



APRENDIZAGEM EM FOCO

VALIDAÇÃO DO SOFTWARE: TESTES DE SOFTWARES E APLICAÇÕES DE SEGURANÇA NO SISTEMA



APRESENTAÇÃO DA DISCIPLINA

Autoria: Luís Otávio Toledo Perin

Leitura crítica: Rafael Maltempe da Vanso

Olá, como é bom tê-lo conosco! Você já deve ter utilizado um software, seja ele de qualquer área, mas será que já parou para pensar como é a sua concepção e construção? Ou, ainda, como é realizada a validação de tudo aquilo que foi construído, passando pelo processo de qualidade e pelos testes de software? Pois bem, este é o momento ímpar para aprofundar seus conhecimentos nesta área da engenharia de software, a qual tem por objetivo a busca por um contínuo processo de melhoria em todas as fases do software.

São quatro conteúdos programáticos fascinantes, em que cada um abordará temas de relevância para nossa era tecnológica. No primeiro, vamos ver a qualidade do software, sua documentação de testes e o modelo TMMi, em que será compreendida a importância da aplicação de testes bem realizados, como uma das garantias da qualidade do software.

No segundo conteúdo, verificaremos o gerenciamento dos testes de software com ferramentas CASE *open source*, em que sua utilização contribuirá para controlar aplicações de testes, fazendo com que fique mais organizado e preciso.

Para o terceiro conteúdo, serão analisados os procedimentos para garantir a segurança do software, com foco na aplicação de procedimentos de segurança em projetos de software.

Por fim, o quarto conteúdo contempla as ferramentas CASE para testes de segurança do software, em que relatamos que conhecer

a ferramenta é de extrema importância para garantir a segurança do produto, bem como do usuário e da organização que está fazendo uso.

Viu como tudo é interessante e atual? Então não perca por esperar, porque tudo foi desenvolvido pensando em você, aluno! Também não posso deixar de mencionar o formato dinâmico no qual a disciplina foi estruturada, com leituras digitais, aulas gravadas com slides, dicas de leituras, enfim, uma gama de objetos de aprendizagem bem diferenciados que tornarão seus estudos cada vez mais agradáveis. Seja muito bem-vindo à nossa disciplina!

INTRODUÇÃO

Olá, aluno (a)! A *Aprendizagem em Foco* visa destacar, de maneira direta e assertiva, os principais conceitos inerentes à temática abordada na disciplina. Além disso, também pretende provocar reflexões que estimulem a aplicação da teoria na prática profissional. Vem conosco!

TEMA 1

Qualidade do software, documentação de testes e o modelo TMMi

Autoria: Luís Otávio Toledo Perin

Leitura crítica: Rafael Maltempe da Vanso






DIRETO AO PONTO

Olá! Vamos relembrar os conceitos principais apresentados a partir deste *Direto ao Ponto*? Quando pensamos em tecnologia ou situações relacionadas a ela, logo nos vêm à mente alguns questionamentos, como: “Qual o objetivo deste programa?”, “Ele vai apresentar erros durante a sua execução?”; ou ainda: “Posso confiar neste software? Questões dessa magnitude tendem a colocar em dúvida a funcionalidade do sistema ou aplicativo criado, fazendo com que haja uma desconfiança primária sobre ele. Tal insegurança, a qual gera essas dúvidas, pode vir de uma experiência não tão boa que o usuário teve com um sistema do mesmo segmento no passado.

Neste aspecto, o papel da qualidade do software torna-se de suma importância! Ao assegurar com métricas e mecanismos já reconhecidos, inclusive internacionalmente, o produto tende a possuir maior nível de qualidade, fazendo com que reclamações e queixas diminuam ou até deixem de existir! Segundo Pressman (2006), há uma conexão entre o grau de importância da aplicação e suas exigências com o papel que a engenharia de software representa, já que são necessárias técnicas e abordagens à altura daquilo que se quer produzir, fazendo com que a qualidade se equipare ao mesmo nível de suas cobranças. Com isso, devemos entender que o fato de algo possuir qualidade não significa que a perfeição ali foi alcançada, mas que há uma busca contínua do processo de qualidade, alinhado ao bom planejamento e à fase de testes em todo o ciclo de desenvolvimento, garantindo maior clareza e rapidez quando as falhas forem encontradas.

Quanto ao planejamento de qualidade de software, ele deve possuir uma garantia mínima de execução, ou seja, deverá assegurar que o básico foi analisado e que o software está




funcionando. É importante entendermos alguns pontos antes do planejamento propriamente dito, os quais envolvem aquilo que foi definido como qualidade com o cliente, chamamos de política de qualidade. Depois, é de suma importância conhecer e entender o escopo do projeto, ou seja, quais as necessidades e os problemas que o sistema deve resolver para aquilo que foi planejado. Compreender os processos, procedimentos e padrões também é relevante para o bom desenvolvimento, já que tornam a aplicação mais próxima da realidade. Por fim, mas não menos importante, conhecer as especificações do produto transparece segurança e conhecimento daquilo que se quer implantar, deste modo, falando em termos claros, conhecer o que o sistema faz e o que ele não faz é essencial, principalmente para a qualidade.

Mas apenas planejar a qualidade não é suficiente, devemos garantir que ela ocorra de maneira satisfatória e ininterrupta. Ela tem por objetivo assegurar que o software possua os padrões básicos de qualidade, sendo segurança, eficiência, confiabilidade, integridade e até a usabilidade. Todos esses fatores, juntamente com algumas atividades a serem desenvolvidas, atestam a garantia do produto.

Agregada à qualidade, e muito importante para a rastreabilidade e estatísticas de erros, a documentação dos testes é extremamente útil. É com essa referência que os ajustes poderão ser efetuados, já que o software passa por testes, os quais são documentados e, posteriormente, seus resultados analisados pelo profissional adequado, neste caso, o analista de testes. Mas para que isso seja possível, técnicas e ferramentas devem ser utilizadas, garantindo, assim, uma padronização nas atividades desenvolvidas.

Neste contexto, temos uma organização sem fins lucrativos, o IEEE – Institute of Electrical and Electronic Engineers, que define diversos padrões e práticas presentes na engenharia de software,



os quais contribuem significativamente, principalmente para a documentação em testes de software. Eles partem de unitários até de aceitação, além de incluir documentos consistentes e adequados para definir, registrar e prover condições de análise dos resultados obtidos ao longo do processo.

Por fim, mas não menos importante e ainda referenciando o assunto qualidade, os modelos de apoio à gestão têm sido de grande valia para o alto grau de qualidade daquilo que se desenvolve, já que são essas especificações que tornam o ciclo de desenvolvimento alinhado a padrões nacionais e internacionais de excelência de qualidade. Um exemplo disso, é o TMMi Foundation, o qual desenvolveu o TMMi (Test Maturity Model and Integration), que vem para somar com o CMMi, que nasceu para implementação e melhoria dos testes de software.

Este modelo baseia-se em classificações e níveis, ou seja, a organização vai se classificando e evoluindo dentro da atividade avaliada, podendo evoluir conforme o cumprimento de processos para atingir determinado nível de maturidade. Com a implantação do TMMi, o número de defeitos e falhas tende a diminuir antes, durante e após o ciclo de desenvolvimento, já que gera um produto com melhor qualidade.

Procure compreender como a atualidade exige de maneira deliberada inúmeras atividades de uma mesma pessoa. Neste contexto, os sistemas vêm para realizar esse gerenciamento, desde que possam ser utilizados. Por isso, o papel da qualidade, manutenção e dos testes de software se torna essencialmente indispensável para que o produto possa ser utilizado com alto grau de satisfação. Observe na Figura 1, em que a construção de um sistema informatizado é pensado minuciosamente, além de ser composto por uma equipe, a qual passou por treinamento e segue um modelo de gestão e execução.

