UNIP
UNIVERSIDADE PAULISTA
Interativa

Engenharia de *Software* II

Autor: Prof. André Luiz Dias Ribeiro
Colaboradores: Prof. Luciano Soares de Souza
Profa. Adriane Paulieli Colossetti
Profa. Jorge Rodolfo Beingolea Garay

Professor conteudista: André Luiz Dias Ribeiro

Mestre em Engenharia Elétrica pela Escola Politécnica da USP em 2006, certificado em Project Management Professional (PMP), Personal Professional Coaching (PPC) e profissional da área de Tecnologia da Informação há 25 anos, exercendo diversos níveis profissionais da área, desde programador de computadores a diretor da área de informática.

Há dez anos atua como professor dos cursos de graduação em Sistemas de Informação, Ciência da Computação e Engenharia de Produção e em cursos de pós-graduação em Gerenciamento de Projetos e Engenharia de *Software*. Responsável pela coordenação interdisciplinar de Engenharia de *Software* nos cursos de graduação e pela orientação de trabalhos de conclusão dos cursos de graduação e pós-graduação.

Dados Internacionais de Catalogação na Publicação (CIP)

R484t Ribeiro, André Luiz.

Engenharia de Software II. / André Luiz Ribeiro. – São Paulo: Editora Sol, 2015.
188 p., il.

Nota: este volume está publicado nos Cadernos de Estudos e Pesquisas da UNIP, Série Didática, ano XVII, n. 2-025/15, ISSN 1517-9230.

1. Qualidade de software. 2. Processos de manutenção. 3. Gestão da configuração. I. Título.

CDU 681.3.02

U500.53 – 19

Sumário

Engenharia de *Software* II

Unidade IV

APRESENTAÇÃO

O objetivo da disciplina *Engenharia de Software II* é apresentar a você características importantes no desenvolvimento de um *software* que, normalmente, são esquecidas e/ou abandonadas pela incontrolável vontade de programar inerente aos profissionais ligados á área de Tecnologia da Informação. Nessa disciplina são descritos os requisitos de qualidade indispensáveis a um bom *software*, como devem estar inseridos no ciclo de desenvolvimento, a importância das atividades de testes, como controlar os diversos artefatos e o código do *software* e suas consequências na manutenção de *software* produzido.

O livro-texto está dividido em quatro unidades, em que são apresentados os principais conceitos e modelos de qualidade para o desenvolvimento de *software*, as principais técnicas para garantir e controlar a qualidade, as características do processo de manutenção do *software* e como controlar tudo o que é produzido durante a construção de um sistema.

Na unidade I, são apresentados os principais conceitos sobre qualidade de *software*; o Sistema de Gestão da Qualidade (SGQ) baseado na norma ISO9000, pioneiro na questão de qualidade; e as principais normas para avaliação da qualidade do produto de *software*.

Na unidade II, são apresentadas as principais normas de padronização para avaliação da qualidade e os modelos de qualidade reconhecidos e praticados pelo mercado de Tecnologia da Informação para melhoria dos processos de desenvolvimento.

Na unidade III, são descritas a principais técnicas e ferramentas de verificação e validação que podem ser utilizadas pelas equipes durante o desenvolvimento do *software* para garantir e controlar a qualidade daquilo que é produzido e o processo de testes de *software*.

Na unidade IV, são discutidos os impactos de tempo e custo na manutenção de *softwares* desenvolvidos com e sem qualidade, mostrando claramente as vantagens de construir sistemas fundamentados em qualidade. Também são apresentadas técnicas para gerenciar e controlar os artefatos e códigos produzidos durante o ciclo de vida, visando garantir a integridade das informações por meio da gestão da configuração.

O objetivo é mostrar que é possível produzir um *software* com qualidade, em detrimento do processo de "fazer para entregar no prazo", tornando o desenvolvimento claro, com o menor retrabalho possível e a garantia da satisfação da equipe e do cliente com o produto final.

Divirtam-se!

INTRODUÇÃO

Quando se fala em qualidade de *software*, a primeira ideia que vem à cabeça é: "Que coisa chata", ou "Isso não adianta nada". Em parte, isso é verdade, pois produzir qualquer coisa com qualidade demanda mais cuidado, mais esmero, mais atenção e mais disciplina, logo, não é uma atividade corriqueira que traz prazer ao ser executada. O prazer vem ao final, quando o cliente recebe o produto de *software*

com poucos erros, o programador faz poucas alterações no código que produziu e alguém, ao efetuar a manutenção desse *software*, diz: "Nossa! Como está fácil fazer essa alteração!". Enfim, nem sempre você poderá presenciar esses resultados, mas saberá que fez o seu melhor, e isso é o mais importante.

A partir desse cenário, ao conhecer as características que traduzem um *software* de qualidade, saber quais são os processos e modelos de qualidade mais utilizados e praticados no mercado de trabalho torna você um profissional diferenciado, uma vez que a maioria dos profissionais está apenas preocupada com a programação dos computadores e se esquece de que esta é somente a atividade-fim de um processo de construção.

Dominando esses requisitos básicos **do que é** produzir um *software* de qualidade, passamos a explanar e conhecer as técnicas e ferramentas que permitem definir o **como fazer**, que irá direcionar as atividades de qualidade durante todo o processo de desenvolvimento, desde o levantamento de requisitos até a programação, para que os resultados sejam avaliados constantemente durante o ciclo de vida, e não apenas ao seu final, como é feito na maioria das empresas.

Como resultado dessa preocupação com a qualidade do que é produzido, as manutenções preventivas e evolutivas do sistema tornam-se mais fáceis e rápidas, gerando satisfação a todos os envolvidos no trabalho.

Para dar suporte a esse processo de construção de um *software* de qualidade e com facilidade de manutenção, faz-se necessário um processo bem-definido para controlar todos os artefatos e códigos produzidos, a fim de que não se percam versões, documentos e outros elementos essenciais para manter a integridade e a confiança no produto e permitir que apenas pessoas autorizadas façam modificações no *software* final.

Enfim, dominando esses assuntos e aplicando as técnicas apresentadas, você terá um grande diferencial em relação aos outros profissionais: saberá produzir *software* com qualidade.

Unidade I

1 CONCEITOS SOBRE QUALIDADE DE *SOFTWARE*

Na Engenharia de *Software*, a área de qualidade tem como objetivo garantir que, ao final, o *software* esteja de acordo com as características definidas pelos usuários no início e no decorrer do processo de desenvolvimento da aplicação.

Segundo a norma NBR ISO (2000a), qualidade de *software* é definida como um conjunto de características que devem ser alcançadas em um determinado grau para que o produto atenda às necessidades de seus usuários. A totalidade de características de uma entidade que lhe confere a capacidade de satisfazer a necessidades explícitas e implícitas.

Há um consenso entre os diversos autores da área de qualidade de que o objetivo principal da qualidade é proporcionar a satisfação dos clientes por meio do atendimento das necessidades especificadas e também dos requisitos implícitos do *software*.

Conforme ilustrado na Figura 1, para Crosby (1990), existem cinco princípios básicos da qualidade que, se seguidos, produzirão melhores resultados:

1) **Fazer certo da primeira vez economiza tempo e dinheiro.**
   Ao se preocupar em produzir com qualidade desde o primeiro momento, as atividades de correção de erros diminuirão e, por consequência, haverá redução dos custos e cumprimento dos prazos estabelecidos.

2) **Qualidade é um processo preventivo.**
   A qualidade deve ser aplicada desde o primeiro momento, e não só após o produto estar pronto.

3) **Qualidade é incorporada ao produto como resultado da atenção dedicada às necessidades dos clientes.**
   Logo no início do processo de desenvolvimento, devem-se identificar e definir os padrões de qualidade esperados pelos clientes, a fim de construir o *software* alinhado a essa expectativa.

4) **Qualidade é responsabilidade de todos os envolvidos.**
   Não basta que a gerência esteja preocupada com a qualidade. Cada membro da equipe deve ter a consciência de que deve fazer o melhor possível sempre, bem como assumir a responsabilidade por isso.

5) **Qualidade é um processo de melhoria contínua.**
Em todos os processos, sempre há o que pode ser melhorado. A qualidade não foge à regra. Cada vez que produzimos algo, aprendemos e aperfeiçoamos, sempre em busca de fazer melhor da próxima vez.

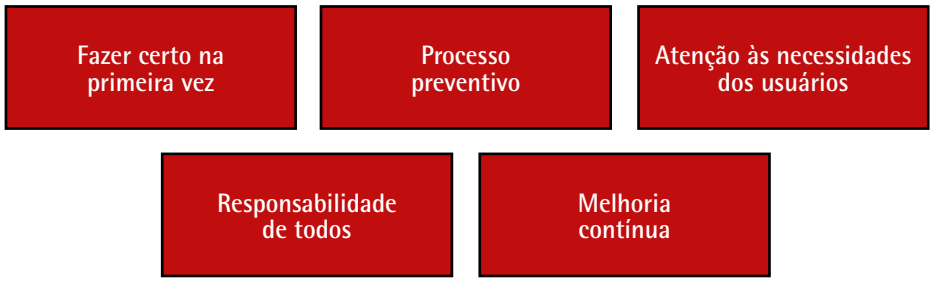

Figura 1 – Princípios básicos da qualidade

Observação

Envolvidos no projeto são todos aqueles que participam do projeto de *software* direta ou indiretamente. Esses envolvidos também são chamados de interessados ou *stakeholders*.

1.1 Benefícios da qualidade

Embora existam várias iniciativas sobre a qualidade de *software*, muitas empresas de Tecnologia da Informação ainda permanecem na situação denominada caos, significando que o *software* é produzido com base em pessoas, e não em processos. A partir da conscientização de todos de que a qualidade pode transformar o cenário atual, processos e métodos são introduzidos gradativamente para alcançar o grau de organização necessário para que a empresa usufrua dos benefícios da qualidade, conforme ilustrado na Figura 2.

Observação

OS envolvidos devem ser identificados logo nas fases iniciais do projeto. Os principais são: o patrocinador, os clientes, os usuários finais e os fornecedores.

Alguns benefícios podem ser observados como resultado direto da produção de um *software* com qualidade:

- aumento da produtividade;
- redução de defeitos no produto;
- aumento da confiabilidade do produto;

- menos retrabalho;
- menos horas extras de trabalho;
- maior satisfação dos clientes.

