QUALIDADE DE SOFTWARE

SOLUÇÕES EDUCACIONAIS INTEGRADAS

Qualidade de *software* e aplicações

Diego Martins Polla de Moraes

OBJETIVOS DE APRENDIZAGEM

- Explicar as atividades de verificação e validação de *software*.
- Caracterizar as revisões, inspeções e auditorias de *software*.
- Identificar as normas IEEE 1012 e IEEE 1028.

Introdução

Com o início da transformação digital, nossa sociedade tem estado cada vez mais dependente de *software*. Todos os benefícios que a automatização de processos, tarefas e negócios promove na vida das pessoas e das organizações podem ser prejudicados por *software* não confiáveis, que contêm falhas ou que não fazem o que supostamente se propõem a fazer. Para Pressman e Maxim (2016), os órgãos de governo e as empresas vêm se preocupando mais com possíveis falhas de *software* capazes de causar prejuízos na casa de bilhões de dólares. Para lidar com esses desafios, a qualidade de *software* precisa estar presente em todo o ciclo de vida do *software*, com ferramentas e técnicas capazes de diminuir esses riscos.

Neste capítulo, você vai conhecer os processos de verificação e validação, bem como as atividades de inspeção, revisão e auditoria de *software* e as normas IEEE 1012:2016 e IEEE 1028:2008.

Verificação e validação

Dentro das atividades de garantia da qualidade de *software*, a verificação e a validação atuam sobre todo o ciclo de vida do *software*, possibilitando se minimizem os desvios (GALLOTTI, 2016).

É muito comum o equívoco de que verificação e validação são sinônimos, mas, na verdade, compreendem atividades diferentes e complementares. Para Gallotti (2016), a verificação consiste na checagem da conformidade do sistema aos seus requisitos, e a validação, por sua vez, refere-se à constatação de que o sistema atende às necessidades do cliente.

Segundo Schach (2009), a verificação se vincula ao processo de determinar se um fluxo de trabalho foi executado adequadamente, sempre ao final do fluxo de trabalho. Já a validação tem como intuito avaliar se o produto pronto atende às necessidades do respectivo cliente.

Para compreender melhor essa diferença, analisemos o seguinte exemplo. Um cliente solicitou um *software* para registrar suas vendas. Todo o processo de desenvolvimento do *software* foi realizado de acordo com as boas práticas de engenharia de *software*, com uma série de etapas de verificação durante sua construção. Quando o *software* foi entregue, o cliente, ao realizar a validação, verificou que um detalhe importante não foi considerado. Seus vendedores, os futuros usuários do sistema, se deslocavam de motocicleta, o que inviabilizaria a utilização de um sistema desenvolvido para *web*, sem considerar os detalhes de responsividade para utilização em dispositivos mais portáteis, como *tablets* ou celulares. Nesse caso, a verificação foi realizada e os ajustes necessários feitos durante a etapa de desenvolvimento do *software*. Na validação (que, no exemplo, ocorreu de forma tardia), constatou-se essa deficiência: embora o *software* funcionasse e tivesse sido feito de acordo com os requisitos definidos para ele, não atendia à necessidade daquela organização.

Pode-se dizer, então, que a verificação responde à seguinte pergunta: o *software* está sendo construído da maneira correta? E a validação responde: estamos construindo o *software* de que precisamos (PRESSMAN; MAXIM, 2016)?

A verificação e a validação fazem parte de um amplo rol de atividades de garantia de qualidade de *software*, já que, mesmo que os testes tenham uma missão muito importante para ambas, muitas outras atividades são essenciais (PRESSMAN; MAXIM, 2016):

- revisões técnicas;
- auditorias de qualidade e configuração;
- monitoramento de desempenho;

- simulação;
- estudo de viabilidade;
- revisões de documentação e de base de dados;
- análise de algoritmo;
- teste de desenvolvimento, de usabilidade, de qualificação, de aceitação e de instalação.