Figura 1 – Construção de sistema em equipe



Fonte: aurielaki/iStock.com.

Referências bibliográficas

PRESSMAN, R. S. **Engenharia de software**: uma abordagem profissional. Tradução João Eduardo Nóbrega Tortello. Revisão técnica Reginaldo Arakaki, Julio Arakaki, Renato Manzan de Andrade. 8. ed. Porto Alegre: AMGH, 2016.

▶ PARA SABER MAIS

Caso tenha lido o seu conteúdo programático com mais atenção, perceberá que, neste momento, o assunto qualidade de software tem sido bastante trabalhado e exemplificado. Isso porque todo o processo que fazemos, neste caso com a tecnologia, merece ao menos um nível mínimo de qualidade, garantindo, assim, que alguns padrões sejam seguidos e que o usuário possa sair satisfeito com o produto.

Mas por que ficar apenas no básico quando se pode ultrapassar esta barreira? O básico exigido serve apenas como uma linha

niveladora, fazendo com que o mercado ofereça soluções que atendam, em parte, o cliente almejado.

Após a crise de software, ocorrida em meados dos anos 1980, novas ideias e padrões foram surgindo, fazendo com que as metodologias e métricas que hoje existem, partissem da ideia de que um sistema deve possuir qualidade suficiente para ser ofertado a seu usuário. Neste aspecto, alguns padrões são indispensáveis, como o CMMi (Capability Maturity Model and Integration) e MPS.BR (Melhoria de Processo do Software Brasileiro), além do TMMi Foundation, que desenvolveu o TMMi (Test Maturity Model and Integration), os quais ofertam inúmeros recursos e exigem a melhoria contínua daquilo que se está produzindo e será ofertado.

Portanto, procurar fazer sempre o melhor e buscar por novos padrões, métricas e ferramentas de trabalho faz com que aquilo que se está desenvolvendo torne-se algo melhor e mais bem preparado frente aos concorrentes, agregando, desta maneira, valor ao produto.

Referências bibliográficas

VASCONCELOS, A. M. L. *de et al.* **Introdução à engenharia de software e à qualidade de software**. 2006. 157 f. (Curso de Pós-graduação “Lato Sensu” (Especialização) à Distância – Melhoria de Processo de Software. Lavras: UFLA/FAEPE, 2006.

TEORIA EM PRÁTICA

Reflita sobre a seguinte situação: imagine que você possua formação na área de tecnologia, mais especificamente em desenvolvimento de software, além de estar cursando uma

especialização em engenharia de software. Sabendo que você é um profissional dedicado e atento às novas tecnologias e os processos de melhorias, o corporativo da empresa na qual trabalha convidou-o a integrar o time de desenvolvimento de software da empresa. Após receber as boas-vindas dos demais colegas e o treinamento de como o setor é estruturado, chegou a sua vez de fazer a coisa acontecer! Você é um programador e deve criar soluções. Após alguns dias no setor, você se depara com muitos casos de retorno de problemas já solucionados, alguns por você mesmo e outros pelos demais integrantes da equipe. Ao analisar cada situação, percebe que a falta de um modelo de métrica e técnicas a ser seguido, e também de testes de software, estava ocasionando o problema. Analisando o ocorrido, como você se comportaria com tal problemática? Como pós-graduando, qual deverá ser a sua atitude? Qual a melhor técnica/métrica a ser sugerida para a gerência?

Para conhecer a resolução comentada proposta pelo professor, acesse a videoaula deste *Teoria em Prática* no ambiente de aprendizagem.




LEITURA FUNDAMENTAL

Indicações de leitura

Indicação 1

A respeito da importância da qualidade de software, apresentamos o livro *Qualidade de software*, do autor Giocondo Marino Antônio Gallotti, que retrata assuntos da atualidade, bem como questões éticas, as quais devem ser levadas em consideração para a criação do software. Também trata de valores



e custos para a criação de um sistema, em que a qualidade está diretamente relacionada, já que, com o retrabalho, os custos aumentam de maneira significativa. Por fim, discorre sobre modelos e característica de qualidade, citando modelos e métricas já consagrados no mercado nacional e internacional.

Para realizar a leitura, acesse a plataforma Pearson / Biblioteca Virtual 3.0, na Biblioteca Virtual da Kroton, e busque pelo título da obra.


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

Indicação 2

O livro *Teste de software*, de Pedro Henrique Braga, é bem didático e prático quanto às técnicas e métricas a serem seguidas para se trabalhar com testes de software. Entender o processo e sua importância, assim como sua influência, é de vital importância para a concepção do produto. Como indicação, a unidade 1, “Introdução ao teste de software”, é bem clara quanto à necessidade de testes e quais são os tipos existentes. Também traz questões importantes, como as principais causas de defeito e o papel do analista nesse meio.

Para realizar a leitura, acesse a plataforma Pearson / Biblioteca Virtual 3.0, na Biblioteca Virtual da Kroton, e busque pelo título da obra.

BRAGA, P. H. C. **Teste de software**. São Paulo: Pearson Education do Brasil, 2016.



QUIZ

Prezado aluno, as questões do Quiz têm como propósito a verificação de leitura dos itens *Direto ao Ponto, Para Saber Mais, Teoria em Prática e Leitura Fundamental*, presentes neste *Aprendizagem em Foco*.

Para as avaliações virtuais e presenciais, as questões serão elaboradas a partir de todos os itens do *Aprendizagem em Foco* e dos slides usados para a gravação das videoaulas, além de questões de interpretação com embasamento no cabeçalho da questão.

1. Sabemos que garantir a qualidade do produto é muito importante por diversas situações. O principal objetivo da garantia da qualidade é assegurar que _____, procedimentos e métricas sejam utilizados durante todo o ciclo de desenvolvimento do software, devendo prover o nível de confiança requerido na entrega do produto.
Complete a frase com a alternativa correta:
 - a. Padrões.
 - b. Situações.
 - c. Sistemas.
 - d. Problemas.
 - e. Dúvidas.

2. Sabemos que, para a melhor execução de um conjunto de testes, estes são fracionados em testes menores, ou então divididos. Assinale a alternativa que explique por que isso é necessário:

- a. Cada fase do teste tem um propósito diferente.
- b. Pois podem ser executados testes diferentes em ambientes diferentes.
- c. Testes em estágio são considerados mais fáceis de se lidar.
- d. Quanto maior a divisão do teste, melhor ele é.
- e. A divisão pouco importa, bastando apenas executar o teste.



GABARITO

Questão 1 - Resposta A

Resolução: Fica evidente que, para o cumprimento do modelo desejado, bem como a efetivação daquilo que se quer construir com qualidade, padrões devem ser seguidos.

Questão 2 - Resposta C

Resolução: Os testes são divididos em estágios ou separados para que o tratamento e a própria execução aconteçam melhor.

TEMA 2

Gerenciamento dos testes de software com ferramentas CASE open source

Autoria: Luís Otávio Toledo Perin

Leitura crítica: Rafael Maltempe da Vanso






DIRETO AO PONTO

Olá, vamos relembrar os principais conceitos sobre Teste de Software? Sabemos que eles estão diretamente relacionados a qualidade, seja ela antes, durante ou depois da concepção do produto. Mas o fato de realizar testes em um sistema sempre gera alguns questionamentos, como: “o que devo observar?”, “como devo testar?” ou “o que é considerado certo ou errado?” Perguntas como essas são importantes para nos nortear quanto a execução desta atividade, bem como refletir sobre o comportamento do sistema perante seus requisitos.

Agora, te faço uma pergunta caro aluno: qual a utilidade do teste de software? Se esse teste for executado apenas com o intuito de simplesmente testar, ou seja, ver como algo se comporta de maneira superficial, não há utilidade nenhuma, pois os resultados obtidos não serviram de padrão para a melhoria dele. Agora, se esse teste for executado com métricas e padrões já existentes no mercado, seu resultado será convertido em melhoria contínua do produto, haja vista que saber o que está fazendo é de suma importância para o sucesso da atividade.