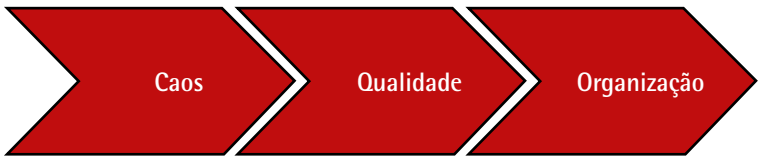

Figura 2 – Evolução com a conscientização sobre a qualidade

1.2 Obstáculos da qualidade

Fazer *software* com qualidade, porém, não é uma tarefa fácil. Há sempre um conjunto de fatores internos e externos que são opostos às boas práticas e que acabam por criar dificuldades à implementação do processo de qualidade em uma empresa e até mesmo a ações individuais de melhoria. Alguns desses fatores são descritos na Figura 3.

Figura 3 – Obstáculos à qualidade de *software*

Lembrete

A cultura da organização é um dos principais obstáculos à qualidade. O conceito "Se está dando certo, por que mudar?" contribui para uma resistência ainda maior da equipe de desenvolvimento.

As aplicações de *software* estão a cada dia mais complexas para construir, testar e navegar, em virtude da evolução de tecnologias como *tablets*, *smartphones*, dentre outros, bem como das interfaces para a internet que são cada vez mais interativas com o usuário, o que aumenta o grau de dificuldades das aplicações.

Custos e prazos maldefinidos são uma constante em virtude da pressão, do mercado e da própria concorrência que os clientes vêm sofrendo para o lançamento de novos produtos. Esses dois fatores levam as equipes a abandonarem os processos de qualidade para atender ao tempo e aos custos do projeto que são prioridades para a direção da empresa.

Por último, um novo produto de *software* é iniciado sem o correto esclarecimento a todos os envolvidos, e, muitas vezes, estes não são nem notificados sobre esses produtos, o que causa surpresas e problemas no atendimento às necessidades desses envolvidos.

1.3 Visões da qualidade

Como mencionado nos obstáculos à qualidade, entender a forma pela qual cada envolvido percebe a qualidade de um produto de *software* é muito importante, pois há interesses que não convergem e causam uma série de conflitos durante o desenvolvimento do *software*, seja por questões de dinheiro e prazo, seja para ver o *software* funcionando corretamente.

As visões no desenvolvimento de um *software* envolvem os gerentes, os desenvolvedores, os clientes e os usuários, e esses interesses são ilustrados na Figura 4.

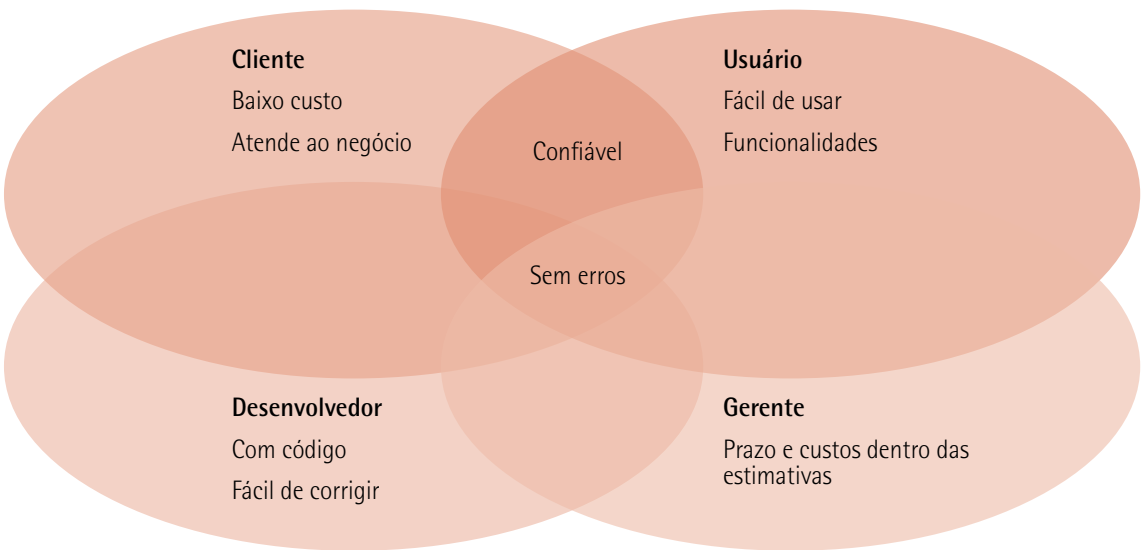

Figura 4 – Visões da qualidade de *software*

1.4 Importância da qualidade

A importância de se produzir *software* com qualidade é inegável, mas na área de Tecnologia da Informação esse aspecto não vem recebendo a atenção devida. Exceção feita aos *softwares* que podem causar a perda de vidas humanas, que recebem a atenção e a preocupação que deve ser uma constante na cabeça dos desenvolvedores.

Hoje, os sistemas computacionais são a base de controle de todas as empresas. A dependência da tecnologia é visível a todos: usuários e empresas. Imagine o que o mau funcionamento de um *software* pode causar:

- bancos perdem milhões;
- clientes veem saldos sumirem de suas contas de repente;
- os telefones param de funcionar;
- aviões têm rotas desviadas;
- trens de metrô são colocados no mesmo trilho.

Imagine o caos nas empresas. Contudo, isto não são previsões. Já está ocorrendo em diversas situações do mundo atual e que realmente fazem refletir sobre qualidade na produção de *software*, independentemente do fato de causar perda de vidas humanas, ou não. Veja alguns exemplos de *bugs* de *software* famosos:

- **Derrubada do Airbus A320, 1988**
  - Ocorrência: o navio da Marinha americana USS Vincennes derrubou um Airbus A320, que foi confundido com um F-14. Morreram 290 pessoas.
  - Motivo: o *software* do radar não diferenciou um avião de passageiros de um caça inimigo de ataque.
- **Pentium Floating-Point, 1994**
  - Ocorrência: em 1994 a Intel apresentou ao mercado a primeira versão do processador Pentium, o qual possuía uma unidade de processamento de ponto flutuante bastante rápida, mas que fazia cálculos errados. Exemplo: 4,195,835 – (4,195,835/3,135,727) * 3,135,727 era igual a zero no processador 486, mas igual a 256 no Pentium.
  - Motivo: o *bug* foi causado por um erro na tabela que era utilizada para acelerar o algoritmo de processamento de multiplicação de ponto flutuante. A Intel afirmou que um usuário típico do Pentium somente perceberia o problema uma vez em 27 mil anos. Ao final, a Intel foi forçada a vir a público e anunciar um programa de troca de *chips*, o qual teve um custo de cerca de US$ 475 milhões.
- ***Recall* de veículos, 2003**
  - Ocorrência: uma fabricante de caminhões e tratores americana anunciou um grande programa de *recall* dos seus veículos no inverno de 2003.

– Motivo: a fabricante de veículos detectou pelo seu controle de qualidade e informou a existência de um problema de *software* que poderia causar instabilidade nos veículos em determinadas circunstâncias.

Saiba mais

Para ver mais exemplos de *bugs* famosos de *software*, visite o *site*: <http://www.softwareqatest.com/>. Acesso em: 9 dez. 2014.

No processo de qualidade existem dois conceitos que sempre causam confusão em relação à sua definição e à sua finalidade. São estes: a garantia da qualidade e o controle da qualidade. Vejamos a diferença.

1.5 Garantia da qualidade

São ações planejadas e sistemáticas de qualidade realizadas durante o processo de desenvolvimento cujo objetivo é atuar de forma preventiva para se atingir a qualidade do produto de *software*. A garantia da qualidade avalia se as características do produto estão de acordo com os padrões estabelecidos e se as atividades estão ocorrendo conforme o planejado. Algumas atividades de garantia da qualidade devem envolver:

- uso, pelos desenvolvedores, de métodos e ferramentas que ajudem a conseguir especificações, projeto e codificação de maior qualidade;
- estabelecimento de padrões para documentos, código e estilo de codificação (como usar linguagem de programação); esses padrões podem ser determinados pelo cliente, por normas internacionais ou pela empresa de desenvolvimento;
- realização das atividades de verificação e validação, como revisões, inspeções, dentre outras.

Observação

Segundo a NBR ISO 9000:2005 (Sistemas de Gestão da Qualidade – Fundamentos e Vocabulário), não conformidade é o não atendimento a um requisito de qualidade (necessidade ou expectativa implícita ou obrigatória).

1.6 Controle da qualidade

São atividades de qualidade realizadas após o produto de *software* estar pronto. O objetivo do controle da qualidade é permitir o aceite do produto, uma vez que se caracteriza por um "selo" atestando que a aplicação está de acordo com as especificações. Por meio do controle da qualidade, evita-se que

produtos defeituosos sejam entregues aos clientes. A principal atividade de controle da qualidade são os testes funcionais do *software*.

No controle da qualidade, uma atividade deve ser executada visando avaliar se as ações de qualidade planejadas estão sendo executadas de acordo com o processo estabelecido. Essa atividade chama-se auditoria.

Tais auditorias podem geram ações corretivas, no caso de encontrar não conformidades, e podem ser classificadas em três tipos:

- auditorias de produto: foco em verificar a conformidade de produtos com os padrões estabelecidos;
- auditorias de processo: verificam se as ações de qualidade planejadas estão sendo executadas;
- auditorias de sistemas de qualidade: avaliam a eficácia da implementação desse sistema e determinam o grau em que os objetivos do sistema estão sendo atingidos.

Lembrete

A garantia da qualidade é feita durante o desenvolvimento do *software*, e o controle da qualidade é realizado após o produto estar pronto.

Alinhados os conceitos iniciais de qualidade de *software*, veremos no próximo tópico os sistemas de gestão da qualidade que orientam e definem os procedimentos de qualidade nas diversas organizações.

Saiba mais

Informações detalhadas sobre qualidade de *software* podem ser obtidas no livro:

GUERRA, A. C.; COLOMBO, R. M. *Tecnologia da Informação*: qualidade de produto de software. Brasília: PBQP Software, 2009.

1.7 Sistemas de Gestão da Qualidade (SGQ)

Um SGQ tem como objetivo padronizar os processos de uma empresa para a criação de seu produto final, proporcionando a satisfação de seus clientes e a melhoria contínua dos seus processos (ANTONIONI, 1995).

