenho do software e os processos de desenvolvimento.

Níveis crescentes de risco implicam níveis crescentes de garantia de qualidade de software e técnicas de controle, como inspeções. Níveis de risco mais altos podem exigir inspeções mais completas de requisitos, design e código ou o uso de técnicas analíticas mais formais. Outra técnica para gerenciar e controlar o risco de software é criar casos de garantia. Um caso de garantia é um artefato razoável, auditável, criado para apoiar a alegação de que suas reivindicações estão satisfeitas. Ele contém os seguintes relacionamentos: uma ou mais declarações sobre propriedades; argumentos que ligam logicamente as evidências e quaisquer suposições às reivindicações; e um corpo de evidências e suposições que sustentam esses argumentos[9].

[7] D. Galin, Software Quality Assurance: From Theory to Implementation, Pearson Education Limited, 2004.
[8] RTCA DO-178C, Software Considerations in Airborne Systems and Equipment Certification, Radio Technical Commission for Aeronautics, 2011.
[9] IEEE Std. 15026.1-2011 Trial-Use Standard Adoption of ISO/IEC TR 15026-1:2010 Systems and Software Engineering—Systems and Software Assurance—Part 1: Concepts and Vocabulary, IEEE, 2011.

2. Processos de gerenciamento de qualidade de software

O gerenciamento de qualidade de software é a coleção de todos os processos que garantem que os produtos de software, serviços e implementações de processos de ciclo de vida atendam aos objetivos organizacionais de qualidade de software e atinjam a satisfação das partes interessadas[10]. O gerenciamento de qualidade de software define processos, proprietários de processos, requisitos para os processos, medições dos processos e suas saídas e canais de feedback durante todo o ciclo de vida do software. O gerenciamento de qualidade de software compreende quatro subcategorias:

Categoria	Descrição
Planejamento de qualidade de software	determinar quais padrões de qualidade devem ser usados, definir metas de qualidade específicas e estimar o esforço e o cronograma das atividades de qualidade de software. Em alguns casos, o planejamento de qualidade de software também inclui a definição dos processos de qualidade de software a serem usados.
Garantia de qualidade de software	definem e avaliam a adequação dos processos de software para fornecer evidências que estabelecem a confiança de que os processos de software são apropriados e produzem produtos de software de qualidade adequada para os fins pretendidos.
Controle de qualidade de software	examinam artefatos específicos do projeto (documentos e executáveis) para determinar se estão em conformidade com os padrões estabelecidos para o projeto (incluindo requisitos, restrições, projetos, contratos e planos). O SQC avalia produtos intermediários, assim como os produtos finais.
Melhoria de processo de software	lida com melhoria e tem vários nomes dentro da indústria de software, incluindo melhoria de qualidade de software e ação corretiva e preventiva de software. As atividades nessa categoria buscam melhorar a eficácia do processo, a eficiência e outras características com o objetivo final de melhorar a qualidade do software.

Os processos de qualidade de software consistem em tarefas e técnicas para indicar como os planos de software (por exemplo, gerenciamento de software, desenvolvimento, gerenciamento de qualidade ou planos de gerenciamento de configuração) estão sendo implementados e como os produtos intermediários e finais estão atendendo aos requisitos especificados. Os resultados dessas tarefas são reunidos em relatórios para gerenciamento antes que as ações corretivas sejam tomadas. O gerenciamento de um processo de software tem a tarefa de garantir que os resultados desses relatórios sejam precisos. O gerenciamento de riscos também pode desempenhar um papel importante no fornecimento de software de qualidade. A incorporação de técnicas disciplinadas de análise e gerenciamento de risco nos processos do ciclo de vida do software pode ajudar a melhorar a qualidade do produto. A respeito do gerenciamento de riscos, o capítulo 07 do SWEBOK (página 06) apresenta que:

> "Risco e incerteza são conceitos relacionados mas distintos. Incerteza resulta da falta de informação. O risco é caracterizado pela probabilidade de um evento que resultará em um impacto negativo em um projeto. O risco é frequentemente o resultado da incerteza. O inverso do risco é a oportunidade, que é caracterizada pela probabilidade de que um evento com um resultado positivo possa ocorrer. O gerenciamento de riscos envolve a identificação de fatores de risco e a análise do impacto provável e potencial de cada fator

[10] IEEE Std. 12207-2008 (a.k.a. ISO/IEC 12207:2008) Standard for Systems and Software Engineering—Software Life Cycle Processes, IEEE, 2008 e ISO 9000:2005 Quality Management Systems—Fundamentals and Vocabulary, ISO, 2005.

de risco, a priorização de fatores de risco e o desenvolvimento de estratégias de mitigação de riscos para reduzir a probabilidade e minimizar o impacto negativo se um fator de risco se tornar um problema. Os métodos de avaliação de risco (por exemplo, opinião especializada, dados históricos, árvores de decisão e simulações de processo) podem às vezes ser usados para identificar e avaliar os fatores de risco. As condições de abandono do projeto também podem ser determinadas neste ponto em discussão com todas as partes interessadas relevantes. Os aspectos de risco exclusivos do software, como a tendência dos engenheiros de software de adicionar recursos desnecessários ou os riscos relacionados à natureza intangível do software, podem influenciar o gerenciamento de riscos de um projeto de software. Uma atenção especial deve ser dada ao gerenciamento de riscos relacionados a requisitos de qualidade de software, como segurança ou proteção. O gerenciamento de riscos deve ser feito não apenas no início de um projeto, mas também em intervalos periódicos ao longo do ciclo de vida do projeto."