Com o intuito de minimizar tais chances de insucesso, práticas de VV&T (verificação, validação e teste) são executadas durante todo o ciclo de desenvolvimento do *software*, garantindo assim que a menor quantidade de falhas possam ocorrer. Neste aspecto, a garantia da qualidade é forte aliada para a definição de procedimentos, processos e padrões, antes ou depois do desenvolvimento do sistema (Sommerville, 2018).

Devemos atentar-se também para o cronograma de entrega do produto, pois é um dos fatores que pode afetar a qualidade, já que é considerado um dos procedimentos mais caros dentro do




ciclo de desenvolvimento de software, e por esse motivo, suprimir algumas fases ou validações não torna-se atraente para acelerar a entrega para o usuário final, haja vista que falhas ou problemas que poderiam ser evitados, correm o risco de tornarem-se aparentes durante sua utilização.

O ato de testar ainda continua sendo o principal aliado para a descoberta de problemas dentro de uma aplicação, já que ele tem por finalidade averiguar se aquilo que foi projetado para ser feito realmente está de acordo, além de descobrir possíveis defeitos antes do usuário, garantindo assim o entendimento e efetivação do processo. Durante essa atividade, dados fictícios são inseridos no *software*, e os resultados obtidos são utilizados como parâmetros em busca de anormalidades.

Ao falarmos em problemas encontrados durante uma bateria de testes, logo nos vem à mente palavras como defeito, erro ou falha. Mas você sabe a diferença entre cada uma? Se não sabe, vou explicar aqui cada uma. Segundo o International Software Testing Qualifications Board (ISTQB), temos o erro como falha humana, o qual produz um resultado incorreto, podendo ser um erro de sintaxe, por exemplo. O defeito, por sua vez é resultado de um código mal escrito, causando mau funcionamento no sistema, conhecido também como bug. Por fim, a falha é resultado de um código que apresenta defeito e foi executado, causando funcionamento incoerente das funções do software, sendo o usuário diretamente atingido, já que é visível ao mesmo.

Agora que já está ciente de cada definição, e já entendeu a importância em porque testar um sistema, precisa também conhecer quais são os tipos de testes e onde cada um é aplicado. Há uma grande divisão nos testes, que se liga a relação temporal, ou seja, o momento mais adequado para determinado teste ser executado (Crespo et al. ,2004). Sabemos que todos são




importantes, mas a sua utilização e o momento ideal para que isso ocorra, deverá ser determinado pelo analista, o qual tem conhecimento técnico para a tomada de decisão.

Entre todos os testes, selecionei os principais tipos, sendo eles: unitário, que é o mais básico de todos e tem por objetivo avaliar cada módulo do sistema de maneira individual; de integração, que tem o foco em avaliar se a comunicação entre os módulos da aplicação é feita do modo certo; de sistema, que visa, após junção dos módulos, ter o sistema por completo e simular o ambiente de uso final; de aceitação, que avalia a aceitação do público com aquilo que o programa executa; de regressão, que é uma espécie de nova verificação, mas com uma nova versão do sistema, isso para assegurar que as falhas foram corrigidas e que não afetaram outras partes; de integridade de dados, que tem foco na confiabilidade e integridade dos dados do software; de configuração e instalação, que pretendem avaliar a capacidade do software em funcionar corretamente em diversas plataformas; de performance, que é destinado a avaliar o nível de excelência do produto desenvolvido e os não funcionais, que estão relacionados aos requisitos não funcionais do sistema.

Alinhado as boas práticas de teste, a escolha pelo melhor tipo e no melhor momento, é importante a utilização de uma ferramenta que faça esse controle e gerenciamento, já que a possibilidade de falhas durante o processo passa a ser menor, além de gerar relatórios e integrar-se a outras plataformas, garantindo assim a unificação das informações.

Outro fator importante para a utilização de ferramentas e padronizações de práticas no ambiente de teste de software diz



respeito ao custo, já que elas tendem a reduzir o valor do processo como um todo (Maldonado et al, 1998). Por isso, ferramentas do tipo “*open source*” tornasse atraentes, já que possui seu código livre para modificações e leitura do público em geral. Dentre todas, podemos citar as mais famosas, sendo elas: Rth, TestLink e TestMaster.

A primeira atua sob licença GNU (*General Public License*) e é construída em linguagem PHP, sendo baseada na web. Ela permite a criação de casos de testes, sua execução e monitoramento do progresso dos testes, além de importar os resultados obtidos para arquivos com extensão *xlsx*. A segunda é mantida pela *TestLink Community* e desenvolvida em linguagem web PHP, também baseada na web. É possível gerenciar projetos, criar e executar casos e planos de teste, além de uma bateria de teste e definir se um determinado teste foi executado corretamente ou se houve falha. Por fim, a terceira é destinada a gestão de testes de software e permite a criação e execução de casos de teste, possuindo funções de automação de teste, além da geração de relatórios dos casos testados. Ela é baseada na web e foi escrita em PERL, sendo *open source* e utilizando o Apache como servidor web.

Entenda que tudo está interligado e conectado, ou seja, no universo do ciclo de desenvolvimento de software não basta apenas uma ou outra parte estarem corretas, mas sim todas as etapas estarem de acordo para que o sucesso possa ser alcançado. Observe na figura 1, ela ilustra o que foi dito acima. Similar a uma engrenagem, onde as peças devem estar rigorosamente posicionadas, assim acontece com processos como planejamento, requisitos, desenvolvimento de software, teste e outros. Tudo deve estar alinhado para que o produto contenha todos os elementos necessários para possuir a qualidade desejada.

Figura 1: Alinhamento de processos



Fonte: solidcolours/iStock.com.

Referências bibliográficas

CRESPO, A. N. et al. **Uma metodologia para teste de software no contexto da melhoria de processo**. Campinas, SP: CenPRA – Centro de Pesquisar Renato Archer, DMPS – Divisão de Melhoria de Processos de Software, 2004. Disponível em: <https://www.researchgate.net/publication/237497188_Uma_Metodologia_para_Teste_de_Software_no_Contexto_da_Melhoria_de_Processo>. Acesso em: 08 Agos. 2020.

MALDONADO, J.C. et al. **Aspectos teóricos e empíricos de teste de cobertura de software**. VI Escola de Informática da SBC - Regional Sul, 1998.

SOMMERVILLE, Ian. **Engenharia de Software**; tradução Luiz Cláudio Queiroz; revisão técnica Fábio Levy Siqueira. 10. ed. São Paulo: Pearson Education do Brasil, 2018.



PARA SABER MAIS

Caso tenha lido o seu conteúdo programático com mais atenção, irá perceber que neste momento o foco é a gerência dos testes de software. Neste aspecto, e considerando a importância em realizar o processo de avaliação e maturação do software desenvolvido, é importante que técnicas sejam desenvolvidas, principalmente quando práticas se tornam corriqueiras e são executadas repetitivamente.

Deste modo, pensar em automatizar os testes é outro fator que irá agregar na qualidade do produto, já que uma mesma atividade desenvolvida inúmeras vezes por um ser humano pode ser suscetível a erros, mas quando esse mesmo processo é efetuado por “robô” torna-se mais preciso e rápido, fazendo com que haja credibilidade e aperfeiçoamento daquilo que se quer testar.

Outro motivo para automatizar diz respeito ao tempo, mas não o da execução do teste, e sim do tempo que poderia ser empregado em outra atividade caso o procedimento já fosse automático. Com essa prática o testador ou o analista ficaria livre para pensar em outras situações, ou até na criação de novos casos de teste, por exemplo.