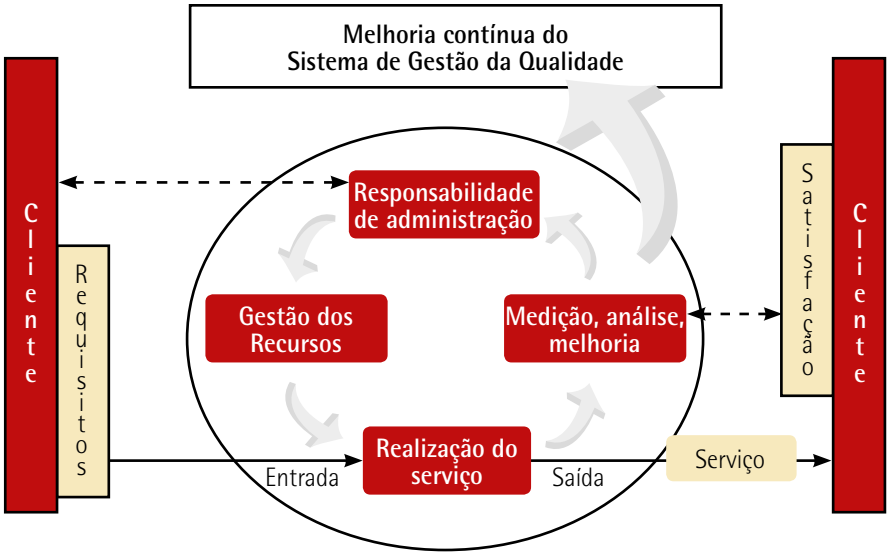

Figura 5 – Melhoria contínua do SGQ (ISO)

ABNT (2000b, p. 2)

Conforme ilustrado na Figura 5, o SGQ se inicia com os requisitos do cliente, a partir dos quais o produto é construído, passando pelo processo de medição e análise dos padrões definidos para a garantia da qualidade. Nesse momento, são tomadas as ações corretivas para melhoria do processo. Em todo o processo, a alta administração e a gerência são responsáveis pelo patrocínio do SGQ, pela gestão dos recursos necessários e pela comunicação com o cliente.

O objetivo final do SGQ é proporcionar a satisfação do cliente.

1.7.1 Fatores para implantação de um SGQ

Conforme a Figura 6, diversos fatores podem motivar uma empresa a implantar um SGQ. Dentre estes, podem-se destacar:

- Conscientização da alta administração: a alta direção reconhece que a qualidade é um diferencial e patrocina o processo. É o fator mais eficaz.

- Razões contratuais: a fim de manter ou iniciar o fornecimento de produtos ou serviços para outros países, para o governo e para algumas organizações da iniciativa privada. O tempo para alcançar a maturidade é maior, mas normalmente se alcança a conscientização.

- Competitividade: necessária para manter a empresa concorrendo no mercado.

- Modismo: fazer porque estão todos fazendo. Bastante ineficaz, normalmente, não atinge a conscientização da alta administração e, com isso, o processo é abandonado no meio do caminho.

Figura 6 – Fatores motivacionais para a implantação do SGQ

1.7.2 A NBR ISO 9000 – norma-padrão

O SGQ mais comum e conhecido no mercado é a NBR ISO 9000 – Normas de Gestão da Qualidade e Garantia da Qualidade –, lançada no fim da década de 1980. Serve como arcabouço-padrão para definir como as demais normas específicas devem ser utilizadas.

A NBR ISO 9000 auxilia a empresa na seleção da norma mais apropriada para o seu negócio e na sua utilização. Trata-se de um documento não normativo (ABNT, 2000a).

Observação

A *International Organization for Standardization* (ISO) é um órgão da ONU e tem o objetivo de fixar normas técnicas essenciais, de âmbito internacional.

NBR ISO 9000 é o nome genérico que se dá às diversas normas que abrangem o assunto. As normas básicas para garantia da qualidade são as contratuais – ISO 9001, ISO 9002 e ISO 9003 – e as organizações só podem ser certificadas em relação a essas normas.

Observação

O representante brasileiro da ISO no Brasil é a Associação Brasileira de Normas Técnicas (ABNT), e as normas são descritas como Normas Brasileiras (NBRs).

1.7.2.1 Descrição básica das normas

Para o estudo relacionado à qualidade de *software*, são relevantes as seguintes normas relacionadas à ISO 9000 e apresentadas na Figura 7:

- ISO 9000 – Normas de Gestão da Qualidade e Garantia da Qualidade: um guia de como as demais normas devem ser usadas.

- ISO 9001 – Sistemas da Qualidade – Modelo para Garantia da Qualidade em Projeto, Desenvolvimento, Produção, Instalação e Assistência Técnica: para uso quando a conformidade com os requisitos especificados tiver de ser garantida pelo fornecedor desde o projeto até a manutenção.

- ISO 9000-1 – Normas de Gestão da Qualidade e Garantia da Qualidade – Parte 1: estabelece os princípios gerenciais que permeiam toda a série de normas.

- ISO 9000-3 – Normas de Gestão da Qualidade e Garantia da Qualidade – Parte 3: esta norma define diretrizes para facilitar a aplicação da norma ISO 9001 a organizações que desenvolvem, fornecem e mantêm *software*. Destina-se a fornecer orientação quando um contrato entre duas partes exigir a demonstração da capacidade do fornecedor em desenvolver, fornecer e realizar manutenção em produtos de *software*.

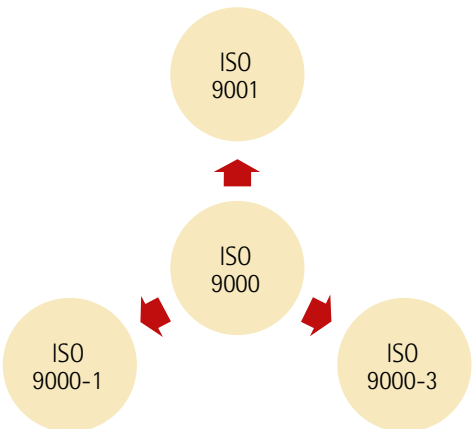

Figura 7 – Normas ISO relacionadas a organizações que produzem *software*

1.7.3 NBR ISO 9000-3 – Norma para empresas de desenvolvimento de *software*

Em junho de 1993 foi criado o guia ISO 9000-3, com diretrizes para a aplicação da ISO 9001 ao desenvolvimento, ao fornecimento e à manutenção de *software*.

Para cada item da NBR ISO 9001 existe um correspondente na NBR ISO 9000-3 que a detalha e a adequa às empresas de *software*.

Lembrete

A NBR ISO 9000-3 é a norma da qualidade do SGQ aplicada às empresas de desenvolvimento de *software*.

A NBR ISO 9000-3 abrange questões relacionadas ao entendimento dos requisitos funcionais, ao uso de metodologias consistentes para desenvolvimento de *software* e ao gerenciamento do projeto desde a concepção até a manutenção. Uma das principais limitações da NBR ISO 9000-3 é não abordar os aspectos relacionados à melhoria contínua do processo de *software* que são tratados pelo modelo *Capability Maturity Model Integration* (CMMI) e pela NBR ISO IEC 15504 – Spice (*Software Process Improvement and Capability Determination* – Melhoria do Processo de *Software* e Determinação da Capacidade). Portanto, o objetivo da NBR ISO 9000-3 é trazer quais processos a organização deve ter e manter para o desenvolvimento do *software* (RAPCHAN, 2005).

Observação

O Modelo CMMI e a norma ISO/IEC 15504 serão detalhados no tópico 4.

1.7.3.1 A estrutura da NBR ISO 9000-3

A NBR ISO 9001 baseia-se em vinte diretrizes que englobam vários aspectos da garantia da qualidade. A NBR ISO 9000-3 exige que 18 critérios estejam presentes no sistema da qualidade e agrupa essas diretrizes em três partes principais, que são apresentadas na Figura 8:

- estrutura: aspectos organizacionais, relacionados ao SGQ;
- ciclo de vida: descreve as atividades de desenvolvimento de *software*;
- suporte: descreve operações que apoiam as atividades do ciclo de vida.

1.7.3.2 O processo de certificação da NBR ISO série 9000

A certificação ISO 9000 é reconhecida no mercado por muitas organizações e traz uma série de benefícios às empresas que adotam o sistema. Dentre estes, podemos destacar:

- abertura de novos mercados;
- maior conformidade e atendimento às exigências dos clientes;
- maior integração entre os setores da organização;
- melhores condições para acompanhar e controlar os processos;
- diminuição dos custos de desenvolvimento.

O primeiro passo para realizar o processo de certificação é selecionar, do modelo ISO 9000, a norma mais adequada aos propósitos da organização (ISO 9001, ISO 9002 ou ISO 9003). Para empresas de desenvolvimento de *software*, é a norma NBR ISO 9000-1.

Após a escolha, a empresa deverá analisar o seu processo e treinar os funcionários para a conscientização sobre a necessidade da qualidade. Na sequência, deverá desenvolver e implementar os procedimentos necessários ao sistema da qualidade, selecionar um órgão certificador que irá avaliar se o sistema está de acordo com a norma, fazer pré-auditorias e realizar a auditoria final de aderência ao SGQ para a certificação.

Durante a definição do processo, deve-se tomar cuidado para que este não se torne burocrático. Ele deve adaptar-se ao dia a dia da empresa, e não o contrário. Além disso, a organização será reavaliada a cada dois anos e deverá cuidar para não perder o certificado, pois isso pode ser muito custoso em termos financeiros e de imagem no mercado.

Figura 8 – Melhoria contínua do SGQ

O processo de certificação pode se estender de alguns meses a dois anos, dependendo do nível de maturidade da qualidade em que a organização se encontra.

1.7.3.3 Custo da certificação da NBR ISO série 9000

Pesquisa realizada pelo National ISO 9000 Support Group nos Estados Unidos e pelo seu correlato no Brasil (ABNT, 2000a) avaliou que os valores para obter a certificação ficam numa faixa de US$ 6.500,00 a US$ 30.000,00, dependendo do grau inicial em que a empresa se encontra.