2.1. Garantia de Qualidade de Software

Para acabar um mal-entendido generalizado, vamos começar definindo que a garantia de qualidade de software não é apenas testar o software. Garantia de qualidade de software é um conjunto de atividades que definem e avaliam a adequação dos processos de software para fornecer evidências que estabelecem a confiança de que os processos de software são apropriados e produzem produtos de software de qualidade adequada aos fins pretendidos.

Um atributo-chave da garantia de qualidade de software é seu objetivo em relação ao projeto. O objetivo também pode ser organizacional independente do projeto; isto é, livre de pressões técnicas, gerenciais e financeiras do projeto[11]. A garantia de qualidade de software possui dois aspectos: garantia do produto e garantia do processo, que são explicados na seção 2.3 (mais abaixo). O plano de qualidade de software (em alguns setores da indústria denominado plano de garantia de qualidade de software) define as atividades e tarefas empregadas para garantir que o software desenvolvido para um produto específico satisfaça os requisitos estabelecidos do projeto e as necessidades do usuário dentro das restrições de custo e cronograma do projeto e seja proporcional com riscos do projeto. Primeiro garante que as metas de qualidade sejam claramente definidas e compreendidas. As atividades e tarefas de qualidade do plano de garantia de qualidade de software são especificadas com seus custos, requisitos de recursos, objetivos e cronograma em relação aos objetivos relacionados nos planos de gerenciamento de engenharia, desenvolvimento e manutenção do software.

O plano de garantia de qualidade de software deve ser consistente com o plano de gerenciamento de configuração de software (Capítulo 06 do SWEBOK). O plano de garantia de qualidade de software identifica documentos, normas, práticas e convenções que regem o projeto e como esses itens são verificados e monitorados para garantir adequação e conformidade. O plano de garantia de qualidade de software também identifica medidas; técnicas estatísticas; procedimentos para relato de problemas e ações corretivas; recursos como ferramentas, técnicas e metodologias; segurança para mídia física; treinamento; relatórios e documentação. Além disso, o plano de garantia de qualidade de software aborda as atividades de garantia de qualquer outro tipo de atividade descrita nos planos de software - tais como aquisição de software de fornecedor para o projeto, instalação comercial de software de prateleira e serviço após a entrega do software. Também pode conter critérios de aceitação, bem como atividades de relatórios e gerenciamento que são críticos para a qualidade do software.

[11] IEEE P730™/D8 Draft Standard for Software Quality Assurance Processes, IEEE, 2012.

2.2. Validação de verificação

O objetivo da tarefa de validação e verificação é ajudar a organização de desenvolvimento a criar qualidade no sistema durante o ciclo de vida. Os processos de validação e verificação fornecem uma avaliação objetiva de produtos e processos ao longo do ciclo de vida do software. Essa avaliação demonstra se os requisitos estão corretos, completos, precisos, consistentes e testáveis. Os processos de validação e verificação determinam se os produtos de desenvolvimento de uma determinada atividade estão em conformidade com os requisitos dessa atividade e se o produto satisfaz seu uso pretendido e as necessidades do usuário.

A verificação é uma tentativa de garantir que o produto seja construído corretamente, no sentido de que os produtos de saída de uma atividade atendem às especificações impostas a eles em atividades anteriores. A validação é uma tentativa de garantir que o produto certo seja construído, ou seja, o produto atenda ao seu propósito específico.

Tanto o processo de verificação quanto o processo de validação começam no início da fase de desenvolvimento ou manutenção. Eles fornecem um exame das principais características do produto em relação ao antecessor imediato do produto e às especificações a serem cumpridas. O propósito do planejamento de validação e verificação é garantir que cada recurso, função e responsabilidade seja claramente atribuído.

Os documentos do plano de validação e verificação resultantes descrevem os vários recursos e suas funções e atividades, bem como as técnicas e ferramentas a serem usadas. Uma compreensão dos diferentes propósitos de cada atividade de validação e verificação ajuda no planejamento cuidadoso das técnicas e recursos necessários para cumprir seus propósitos. O plano também aborda o gerenciamento, a comunicação, as políticas e os procedimentos das atividades de validação e verificação e sua interação, bem como os requisitos de relatórios e documentação de defeitos.

2.3. Revisões e Auditorias

Revisões e processos de auditoria são amplamente definidos como estáticos - o que significa que nenhum programa ou modelo de software é executado - exame de artefatos de engenharia de software com respeito a padrões que foram estabelecidos pela organização ou projeto para esses artefatos. Diferentes tipos de revisões e auditorias distinguem-se por seu propósito, níveis de independência, ferramentas e técnicas, papéis e pelo assunto da atividade. As auditorias de garantia de produto e garantia de processo são normalmente conduzidas por pessoal de garantia de qualidade de software que são independentes das equipes de desenvolvimento. Revisões gerenciais são conduzidas por gerenciamento organizacional ou de projeto. A equipe de engenharia realiza revisões técnicas.