Para Gallotti (2016), a verificação vincula-se à equipe que desenvolve o *software* e a validação reside na interação entre o usuário e o *software*. Isso significa que os testes unitários, de integração e de sistema, fazem parte da verificação, e os de aceitação compõem a validação.

Por se tratar de uma atividade interna da equipe de desenvolvimento, a verificação de *software* tem condições de ser realizada em sua plenitude, desde que seja planejada no processo de desenvolvimento de *software* e priorizada no planejamento do projeto. Já a atividade de validação depende muito do cliente que receberá o *software* — se os usuários não se organizarem para realizar uma validação do *software* adequada, possíveis problemas serão levados até a fase de uso em produção do *software*. Portanto, se os usuários responsáveis pelo sistema que está sendo entregue não tiverem um processo de validação estruturado, cabe à equipe de desenvolvimento apresentar a justificativa da importância do cliente executar uma fase de validação e os benefícios dessa execução, além de oferecer apoio nesse processo. Não há como a equipe de desenvolvimento trazer para si esta atividade, pois depende diretamente da experiência dos usuários-chave do cliente, que precisarão percorrer os fluxos do sistema confirmando se o sistema atende às necessidades daquela organização.

As atividades de verificação se dividem em dois tipos (GALLOTTI, 2016): estática, que não exige que o *software* esteja sendo executado e é feita sobre os artefatos de documentação do *software* (p. ex., protótipos, casos de uso, listas de requisitos, histórias de usuário, documentação de visão e outros artefatos de documentação); e dinâmica, que envolve a execução do *software*, do produto funcionando, e na qual todos os tipos de testes de *software* se enquadram (GALLOTTI, 2016).

Fique atento

Um processo de validação de *software* pode estar atrelado a leis ou normas específicas de determinada área, como no caso de *software* de gestão pública, jurídicos, bancários e de saúde. Confira no *site* da Harbor informática industrial o artigo "Preciso validar meu software. Por onde começo?" (MELO, 2018), um exemplo de validação de *software* que deve seguir as regras da Agência Nacional de Vigilância Sanitária (Anvisa).

Revisões, inspeções e auditorias de *software*

As revisões e inspeções são atividades de controle de qualidade que aferem a qualidade de um projeto, o que inclui examinar o *software* em si (o produto funcionando), seu código-fonte, sua documentação e os registros de execução do processo de desenvolvimento (SOMMERVILLE, 2011).

De acordo com Koscianski e Soares (2007), na atividade de revisão, uma pessoa ou uma equipe revisará um ou um conjunto de artefatos, por exemplo, o código-fonte, diagramas de banco de dados, diagramas UML e manuais, verificando sua aderência a uma série de aspectos predeterminados. A atividade de revisão pode ser aplicada às mais variadas fases do processo de desenvolvimento de *software*.

Indica-se que esses processos de revisão/inspeção sejam realizados por pessoas diferentes das responsáveis pela construção desses artefatos. Geralmente, quem foi responsável pela elaboração de um documento ou de um código-fonte terá maiores dificuldades em encontrar erros. Cada pessoa tem uma forma de trabalho, mas, com certeza, todo profissional aplica um filtro de qualidade ao concluir seu trabalho. Considerando esse fator, é mais produtivo que outra pessoa faça o trabalho de revisão/inspeção.

É importante salientar que o trabalho de revisão/inspeção deve ser realizado com impessoalidade, o que significa afirmar que o que está sendo revisado é o produto, o documento, o trabalho ou o processo, e não as pessoas que os construíram. Logicamente, sempre que um erro é encontrado, em um processo estruturado, pode-se rastrear os profissionais responsáveis. Mas a intenção consiste em relatar os problemas vinculados aos artefatos produzidos, e não ao desempenho das pessoas — a verificação do desempenho e a orientação devem constituir um processo à parte. Outro ponto importante da impessoalidade reside no fato de que eu devo apontar todas as falhas descobertas mesmo que o trabalho tenha sido realizado por um colega muito próximo, como um amigo.