Saiba mais

Mais detalhes sobre as NBRs ISO da série 9000 podem ser obtidas nos *sites* oficiais: <www.iso.org> ou <www.abnt.org.br>.

2 GESTÃO DA QUALIDADE DO PRODUTO DE *SOFTWARE*

A partir da padronização de processos por meio das normas ISO, foram criadas normas específicas para o desenvolvimento de *software*, com o objetivo de garantir a aderência dos produtos de *software* a padrões preestabelecidos, e outras para avaliação desses produtos.

Embora existam diversas normas relacionadas a esse assunto, aqui são selecionadas as normas mais utilizadas pelo mercado de Tecnologia da Informação, mais o Modelo de McCall, que foi o primeiro a abordar os assuntos relacionados à qualidade de um produto de *software*.

Essas normas principais são: ISO/IEC 9126, ISO/IEC 12207, ISO/IEC 14598 e ISO/IEC 25000.

Apresentamos no Quadro 1 uma relação das normas que são abordadas neste tópico.

Quadro 1 – Normas e modelo de qualidade para produtos de *software*

Norma	Objetivo
Modelo de McCall	Define fatores e critérios de qualidade para o produto de *software*
ISO/IEC 9126 ou NBR 13596	Define as características da qualidade que devem estar presentes em um produto de *software*
ISO/IEC 12207	Define as tarefas para o ciclo de vida do *software*
ISO/IEC 14598	Estabelece um plano para avaliação do produto de *software*
ISO/IEC 25000	Define uma especificação e uma avaliação da qualidade do produto de *software* É a nova geração das séries ISO/IEC 9126 e ISO/IEC14598

2.1 Modelo de McCall

Em 1977, Jim McCall desenvolveu um modelo de qualidade para o Departamento de Defesa Americano em que a qualidade é definida por um conjunto de características internas e externas de um *software*, tornando-se o primeiro modelo de qualidade a ser amplamente divulgado e utilizado.

Essas características estão divididas em fatores de qualidade que definem os atributos externos e que só podem ser medidos indiretamente e critérios de qualidade que são os atributos internos e que podem ser medidos diretamente. A combinação de ambos permite chegar a um fator quantitativo de qualidade que um *software* apresenta.

Observação

McCall definiu 11 fatores de qualidade e 23 critérios que, inter-relacionados, permitem a avaliação da qualidade de um produto de *software*.

O Modelo de McCall define claramente quais são esses fatores de qualidade que podem ser avaliados e os dividiu em três visões, conforme ilustrado na Figura 9.

Essas três visões são: a visão de revisão, que avalia a capacidade de sofrer manutenção; a visão de operação, a qual verifica as condições de utilização do *software*; e a visão de transição, cujo foco está na avaliação da capacidade do *software* de adaptar-se a outros ambientes.

A seguir são descritos os conceitos fundamentais dos 11 fatores do modelo (McCALL; RICHARDS; WALTERS, 1977):

Figura 9 – Fatores de qualidade de McCall

2.1.1 Visão de operação

- Correção: quanto o sistema satisfaz a sua especificação e cumpre os objetivos esperados pelo cliente. Faz-se a pergunta: "O *software* faz o que queremos?".
- Confiabilidade: quanto o sistema executa sua função com a precisão exigida. Faz-se a pergunta: "O *software* é estável?".
- Eficiência: quanto o sistema usa os recursos computacionais. Faz-se a pergunta: "O *software* usa o *hardware* de maneira adequada?".
- Integridade: quanto o acesso ao *software* ou aos dados por pessoas não autorizadas pode ser controlado. Faz-se a pergunta: "O *software* é seguro?".
- Usabilidade: quanto de esforço é necessário para aprender e usar o sistema. Faz-se a pergunta: "O *software* é fácil de usar?".

2.1.2 Visão de revisão

- Manutenibilidade: quanto de esforço é necessário para localizar e eliminar erros no sistema. Faz-se a pergunta: "O *software* é fácil de corrigir?".

- Flexibilidade: quanto de esforço é necessário para modificar um programa. Faz-se a pergunta: "É fácil incluir novas funcionalidades?".

- Testabilidade: quanto de esforço é necessário para testar um programa a fim de garantir que ele atenda às necessidades. Faz-se a pergunta: "É fácil testar o sistema?".

2.1.3 Visão de transição

- Portabilidade: quanto de esforço é necessário para transferir um sistema de uma plataforma de *hardware* ou *software* para outra. Faz-se a pergunta: "O *software* pode ser usado em outra máquina?".

- Reusabilidade: quanto de um sistema pode ser reutilizado por outros sistemas. Faz-se a pergunta: "Alguma parte pode ser reutilizada?".

- Interoperabilidade: quanto de esforço é necessário para estabelecer conectividade entre o sistema e os outros sistemas. Faz-se a pergunta: "O *software* está apto para fazer interface com outros sistemas?".

McCall, Richards e Walters (1977) também definiram 23 critérios. Atribuindo peso a cada um deles e combinando-os com os fatores de qualidade, chegaram a uma classificação quantitativa para a verificação da aderência do *software* aos padrões de qualidade esperados.

A atribuição de pesos aos fatores e critérios de qualidade decorre do fato de que cada tipo de *software* possui suas próprias características e seus próprios requisitos de qualidade. Portanto, a importância de cada fator e cada critério de qualidade varia conforme o tipo do *software*.

Por exemplo, para um *software* de locação de *games*, os fatores de integridade, confiabilidade, usabilidade e facilidade de manutenção são essenciais. Porém, para um *software* embarcado para um automóvel, a eficiência, a correção e a confiabilidade são os fatores mais importantes. Nesses dois cenários, cada fator receberá seu peso equivalente à importância estabelecida.

Cada critério de qualidade está associado a um ou mais fatores de qualidade que permitem a correta avaliação de cada fator para cada tipo de *software*, conforme ilustrado no Quadro 2.

Quadro 2 – Correlação entre fatores e critérios de qualidade de McCall

	Correção	Confiabilidade	Eficiência	Integridade	Usabilidade	Manutenibilidade	Testabilidade	Flexibilidade	Portabilidade	Reusabilidade	Interoperabilidade
Acesso a auditoria				X							
Controle de acesso				X							
Acurácia		X									
Conectividade											X
Completude	X										
Comunicabilidade					X						
Clareza e concisão						X					
Consistência	X	X				X					
Compartilhamento de dados											X
Tolerância a erros		X									
Eficiência na execução			X								
Capacidade de expansão								X			
Generalista								X			
Independência de *hardware*										X	
Facilidade de manipulação							X		X	X	
Modularidade						X	X	X	X	X	X
Operabilidade					X						
Autodocumentação						X	X	X	X	X	
Simplicidade		X				X	X				
Independência de *software*									X	X	
Eficiência de armazenamento			X								
Rastreabilidade	X										
Treinamento					X						

Adaptado de: McCall; Richards; Walters (1977).

Observação

Os critérios de qualidade são denominados de requisitos não funcionais e estão presentes em qualquer tipo de *software*.

Exemplo de aplicação

Vamos considerar a relação entre os fatores e os critérios de qualidade apresentados no Quadro 2 para simularmos a avaliação de um *software* de biblioteca que atende aos alunos de

uma universidade pela internet e controla os empréstimos e as devoluções dos livros. A avaliação mínima deve ser 7,0.

Em primeiro lugar, vamos identificar os fatores que precisam ser avaliados para esse tipo de *software*. Dos 11 fatores disponíveis, vamos selecionar e atribuir um peso de 0 a 5 para sua importância no contexto:

Tabela 1 – Exemplo 1

Fator	Peso
• Correção	4
• Confiabilidade	5
• Eficiência	5
• Integridade	3
• Usabilidade	5
• Manutenibilidade	3
• Testabilidade	3

Os pesos estão relacionados ao grau de importância do fator: **zero** significa nenhuma importância, e **cinco**, importância muito alta.

Agora, vamos selecionar os critérios relevantes dentro de cada fator de qualidade e distribuir o peso atribuído ao fator entre os critérios. Não é necessário que os fatores contenham todos os critérios previstos no Quadro 2.

Tabela 2 – Exemplo 2

Fator	Peso
• Correção	**4**
– Completude	2
– Rastreabilidade	2
• Confiabilidade	**5**
– Tolerância a erros	3
– Simplicidade	2
• Eficiência	**5**
– Na execução	3
– No armazenamento	2
• Integridade	3
– Controle de acesso	3
• Usabilidade	5
– Comunicabilidade	4
– Treinamento	1

• Manutenibilidade	3
– Modularidade	2
– Documentação	1
• Testabilidade	3
– Facilidade de manipulação	3

Temos todos os fatores e critérios selecionados para o tipo de *software* que vamos avaliar. O passo seguinte é fazer uma avaliação de 0 a 10 para cada critério de qualidade. Para atribuir essa nota, podem-se usar critérios subjetivos do próprio avaliador ou criar um *checklist* com os itens que são considerados na atribuição da nota para cada um dos critérios listados para a aplicação. No exemplo, para fins didáticos, são utilizados os critérios subjetivos.

Geradas as notas individuais para cada critério, calculamos a nota final do fator pelo somatório dessas. Após isso, calculamos a nota final da aplicação somando as notas dos fatores de qualidade do mapa que elaboramos.

Vejamos o exemplo:

Tabela 3 – Exemplo 3

Fator	Peso	Nota	Cálculo
• Correção	4	2,5	25/10
– Completude	2	7,5	7,5*2
– Rastreabilidade	2	5	5*2
• Confiabilidade	5	4,5	45/10
– Tolerância a erros	3	10	10*3
– Simplicidade	2	7,5	2*7,5
• Eficiência	5	4,4	44/10
– Na execução	3	8	8*3
– No armazenamento	2	10	10*2
• Integridade	3	3	30/10
– Controle de acesso	3	10	10*3
• Usabilidade	5	4,0	40/10
– Comunicabilidade	4	7,5	7,5*4
– Treinamento	1	10	1*10
• Manutenibilidade	3	1,9	19/10
– Modularidade	2	7	7*2
– Documentação	1	5	5*1
• Testabilidade	3	1,8	18/10
– Facilidade de manipulação	3	6	6*3

O próximo passo é calcular a média ponderada da matriz de acordo com as notas de cada fator:

Tabela 4 – Exemplo 4

Fator	Peso	Nota	Final
• Correção	4	2,5	10
• Confiabilidade	5	4,5	22,5
• Eficiência	5	4,4	22
• Integridade	4	3	12
• Usabilidade	5	4	20
• Manutenibilidade	3	1,9	5,7
• Testabilidade	3	1,8	5,4
• Soma total	28	22,1	97,6

Calculando a nota final da aplicação, temos: 97,6 dividido por 28, que resulta em uma nota final de 3,48 de um total de 5, significando que a aplicação tem uma nota final de 6,96.