Portanto, após o processo de teste estar maduro, é de suma importância cogitar a automatização dos mesmos, haja vista que além de tornar-se um processo mais seguro e robusto, o tempo empregado para o teste manual poderia ser atribuído à outra atividade, garantindo assim o dinamismo do processo.

Referências bibliográficas

PRESSMAN, Roger S. **Engenharia de Software: uma abordagem profissional**; tradução João Eduardo Nóbrega Tortello; revisão técnica Reginaldo Arakaki, Julio Arakaki, Renato Manzan de Andradel. 8. ed. Porto Alegre: AMGH, 2016.



TEORIA EM PRÁTICA

Refleta sobre a seguinte situação: Você é recém formado e enviou diversos currículos para diversas empresas de TI de sua cidade. Um belo dia é chamado para uma entrevista e nela te ofertam a vaga de testador de software, claro, como júnior, mas já é um começo. Apesar de almejar ser programador, resolve aceitá-la, já que não tem nada a perder e ainda por cima exercerá a função na área. Pois bem, após o treinamento dos sistemas desenvolvidos pela empresa e a apresentação da equipe que irá trabalhar, suas atividades começam. Logo no início você observa algumas situações que fogem das práticas e métricas que aprendeu em sua graduação, como testes sendo realizados sem um planejamento prévio, ou registros e resultados deles sendo marcados em documento físico, neste caso papel A4. Diante da situação relatada acima, qual seria a sua atitude? Iria permanecer com tais métricas e práticas não recomendáveis ou iria solicitar uma reunião e expor um plano de melhoria do processo?

Para conhecer a resolução comentada proposta pelo professor, acesse a videoaula deste *Teoria em Prática* no ambiente de aprendizagem.

LEITURA FUNDAMENTAL

Indicações de leitura

Indicação 1

Uma superleitura que complementar seu entendimento sobre ferramentas *open source* para teste de software é o artigo “Utilização da ferramenta Mantis para gerenciamento de defeitos

de testes de software”, dos autores Marcos Flávio de Souza Reis *et al.*, que pode ser encontrado no site do Inpe ou por meio do Google Acadêmico.

Em seu conteúdo, o artigo pretende apresentar uma alternativa para pequenas e grandes empresas quando o assunto é gerenciamento de defeitos de seus produtos, por meio de ferramentas gratuitas, além de trazer experiências vivenciadas em testes funcionais durante os testes realizados para a produção do artigo. Vale a pena ler!

REIS, M. F. S. *et al.* **Utilização da ferramenta Mantis para gerenciamento de defeitos de testes de software**. INPE. Dissertação (Mestrado em Engenharia e Tecnologia Espaciais/ Gerenciamento de Sistemas Espaciais) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2012.

Indicação 2

A indicação é de grande valia para despertar o instinto curiosidade, ou seja, para ir além do que foi apresentado. Neste quesito, o artigo denominado “Test-Module: uma ferramenta para gerenciamento de testes de software integrada ao FireScrum”, de Audrey B. Vasconcelos *et al.*, cumpre o seu papel, podendo ser obtido por meio do Google Acadêmico.

Nesta leitura, serão abordados temas como o teste de software com ferramentas gratuitas, além da integração com a prática Scrum. O autor ainda ressalva a importância do desenvolvimento de aplicação com a prática de testes em paralelo, argumentando que o mercado está cada vez mais exigente. Espero que goste da indicação!

AMORIM D. G. *et al.* **Gerenciamento de Teste de Software:** Um Comparativo entre Ferramentas Open Source Revista Gestão.Org, Recife, v. 14, Edição Especial, 2016. p. 296-302.

QUIZ

Prezado aluno, as questões do Quiz têm como propósito a verificação de leitura dos itens *Direto ao Ponto, Para Saber Mais, Teoria em Prática e Leitura Fundamental*, presentes neste *Aprendizagem em Foco*.

Para as avaliações virtuais e presenciais, as questões serão elaboradas a partir de todos os itens do *Aprendizagem em Foco* e dos slides usados para a gravação das videoaulas, além de questões de interpretação com embasamento no cabeçalho da questão.

1. Sabemos que inúmeros são os testes em um software, bem como o objetivo proposto por cada um deles, mas é a relação temporal que deve prevalecer nesta escolha, já que saber o momento mais adequado para determinado teste é de extrema importância para obter o máximo rendimento. Ciente da leitura efetuada anteriormente, quem é o responsável que detém o conhecimento técnico para escolher o melhor teste em determinado momento?.
- a. O usuário.
- b. O gerente.
- c. O analista.
- d. O desenvolvedor.
- e. O testador.

2. Sabemos que em todo projeto há orçamento, e para projetos de software não é diferente. Neste sentido, devemos estabelecer um ponto de equilíbrio entre o que foi orçado e as atividades envolvidas durante a execução dele, para que um não afete o outro. Mas há um fator que pode comprometer, e muito, a qualidade. Assinale a alternativa correta:
- a. Equipe.
 - b. Linguagem a ser desenvolvida.
 - c. Cliente.
 - d. Analista.
 - e. Cronograma.



GABARITO

Questão 1 - Resposta C

Resolução: Fica evidente que o responsável por essa tarefa é o analista, já que possui conhecimento técnico sobre o assunto, além de estar diretamente ligado às decisões sobre teste e qualidade.

Questão 2 - Resposta E

Resolução: Assim como em outros projetos, para a construção de um software, o cronograma é um fator decisivo na qualidade, já que, se for “atropelado”, pode omitir alguma fase do teste, comprometendo, assim, o produto.

TEMA 3

Procedimentos para garantir a segurança do software

Autoria: Luís Otávio Toledo Perin

Leitura crítica: Rafael Maltempe da Vanso





DIRETO AO PONTO

Vamos lembrar os conceitos principais apresentados a partir deste *Direto ao Ponto*? Já parou para pensar sobre como manter a segurança de um software antes, durante e após o seu desenvolvimento? Quais são as práticas a serem realizadas e como isso afeta diretamente a equipe de desenvolvimento? Pois bem, são esses os questionamentos que serão esclarecidos ao longo desta leitura. Então, vamos lá!

Sempre que desenvolvemos algo, ou alguma coisa específica, devemos pensar no produto e em sua utilização, ou seja, quem irá utilizar e como aquilo será benéfico para o ambiente. Todo o ciclo de desenvolvimento de software deve estar voltado, única e exclusivamente para projetar e conceber aplicações que facilitem a vida do usuário, além de possuir um nível satisfatório de segurança, já que dados estão em jogo e a proteção deles é importantíssima para a continuidade dos negócios.

Focar na qualidade é essencial para eliminar as falhas de um sistema e, por consequência, torná-lo mais seguro ao seu usuário. Mas aí você deve estar se perguntando: qual a melhor técnica a ser utilizada? Ou, ainda, o que devo fazer para tornar a aplicação mais segura? Por meio dessas preocupações, padrões reconhecidos mundialmente foram criados, sendo eles: NBR ISO/IEC 27.001, NBR ISO/IEC 27.002 e ISO/IEC 15.408 (ABNT, 2016). Eles estabelecem normas e técnicas padronizadas para que os processos possam ser mais bem trabalhados, cada qual com sua particularidade.


As normas NBR ISO/IEC 27.001 e NBR ISO/IEC 27.002 (ABNT, 2016) têm o princípio de “estabelecer diretrizes e princípios gerais para iniciar, implementar, manter e melhorar a gestão da segurança da informação em uma organização” (ABNT NBR ISO/IEC 27.017,

2016). Se trabalhadas de maneira concomitante, tornam-se o mais completo padrão de segurança da informação, pois possuem métricas internacionais em sua concepção.