Quando aplicadas sistemicamente, as técnicas de revisão representam um ótimo meio de nivelamento de conhecimento. Se executada por um profissional experiente, conhecedor dos padrões e das boas práticas do projeto, poderá orientar o trabalho daqueles que estão iniciando, transferindo de mais rapidamente toda uma bagagem obtida durante sua trajetória, fazendo com que a equipe tenha uma curva de aprendizado menor.

Para Sommerville (2011), o processo de revisão geralmente se divide em três fases (Figura 1).

Atividades pré revisão: fase de planejamento da revisão. Aqui, serão determinados os membros da equipe que realizará a revisão, além de organizados os documentos e as permissões de acesso aos artefatos que serão revisados.

Reunião de revisão: ocorrência da revisão em si, que pode ser realizada em um formato em que o autor do artefato apresenta a equipe de revisores, ou que a equipe de revisores faça isso de maneira autônoma. O responsável pela revisão deve registrar todos os comentários e ações corretivas combinados durante a revisão.

Atividades pós-revisão: aqui, a equipe responsável pela produção dos artefatos que foram revisados e tiveram problemas identificados deve agir para sanar as ocorrências. Isso pode incluir correções de *bugs* no código-fonte, alterações em documentações, diagramas, enfim, a correção do que for necessário para trazer conformidade para os artefatos revisados. Por fim, os revisores devem identificar, para uma aprovação final, se todos os pontos de ajuste foram realizados.

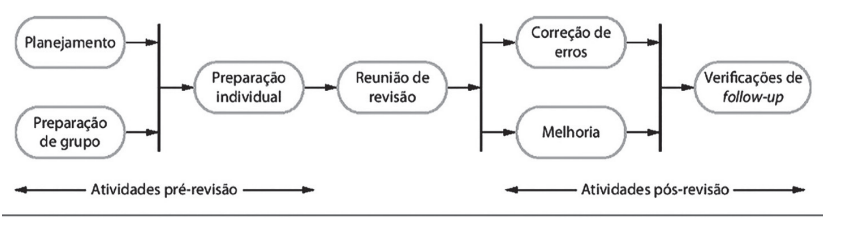

Figura 1. Processo de revisão de *software*.
Fonte: Adaptada de Sommerville (2011).

Inspeções de programa

Geralmente feitas em pares, consistem em realizar revisões minuciosas no código-fonte com o intuito de identificar possíveis erros, normalizar padrões e otimizar o código-fonte. Destacam-se os seguintes benefícios quanto a essa prática (ALONSO, 2018):

Compartilhamento de conhecimento: em equipes de desenvolvimento, é muito comum que um profissional tenha ficado mais envolvido na construção de determinada funcionalidade. A prática de *code-review* (é bem popular entre as equipes usar o termo em inglês) permite que outros membros da equipe conheçam detalhes de como tal funcionalidade foi construída, possibilitando compartilhar o conhecimento entre um maior número de pessoas.

Criação de soluções alternativas para os problemas: ao realizar um *code--review*, o revisor, em conjunto com quem construiu a funcionalidade, identifica a existência de outras formas de resolver aquele mesmo problema, talvez com menos código, de maneira mais segura ou até mesmo mais performática, o que promoverá um ganho para o projeto.

Aumento do senso de equipe: ocorre pelo fato de que, se ocorrer um problema em uma funcionalidade, todos são corresponsáveis. O código está sendo construído e revisado por todo o time. Esse sentimento é construído aos poucos, mas, quando se consolida, traz muitos benefícios para a qualidade e a responsabilidade geral sobre o produto.

Uma das maneiras de padronizar o processo de revisão de código consiste na criação de um *checklist*, o que faz com que o processo tenha sempre o mesmo nível de acurácia, independentemente da experiência do revisor. No Quadro 1, você pode verificar um exemplo de *checklist* de inspeção de código.