• Revisões gerenciais avaliam os resultados reais do projeto em relação aos planos.

• Revisões técnicas (incluindo inspeções, instruções passo a passo e verificação de bancada) examinam os produtos de trabalho de engenharia.

• Auditorias de garantia de processo. As atividades de garantia de processos de software asseguram que os processos usados para desenvolver, instalar, operar e manter software estejam em conformidade com os contratos, cumpram com quaisquer leis, regras e regulamentos impostos e sejam adequados, eficientes e eficazes para o propósito pretendido.

• Auditorias de garantia de produto. As atividades de garantia do produto asseguram a comprovação de que os produtos de software e a documentação relacionada são identificados e estão em conformidade com os contratos; e assegurar que as não conformidades sejam identificadas e abordadas[12].

[12] IEEE P730™/D8 Draft Standard for Software Quality Assurance Processes, IEEE, 2012.

2.3.1. Revisões de gerenciamento

O objetivo de uma revisão de gerenciamento é monitorar o progresso, determinar o status dos planos e cronogramas e avaliar a eficácia dos processos, ferramentas e técnicas de gerenciamento. As análises de gerenciamento comparam os resultados reais do projeto com os planos para determinar o status dos projetos ou esforços de manutenção. Os principais parâmetros das revisões gerenciais são o custo, o cronograma, o escopo e a qualidade do projeto. Revisões gerenciais avaliam decisões sobre ações corretivas, mudanças na alocação de recursos ou mudanças no escopo do projeto. Insumos para revisões de gerenciamento podem incluir relatórios de auditoria, relatórios de progresso, relatórios de validação e verificação e planos de vários tipos, incluindo gerenciamento de riscos, gerenciamento de projetos, gerenciamento de configuração de software, segurança de software e avaliação de risco, entre outros. (Consulte os capítulos 06 - Software Engineering Management e 07 - Software Configuration Management do SWEBOK para mais detalhes.)

2.3.2. Revisões Técnicas

O propósito de uma revisão técnica é avaliar um produto de software por uma equipe de pessoal qualificado para determinar sua adequação para o uso pretendido e identificar discrepâncias de especificações e padrões. Ele fornece à administração evidências para confirmar o status técnico do projeto.

Embora qualquer produto de trabalho possa ser revisado, revisões técnicas são realizadas nos principais produtos de engenharia de software de requisitos de software e design de software. Propósito, funções, atividades e, mais importante, o nível de formalidade distinguem diferentes tipos de revisões técnicas. As inspeções são as mais formais, menos detalhadas e as revisões de pares ou cheques de mesa são as menos formais. Exemplos de funções específicas incluem um tomador de decisões (ou seja, líder de software), um líder de revisão, um gravador e verificadores (membros da equipe técnica que examinam os produtos de trabalho).

As revisões também são diferenciadas pelas reuniões (face a face ou eletrônicas) que são incluídas no processo. Em alguns métodos de revisão, os verificadores examinam solitariamente os produtos de trabalho e enviam seus resultados de volta a um coordenador. Em outros métodos, os verificadores trabalham cooperativamente em reuniões.

Uma revisão técnica pode exigir que entradas obrigatórias sejam implementadas para prosseguir:

- Declaração de objetivos
- Produto de software específico
- Plano de gerenciamento de projeto específico
- Lista de problemas associados a este produto
- Procedimento de revisão técnica.

A equipe segue o procedimento de revisão documentado. A revisão técnica é concluída assim que todas as atividades listadas no exame forem concluídas. As revisões técnicas do código-fonte podem incluir uma ampla variedade de preocupações, como análise de algoritmos, utilização de recursos críticos do computador, aderência aos padrões de codificação, estrutura e organização do código para testabilidade e considerações críticas de segurança.

Observe que as revisões técnicas do código-fonte ou modelos de design, como UML, também são denominadas análise estática (consulte o tópico 3, Considerações práticas, para mais detalhes).

2.3.3. Inspeções

O objetivo de uma inspeção é detectar e identificar anomalias de produtos de software[13]. Alguns importantes diferenciadores de inspeções em comparação com outros tipos de revisões técnicas são:

Tipo de Revisão	Descrição
Regras	As inspeções são baseadas no exame de um produto de trabalho com relação a um conjunto definido de critérios especificados pela organização. Conjuntos de regras podem ser definidos para diferentes tipos de produtos de trabalho (por exemplo, regras para requisitos, descrições de arquitetura, código-fonte, etc)
Amostragem	Em vez de tentar examinar cada palavra e figura em um documento, o processo de inspeção permite que os revisores avaliem subconjuntos definidos (amostras) dos documentos em análise
Pares	Indivíduos que ocupam cargos de gerência sobre membros da equipe de inspeção não participam da inspeção. Essa é uma distinção fundamental entre revisão por pares e revisão gerencial
Moderador	Um moderador imparcial treinado em técnicas de inspeção conduz reuniões de inspeção
Reunião	O processo de inspeção inclui reuniões (presenciais ou eletrônicas) conduzidas por um moderador de acordo com um procedimento formal no qual os membros da equipe de inspeção relatam as anomalias encontradas e outras questões. As inspeções de software sempre envolvem o autor de um produto intermediário ou final. As inspeções também incluem um líder de inspeção, um gravador, um leitor e alguns (dois a cinco) verificadores (inspetores)

Os membros de uma equipe de inspeção podem possuir conhecimentos diferentes, como experiência de domínio, conhecimento de método de design de software ou experiência em linguagem de programação. As inspeções são normalmente realizadas em uma seção relativamente pequena do produto por vez (amostras). Cada membro da equipe examina o produto de software e outras entradas de revisão antes da reunião de revisão, talvez aplicando uma técnica analítica (consulte a seção 3.3.3) a uma pequena seção do produto ou a todo o produto com foco em apenas um aspecto, por exemplo, interfaces.