São divididas em parte um (NBR ISO/IEC 27.002) e parte dois (NBR ISO/IEC 27.001), abrangendo esta segunda o ciclo PDCA (*plan-do-check-act*) (FERREIRA; ARAÚJO, 2008) e sua estrutura está dividida em escopo, termos e definições, estrutura da norma, avaliação e tratamento dos riscos, política de segurança da informação, organizando a segurança da informação, gestão de ativos, segurança em recursos humanos, segurança física e do ambiente, gestão das operações e comunicações, controle de acesso, aquisição, desenvolvimento e manutenção de sistemas de informação, gestão de incidentes de segurança da informação, gestão da continuidade do negócio e conformidade.

Com toda essa estrutura, temos uma organização bem definida e, como resultado, obtemos um ISMS (Information Security Management System) ou SGSI (Sistema de Gestão de Segurança da Informação), o qual tem por objetivo definir ações planejadas que contemplem práticas, diretrizes, procedimentos, modelos e qualquer outro tipo de medida que vise a reduzir os riscos para a segurança da informação. Ele pode ser construído com o padrão do ciclo PDCA, em que, para cada atividade, temos um objetivo a ser realizado, sendo: *plan*, ou planejar, para estabelecer o SGSI; o *do*, ou fazer, para implementar o SGSI; o *check*, ou checar, para monitorar e melhorar o SGSI; e por fim o *act*, ou agir, para manter o SGSI.

Seguindo esse modelo de normativas, a ISO/IEC 15.408, ou *common criteria*, passa a ser criada para a garantia da segurança, por meio de especificações do ambiente da aplicação. Com base nos resultados obtidos, é possível determinar se tais produtos atendem ou não às suas necessidades de segurança, podendo ser utilizada como guia ou norte para desenvolvimento, avaliação ou aquisição de produtos de TI voltados para a segurança.



Para Albuquerque e Ribeiro (2002), a norma foi implementada com o intuito da melhoria do processo e possui sete níveis para a garantia da segurança, os quais denominamos EAL – *evaluation assurance level*, entretanto, apenas os quatro primeiros níveis são considerados, já que os demais são rígidos e acabam tornando-se inviáveis para sua aplicabilidade. Deste modo, temos: EAL 1: funcionalmente testado; EAL 2: estruturalmente testado; EAL 3: metodicamente testado e verificado; EAL 4: metodicamente projetado, testado e verificado; EAL 5: semiformalmente projetado e testado; EAL 6: semiformalmente projetado, testado e verificado; e EAL 7: formalmente projetado, testado e verificado.

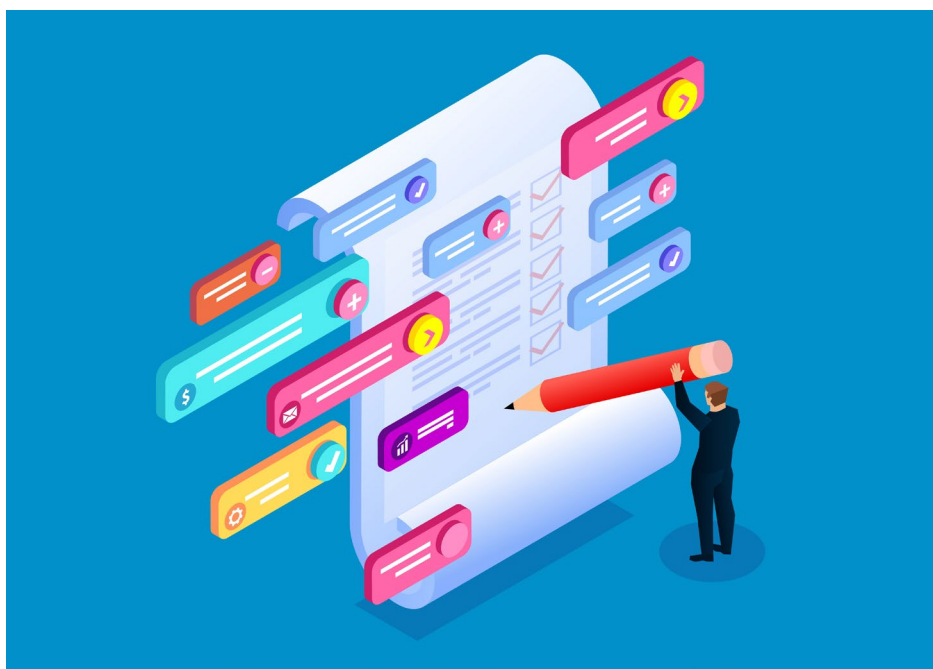
Para aprimorar o modelo de segurança com foco na aplicação e nos resultados para o usuário, a Microsoft, em 2002, traça um modelo denominado SDL (Security Development Lifecycle), o qual passa a ser implementado em suas aplicações. Ele envolve uma série de atividades e produtos com foco no software seguro, os quais são trabalhados durante a fase de desenvolvimento do software.

É com base na integração de medidas e adição de pontos de verificação que o SDL propõe modificar a maneira como o software é comumente desenvolvido, tendo como resultado produtos com alto padrão de segurança. Para isso, são propostas algumas fases, sendo elas: treinamento de segurança, que tem por base treinar e formar a equipe de desenvolvimento sobre aspectos como princípios, tendências em segurança e privacidade; requisitos, que focam em segurança e privacidade da aplicação, bem como a realização de uma análise de custo e rastreamento de bugs; design, que foca na análise da superfície de ataque da aplicação e posterior modelo de ameaças; implementação (codificação segura), que basicamente tem o objetivo de produzir codificação segura, utilizando-se de padrões; verificação, que é a fase destinada a testes, inspeção de código e análise da documentação do software; release (liberação de versões), que

cria um plano de ação, o qual tem por objetivo nortear a equipe de tratamento de incidentes sobre eventuais falhas de segurança no software; e, por fim, a resposta, que tem por objetivo nortear quanto ao tratamento de incidentes relacionados à aplicação.

Devemos compreender que a qualidade e segurança devem caminhar lado a lado durante o ciclo de desenvolvimento de software, haja vista que técnicas e métricas bem aplicadas, ou não, durante a gestão da qualidade afetam o quesito segurança, isso porque repercutem na usabilidade final do produto.

Figura 1 – Validação de processos



Fonte: sesame/iStock.com.

Referências bibliográficas

ABNT – Associação Brasileira de Normas Técnicas. **ABNT NBR ISO/IEC 27.002**: tecnologia da informação: técnicas de segurança: código de prática para a gestão de segurança da informação. Rio de Janeiro: ABNT, 2016.

ABNT – Associação Brasileira de Normas Técnicas. **“ABNT NBR ISO/IEC 27017**: 2016 - Tecnologia da informação - Técnicas de segurança - Código de

prática para controles de segurança da informação com base ABNT NBR ISO/IEC 27002 para serviços em nuvem”. ABNT, 2016

ABNT – Associação Brasileira de Normas Técnicas. “**ISO/IEC 15408-1:2009** - Information technology — Security techniques — Evaluation criteria for IT security — Part 1: Introduction and general model”. ABNT, 2009

ALBUQUERQUE, R.; RIBEIRO, B. **Segurança no desenvolvimento de software**: como garantir a segurança do sistema para seu cliente usando a ISO/IEC. Rio de Janeiro: Editora Campus, 2002.

FERREIRA, F. N. F.; ARAÚJO, M. T. de. **Política de segurança da informação** – Guia prático para elaboração e implementação. 2. ed. revisada. Rio de Janeiro: Ciência Moderna, 2008.



PARA SABER MAIS

Ao ler o seu conteúdo programático, deve ter percebido que o assunto tratado diz respeito à segurança do software. Pois bem, este é um fator que merece extrema atenção, já que a aplicação deve ser considerada segura para que o usuário possa utilizá-la sem medo.