Quadro 1. Exemplo de um *checklist* de inspeção

Classe de defeito	Verificação de inspeção
Defeitos de dados	■ Todas as variáveis de programa são iniciadas antes que seus valores sejam usados? ■ Todas as constantes foram nomeadas? ■ O limite superior de vetores deve ser igual ao tamanho do vetor ou ao tamanho –1? ■ Se as *strings* de caracteres são usadas, um delimitador é explicitamente atribuído? ■ Existe alguma possibilidade de *overflow* de *buffer*?
Defeitos de controle	■ Para cada instrução condicional, a condição está correta? ■ É certo que cada *loop* vai terminar? ■ As declarações compostas estão posicionadas corretamente entre colchetes? ■ Em declarações *case*, todos os *cases* possíveis são considerados? ■ Se um *break* é requerido após cada *case* em declarações *case*, este foi incluído?
Defeitos de entrada e saída	■ Todas as variáveis de entrada são usadas? ■ Todas as variáveis de saída receberam um valor antes de serem emitidas? ■ Entradas inesperadas podem causar corrupção de dados?

(Continua)

(Continuação)

Classe de defeito	Verificação de inspeção
Defeitos de interface	■ Todas as chamadas de funções e métodos têm o número correto de parâmetros? ■ Os parâmetros formais e reais correspondem? ■ Os parâmetros estão na ordem correta? ■ Se os componentes acessam memória compartilhada, eles têm o mesmo modelo de estrutura de memória compartilhada?
Defeitos de gerenciamento de armazenamento	■ Se uma estrutura ligada é modificada, todas as ligações foram corretamente reatribuídas? ■ Se o armazenamento dinâmico é usado, o espaço foi alocado corretamente? ■ O espaço é explicitamente deslocado depois de não ser mais necessário?
Defeitos de gerenciamento de exceção	■ Foram levadas em consideração todas as condições possíveis de erro?

Fonte: Adaptado de Sommerville (2011).

Auditoria

Um processo de auditoria é composto pela validação da execução das atividades previstas em determinado processo, devendo ter como resultado uma documentação que sinaliza os pontos de atenção e as sugestões de melhoria detectadas pela equipe de auditoria (GALLOTTI, 2016).

De acordo com Cardoso (2015), as auditorias podem ser realizadas em nível de sistemas, processos e produtos, além do fato de que devem conter um planejamento detalhado, com suas constatações obtidas por meio de fatos, evidências objetivas e registros de processos, e não apenas na entrevista das pessoas auditadas.

Entre os termos técnicos comumente utilizados em um processo de auditoria, destacam-se (CARDOSO, 2015):

- Ação corretiva: para resolver uma não conformidade identificada;
- Ação preventiva: para resolver uma potencial não conformidade identificada;
- Alta direção: corpo de gestão da organização auditada;

- Ambiente de trabalho: condições na quais se realiza o trabalho de auditoria;
- Auditoria: o próprio processo sistemático e independente para obter e documentar a auditoria realizada;
- Auditado: organização que está sendo auditada;
- Auditor: responsável pelo processo de auditoria;
- Auditoria de primeira parte ou interna: auditoria realizada pela própria organização, com o intuito de obter a autodeclaração de conformidade;
- Auditoria de segunda parte: auditoria realizada por alguma parte interessada envolvida no processo auditado (p. ex., cliente, fornecedor);
- Auditoria de terceira parte: realizada por uma organização independente.

A auditoria se dá em três partes: planejamento, auditoria e finalização. Na primeira, define-se como o processo de auditoria funcionará, desde os documentos que serão utilizados até como as pessoas que serão envolvidas, em que serão definidos os limites da auditoria: o escopo é essencial para que o processo de auditoria tenha um início, um meio e um fim. Na fase de auditoria, haverá a execução da auditoria em si, em que serão entrevistados os auditados, que devem ser sempre apresentados aos objetivos da auditoria. Os documentos que detalham o processo serão analisados, identificando-se sua aderência aos processos. Na finalização da auditoria, serão apresentados os resultados — as boas práticas, as não conformidades e oportunidades de melhoria com as ações corretivas e preventivas (GALLOTTI, 2016).