Durante a inspeção, o moderador conduz a sessão e verifica se todos se prepararam para a inspeção e inicia a sessão. O registrador de inspeção documenta anomalias encontradas. É estabelecido um conjunto de regras, com critérios e questões pertinentes. Esta é uma ferramenta comum usada em inspeções. A lista resultante geralmente classifica as anomalias (consulte a seção 3.2, Caracterização de defeitos) e é revisada quanto à integridade e precisão pela equipe. A decisão de saída de inspeção corresponde a uma das seguintes opções:

1. Aceitar sem retrabalho ou aceitar com retrabalho menor;
2. Aceitar com verificação de retrabalho;
3. Reinspectar.

2.3.4. Passo a passo

O objetivo de uma análise sistemática é avaliar um produto de software. Um passo a passo pode ser conduzido com a finalidade de educar um público em relação a um produto de software. São diferenciados os passos das inspeções. A principal diferença é que o autor apresenta o produto de trabalho aos outros participantes em uma reunião (face a face ou eletrônica). Ao contrário de uma inspeção, os participantes da reunião podem não ter necessariamente visto o material antes da reunião. As reuniões podem ser conduzidas de

[13] IEEE Std. 1028-2008, Software Reviews and Audits, IEEE, 2008.

maneira menos formal. O autor assume o papel de explicar e mostrar o material aos participantes e solicita feedback. Assim como as inspeções, as orientações podem ser conduzidas em qualquer tipo de produto de trabalho, incluindo plano de projeto, requisitos, design, código-fonte e relatórios de teste.

2.3.5. Auditorias de Garantia de Processo e Garantia de Produto

O objetivo de uma auditoria de software é fornecer uma avaliação independente da conformidade dos produtos e processos de software com os regulamentos, normas, diretrizes, planos e procedimentos aplicáveis. As auditorias de garantia de processo determinam a adequação dos planos, cronogramas e requisitos para atingir os objetivos do projeto[14]. A auditoria é uma atividade formalmente organizada com participantes com funções específicas - como auditor líder, outro auditor, um gravador ou um iniciador - e incluindo um representante da organização auditada. As auditorias identificam casos de não conformidade e produzem um relatório exigindo que a equipe tome ações corretivas.
Embora possa haver muitos nomes formais para revisões e auditorias, como os identificados na norma[15], o ponto importante é que eles podem ocorrer em praticamente qualquer produto em qualquer estágio do processo de desenvolvimento ou manutenção.

3. Considerações Práticas

3.1. Requisitos de qualidade de software

Este tópico está preocupado com a avaliação da qualidade e melhoria do processo de requisitos. Sua finalidade é enfatizar o papel fundamental que o processo de requisitos desempenha em termos de custo e pontualidade de um produto de software e da satisfação do cliente com ele. Isso ajudará a orientar o processo de requisitos com padrões de qualidade e modelos de melhoria de processos para software e sistemas. A qualidade e o aprimoramento do processo inclui:

- cobertura de processos de requisitos por padrões e modelos de melhoria de processos;
- medidas de processo de requisitos e benchmarking;
- planejamento e implementação de melhorias;
- segurança / melhoria da CIA / planejamento e implementação.

3.1.1. Fatores de Influência

Vários fatores influenciam o planejamento, o gerenciamento e a seleção de atividades e técnicas de Gerenciamento da Qualidade do Software, incluindo:

- o domínio do sistema no qual o software reside; as funções do sistema podem ser críticas para a segurança, de missão crítica (críticas tanto para os negócios quanto para a segurança);
- o ambiente físico no qual o sistema de software reside;
- sistema e software funcional (o que o sistema faz) e qualidade (quão bem o sistema executa suas funções) requisitos;
- os componentes comerciais (externos) ou padrão (internos) a serem usados no sistema;
- os padrões específicos de engenharia de software aplicáveis;
- os métodos e ferramentas de software a serem usados para desenvolvimento e manutenção e para avaliação e melhoria da qualidade;
- orçamento, pessoal , organização do projeto, planos e programação de todos os processos;
- os usuários previstos e o uso do sistema;
- o nível de integridade do sistema.

[14] IEEE P730™/D8 Draft Standard for Software Quality Assurance Processes, IEEE, 2012.
[15] IEEE Std. 1028-2008, Software Reviews and Audits, IEEE, 2008.