Sem o critério estabelecido para a escolha da aplicação de biblioteca, nesse cenário apresentado no exemplo, essa aplicação estaria fora do processo de seleção por não atender as condições mínimas de aceitação.

O importante nessa avaliação é estabelecer critérios claros por meio de um *checklist* para a atribuição correta das notas, a fim de que não gere dúvidas se a nota é justa ou não e permita uma conclusão imparcial da aplicação.

2.2 ISO/IEC 9126 – Características de qualidade do produto de *software*

A norma ISO/IEC 9126 é uma referência técnica mundial para a qualidade de um produto de *software* elaborada pela ISO e pelo IEC em 1991 e publicada no Brasil em 1996 como NBR/13596. Fornece um modelo geral que define seis categorias de características de qualidade do produto de *software* divididas em subcaracterísticas. Estas podem ser avaliadas por meio de métricas quantitativas. Tal conjunto permite dizer se o *software* satisfaz as necessidades e os padrões estabelecidos pelos desenvolvedores e pelos usuários (ISO, 2001).

Observação

IEC significa International Electrotechnical Commission e é uma organização internacional sem fins lucrativos que desenvolve padrões sobre tecnologias elétricas e eletrônicas, inclusive sobre *software*, presente em mais de 150 países.

De acordo com a norma ISO/IEC 9126, o documento propõe um conjunto de atributos de qualidade, distribuídos em seis características, que, por sua vez, são divididas em subcaracterísticas, chamado de modelo de referência e ilustrado na Figura 10.

Os conceitos e detalhes de cada característica e sua composição são descritos a seguir (ISO, 2001):

- Funcionalidade: descreve o que o *software* faz. Pergunta-se: "Satisfaz as necessidades dos usuários?".
  - Subcaracterísticas:
    - Adequação: todas as funções especificadas estão construídas?
    - Acurácia: o produto gera resultados precisos ou dentro do esperado?
    - Interoperabilidade: interage com outros sistemas?
    - Segurança de acesso: previne acesso não autorizado ao sistema?
    - Conformidade: está de acordo com padrões, convenções ou regras?
- Confiabilidade: descreve a capacidade do *software* de realizar suas funções sem falhas. Pergunta-se: "O *software* é tolerante a falhas?".
  - Subcaracterísticas:
    - Maturidade: qual a frequência em que falhas ocorrem?
    - Tolerância a falhas: como o sistema se comporta na presença de problemas?
    - Recuperabilidade: é capaz de se restabelecer em caso de ocorrência de falha?
- Usabilidade: descreve quão fácil um sistema é de ser utilizado. Pergunta-se: "O *software* é fácil de usar?".
  - Subcaracterísticas:
    - Inteligibilidade: o *software* é de fácil entendimento para o uso?
    - Apreensibilidade: o sistema é fácil de aprender a usar?
    - Operacionalidade: é fácil de utilizar?
    - Atratividade: quão agradáveis são a navegação e o uso do *software*?

- Eficiência: descreve qual é a relação entre o desempenho e os recursos utilizados. Pergunta-se: "O *software* é fácil de usar?".

  – Subcaracterísticas:

    – Comportamento em relação ao tempo de resposta: o tempo de resposta do sistema está de acordo com as expectativas?

    – Utilização de recursos: quanto tempo o sistema utiliza dos recursos computacionais (CPU, disco e memória)?

- Manutenibilidade: descreve qual é a facilidade de fazer as alterações. Pergunta-se: "O *software* é fácil de alterar?".

  – Subcaracterísticas:

    – Analisabilidade: qual o esforço para identificar as causas de falhas?

    – Modificabilidade: qual o esforço para realizar as alterações a eventuais mudanças no sistema?

    – Estabilidade: qual o risco de efeitos inesperados de alterações?

    – Testabilidade: qual o esforço para testar o *software* alterado?

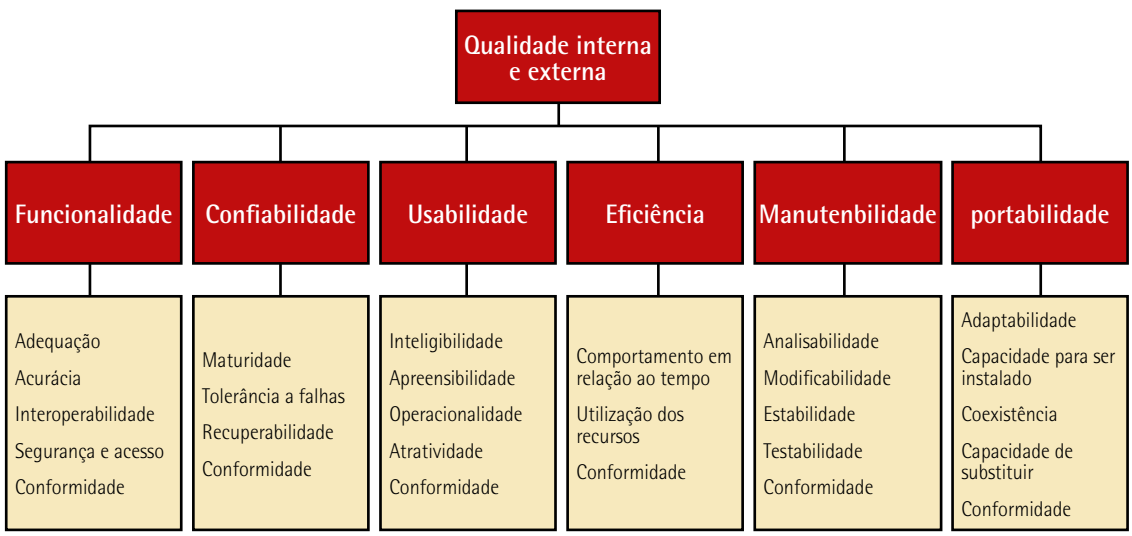

Figura 10 – Modelo de referência da norma ISO/9126

- Portabilidade: descreve qual é a facilidade de transferir a aplicação para outro ambiente. Pergunta-se: "O *software* é utilizável em outro ambiente?".

  – Subcaracterísticas:

    – Adaptabilidade: qual a facilidade de se adaptar o produto para funcionar em outros ambientes operacionais?

    – Capacidade para ser instalado: qual o esforço para instalar o *software* em outros ambientes?

    – Coexistência: o *software* está de acordo com padrões referentes à portabilidade?

    – Capacidade de substituir: qual o esforço de utilizar o *software* em substituição a outro?

2.2.1 Métricas de qualidade

A norma ISO/IEC 9126 não define métricas para utilização nas características propostas, mas define três classes de aplicação de métricas que auxiliam as empresas na criação de suas próprias métricas.

Com base na estrutura apresentada na Figura 11, tem-se as classes de métricas externas, de métricas internas e a qualidade em uso definidas pela norma.

As métricas externas definem indicadores para avaliar um *software* por meio da medição do comportamento do sistema quando da sua execução. O objetivo das métricas externas é medir se o *software* satisfaz as necessidades de operação do sistema pelos usuários. Por exemplo: avaliando a característica de eficiência e a subcaracterística de tempo de resposta, cria-se a métrica X para medir o tempo que a aplicação leva desde o acesso até a abertura da página inicial. Com essa métrica é possível avaliar se a aplicação atende aos requisitos de desempenho esperados.

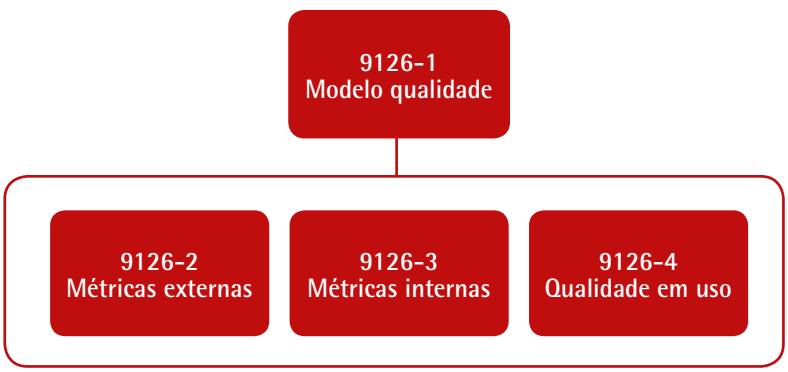

Figura 11 – Estrutura da série de normas da ISO/9126

As métricas internas são aplicáveis para avaliar o *software* durante sua construção, ou seja, para a garantia da qualidade. Por meio das métricas internas é possível medir a qualidade de produtos

intermediários produzidos e predizer a qualidade final do *software*. Por exemplo: avaliar o número de erros encontrados em revisões.

A qualidade em uso avalia a qualidade do *software* na perspectiva do uso cotidiano pelos usuários. Nesse cenário, são avaliadas a efetividade, a produtividade, a segurança e a satisfação do usuário com o *software*.

A norma ISO/IEC 9126 pode ser utilizada para definir os padrões de qualidade esperados pelos usuários e para a avaliação da garantia da qualidade durante a construção do *software*, bem como para o controle de qualidade por meio da verificação antes da entrega ao usuário final.

Sua aplicação é idêntica aos requisitos de McCall, porém com mais características que devem ser avaliadas. O exemplo de aplicação demonstrado no tópico referente ao Modelo de McCall serve de referência para a aplicação da norma ISO/IEC 9126, utilizando o mesmo princípio de adaptação ao tipo de *software* e posteriores pontuação e avaliação da qualidade do produto de *software*.

Lembrete

A norma ISSO/IEC 9126 pode ser utilizada como *checklist* para definir os requisitos não funcionais e para avaliar se estes estão atendidos ao final da construção do *software*.

2.3 Norma ISO/IEC 12207 – Ciclo de vida do *software*

A norma ISO/IEC 12207 descreve os processos de ciclo de vida de um produto de *software* e foi publicada em 1995. A norma define um conjunto de processos que padroniza as atividades e orienta o desenvolvimento, a manutenção e a aquisição para as empresas de desenvolvimento de *software*.