Saiba mais

Veja, no Guia de revisão de código do desenvolvedor do Google, disponível no *link* a seguir, quais práticas uma das maiores empresas de *software* do mundo utiliza para realizar suas *code-reviews*.

https://qrgo.page.link/r9drJ

Normas IEEE vinculadas a verificação, validação e inspeção de *software* (IEEE 1012:2016 e IEEE 1028:2008)

O Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE) é a maior organização profissional técnica do mundo, cujas atividades dedicam-se ao avanço da tecnologia para benefício da humanidade (IEEE, 2020). Por essas caracte-

rísticas, entre seus membros estão envolvidos, há muito tempo, engenheiros, cientistas e outros profissionais correlatos, rol no qual se incluem cientistas da computação, desenvolvedores de *software*, profissionais de tecnologia da informação, físicos, médicos, etc., além do núcleo de engenharia elétrica e eletrônica do IEEE. Por isso, a organização não usa mais o nome completo, exceto quando é necessário por questões legais e formais, sendo amplamente chamada apenas de IEEE.

Entre os diversos padrões mantidos pelo IEEE, estão o IEEE 1012:2016, um padrão para verificação e validação de sistema, *software* e hardware, e o IEEE 1028:2008, um padrão para análises e auditorias de *software*.

IEEE 1028:2008

Seus objetivos consistem em:

- definir uma estrutura comum dos processos, atividades e tarefas de verificação e validação em apoio a todo o sistema, processos de ciclo de vida de *software* e *hardware*;
- definir as tarefas de verificação e validação, as entradas e saídas necessárias em cada processo do ciclo de vida;
- identificar as tarefas mínimas de verificação e validação que permitirão a utilização de um esquema de integridade de quatro níveis;
- definir o conteúdo do plano de verificação e validação.

Conforme a IEEE 1028:2008, os processos de verificação e validação entregam uma avaliação objetiva de produtos e processos ao longo do ciclo de vida, a qual demonstra se os requisitos são corretos, completos, precisos, consistentes e testáveis.

Os processos de verificação e validação determinam se os produtos de determinada atividade estão em conformidade com os requisitos desta e se o produto satisfaz ao uso pretendido e às necessidades do usuário. Tal determinação inclui avaliação, análise, avaliação, revisão, inspeção e teste de produtos e processos. As atividades de verificação e validação são realizadas em paralelo a todas as etapas do ciclo de vida, e não em sua conclusão.

Os resultados da verificação e validação promovem os seguintes benefícios:

- facilitam a detecção e a correção precoces de anomalias;
- aprimoram a percepção de gerenciamento dos riscos do processo e do produto;

- apoiam os processos do ciclo de vida para garantir a conformidade com o desempenho do programa, o cronograma e as despesas;
- fornecem uma avaliação inicial do desempenho;
- fornecem evidências objetivas de conformidade para apoiar um processo formal de certificação;
- melhoram os produtos desde os processos de aquisição, fornecimento, desenvolvimento e manutenção;
- apoiam atividades de melhoria de processos.

A norma IEEE 1012:2016 abrange o ciclo de vida completo de um *software*, conforme você pode identificar no Quadro 2, que resume as tarefas por ela previstas.

Quadro 2. Principais atividades de verificação e validação previstas na IEEE 1012:2016

Atividade	Tarefa
Gerência de *software*	■ Planejamento ■ Monitoramento ■ Avaliação dos resultados do monitoramento e verificação dos impactos ■ Relatórios gerenciais
Requisitos de *software*	■ Elaboração da documentação de especificação de requisitos de *software* ■ Análise de impactos ■ Análise de interface ■ Planejamento dos testes de sistemas
Projeto do *software*	■ Análise de impacto ■ Desenvolvimento do projeto de *software* ■ Análise da interface ■ Testes de unidade ■ Testes de integração
Codificação	■ Desenvolvimento do código ■ Análise da interface ■ Complementação dos testes de unidade
Testes de unidade	■ Execução dos testes de unidade ■ Registros dos testes executados
Testes de integração	■ Execução dos testes de integração ■ Registro dos testes de integração
Testes de sistema	■ Execução dos testes de sistema ■ Registro dos testes de sistema

Fonte: Adaptado de Gallotti (2016).