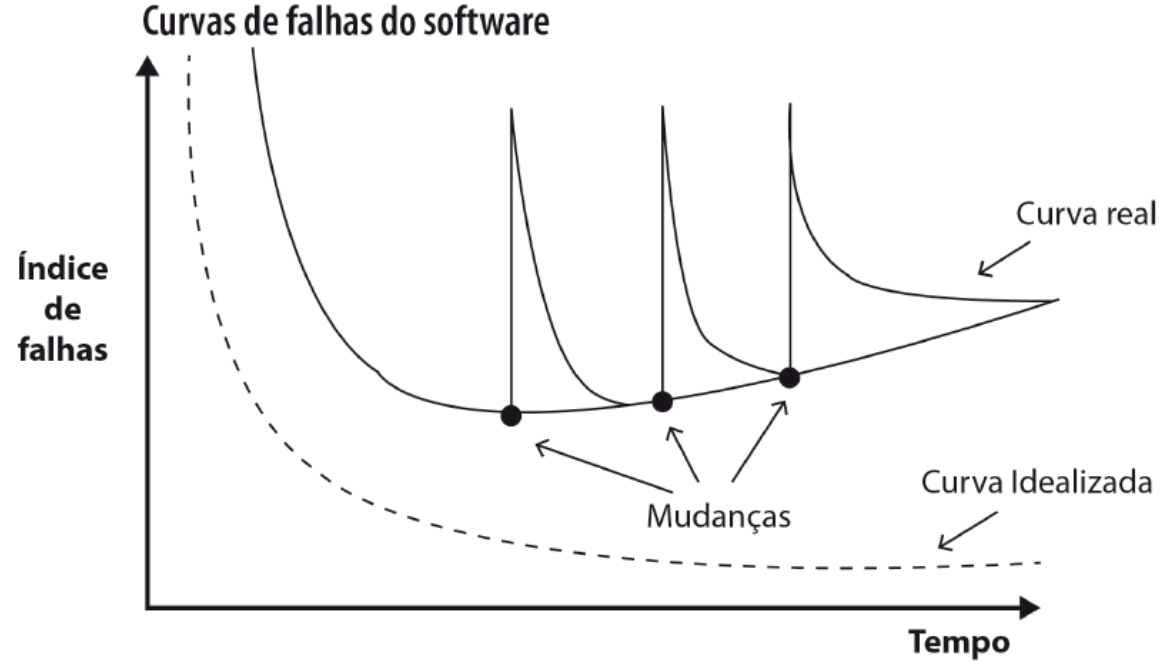
As informações sobre esses fatores influenciam como os processos de Gerenciamento da Qualidade do Software são organizados e documentados, como atividades específicas de Gerenciamento da Qualidade do Software são selecionadas, quais recursos são necessários e quais recursos impõem limites aos esforços.

3.1.2. Confiabilidade

Nos casos em que a falha do sistema pode ter conseqüências extremamente graves, a confiabilidade geral (hardware, software e recursos humanos ou operacionais) é o principal requisito de qualidade, além da funcionalidade básica, pelas seguintes razões:

• falhas no sistema afetam um grande número de pessoas;
• os usuários frequentemente rejeitam sistemas que não são confiáveis, não são seguros ou são inseguros;
• os custos de falha do sistema podem ser enormes;
• sistemas não confiáveis podem causar perda de informações.

A confiabilidade do sistema e do software inclui características como disponibilidade, confiabilidade e segurança. Ao desenvolver software confiável, ferramentas e técnicas podem ser aplicadas para reduzir o risco de injetar falhas nos produtos intermediários ou no produto de software final. Verificação, validação e testes de processos, técnicas, métodos e ferramentas identificam falhas que afetam a confiabilidade o mais cedo possível no ciclo de vida. Além disso, podem ser necessários mecanismos no software para proteger contra ataques externos e para tolerar falhas.



Fonte: Adaptado de Pressman (2006,p.15).

3.1.3. Níveis de Integridade do Software

Definir níveis de integridade é um método de gerenciamento de riscos. Níveis de integridade de software são uma gama de valores que representam a complexidade do software, criticidade, risco, nível de segurança, desempenho desejado, confiabilidade ou outras características do projeto exclusivas que definem a importância do software para o usuário e adquirente. As características usadas para determinar o nível de integridade do software variam dependendo da aplicação pretendida e do uso do sistema. O software é uma parte do sistema e seu nível de integridade deve ser determinado como parte desse sistema.

Os níveis de integridade de software atribuídos podem mudar conforme o software evolui. Recursos de design, codificação, procedimentos e tecnologia implementados no sistema ou software podem aumentar ou diminuir os níveis de integridade de software atribuídos. Os níveis

de integridade de software estabelecidos para um projeto resultam de acordos entre as autoridades de aquisição, fornecedor, desenvolvedor e de avaliação independente. Um esquema de nível de integridade de software é uma ferramenta usada para determinar os níveis de integridade de software[16], ou seja, os níveis de integridade podem ser aplicados durante o desenvolvimento para alocar esforços adicionais de verificação e validação para componentes de alta integridade[17].