Observação

Ciclo de vida é uma estrutura que contém atividades aplicadas ao desenvolvimento, à operação e à manutenção de *software*, desde a definição de requisitos até o término de seu uso (ISO, 2008).

A norma ISO/IEC 12207 está estruturada em três grupos de processo: os processos fundamentais, que abrangem a execução do desenvolvimento do *software*; os processos de apoio, que são as atividades de suporte e qualidade do *software*; e os processos organizacionais, que são as atividades que permitem a manutenção e a melhoria dos processos. A organização desses processos é apresentada na Figura 12.

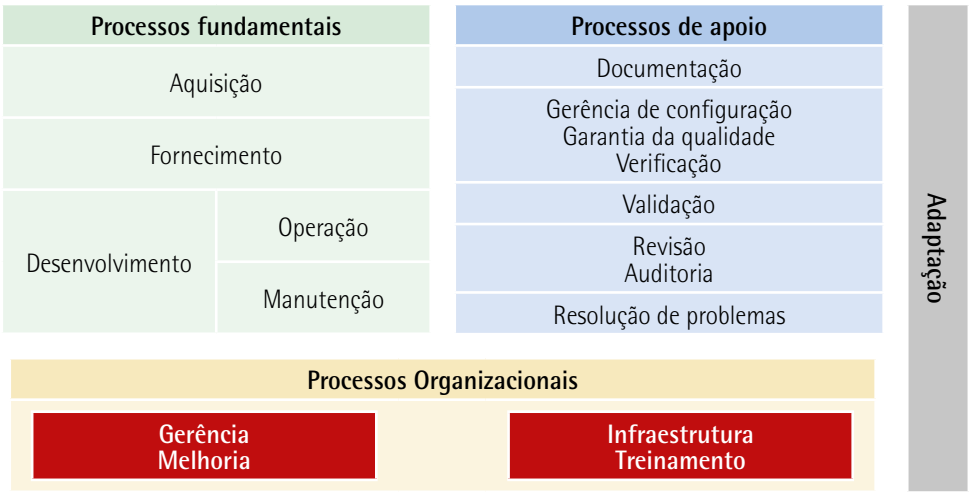

Figura 12 – Estrutura da norma ISO/IEC 12207, publicada em 1995

2.3.1 Processos fundamentais

Execução do desenvolvimento, da operação e da manutenção do *software* durante o seu ciclo de vida.

- **Aquisição**
  Define as tarefas do adquirente de *software*. Inclui as atividades de definição das necessidades de adquirir um *software*, pedido de proposta, seleção de fornecedores, gerência da aquisição e aceitação do *software*.

- **Fornecimento**
  Define as tarefas do fornecedor ou da organização que provê o *software*. Abrange as atividades de preparação da proposta, assinatura de contrato, determinação dos recursos necessários, planejamento do projeto e entrega do *software*.

- **Desenvolvimento**
  Define as tarefas do desenvolvedor ou da organização que desenvolve o *software*. Inclui as atividades de definição de requisitos, análise, projeto, codificação, integração, testes, instalação e aceitação do *software*.

- **Operação**
  Define as tarefas a serem executadas pelo operador ou pela organização que presta serviço de manutenção do seu ambiente computacional para seus usuários.

- **Manutenção**
  Define as tarefas a serem realizadas por quem presta serviços de manutenção do *software*. Inclui as atividades de gerenciamento das alterações, migração e descontinuação do *software*. Esses tópicos são detalhados no tópico 8.

2.3.2 Processos de apoio

Definem as tarefas que auxiliam outros processos, principalmente, as relacionadas à qualidade do *software*.

- **Documentação**
  Define as tarefas para registrar as informações produzidas por um processo. Inclui as atividades de produção, edição, distribuição e manutenção dos documentos necessários a gerentes, desenvolvedores e usuários do *software*.

- **Configuração**
  Define as tarefas de identificação e controle dos artefatos produzidos no desenvolvimento do *software*. Engloba as atividades de identificação, controle, armazenamento, liberação, manipulação, distribuição e modificação dos artefatos que compõem o *software*.

- **Garantia da qualidade**
  Define as tarefas para garantir que o *software* esteja seguindo os padrões de qualidade estabelecidos. Técnicas de revisão, auditoria, verificação e validação podem ser utilizadas.

- **Verificação**
  Define as tarefas para verificar se os artefatos de *software* atendem aos requisitos estabelecidos.

- **Validação**
  Define as tarefas para validação dos requisitos e do *software* final e se estão de acordo com a proposição inicial.

- **Revisão**
  Define as tarefas para avaliação da situação e dos produtos de uma atividade. Atividade essencial para a garantia da qualidade.

- **Auditoria**
  Define as tarefas para determinar a conformidade de requisitos, planos e contrato aos padrões estabelecidos.

- **Resolução de problemas**
  Define as tarefas para análise e eliminação dos problemas identificados durante a execução dos processos de desenvolvimento, de operação e de manutenção do *software*.

2.3.3 Processos organizacionais

Definem as tarefas que permitem a manutenção ou a melhoria contínua dos processos.

- **Gerência**
  Define as tarefas de gerenciamento do projeto durante o ciclo de vida.

- **Infraestrutura**
  Define as tarefas de fornecimento de recursos para outros processos, tais como *hardware*, *software*, ferramentas e técnicas.

- **Melhoria**
  Estabelece as tarefas que visam medir, controlar e melhorar o seu ciclo de vida.

- **Treinamento**
  Estabelece as tarefas necessárias para manter o pessoal treinado e o treinamento dos novos funcionários.

2.3.4 Processos de adaptação

Camada que proporciona a flexibilidade necessária a todo processo. Estão definidas as tarefas para adequar a aplicação da norma na organização ou em projetos de *software* que possuem características específicas. Esses processos permitem que a norma seja adaptável a qualquer empresa de desenvolvimento.

Para exemplificar a abrangência de utilização da norma, a seguir são descritas as atividades envolvidas no processo fundamental de desenvolvimento da norma ISO/IEC 12207 (1995):

- análise dos requisitos do sistema;
- projeto da arquitetura do sistema;
- análise dos requisitos do *software*;
- projeto de arquitetura do *software*;
- projeto detalhado do *software*;
- codificação e testes do *software*;
- integração do *software*;
- testes de qualificação do *software*;
- integração do sistema;
- teste de qualificação do sistema;
- instalação do *software*;
- apoio à aceitação do *software*.

A norma ISO/IEC 12207 define os requisitos básicos, a partir desse conjunto, para que as organizações elaborem e definam a sua metodologia de desenvolvimento de *software*. Vale lembrar que metodologias são amplas e atendem a qualquer tipo de projeto, portanto deve haver, no início do projeto, um processo de customização da metodologia para adequá-la às condições de execução de cada projeto.

2.4 ISO/IEC 14598 – Avaliação de produto de *software*

A norma ISO/IEC 14598 foi publicada em 1999 e estabelece um conjunto de tarefas com o objetivo de padronizar a avaliação da qualidade do produto de *software*. Trata-se de um complemento da norma ISO/IEC 9126 e deve ser utilizada em conjunto com esta. A norma ISO/IEC 14598 possui atividades para medir as características de um produto de *software*, relatórios e documentos de avaliação (1999).

A norma fornece subsídios para a avaliação de um produto de *software* em desenvolvimento ou já existente e descreve detalhadamente como avaliar cada característica do produto de forma quantitativa. A avaliação não se limita a avaliar o código ou o produto pronto, mas todos os artefatos produzidos durante o ciclo de vida do *software*, tais como: especificação de requisitos, protótipo de telas, modelo de dados, roteiro de testes, entre outros.

A norma ISO/IEC 14598 está subdividida em seis partes, e o objetivo de cada série está detalhado no Quadro 3.

Quadro 3 – Objetivo da série de normas da ISO/IEC 14598 (1999)

Norma	Título	Objetivo
14598-1	Parte 1 – Visão geral	Descrição geral da estrutura da norma e dos processos de avaliação
14598-2	Parte 2 – Planejamento e gestão	Tarefas de planejamento e gestão do processo de avaliação
14598-3	Parte 3 – Processo para desenvolvedores	Atividades de avaliação durante o processo de desenvolvimento de *software*
14598-4	Parte 4 – Processo para adquirentes	Atividades de avaliação do processo de seleção para aquisição do *software*
14598-5	Parte 5 – Processo para avaliadores	Descrição do processo e das atividades de avaliação para os avaliadores
14598-6	Parte 6 – Documentação de módulos de avaliação	Descrição de métodos e ferramentas para apoio ao processo de avaliação

A série de normas descreve o processo de avaliação do produto de *software* sob três perspectivas: desenvolvedor, adquirente e avaliador. Cada perspectiva possui quatro fases distintas no processo de avaliação: análise, especificação, projeto e execução, conforme exibido no Quadro 4.

Quadro 4 – Definição do processo de avaliação segundo a norma ISO/IEC 14598 (1999a)

	Análise	Especificação	Projeto	Execução
Processo para desenvolvedor	Definir as métricas dos requisitos de qualidade	Criar requisitos de qualidade	Planejar a avaliação durante a construção	Monitorar e controlar a qualidade durante a construção
Processo para adquirente	Estabelecer o escopo da avaliação	Definir métricas externas a serem realizadas	Planejar, programar e documentar a avaliação	Monitorar e controlar a avaliação
Processo para avaliador	Descrever os objetivos da avaliação	Definir escopo da avaliação e das medições	Documentar os processos a serem usados pelo avaliador	Obter os resultados a partir das ações de medição e verificação

Isso se faz necessário, uma vez que cada um desses envolvidos tem um papel diferente na construção do produto. O avaliador deve se preocupar em estabelecer as métricas que serão aplicadas considerando os padrões estabelecidos. O desenvolvedor deve se preocupar em criar processos e formas de medição da qualidade do produto durante sua construção. O adquirente tem a responsabilidade de avaliar um produto de *software* que está sendo adquirido e, para isso, precisa criar parâmetros e métricas que deixem claros os critérios adotados no seu parecer.

Para melhor descrever o conteúdo da norma ISO/IEC 14598, a seguir é descrito em detalhes o processo de avaliação de produtos de *software* sob a abordagem da série 14598-1, pois esse ciclo se repete nas demais séries incluindo as tarefas específicas de cada uma delas.