IEEE 1028:2008

Trata-se de um padrão que fornece requisitos mínimos aceitáveis para revisões sistemáticas de *software*, incluindo os seguintes atributos:

- participação da equipe;
- resultados documentados da revisão;
- procedimentos documentados para conduzir a revisão.

O IEEE 1028:2008 apresenta cinco tipos de revisão de *software*, com seus respectivos procedimentos necessários para execução:

- revisões gerenciais;
- revisões técnicas;
- inspeções;
- *walk-throughs* (ensaios de uso dos artefatos do *software*);
- auditorias.

Essa norma se preocupa apenas com as revisões e auditorias, ou seja, não apresenta procedimentos para determinar a necessidade de uma revisão ou auditoria, nem especifica a disposição dos resultados da revisão ou auditoria. No Quadro 3, são apresentadas as principais características desses tipos de revisão.

Quadro 3. Quadro comparativo das características dos tipos de inspeções da IEEE 1028:2008

Característica	Revisão gerencial	Revisão técnica	Inspeção	*Walk-throughs*	Auditoria
Objetivo	Garantir o progresso, recomendar ações corretivas e garantir alocação correta dos recursos	Avaliar a conformidade do estado atual com as especificações e planos, garantir integridade da mudança	Encontrar anomalias, verificar decisões e verificar a qualidade do produto	Encontrar anomalias, examinar alternativas e melhorar o produto, fórum para aprendizado	Avaliação independente de cumprimento com os objetivos de padrões e regulamentos

(Continua)

(Continuação)

Característica	Revisão gerencial	Revisão técnica	Inspeção	*Walk-throughs*	Auditoria
Tomada de decisão	A equipe de gerenciamento traça o curso da ação, decisões são feitas na reunião ou como resultado das recomendações	A equipe de revisão solicita aos gerentes ou a liderança técnica que atuem nas recomendações	A equipe de revisão escolhe as disposições predefinidas do produto, os defeitos devem ser removidos	A equipe concorda com as mudanças para serem feitas pelo autor	Organização auditada, iniciador, comprador, cliente ou usuário
Verificação das mudanças	O líder verifica que itens são fechados, a verificação das mudanças é deixada para outros controles do projeto	O líder verifica que itens são fechados, a verificação das mudanças é deixada para outros controles do projeto	O líder verifica que itens são fechados, a verificação das mudanças é deixada para outros controles do projeto	O líder verifica que itens são fechados, a verificação das mudanças é deixada para outros controles do projeto	Responsabilidade da organização auditada
Tamanho recomendado do grupo	Duas ou mais pessoas	Três ou mais pessoas	Três a seis pessoas	Duas a sete pessoas	Uma a sete pessoas
Quem participa	Gerentes, liderança técnica e algumas pessoas de outras áreas	Liderança técnica e algumas pessoas de outras áreas	Pessoas da área com acompanhamento documentado	Liderança técnica e algumas pessoas de outras áreas	Auditores, organização auditada, pessoal de gerência e técnico
Grupo da liderança	Normalmente, o gerente responsável	Normalmente, o engenheiro líder	Um facilitador treinado	O facilitador ou o autor	O auditor líder
Volume de materiais	Moderado para muito, dependendo dos objetivos da reunião	Moderado para muito, dependendo dos objetivos da reunião	Relativamente baixo	Relativamente baixo	Moderado para muito, dependendo dos objetivos da reunião

(Continua)

(Continuação)