Diante desse fato, uma norma que não foi tratada na leitura, mas é muito importante para a melhoria da segurança, é a ISO/IEC 27.034. Ela trata de todos os aspectos voltados para a segurança, ou seja, desde os requisitos de segurança das informações até a proteção das informações acessadas por um aplicativo, além da prevenção de invasores e/ou ações que podem interferir na confiabilidade e integridade da aplicação.

Juntamente ao quesito segurança, temos a qualidade como grande aliada nesta tarefa, haja vista que é ela quem irá propiciar que os processos sejam gerados com o maior nível de maturidade e, conseqüentemente, de segurança.

Deste modo, estar atento às práticas e técnicas que o mercado oferta é de extrema importância para que você, profissional, possa

desempenhar sua função com toda a qualidade que se espera. Por isso, aprofunde-se nos temas aqui tratados, haja vista que existem manuais e normas recheadas de informações que irão ajudá-lo na organização do software ou da fase a ser desenvolvida.

Referências bibliográficas

ABNT – Associação Brasileira de Normas Técnicas. “**ISO/IEC 27034-1:2011** - Information technology — Security techniques — Application security — Part 1: Overview and concepts”. ABNT, 2011

PRESSMAN, R. S. **Engenharia de software**: uma abordagem profissional. Tradução Ariovaldo Griesi. Revisão técnica Reginaldo Arakaki, Julio Arakaki, Renato Manzan de Andrade. 7. ed. Porto Alegre: AMGH, 2011.

TEORIA EM PRÁTICA

Refleta sobre a seguinte situação: imagine que você já seja formado na área de TI há algum tempo, assim como trabalha na mesma empresa há cerca de dez anos, e também sabe das tecnologias utilizadas e tem domínio sobre o ciclo de desenvolvimento de software dela. Sabe-se, ainda, que a empresa não utiliza práticas bem definidas sobre análise, planejamento e criação dos produtos que são ofertados aos clientes. Como a cidade em que reside é de porte médio e você tem conhecidos em outras empresas do mesmo setor, fica sabendo de ataques de *crackers* sobre sistemas de concorrentes, mas que trabalham com a mesma tecnologia e situação de desenvolvimento que a empresa em que trabalha. Diante do exposto qual seria a atitude que você tomaria, sabendo que um ataque da mesma magnitude pode ocorrer a qualquer momento?

Para conhecer a resolução comentada proposta pelo professor, acesse a videoaula deste *Teoria em Prática* no ambiente de aprendizagem.

LEITURA FUNDAMENTAL

Indicações de leitura

Indicação 1

Em seu conteúdo programático, mais especificadamente o capítulo 12 – “Especificação de confiança e proteção”, o livro *Engenharia de software*, de Ian Sommerville, retrata assuntos ligados à temática que pudemos perceber nesta leitura, ou seja, retrata os requisitos de risco, segurança, confiabilidade e proteção que um software precisa ter.

Para realizar a leitura, acesse a plataforma Pearson / Biblioteca Virtual 3.0, na Biblioteca Virtual da Kroton, e busque pelo título da obra.

SOMMERVILLE, I. **Engenharia de software**. 9. ed. São Paulo, Pearson, 2011. 548 p.

Indicação 2

O livro *Qualidade de software*, de Aline Zanin *et al.*, é bem didático e prático quanto às técnicas e métricas a serem seguidas para se trabalhar com os padrões de maturidade do software. Como indicação, a página 75, a qual trata do CMM e CMMI, é bem clara quanto à necessidade de níveis de maturidade durante o ciclo de desenvolvimento do software.

Para realizar a leitura, acesse a plataforma Pearson / Biblioteca Virtual 3.0, na Biblioteca Virtual da Kroton, e busque pelo título da obra.

ZANIN, A. *et al.* **Qualidade de software**. Porto Alegre: SAGAH, 2018.

QUIZ

Prezado aluno, as questões do Quiz têm como propósito a verificação de leitura dos itens *Direto ao Ponto, Para Saber Mais, Teoria em Prática e Leitura Fundamental*, presentes neste *Aprendizagem em Foco*.

Para as avaliações virtuais e presenciais, as questões serão elaboradas a partir de todos os itens do *Aprendizagem em Foco* e dos slides usados para a gravação das videoaulas, além de questões de interpretação com embasamento no cabeçalho da questão.

1. Sabemos que projetar sistemas robustos e que contenham princípios de qualidade e segurança é fundamental para o sucesso do produto. Também temos em mente que padrões e métricas devem ser adotados para facilitar e organizar o processo.
Ciente da leitura efetuada anteriormente, quem mais é afetado com as boas práticas citadas?
 - a. O usuário.
 - b. O gerente.
 - c. O analista.
 - d. O desenvolvedor.

e. O testador.

2. Sabemos que em todo projeto é necessário possuir normas para que a segurança seja obtida de acordo com os padrões do mercado, bem como priorizar a qualidade do produto. Assinale a alternativa que apresenta corretamente a norma NBR ISO/IEC correspondente à primeira parte.

- a. NBR ISO/IEC 27.001.
- b. NBR ISO/IEC 27.002.
- c. ISO/IEC 15.408.
- d. NBR ISO/IEC 27.300.
- e. NBR ISO/IEC 27.023.



GABARITO

Questão 1 - Resposta A

Resolução: Todo o ciclo de desenvolvimento de software deve estar voltado, única e exclusivamente, para projetar e conceber aplicações que facilitem a vida do usuário.

Questão 2 - Resposta B

Resolução: As normas NBR ISO/IEC 27.001 e NBR ISO/IEC 27.002 têm o princípio de “estabelecer diretrizes e princípios gerais para iniciar, implementar, manter e melhorar a gestão da segurança da informação em uma organização”. São divididas em parte um (NBR ISO/IEC 27.002) e parte dois (NBR ISO/IEC 27.001).

TEMA 4

Ferramentas CASE para testes de segurança do software

Autoria: Luís Otávio Toledo Perin

Leitura crítica: Rafael Maltempe da Vanso






DIRETO AO PONTO

Olá. Vamos relembrar os principais conceitos apresentados até aqui, neste *Direto ao Ponto*? Já parou para pensar sobre como testar a segurança de um software antes, durante e após o seu desenvolvimento? Quais são as práticas a serem realizadas? E como isso afeta diretamente a equipe de desenvolvimento? Pois bem, são esses os questionamentos que serão esclarecidos ao longo desta leitura. Então, vamos lá!

A definição de software vai além de um programa computacional desenvolvido para uma determinada atividade, ou seja, diversos outros fatores estão agregados àquela prática, como a inserção de dados para organização do negócio propriamente dito, ou a exibição e impressão de relatórios gerenciais para a tomada de decisão, por exemplo. Tudo isso é parte do produto software, o qual passa por diversas etapas até a sua finalização.

Dentro deste contexto, sete grandes categorias de software têm papel fundamental dentro de sua engenharia, mais precisamente para os engenheiros, já que são eles que deverão lidar com essa manipulação, sendo elas: de sistema, de aplicação, científico/de engenharia, embutido, para linha de produtos, para a web e de inteligência artificial, possuindo cada um sua particularidade e especificação técnica, o que deve ser levado em consideração durante o seu desenvolvimento, teste e manutenção da aplicação.

Dentro do princípio da segurança da informação, manter a informação segura é parte inerente das atribuições que um software deve possuir, garantindo que não haja vulnerabilidade, e sim proteção contra invasores e acesso não autorizado, sendo essa eventual vulnerabilidade oriunda da forma como são




armazenados ou transmitidos os dados, por exemplo (GOERTZEL *et al.*, 2007).

Outro fator que deve ser levado em consideração durante o processo é o humano, já que atos inseguros podem ocorrer de maneira intencional ou não, devendo a segurança do software preferencialmente evitar ou minimizar possíveis ataques ou tentativas deles. Entre os fatores que colaboram com a exploração de sistemas, podemos citar (GOERTZEL *et al.*, 2006): a exposição da aplicação, como sendo uma “janela” a invasores; a complexidade e o tamanho do software, além das diversas interfaces que ele pode possuir, já que isso pode dificultar a realização e validação de testes corretos e precisos; a terceirização durante o desenvolvimento do software, além do uso de software não verificado para sua construção; e, por fim, a mesclagem de aplicação legada para outras aplicações e em novos ambientes.

Assim como em qualquer outro produto, a segurança é essencial para que haja aceitação por parte do consumidor, neste caso, do usuário ou do cliente, haja vista que a segurança dos dados manipulados é de extrema importância para a organização. Deste modo, garantir que o sistema possa ser acessado em todo e qualquer lugar (on-line) e com segurança é um requisito não funcional que deve ser cumprido à risca.

O software considerado seguro, ou seja, resistente à invasão, é aquele que reduz a probabilidade de um ataque ser bem-sucedido, e se caso este ocorrer, que os danos possam ser suavizados, sendo a aplicação menos afetada possível. Mas isso só será possível se os conceitos de segurança de software forem aplicados de maneira eficiente durante seu ciclo de desenvolvimento (PAUL, 2011).



Neste sentido, as propriedades que caracterizam a segurança do software são as falhas exploráveis e outras fraquezas que são evitadas por desenvolvedores bem intencionados; a probabilidade de que desenvolvedores mal intencionados implantem falhas exploráveis ou lógicas maliciosas no software é baixa ou nula e, por fim, o software sendo resistente, tolerante ou resiliente a ataques.

Outro fator determinante durante a concepção do software é o seu ciclo de vida, o qual deve possuir métricas e práticas bem definidas, acarretando, assim, a sua correta criação. Para McGraw (2006), cada atividade desenvolvida ao longo do ciclo de vida do software é importante para a segurança e, deste modo, é mais bem explicada a seguir:


- a. Casos de abuso:** atividade voltada para avaliar se algum padrão de ataque se encaixa no sistema ou em seus requisitos.
- b. Requisitos de segurança:** fase focada no levantamento das necessidades de segurança do software, além do mapeamento para garantir sua correta implementação.
- c. Análise de risco arquitetural:** fase relacionada com o levantamento dos riscos de segurança que podem estar presentes na arquitetura da aplicação que será construída.
- d. Teste de segurança baseado em riscos:** os testes desenvolvidos precisam absorver dois itens principais, sendo os testes dos requisitos de segurança com técnicas de teste para requisitos funcionais e testes de segurança baseados em riscos, os quais foram anteriormente levantados pelos casos de abuso e pela validação dos padrões de ataque.
- e. Revisão de código-fonte:** após o desenvolvimento das linhas de código propriamente ditas, e antes da fase de testes, é importante revisar a codificação desenvolvida,

isso para averiguar se os requisitos de segurança foram bem implementados, além das vulnerabilidades listadas na análise de casos de abuso.

- f. Testes de penetração:** fase direcionada para testar, de maneira dinâmica, falhas de projeto ou vulnerabilidades que a aplicação possa ter.
- g. Operação segura:** destinado a assegurar que as atividades desenvolvidas pelo usuário possam ser rastreadas, além de protegê-las em um eventual ataque, fazendo com que estejam em segurança.

O teste de segurança é importante para detectar possíveis vulnerabilidades presentes no software. Assim como nos afirma Myers (2004): “Testar é analisar um programa com a intenção de descobrir erros e defeitos”. Testar a segurança de um software é de extrema importância para prever o comportamento da aplicação quanto a possíveis ataques de invasores, além de averiguar qual será o comportamento frente ao inimigo, já que o sistema deve estar preparado para suportar a invasão, bem como assegurar que os dados estejam protegidos.

A fim de contribuir com o processo de verificação de testes de segurança, o documento *Special Publication 800-115: Technical Guide to Information Security Testing and Assessment (NIST, 2008)* define técnicas de teste de segurança de rede, o qual está organizado em oito seções, sendo elas: seção 1, contendo introdução, propósito e escopo do documento; seção 2, a visão geral das avaliações de segurança da informação; seção 3, fornecendo uma descrição detalhada das várias técnicas de avaliação; seção 4, a qual relata técnicas para a identificação de alvos e procedimentos para mapear possíveis vulnerabilidades; seção 5, com técnicas usadas para validar a existência de vulnerabilidades; seção 6, com abordagem e processo



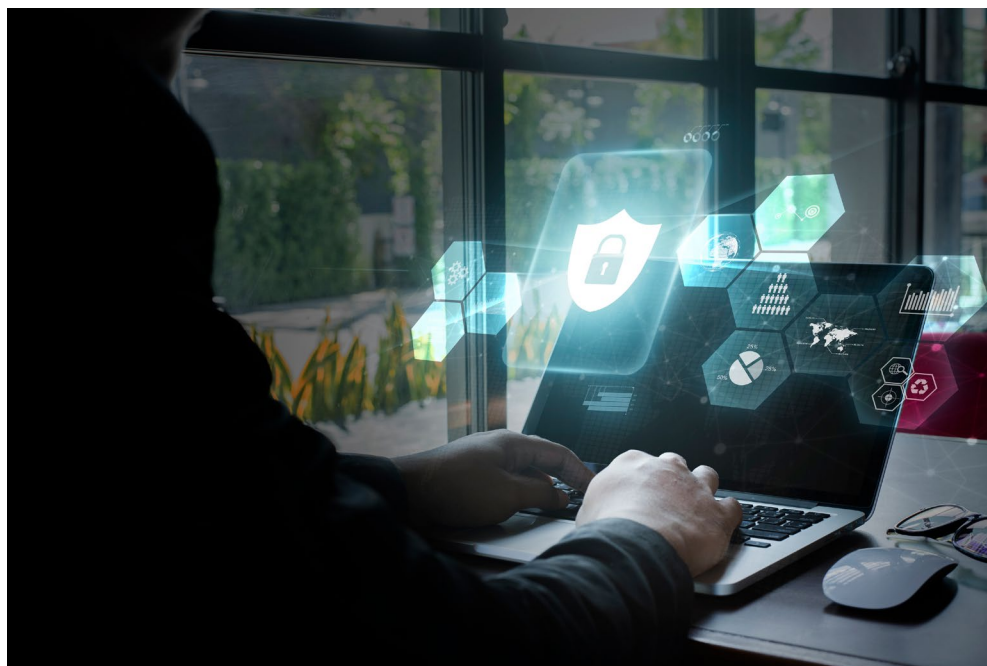
de planejamento de uma avaliação de segurança; seção 7, com os fatores que são fundamentais para a execução de avaliações de segurança; e, por fim, a seção 8, que descreve a apresentação dos resultados da avaliação e fornece uma visão geral das ações para mitigar os riscos.

Neste aspecto, a metodologia Secure Tropos, que é uma extensão de Tropos (que, por sua vez é um *framework* de análise com preocupação na segurança ao longo das fases de desenvolvimento do sistema), é apresentada com a finalidade de modelar problemas de segurança durante todo o ciclo de desenvolvimento do produto software (MOURATIDIS, 2009). Ela consiste em um conjunto de conceitos de requisitos da engenharia de domínio e engenharia de segurança, formando, assim, parâmetros de restrição de segurança, objetivos de segurança e ataques.

O Secure Tropos possui várias etapas durante o seu processo de desenvolvimento, sendo elas: modelagem de atores; modelagem de dependências; identificação de restrições de segurança; delegação de restrições de segurança e objetivos; modelagem de objetivos; modelagem de restrições de segurança; modelagem de entidades de segurança; análise de ameaças; análise de vulnerabilidades; modelagem de atacantes e seleção de provedor de nuvem.