3.2. Caracterização de Defeitos

As técnicas de avaliação de qualidade de software (isto é, controle de qualidade de software) encontram defeitos, faltas e falhas. A caracterização dessas técnicas leva a uma compreensão do produto, facilita correções no processo ou no produto e informa à gerência e às outras partes interessadas o status do processo ou produto.
Existem muitas taxonomias e, embora tenham sido feitas tentativas para obter consenso, a literatura indica que existem algumas em uso. A caracterização de defeitos também é usada em auditorias e revisões, com o líder de revisão frequentemente apresentando uma lista de problemas fornecidos pelos membros da equipe para consideração em uma reunião de revisão. À medida que novos métodos de design e linguagens evoluem, juntamente com os avanços nas tecnologias gerais de software, novas classes de defeitos aparecem, e um grande esforço é necessário para interpretar as classes previamente definidas.
Ao rastrear defeitos, o engenheiro de software está interessado não apenas no número de defeitos, mas também nos tipos. A informação sozinha, sem alguma classificação, pode não ser suficiente para identificar as causas subjacentes dos defeitos. Tipos específicos de problemas precisam ser agrupados para identificar tendências ao longo do tempo. O objetivo é estabelecer uma taxonomia de defeitos que seja significativa para a organização e para os engenheiros de software. As atividades de controle de qualidade de software descobrem informações em todos os estágios de desenvolvimento e manutenção de software.
Em alguns casos, a palavra defeito é sobrecarregada para se referir a diferentes tipos de anomalias. No entanto, diferentes culturas e padrões de engenharia podem usar significados um pouco diferentes para esses termos. A variedade de termos solicita que esta seção forneça um conjunto amplamente usado de definições[18]:

Termo	Descrição
Erro computacional	A diferença entre um valor ou condição calculado, observado ou medido e o valor ou condição verdadeira, especificada ou teoricamente correta
Erro	Uma ação humana que produz um resultado incorreto. Um deslize ou erro cometido por uma pessoa. Também chamado de erro humano
Defeito	Uma imperfeição ou deficiência em um software quando não atende a seus requisitos ou especificações e precisa ser corrigido
Falta	Um defeito no código-fonte. Uma etapa, processo ou definição de dados incorreta no programa de computador. A codificação de um erro humano no código-fonte. Falta é o nome formal de um bug
Falha	Um evento no qual um sistema ou componente do sistema executa uma função com falta (erro, bug). Uma falha é produzida quando uma falta é encontrada pelo processador sob condições especificadas

Usando essas definições, três medições de qualidade de software amplamente usadas são:
1) Densidade de defeitos: número de defeitos por tamanho funcional (pontos de função);

[16] IEEE P730™/D8 Draft Standard for Software Quality Assurance Processes, IEEE, 2012.
[17] J.W. Moore, The Road Map to Software Engineering: A Standards-Based Guide, Wiley-IEEE Computer Society Press, 2006.
[18] ISO/IEC/IEEE 24765:2010 Systems and Software Engineering—Vocabulary, ISO/ IEC/IEEE, 2010.

2) Densidade de falhas: número de falhas por 1K linhas de código;
3) Intensidade de falhas: número de falhas por hora de uso ou por hora de teste.
Os modelos de confiabilidade são construídos a partir de dados de falha coletados durante o teste de software ou do software em serviço e, portanto, podem ser usados para estimar a probabilidade de falhas futuras e para auxiliar nas decisões sobre quando interromper o teste.
Uma ação provável resultante das descobertas do gerenciamento da qualidade de software é remover os defeitos do produto em exame (por exemplo, localizar e corrigir bugs, criar nova compilação). Outras atividades tentam eliminar as causas dos defeitos - por exemplo, a análise de causa raiz. As atividades de análise de causa raiz incluem analisar e resumir as descobertas para encontrar as causas básicas e usar técnicas de medição para melhorar o produto e o processo, bem como rastrear os defeitos e sua remoção. A melhoria de processos é discutida principalmente no capítulo 08 - Processos de Engenharia de Software, com o processo de gerenciamento da qualidade de software sendo uma fonte de informações. Os dados sobre inadequações e defeitos encontrados pelas técnicas de controle de qualidade de software podem ser perdidos, a menos que sejam registrados. Para algumas técnicas (por exemplo, revisões técnicas, auditorias, inspeções), os gravadores estão presentes para definir essas informações, juntamente com questões e decisões. Quando ferramentas automatizadas são usadas (consulte o tópico 4, Ferramentas de qualidade de software), a saída da ferramenta pode fornecer as informações sobre defeitos. Relatórios sobre defeitos são fornecidos para o gerenciamento da organização.

3.3. Técnicas de gerenciamento de qualidade de software

As técnicas de controle de qualidade de software podem ser categorizadas de várias maneiras, mas uma abordagem direta usa apenas duas categorias:

Técnica	Descrição
Estática	Técnicas estáticas envolvem a análise de documentos e código-fonte, mas não a execução do software
Dinâmica	Técnicas dinâmicas envolvem a execução do software

3.3.1. Técnicas Estáticas

As técnicas estáticas examinam a documentação do software (incluindo requisitos, especificações de interface, design e modelos) e o código-fonte do software sem executar o código. Existem muitas ferramentas e técnicas para examinar estaticamente os produtos de trabalho de software (ver seção 2.3.2). Além disso, as ferramentas que analisam o fluxo de controle de código-fonte e a pesquisa de comentários no código são consideradas ferramentas de análise estática porque não envolvem a execução do código do software.
Outros tipos de técnicas analíticas, mais formais, são conhecidas como métodos formais. Eles são notavelmente usados para verificar os requisitos e projetos de software. Eles têm sido usados principalmente na verificação de partes cruciais de sistemas críticos, como requisitos específicos de segurança e proteção. (Veja também Métodos Formais no capítulo 09 do SWEBOK - Modelos e Métodos de Engenharia de Software)