São quatro fases distintas que são apresentadas na Figura 13.

1) Estabelecer os requisitos de avaliação.

2) Especificar a avaliação.

3) Projetar a avaliação.

4) Executar a avaliação.

Figura 13 – Visão geral do processo de avaliação da série ISO/IEC 14598

1) **Estabelecer os requisitos de avaliação**
Consiste em dizer quais características do produto de *software* fazem parte do processo e devem estar associadas às características de qualidade definidas na norma ISO/IEC 9126-1. Nessa fase, é importante adequar as características da qualidade ao produto a ser avaliado, pois cada *software* tem suas características específicas e que devem ser respeitadas.

O resultado dessa fase consiste em dizer quais são os padrões de qualidade esperados.

2) **Especificar a avaliação**
Essa fase consiste em definir as medidas quantitativas para as características selecionadas na fase de requisitos. Além disso, devem-se definir as metas e os critérios de avaliação de cada uma das medidas. Novamente as séries da norma ISO/IEC 9126-2 e ISO/IEC 9126-3 que definem as métricas internas e externas podem ser utilizadas como referência.

3) **Projetar a avaliação**
O objetivo dessa fase é descrever quem, como, o que, quando e onde as avaliações são aplicadas e documentadas para permitir a análise comparativa da evolução das medidas realizadas.

4) **Executar a avaliação**
Nessa fase, a avaliação é aplicada e se obtêm os resultados quantitativos das medidas. Esses resultados são comparados com as metas estabelecidas de acordo com os critérios definidos.

2.4.1 Relação entre as séries das normas ISO/IEC 9126 e ISO/IEC 14598

A partir da descrição das fases do processo de avaliação é possível verificar que há uma forte relação entre as normas ISO/IEC 9126 e ISO/IEC 14598. Portanto, ambas devem ser utilizadas em conjunto para a obtenção de melhores resultados no processo de avaliação. Essa relação é apresentada na Figura 14.

Observa-se que há convergência entre as normas no que tange às características e subcaracterísticas definidas na série ISO/IEC 9126-1 e, principalmente, na definição das métricas internas, das externas e na qualidade em uso descrita nas séries ISO/IEC 9126-2, ISO/IEC 9126-3 e ISO/IEC 9126-4. As métricas são essenciais para que o processo de avaliação seja realizado.

A série de normas da ISO/IEC 14598 define detalhadamente o processo de avaliação e como essas métricas são utilizadas durante a aplicação do modelo de avaliação do processo de *software*.

Lembrando que a avaliação da qualidade do produto de *software* também está envolvida nesse cenário de avaliação.

O processo de avaliação é iniciado com a definição do produto de *software* que faz parte da avaliação, a elaboração do plano de avaliação e a definição dos papéis e responsabilidades das pessoas envolvidas no processo.

O passo seguinte é escolher, do conjunto de características e atributos de qualidade da norma ISO/IEC 9126, aqueles que se aplicam à avaliação em questão. A seguir, procede-se à avaliação do produto, considerando que diagramas, modelos e documentos serão avaliados como requisitos de qualidade interna (ISO/IEC 9126-3), enquanto o código e a aplicação serão avaliados como requisitos externos (ISO/IEC 9126-2). Para cada um desses requisitos de qualidade definidos devem ser estabelecidas métricas que são a referência das metas a serem atingidas. Estas podem variar em uma escala atribuída pelo avaliador, em que será definido o valor mínimo aceitável de avaliação para cada requisito.

Ao final, com base nas medidas coletadas pelo processo de avaliação, realizam-se a avaliação dos resultados e o julgamento se o produto atende ao conjunto de requisitos da qualidade definidos.

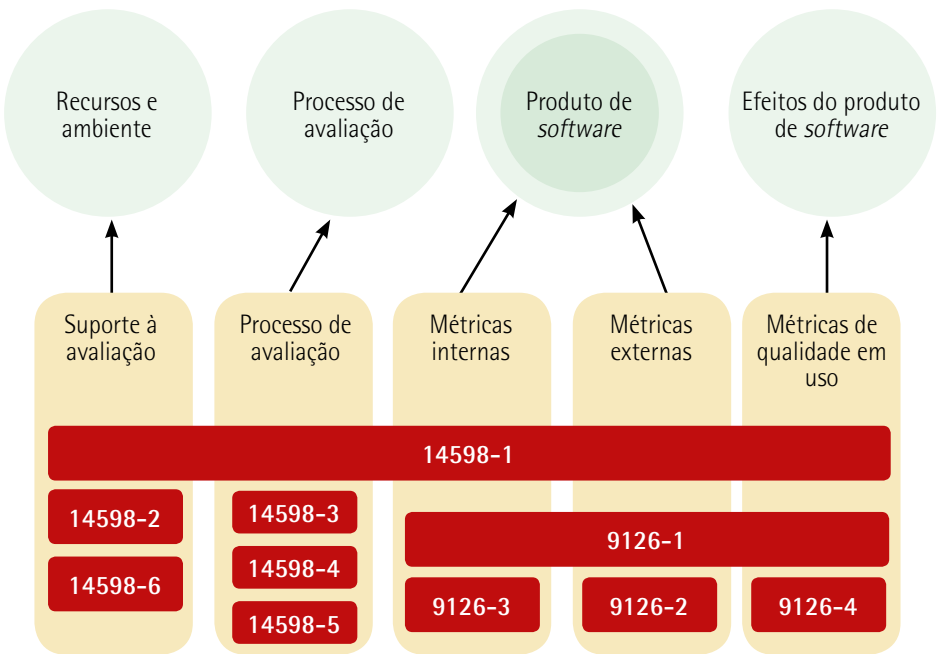

Figura 14 – Relação entre as séries de normas ISO/IEC 9126 e ISO/IEC 14598

Portanto, a utilização das normas ISO/IEC 14598 e ISO/IEC 9126 deve ser em conjunto, pois são complementares. A geração do resultado da avaliação da qualidade deve ser realizada sempre com a participação do pessoal do produto que está sendo avaliado e de um avaliador certificado pela ISO.

2.5 ISO/IEC 25000 – SQuaRE (*Software Product Quality Requirements and Evaluation*)

O Modelo SQuaRE **é a nova versão das normas** ISO/IEC 9126 e ISO/IEC 14598 para especificação e avaliação da qualidade do produto de *software* publicada em 2008, com o objetivo de unificar o processo de medição da qualidade do *software*.

O objetivo da criação dessa norma é atualizar as informações de requisitos de qualidade, alinhar com novos conceitos de avaliação da qualidade, aumentar a consistência entre os atributos da qualidade descritos na norma ISO/IEC 9126 e o processo de avaliação da norma ISO/IEC 14598 e incluir procedimentos novos. Com isso, obtém-se um documento único e consistente para todo o processo de qualidade.

A ISO/IEC 25000 contém cinco séries distintas, cuja estrutura é ilustrada na Figura 15.

Figura 15 – Séries de normas ISO/IEC 25000

Observação

A estrutura da norma ISO/IEC 25000 (SQuaRE) está em processo de revisão e deve sofrer alterações.

A série ISO/IEC 25000 faz uma introdução geral sobre a norma e definição gerais sobre os termos utilizados nas séries. Também descreve orientações de como utilizar a norma e o relacionamento entre elas.

A série ISO/IEC 25001 corresponde à norma ISO/IEC 9126, que descreve as características dos requisitos de qualidade externos e internos. Detalha os requisitos de qualidade do produto de *software*.

A série ISO/IEC 25002 fornece um modelo de referência para medição da qualidade e orientações para a aplicação das métricas.

A série ISO/IEC 25003 apoia a especificação dos requisitos de qualidade durante a fase de levantamento de requisitos para um novo produto de *software*, ou seja, auxilia na definição dos padrões de qualidade esperados. Traz da norma ISO/IEC 9126 o conceito da relação de requisitos de qualidade com os requisitos do *software*.

A série ISO/IEC 25004 descreve os requisitos, as orientações para o processo de avaliação de produto de *software* e a definição dos documentos necessários para registrar todo o ciclo de medição realizado durante a avaliação. Apresenta uma estrutura de avaliação da qualidade oriunda das normas ISO/IEC 9126-1 e ISO/IEC 14598.

A Figura 16 mostra o conteúdo básico de cada uma das séries da norma ISO/IEC 25000, em que se pode observar a total interação com as normas de origem ISO/IEC 9126 e ISO/IEC 14598.

Figura 16 – Conteúdo básico da série de normas ISO/IEC 25000

Observação

A norma ISO/IEC 25000 (SQuaRE) é mais recente, atualizada e incorpora todos os conceitos das normas ISO/IEC 9126 e ISO/IEC 14598. Portanto, deve-se priorizar a sua utilização em relação às anteriores.

A norma ISO/IEC 25000 organiza, completa e elimina alguns conflitos que existem entre as normas ISO/IEC 9126 e ISO/IEC 14598, e ainda possui exemplos de utilização da norma que facilitam seu uso. Portanto, torna-se a referência para ser aplicada nas avaliações de produto de *software*.

Resumo

Nesta unidade, verificamos quão importante é o tema Qualidade de *Software*. Embora exista uma forte resistência das equipes de desenvolvimento de *software* e das próprias empresas fornecedoras de *software*, as organizações-clientes estão cada vez mais exigentes com a qualidade do produto que recebem.

Abordamos que o conceito básico de qualidade sobre fazer certo da primeira vez é muito pouco utilizado, prevalecendo a técnica de tentativa e erro. Não se atua de forma preventiva diante dos problemas que podem ocorrer e que, embora conhecidos, não são tratados, e as atitudes tornam-se reativas. O ato de revisar todo e qualquer artefato antes de entrega ao cliente simplesmente é ignorado na maioria das vezes. Além disso, os desenvolvedores continuam a pensar que o gerente é o responsável pela qualidade do *software*, quando a responsabilidade é de toda a equipe. Seguindo e praticando esses princípios básicos, os resultados colhidos estão cada vez mais alinhados com as expectativas e com a satisfação dos usuários finais.

Vimos também que os obstáculos à qualidade são muitos. Vão desde a cultura da organização que fornece o *software* e atua segundo o modelo corrente, que considera: "Se está dando certo até agora, para que alterar?", quando os resultados poderiam ser muito melhores, até os próprios clientes, que continuam a contratar essas empresas que fornecem *software* ruim em vez de outras com processos de qualidade, porque simplesmente "Todo projeto de *software* é assim mesmo. Sempre está tudo com problema."; além disso, os preços dessas empresas são mais baixos.

Aprendemos ainda que outro fator relevante como obstáculo à qualidade são as características dos *softwares* atuais que aumentaram

muito a complexidade da interface com o usuário, as soluções técnicas cada vez mais detalhadas para permitir uma infinidade de mídias a fim de consumir as informações e as necessidades de prazo cada vez menores para a construção do sistema. Além disso, temos a divergência das visões dos envolvidos no processo de desenvolvimento, em que cada um quer ver os resultados à sua maneira, mas são antagônicos; por exemplo: o cliente quer um produto rápido e barato, o usuário quer que atenda a todas as funções que ele deseja e que o sistema esteja sem erros, o desenvolvedor quer apenas codificar e ver seu programa funcionando e o gerente do projeto precisa que o sistema funcione dentro do prazo e do custo orçado. Essas diferentes visões tornam a qualidade do *software* um elemento deixado em segundo plano.

Abordamos ainda que, para ter qualidade em tudo o que é produzido, não basta testar no final para ver se está tudo certo. Como visto, a qualidade é um processo preventivo e, para tal, requer que sejam tomadas ações de garantia da qualidade durante o desenvolvimento, com verificações e validações de tudo o que for produzido. Ao final, tem-se o controle da qualidade, que é a aplicação de testes para ter o aceite do *software* produzido. Portanto, vimos que para minimizar os problemas de qualidade, as empresas precisam se dedicar mais à garantia dessa qualidade, o que é realizado durante o processo de desenvolvimento, buscando atuar fortemente na prevenção de problemas e ter o controle de qualidade, realizado ao final, apenas como um atestado de que o produto atende aos requisitos do usuário.

Aprendemos também que a norma ISO/IEC 9000 trouxe para a área de Tecnologia da Informação os fundamentos de um SGQ, em que a preocupação está em transformar as atividades realizadas por pessoas em atividades perenes e que possam ser repetidas por qualquer indivíduo treinado e capacitado no processo. O SGQ define um conjunto de processos para que as empresas formalizem o seu padrão de trabalho, atendendo a um conjunto de requisitos essenciais para a elaboração de um trabalho. A norma NBR/ISO 9000 é o nome genérico que se dá às diversas normas que abordam o assunto. As normas básicas para garantia da qualidade são as contratuais – ISO 9001, ISO 9002 e ISO 9003 –, e as organizações só podem ser certificadas em relação a essas normas.

Conhecemos também a norma NBR/ISO 9000-3 – Norma para Empresas de Desenvolvimento de *Software* –, que define as diretrizes para facilitar a aplicação da norma ISO 9001 a organizações que desenvolvem, fornecem e mantêm *software*. Publicada em junho de 1993, essa norma é dividida em três partes: atividades de estrutura, atividades do ciclo de vida e atividades de suporte. Nas atividades de ciclo de vida, a NBR/ISO 9000-3

abrange questões relacionadas ao entendimento dos requisitos funcionais, o uso de metodologias consistentes para desenvolvimento de *software* e o gerenciamento do projeto, desde a concepção até a implantação do *software*. Portanto, o objetivo da norma NBR/ISO 9000-3 é indicar quais processos a organização deve ter e manter para o desenvolvimento do *software* com qualidade.

Vimos que a certificação ISO 9000 é reconhecida no mercado por muitas empresas e traz uma série de benefícios às organizações que adotam o sistema, como abertura de novos mercados, maior satisfação do cliente e menores custos de desenvolvimento. Para realizar a certificação, a empresa deve analisar o seu processo e treinar os funcionários para a conscientização da necessidade da qualidade. Na sequência, desenvolve os procedimentos necessários ao sistema da qualidade, seleciona um órgão certificador que irá avaliar se o sistema está de acordo com a norma, faz pré-auditorias e realiza a auditoria final de aderência ao SGQ para a certificação.

Abordamos ainda que, durante a definição, deve-se tomar cuidado para que o processo não se torne burocrático, do tipo que ninguém usa. Este deve se adaptar ao dia a dia da empresa, e não o contrário, senão o manual somente será lido quando houver processo de certificação. Além disso, a organização será reavaliada a cada dois anos e deverá manter o pessoal treinado e atualizado quanto aos processos para que não perca o certificado, pois isso pode ser muito custoso em termos financeiros e de imagem junto aos clientes. O gestor do SGQ deve ser o responsável por manter esse processo.

Foram apresentadas também as principais normas para verificação da qualidade de um produto de *software* nas organizações. Essas normas têm como objetivo minimizar os problemas de qualidade resultantes do processo de desenvolvimento de um *software*.

Em seguida, vimos que as primeiras preocupações com qualidade do produto de *software* surgiram em 1977, com McCall, que já destacava a visão de produzir um *software* pensando nas fases seguintes de operação e de manutenção, e não apenas na conclusão do trabalho. Isso mudou a forma de produção de *software* e deu origem a diversas normas que focam avaliar a qualidade dos produtos de *software*. O foco dos requisitos de McCall está em três abordagens. A primeira é a abordagem de revisão, com os requisitos de facilidade de manutenção, flexibilidade e facilidade de testes. Outra está relacionada à operação do produto com os requisitos de correção, confiabilidade, integridade e eficiência. A última abordagem é a de transição, com os requisitos de portabilidade, reúso e interoperabilidade.

Abordamos ainda que a norma de qualidade de produto de *software* pioneira foi a ISO/IEC 9126, que atribuiu características e subcaracterísticas ao *software* para permitir a avaliação de quão alinhada ao padrão de qualidade esperado a aplicação construída está, gerando uma avaliação quantitativa que possibilita a aceitação do produto ou a tomada de ações corretivas. Aprendemos que a norma segue basicamente os requisitos não funcionais definidos por McCall de forma reestruturada e acrescenta uma visão de facilidade de uso e funcional. Essas seis características são subdivididas para facilitar a avaliação. Para a característica de funcionalidade, são avaliadas a adequação, a acurácia, a segurança de acesso e a conformidade. Na confiabilidade, são avaliadas a maturidade, a tolerância a falhas, a recuperabilidade e a conformidade. Para a usabilidade, avaliam-se os requisitos de facilidade de operação da aplicação. Para a eficiência, o foco é a avaliação do desempenho. Para a manutenibilidade, são abordados os requisitos de facilidade para incorporar mudanças. Para a portabilidade, são avaliados os requisitos de múltiplos ambientes.

Em seguida, vimos que a norma ISO/IEC 12207 já traz o foco de processo de desenvolvimento de *software*, abordando o ciclo completo, desde a concepção até a manutenção das aplicações, definindo uma forma estruturada para a construção de um sistema, independentemente do processo de desenvolvimento pela organização: espiral, incremental ou iterativo.

Aprendemos que a norma ISO/IEC 14598 traz os conceitos de avaliação quantitativa do produto de *software* e é utilizada em conjunto com a norma ISO/IEC 9126 para produzir notas relacionadas aos padrões de qualidade esperados e à avaliação de cada requisito de qualidade.

Finalmente, abordamos a norma ISO/IEC 25000 – SquaRE –, que unificou as normas ISO/IEC 9126 e ISO/IEC 14598 em apenas uma, incluindo novas atualizações e proporcionando maior facilidade de uso, por trazer todos os conceitos de requisitos de qualidade do produto de *software* e avaliação na mesma norma.

Exercícios

Questão 1. No controle da qualidade, uma atividade deve ser executada visando avaliar se as ações de qualidade planejadas estão sendo executadas de acordo com o processo estabelecido. Essa atividade chama-se auditoria e pode gerar ações corretivas no caso de não encontrar conformidades. Essas ações corretivas são comumente classificadas em três tipos. Qual das alternativas corresponde ao tipo de ação corretiva que verifica especificamente se as ações de qualidade planejadas estão sendo executadas de forma adequada?

A) Auditorias de produto.

B) Auditorias de processo.

C) Auditorias de sistemas de qualidade.

D) Sistemas de gestão de qualidade.

E) Garantia de qualidade.

Resposta correta: alternativa B.

Análise das alternativas

A) Alternativa incorreta.

Justificativa: essa ação corretiva verifica apenas a conformidade do produto, e não os processos sobre os quais foi elaborado.

B) Alternativa correta.

Justificativa: somente essa ação corretiva de auditoria permite verificar se as ações de qualidade estão sendo executadas de forma adequada. Isso é possível com a verificação de cada ação (processo, etapas) do ciclo planejado.

C) Alternativa incorreta.

Justificativa: essa ação corretiva observa e avalia o processo como um todo e mensura os resultados, avalia a eficácia da utilização do sistema de qualidade implantado, e não especificamente de cada processo executado pelo sistema em questão.

D) Alternativa incorreta.

Justificativa: sistemas de gestão têm como objetivo padronizar processos de uma empresa, e não especificamente verificar as ações de qualidade planejadas.

E) Alternativa incorreta.

Justificativa: não verificam a execução de ações de qualidade planejadas. Essa alternativa está focada na garantia de qualidade do produto gerado com a adequação de padrões.

Questão 2. Um Sistema de Gestão da Qualidade (SGQ) tem como objetivo padronizar os processos de uma empresa para a produção de seu produto final, proporcionando a satisfação de seus clientes e a melhoria contínua dos seus processos (ANTONIONI, 1995). Existem diversos fatores que podem motivar a implantação de um SGQ.

Assinale a alternativa que apresenta o fator mais relevante à necessidade de um SGQ.

A) Diminuir o entrosamento entre empresa e cliente com o intuito de não dificultar os processos produtivos, regras de negócios e o ciclo planejado de entrega do produto.

B) Diminuir o volume de dados relacionados às atividades ou dos processos de qualidade com o intuito de facilitar a tomada de decisão.

C) Diminuir o volume de dados relacionados aos processos de gestão e das etapas de geração de um produto para flexibilizar as correições e melhorias.

D) Evitar medir de forma contínua a satisfação do cliente.

E) A alta direção de uma empresa reconhece que a qualidade é um diferencial e patrocina o processo.

Resposta desta questão na plataforma.