Pense em como a segurança está diretamente relacionada ao produto software, devendo este fazer parte de sua concepção desde o início do projeto, ou seja, já durante o levantamento de requisitos e posterior ciclo de desenvolvimento. Observe a imagem a seguir, ela foca na utilização de uma aplicação, a qual devem estar agregadas diferentes métricas de segurança, fator de fundamental importância para a integridade dos dados do usuário.

Figura 1: Software seguro



Fonte: iBrave/iStock.com.

Referências bibliográficas

GOERTZEL, K. M. *et al.* **Software Security Assurance: A State of the Art Report (SOAR)**, Information Assurance Technology Analysis Center (IATAC), 2007.

MCGRAW, G. **Software security: building security in**, Boston: Addison Wesley Professional, 2006.

MOURATIDIS, H. Secure Tropos: an agent oriented software engineering methodology for the development of health and social care information systems. **International Journal of Computer Science and Security**, v. 3, n. 3, p. 241-271, 2009. Disponível em: <http://www.sts-tool.eu/manuals>. Acesso em: 12 jul. 2020.

MYERS, Glenford J. **The Art of Software Testing**. 2 ed. Nova Jérsei: John Wiley & Sons.

NIST. **Special Publication 800-115**. Technical Guide to Information Security Testing and Assessment. NIST 2008.

PAUL, M. **Official (ISC) 2 Guide to the CSSLP**. Florida: CRC Press, 2011.

WILLOUGHBY, M. Q&A: quality software means more secure software. **Computerworld**, 21 mar. 2005. Disponível em: <https://www.computerworld.com/article/2563708/q-a--quality-software-means-more-secure-software.html>. Acesso em: 12 jul. 2020.



PARA SABER MAIS

Como podemos notar, o conteúdo segurança da aplicação de software é de extrema importância para o usuário e, deste modo, diversas medidas, técnicas e processos devem ser adotados durante o seu ciclo de desenvolvimento, desta forma, a garantia da qualidade seria obtida de maneira satisfatória.

Neste aspecto, a engenharia de software se utiliza de técnicas que auxiliam nesse desenvolvimento mais seguro e planejado, como o teste de caixa branca, preta e de regressão, todos com o objetivo de validar aquilo que está sendo desenvolvido como produto.

O teste caixa branca possui critérios baseados em fluxo de controle e em fluxo de dados, portanto, avalia a parte estrutural do software. Já o teste de caixa preta se preocupa em avaliar o software conforme as entradas e saídas, aplica-se externamente nele sem considerar seu funcionamento interno. Por sua vez, o teste de regressão avalia todas as experiências em testes executados no passado e, a cada nova versão disponibilizada, executam-se os testes já realizados anteriormente e que apresentaram sucesso.

Como este processo envolve diversas outras etapas, é de extrema importância que haja um entendimento sobre o que deve ser feito e qual a tecnologia mais atual para que aquilo possa ser aplicado, garantindo, assim, a atualização dos processos realizados.

Por isso, aprofunde-se nos temas aqui tratados, haja vista que existem manuais e normas técnicas com diversas outras informações, as quais irão ajudá-lo na organização do software ou da fase a ser desenvolvida.

Referências bibliográficas

PRESSMAN, R. S. **Engenharia de software**: uma abordagem profissional. Tradução Ariovaldo Griesi. Revisão técnica Reginaldo Arakaki, Julio Arakaki, Renato Manzan de Andrade. 7. ed. Porto Alegre: AMGH, 2011.

TEORIA EM PRÁTICA

Refleta sobre a seguinte situação: você possui formação e pós-graduação na área de tecnologia da informação, além de sempre estar atualizado sobre os diversos processos do ciclo de desenvolvimento de software. Atualmente, trabalha como programador em uma empresa de sistemas para o setor comercial, além de ser analista em alguns momentos. Já estando na área há certo tempo e sempre participando de fóruns e eventos do meio, ganhou visibilidade, despertando o interesse por um grupo de jovens que sonham com seu negócio próprio. Em uma conversa, a oferta foi apresentada e você seria responsável por gerenciar toda uma equipe de desenvolvimento. Pois bem, diante do desafio e de querer crescer como profissional, resolveu aceitar a proposta, tornando-se agora líder do setor. Como a empresa está se estruturando, além de partir do zero quanto à criação do software, você será o responsável por organizar seu ciclo de desenvolvimento, principalmente focado na segurança. Deste modo, por onde começaria? Quais as técnicas que iria usar quanto à segurança? Com seria o padrão a ser estabelecido?

Para conhecer a resolução comentada proposta pelo professor, acesse a videoaula deste *Teoria em Prática* no ambiente de aprendizagem.



LEITURA FUNDAMENTAL

Indicações de leitura

Indicação 1

Em seu conteúdo programático(mais especificamente a unidade que trata da introdução aos testes de software, na página 13 do livro *Teste de software*), retrata assuntos relacionados a teste de software, dos quais estão ligados à temática aqui apresentada. Vale a pena conferir.

Para realizar a leitura, acesse a plataforma Pearson / Biblioteca Virtual 3.0, na Biblioteca Virtual da Kroton, e busque pelo título da obra.


BRAGA, P. H. C. **Teste de Software**. São Paulo, Pearson, 2016. 139 p.

Indicação 2

O livro *Qualidade de software* é bem didático e prático quanto às técnicas e métricas a serem seguidas para se trabalhar com qualidade e posteriores testes em software. Como indicação, a unidade que descreve as abordagens formais e garantia estatística de qualidade de software, na página 49, vai ao encontro do tema aqui retratado.

Para realizar a leitura, acesse a plataforma Pearson / Biblioteca Virtual 3.0, na Biblioteca Virtual da Kroton, e busque pelo título da obra.

GALLOTTI, G. M. A. **Qualidade de software**. São Paulo, Pearson, 2015.139 p.



QUIZ

Prezado aluno, as questões do Quiz têm como propósito a verificação de leitura dos itens *Direto ao Ponto, Para Saber Mais, Teoria em Prática e Leitura Fundamental*, presentes neste *Aprendizagem em Foco*.

Para as avaliações virtuais e presenciais, as questões serão elaboradas a partir de todos os itens do *Aprendizagem em Foco* e dos slides usados para a gravação das videoaulas, além de questões de interpretação com embasamento no cabeçalho da questão.

1. Sabemos que, ao determinar o que significa software, estamos afirmando que vai além de um programa computacional desenvolvido para uma determinada atividade, ou seja, diversos outros fatores estão agregados àquela prática. Ciente da leitura efetuada anteriormente, para que a construção do produto software tenha êxito, um processo deve ser rigorosamente seguido em seu desenvolvimento. A este processo damos o nome de:
 - a. O ciclo de vida.
 - b. A execução de erros.
 - c. Os requisitos.
 - d. O desenvolvedor.
 - e. O cliente.

2. Ciente de que diversas falhas e vulnerabilidades são descobertas a cada dia, temos que o software está exposto a inúmeras situações, principalmente se ele

estiver em um ambiente on-line. Considerando o exposto, assinale a alternativa que contém a melhor definição de software seguro:

- a. Tem como referência um software sem defeitos.
- b. Um software capaz de aguentar tudo.
- c. Um software 100% on-line e acessível de qualquer lugar.
- d. Desenvolvimento focado em padrões do analista.
- e. Reduz a probabilidade de um ataque ser bem-sucedido.



GABARITO

Questão 1 - Resposta A

Resolução: Todo o ciclo de vida de desenvolvimento do software deve estar de acordo com os padrões estabelecidos, ou seja, seguir as boas práticas que são mencionadas ao longo do curso, além de aderir a métricas e normas.

Questão 2 - Resposta E

Resolução: O software considerado seguro, ou seja, resistente à invasão, é aquele que reduz a probabilidade de um ataque ser bem-sucedido, e caso este ocorra, que os danos possam ser suavizados, sendo a aplicação menos afetada possível.



BONS ESTUDOS!