3.3.2. Técnicas Dinâmicas

As técnicas dinâmicas envolvem a execução do código do software. Diferentes tipos de técnicas dinâmicas são realizadas durante todo o desenvolvimento e manutenção de software. Geralmente, essas são técnicas de teste, mas técnicas como simulação e análise de modelo podem ser consideradas dinâmicas (consulte Modelos e métodos de engenharia de software no capítulo 09 do SWEBOK).
A leitura de código é considerada uma técnica estática, mas engenheiros de software experientes podem executar o código à medida que o lêem. Neste caso, a leitura de código é

considerada uma técnica dinâmica porque ele vai sendo executado. Diferentes grupos podem realizar testes durante o desenvolvimento de software, incluindo grupos independentes da equipe de desenvolvimento. O capítulo 04 do SWEBOK - Software Testing é dedicado inteiramente a este assunto.

3.3.3. Testes

Dois tipos de testes podem ser abrangidos pela validação e verificação devido à sua responsabilidade pela qualidade do projeto:

- Avaliação e testes de ferramentas a serem usadas no projeto
- Testes de conformidade (ou revisão de testes de conformidade) de componentes.

Às vezes, uma organização independente (terceirizada) pode ser encarregada de realizar testes ou monitorar o processo de teste, a adequação de planos, processos e procedimentos, e adequação e precisão dos resultados. Essas empresas terceirizadas não são desenvolvedoras do software e nem estão associadas ao desenvolvimento do produto, em vez disso, são independentes, geralmente credenciadas e certificadas e sua finalidade é testar um produto quanto à conformidade com um conjunto específico de requisitos (consulte o capítulo 04 do SWEBOK - Software Testing para maiores detalhes).

3.4. Medição de Qualidade de Software

Medidas de qualidade de software são usadas para apoiar a tomada de decisões. Com a crescente sofisticação do software, as questões de qualidade vão além do fato de o software funcionar ou não até atingir metas mensuráveis de qualidade.

As decisões apoiadas pela medição de qualidade de software incluem a determinação dos níveis de qualidade de software, principalmente porque os modelos de qualidade de produto de software incluem medidas para determinar o grau em que o produto de software alcança as metas de qualidade. Questões gerenciais sobre esforço, custo e cronograma; determinar quando parar de testar e liberar um produto[19], e determinar a eficácia dos esforços de melhoria de processos.

O custo dos processos de gerenciamento da qualidade de software é um problema frequentemente levantado para decidir como um projeto ou um grupo de desenvolvimento e manutenção de software deve ser organizado. Muitas vezes, são usados modelos genéricos de custo, que são baseados em quando um defeito é encontrado e quanto esforço é necessário para corrigir o defeito em relação à localização do defeito no início do processo de desenvolvimento. Os dados de medição de qualidade de software coletados internamente podem fornecer uma melhor imagem do custo dentro desse projeto ou organização. Embora os dados de medição de qualidade de software possam ser úteis por si só (por exemplo, o número de requisitos defeituosos ou a proporção de requisitos defeituosos), técnicas matemáticas e gráficas podem ser aplicadas para auxiliar na interpretação das medidas[20]. Essas técnicas incluem:

- estatística descritiva baseada (por exemplo, análise de Pareto, gráficos de execução, gráficos de dispersão, distribuição normal)
- testes estatísticos (por exemplo, teste binomial, teste de quadra)
- análise de tendências (por exemplo, gráficos de controle; lista de leituras adicionais)
- previsão (por exemplo, modelos de confiabilidade).

Técnicas e testes baseados em estatísticas descritivas geralmente fornecem um mapa instantâneo das áreas mais problemáticas do produto de software sob exame. Os gráficos resultantes são usados para visualização, que os tomadores de decisão podem usar para concentrar recursos e conduzir melhorias de processos onde elas (melhorias) parecem ser mais necessárias.

[19] Veja seção 5.1 do capítulo 04 - Software Testing do SWEBOK
[20] Veja o capítulo 15 - Software Engineering Foundations do SWEBOK

Os resultados da análise de tendências podem indicar que um cronograma está sendo cumprido, como em testes, ou que certas classes de falhas podem se tornar mais prováveis de ocorrer, a menos que alguma ação corretiva seja tomada no desenvolvimento. As técnicas preditivas auxiliam na estimativa do esforço e cronograma de testes e na previsão de falhas[21].
A medição de qualidade de software inclui a medição de ocorrências de defeitos e a aplicação de métodos estatísticos para entender os tipos de defeitos que ocorrem com mais frequência. Essas informações podem ser usadas pelo aprimoramento do processo de software para determinar métodos para evitar, reduzir ou eliminar sua recorrência. Eles também ajudam a entender as tendências, como as técnicas de detecção e contenção estão funcionando e como os processos de desenvolvimento e manutenção estão progredindo. A partir desses métodos de medição, perfis de defeitos podem ser desenvolvidos para um domínio de aplicação específico. Então, para o próximo projeto de software dentro dessa organização, os perfis podem ser usados para guiar os processos de gerenciamento da qualidade de software - isto é, para gastar o esforço onde os problemas são mais prováveis de ocorrer.
Da mesma forma, os benchmarks, ou contagens de defeitos típicos desse domínio, podem servir como um auxílio para determinar quando o produto está pronto para entrega. A discussão sobre o uso de dados do gerenciamento da qualidade de software para melhorar os processos de desenvolvimento e manutenção aparece no capítulo 7 - gerenciamento de engenharia de software do SWEBOK.

4. Ferramentas de Qualidade de Software

Ferramentas de qualidade de software incluem ferramentas de análise estática e dinâmica. As ferramentas de análise estática inserem o código-fonte, realizam análises sintáticas e semânticas sem executar o código e apresentam os resultados aos usuários.
Há uma grande variedade na profundidade e escopo das ferramentas de análise estática que podem ser aplicadas a artefatos, incluindo modelos, além do código-fonte[22].
Categorias de ferramentas de análise estática incluem:
• Ferramentas que facilitam e automatizam parcialmente as revisões e inspeções de documentos e códigos. Essas ferramentas podem rotear o trabalho para diferentes participantes, a fim de automatizar parcialmente e controlar um processo de revisão. Eles permitem que os usuários insiram defeitos encontrados durante inspeções e revisões para remoção posterior.
• Algumas ferramentas ajudam as organizações a realizar análises de risco de segurança de software. Estas ferramentas fornecem, por exemplo, suporte automatizado para o modo de falha e análise de efeitos e análise de árvore de falhas.
• As ferramentas que suportam o rastreamento de problemas de software fornecem a entrada de anomalias descobertas durante o teste de software e posterior análise, disposição e resolução. Algumas ferramentas incluem suporte para fluxo de trabalho e para rastrear o status da resolução de problemas.
• Ferramentas que analisam dados capturados em ambientes de engenharia de software e ambientes de teste de software e produzem exibições visuais de dados quantificados na forma de gráficos e tabelas. Essas ferramentas às vezes incluem a funcionalidade de realizar análises estatísticas em conjuntos de dados (com o objetivo de discernir tendências e fazer previsões).
Algumas dessas ferramentas fornecem taxas de injeção de defeitos e remoção; densidades de defeitos; rendimentos; distribuição de injeção e remoção de defeitos para cada uma das fases do ciclo de vida.

[21] Veja o capítulo 7 - Software Engineering Management do SWEBOK
[22] Consulte os capítulos 03 - construção de software, 04 - teste de software e 05 - manutenção de software do SWEBOK para descrições de utilização das ferramentas de análise dinâmica

5. Bibliografia

- P.B. Crosby, Quality Is Free, McGraw-Hill,1979.
- W. Humphrey, Managing the Software Process, Addison Wesley, 1989.
- S.H. Kan, Metrics and Models in Software Quality Engineering, 2nd ed., Addison Wesley, 2002.
- ISO/IEC 25010:2011 Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE)—Systems and Software Quality Models, ISO/IEC, 2011.
- IEEE P730™/D8 Draft Standard for Software Quality Assurance Processes, IEEE, 2012.
- F. Bott et al., Professional Issues in Software Engineering, 3rd ed., Taylor & Francis, 2000.
- D. Galin, Software Quality Assurance: From Theory to Implementation, Pearson Education Limited, 2004.
- S. Naik and P. Tripathy, Software Testing and Quality Assurance: Theory and Practice, Wiley-Spektrum, 2008.
- P. Clements et al., Documenting Software Architectures: Views and Beyond, 2nd ed., Pearson Education, 2010.
- G. Voland, Engineering by Design, 2nd ed., Prentice Hall, 2003.
- RTCA DO-178C, Software Considerations in Airborne Systems and Equipment Certification, Radio Technical Commission for Aeronautics, 2011.
- IEEE Std. 15026.1-2011 Trial-Use Standard Adoption of ISO/IEC TR 15026-1:2010
- Systems and Software Engineering—Systems and Software Assurance—Part 1: Concepts and Vocabulary, IEEE, 2011.
- IEEE Std. 12207-2008 (a.k.a. ISO/IEC 12207:2008) Standard for Systems and Software Engineering—Software Life Cycle Processes, IEEE, 2008.
- ISO 9000:2005 Quality Management Systems—Fundamentals and Vocabulary, ISO, 2005.
- IEEE Std. 1012-2012 Standard for System and Software Verification and Validation, IEEE, 2012.
- IEEE Std. 1028-2008, Software Reviews and Audits, IEEE, 2008.
- J.W. Moore, The Road Map to Software Engineering: A Standards-Based Guide, Wiley-IEEE Computer Society Press, 2006.
- K.E. Wiegers, Software Requirements, 2nd ed., Microsoft Press, 2003.
- ISO/IEC/IEEE 24765:2010 Systems and Software Engineering Vocabulary, ISO/IEC/IEEE, 2010.
- N. Leveson, Safeware: System Safety and Computers, Addison-Wesley Professional, 1995.
- T. Gilb, Principles of Software Engineering Management, Addison Wesley Professional, 1988.
- T. Gilb and D. Graham, Software Inspection, Addison Wesley Professional, 1993.
- K. Wiegers, Peer Reviews in Software: A Practical Guide, Addison-Wesley Professional, 2001.
- N.R. Tague, The Quality Toolbox, 2nd ed. ASQ Quality Press, 2010.