Característica	Revisão gerencial	Revisão técnica	Inspeção	*Walk-throughs*	Auditoria
Apresentador	O líder da revisão determina os apresentadores	O líder da revisão determina os apresentadores	Um leitor	Autor	Auditores coletam e examinam a informação fornecida pela organização auditada
Coleta de Dados	Políticas, padrões ou planos	Não há requisitos formais	Obrigatória	Recomendada	Não há requisitos formais
Resultado	Documentação da revisão gerencial, incluindo plano de ação com responsabilidades e datas para resolução	Documentação da revisão técnica, incluindo plano de ação com responsabilidades e datas para resolução	Lista e resumo de anomalias, documentação da inspeção	Lista de anomalias, itens de ação, decisões, propostas de acompanhamento	Relatório de auditoria formal, observação, descobertas e deficiências

Fonte: Adaptado de IEEE (2008).

Como verificamos neste capítulo, existem muitas técnicas e padrões para aplicar qualidade aos projetos de *software* que conduzimos. E, embora às vezes isso possa parecer um pouco distante para as equipes que atualmente não têm práticas visando à qualidade, o grande segredo consiste em realizar uma implantação gradual. Ao escolher um processo, uma ferramenta ou um método por vez, aplicar, colher os resultados e estabelecer um padrão na equipe, ao longo do tempo, várias iniciativas já estarão sendo executadas, trazendo resultados para o processo e, consequentemente, para o produto.

Exemplo

Pela questão da transparência, é muito comum realizar auditorias em órgãos públicos.

No *link* a seguir, você pode verificar o relatório de resultado da auditoria ocorrida na Diretoria de Tecnologia da Informação do Instituto Federal de Rondônia. Observe o detalhamento e a quantidade de informação que podem ser obtidos por meio de um relatório como este.

https://qrgo.page.link/gmTme

Referências

ALONSO, V. *Qual o real valor do code review para uma equipe de desenvolvimento?* 2018. Disponível em: https://imasters.com.br/desenvolvimento/qual-o-real-valor-code-review-para-uma-equipe-de-desenvolvimento. Acesso em: 25 nov. 2020.

CARDOSO, A. *Auditoria de sistema de gestão integrada*. São Paulo: Pearson Education do Brasil, 2015.

GALLOTTI, G. M. A. *Qualidade de software*. São Paulo: Pearson Education do Brasil, 2016.

IEEE. *IEEE 1028-2008*. IEEE standard for software reviews and audits. New York: IEEE, 2008.

IEEE. *IEEE 1012-2016*. IEEE standard for system, software, and hardware verification and validation. New York: IEEE, 2016.

IEEE. *Mission & vision*. New York: IEEE, 2020. Disponível em: https://www.ieee.org/about/vision-mission.html. Acesso em: 25 nov. 2020.

KOSCIANSKI, A.; SOARES, M. S. *Qualidade de software*: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software. São Paulo: Novatec, 2007.

PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de software*: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.

SCHACH, S. R. *Engenharia de software*: os paradigmas clássico e orientado a objetos. 7. ed. São Paulo: McGraw-Hill, 2009.

SOMMERVILLE, I. *Engenharia de software*. 9. ed. São Paulo: Pearson Prentice Hall, 2011

Leituras recomendadas

BRASIL. Ministério da Educação. *Relatório de auditoria nº 003/2018*. Por Velho: MEC, 2018. Disponível em: https://qrgo.page.link/gmTme. Acesso em: 25 nov. 2020.

GOOGLE. *Google's engineering practices documentation*. 2020. Disponível em: https://google.github.io/eng-practices/review/. Acesso em: 25 nov. 2020.

MELO, A. A. *Preciso validar meu software*. Por onde começo? 2018. Disponível em: https://www.harbor.com.br/harbor-blog/2018/03/15/validacao-de-software/. Acesso em: 25 nov. 2020.

Fique atento

Os *links* para *sites* da *web* fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declaram não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS