UMA VISÃO GERAL SOBRE AUTOMAÇÃO DE TESTES

ARTIGO ORIGINAL

SILVA, Mario Luis Moreira da ¹

DALLILO, Felipe Diniz ²

SILVA, Mario Luis Moreira da. DALLILO, Felipe Diniz. **Uma visão geral sobre automação de testes**. Revista Científica Multidisciplinar Núcleo do Conhecimento. Ano 04, Ed. 12, Vol. 01, pp. 117-130. Dezembro de 2019. ISSN: 2448-0959, Link de acesso: <https://www.nucleodoconhecimento.com.br/tecnologia/automacao-de-testes>

RESUMO

As organizações responsáveis pelo desenvolvimento de softwares, no mundo empresarial atual, buscam por técnicas e estratégias inovadoras tendo-se como escopo o aperfeiçoamento dos seus processos bem como a entrega de produtos em um período de tempo apontado pelos consumidores como adequado, para, assim, atender o que almejam esses que consomem, visto que, atualmente, são menos tolerantes à erros e são bastante exigentes frente ao que adquirem. Software em desenvolvimento requer um grande esforço em testes minuciosos por estarem mudando constantemente e através dos testes automatizados é possível obter um *feedback* mais rápido, por exemplo, de visualizar quais testes passaram e quais falharam, qual passo teve erro e qual o motivo. Durante o desenvolvimento, é necessário tomar a decisão de automatizar ou não os cenários de testes, para garantir que não há regressões no *software* e economizar tempo. Este artigo avalia de forma descritiva requisitos de automação de testes, mostrando a importância nas etapas de

¹ Graduando do Curso de Sistema da Informação da Universidade de Araraquara-UNIARA. Araraquara-SP.

² Mestre do Curso de Sistema da Informação da Universidade de Araraquara-UNIARA. Araraquara-SP.

produção e execução, discorrendo, nesse processo, sobre as teorias que orientam a realização de atividades de teste, as técnicas e critérios. Escolheu-se por aqueles que podem ser acionados tanto para verificar quanto para validar um produto.

Palavras-chave: Automação, desenvolvimento de *Software*, *Software* Profissional, testes automatizados.

1. INTRODUÇÃO

Atualmente, há uma procura frequente pela introdução e/ou inclusão dos mais diversos softwares no mercado, prezando-se, nesse processo, pela qualidade. Detectou-se, então, que para tanto é necessária a realização de testes para que defeitos e falhas que apenas poderiam ser descobertos tardiamente sejam identificados mais rápidos, pois essa demora dificulta, diretamente, na manutenção do sistema. Dessa forma, a realização de testes é uma etapa fundamental e indispensável para a elaboração e desenvolvimento de um bom sistema. A automação desses testes surge como um processo facilitador, diminuindo, para isso, tanto o tempo de produção quanto de manutenção. As maiores frustrações em projetos de automação de testes estão relacionadas às expectativas iniciais sobre o que poderá acarretar em uma melhor automatização em termos de custos de implantação e manutenção do sistema, uma vez que é possível automatizar e testar tudo.

Sommerville (2011) afirma que o teste deve mostrar que um programa faz aquilo que lhe foi atribuído/designado bem como se ocupa com buscar por possíveis defeitos antes que esse sistema esteja pronto para uso. Assim, devido a presença dos modelos atuais de desenvolvimento de *software*, o propósito desse artigo é divulgar e incentivar a atividade de teste para oferecer uma atenção especial, antecipando assim, a atividade de teste para o início da elaboração do desenvolvimento desse *software*. Pressman (2011) alude que o conceito de teste de *software* deve ser interpretado a partir de um olhar intuitivo ou, ainda, de maneira formal. O mesmo estudo aponta que, no mundo moderno, existem diversas conceituações sobre essa temática. Compreende-se, de forma geral, que o teste de softwares implica na necessidade de

verificar, a partir de um processo de execução controlada, se o comportamento desse software transcorre conforme o esperado.

Durante o processo de desenvolvimento do mesmo, podem ocorrer diversas falhas e para que estas não sejam percebidas pelos clientes, é necessário ter os testes integrados em todo processo, garantindo assim, a qualidade do produto. Além de encontrar *bugs*, o teste principalmente os previne. Para Sommerville (2011) o processo de teste, envolve uma mistura de testes manuais e automatizados. No teste manual, executa-se o programa com alguns dados para verificar se o resultado corresponde ao proposto, informando aos desenvolvedores se houve erros, e no teste automatizado, ele é codificado em um programa e executado cada vez que o sistema em desenvolvimento avança.

Sommerville (2011), em seu estudo, elucida que durante o processo de desenvolvimento de um dado software, é papel dos responsáveis a criação de testes devido ao seu conhecimento do código gerado e os testam, o custo de corrigir erros ou falhas no sistema por meio de uma mudança é maior que o custo de corrigir durante planejamento ou codificação. O objetivo deste artigo é avaliar de forma descritiva requisitos de automação de testes, comentando sobre sua importância no processo de desenvolvimento e execução, abordando os principais conceitos relacionados às atividades de teste, técnicas e critérios que devem ser acionados para verificar e/ou validar um determinado produto, tendo-se como escopo principal a ilustração acerca do funcionamento de um teste automatizado.

2. REVISÃO BIBLIOGRÁFICA

Sommerville (2011), afirma que os padrões são muito importantes para gerenciar e garantir a qualidade do software, selecionando e definindo quais os padrões que devem ser aplicados durante o processo de desenvolvimento.

Figura 1: A qualidade e as suas etapas

Figura 1 Qualidade baseada em processos

Fonte: SOMMERVILLE(2011)

Fonte: Sommerville (2011)

A Gestão da Qualidade pode ser compreendida como um exercício que coordena as atividades que englobam os processos de produção de serviços e produtos para que cheguem até os consumidores com qualidade. Tem-se em mente que suas dimensões podem ser compreendidas como desempenho, confiabilidade, percepção e durabilidade. Sommerville (2011) comenta que, como parte desse processo, devem ser escolhidas ferramentas e métodos para suportar o uso desses padrões. Uma vez que os padrões foram selecionados para uso, processos específicos de projeto devem ser definidos para monitorá-los e verificar se foram seguidos. Pode-se mencionar dois padrões voltados à engenharia de *software* que, de forma geral, são acionados para gerir a qualidade. São eles:

- Padrões de produto: Aplicam-se ao produto de *software* que está sendo desenvolvido um padrão. Eles incluem padrões de documentos, como a estrutura dos documentos de requisitos; padrões de documentação como um cabeçalho de comentário padrão para uma definição de classe de objeto e padrões de codificação, os quais definem como uma linguagem de programação deve ser usada;
- Padrões de processo: Definem os padrões que precisam ser cumpridos durante a etapa de desenvolvimento do software. Eles devem encapsular as

boas práticas de desenvolvimento. Os padrões de processo podem incluir definições de especificação, projeto e processos de validação, ferramentas de suporte do processo bem como a descrição acerca dos processos desse desenvolvimento em um documento.

Tal administração da qualidade visa a garantia da excelência durante a execução de todas as atividades associadas a este software. As dimensões da qualidade que devem ser garantidas durante tal desenvolvimento são, principalmente, o desempenho, a confiabilidade, a percepção, a durabilidade, a conformidade e o bom atendimento.

2.1 AUTOMAÇÃO DE TESTE DE SOFTWARE

A automação evidencia a busca pela otimização do tempo do teste desse software em desenvolvimento. Tem-se necessidade de tornar esse teste mais rápido e efetivo com tal otimização. O objetivo geral é procurar por possíveis falhas. Recomenda-se que se presta atenção no quanto se deve automatizar esse software e como este pode ser feito sem prejudicar a qualidade. De forma geral, a automação compreende os testes de unidade ou de sistema, e, dessa forma, exige-se um esforço maior logo no início desses testes. Tal fenômeno é decorrente da necessidade de se realizar um planejamento para se identificar o que, como e por que esse software precisa ser automatizado, conforme Sommerville (2011). O mesmo estudo compreende que testes de unidade eficazes estão relacionados com a estrutura do software e com a base do código. Já no caso do teste de sistema, implica-se na necessidade de compreender as regras de negócio, seu funcionamento, seus valores e situações que evidenciam a sua aplicabilidade.

Os testes são considerados como significativos e bem elaborados quando são capazes de contribuir para com o aumento expressivo da qualidade na entrega do produto final. Todavia, a construção de scripts para a aplicação desses testes automáticos implica na necessidade do requerimento de uma padronização nas etapas assim como na compreensão da codificação e análise por parte daqueles responsáveis por testar o sistema. A falta de um processo de testes planejados e

automáticos faz com que ocorra demora na entrega do produto final ao cliente, prejudicando as manutenções, além de poder encarecer o custo do produto para conseguir manter sua qualidade. Uma aplicação mal testada poderá gerar grandes problemas, desde a recodificação total da aplicação, prejuízos financeiros (tanto para a empresa desenvolvedora como para os clientes e/ou usuários) e dependendo da área de aplicação, poderá até causar risco de vida às pessoas (por exemplo, no controle de tráfego aéreo).

2.2 PLANEJAMENTO DE TESTES

O processo de planejamento dos testes evidencia a compreensão acerca das características e funções desse sistema que precisarão ser averiguadas por meio desses testes. Deve-se, ainda, ater-se aos responsáveis por al processo bem como aos riscos envolvidos. Esse planejamento precisa, também, definir o seu objetivo, o custo e o prazo para a execução das atividades que nortearão esses testes. Recomenda-se, para a realização de tal planejamento, que os responsáveis se atenham à referenciação dos requisitos e das prioridades desse sistema a ser desenvolvido. Por fim, ainda nesse momento, é preciso definir estimativas, estratégias e técnicas para a identificação de possíveis problemas, tais como:

- Funções incorretas ou omitidas;
- Erros de *interface*;
- Erros de comportamento ou desempenho;
- Erros de iniciação e término.

O teste é parte de um processo volumoso voltado à verificação e validação (V&V). que “(...) não são a mesma coisa, embora sejam frequentemente confundidas” (SOMMERVILLE, 2011, p. 145). O planejamento dos testes precisa, portanto, tomar forma em níveis distintos bem como deve ocorrer de forma paralela ao desenvolvimento do software. O objetivo desse teste deverá ser capaz de identificar quais os elementos que devem ser testados bem como é preciso especificar qual cobertura se deseja aderir. Os recursos, por sua vez, sejam eles humanos, materiais, de tempo e de custo, devem ser definidos de forma clara e objetiva. O cronograma de

atividades, por conseguinte, estabelecerá as tarefas que deverão ser cumpridas em ordem cronológica para que o andamento não seja prejudicado.

2.3 ESTRATÉGIA DE TESTES

Para Pressman (2011, p. 40) “as ferramentas de engenharia de *software* fornecem apoio automatizado ou semi-automatizado para o processo e para os métodos” Assim, as estratégias são importantes porque são capazes de otimizar o tempo, ou seja, evitam que ele seja desperdiçado assim como o esforço que poderia ser dispensado e evidencia a possibilidade da descoberta de erros que podem ser solucionados logo nos testes. Em certa etapa, faz-se necessário que as atividades, objetos, papéis e padrões sejam apresentados no processo de desenvolvimento. Já as reflexões de Sommerville (2011) apontam que no processo de teste (como nos demais), o aperfeiçoamento constante permite que esse software ganhe mais maturidade bem como amplia o seu campo de atuação e, por fim, os agentes responsáveis acabam tendo uma maior visibilidade.

Almeja-se, nessa etapa, conseguir uma maior agilidade e controle nos procedimentos operacionais durante a execução desses testes. A pesquisa de Sommerville (2011) reitera, ainda, que uma boa estratégia para ser incorporada nesse processo de testes dos softwares é a descrição acerca da abordagem genérica acerca dos objetivos direcionados às tarefas a serem cumpridas pelo teste. Para tanto, tal abordagem deve contemplar níveis ou fases para essas atividades. Assim sendo, precisa-se considerar alguns elementos como os tipos de testes a serem realizados bem como suas técnicas; a descrição clara acerca dos critérios para a sua conclusão e os critérios para que o software atinja o sucesso.

2.4 TESTES DE DESENVOLVIMENTO

Os testes sobre os quais falamos nesse estudo contemplam as tarefas que devem ser feitas pela equipe responsável pelo desenvolvimento do software. Já o seu testador pode ser o programador que o desenvolveu, contudo isso pode variar de acordo com cada organização. Em sistemas críticos é comum que se adira a processos mais

formais, e, dessa forma, pode-se fazer uso de um grupo de testes independente que pode fazer parte da equipe responsável pelo desenvolvimento do software, pela equipe de testes e da equipe que tem como função a realização da manutenção dos dados sobre esses resultados de forma detalhada. Nesse sentido, Sommerville (2011, p. 148) alude que os “testes de desenvolvimento são essencialmente um processo de teste de defeitos, em que o objetivo do teste é descobrir *bugs* no *software*”.

Ao desenvolver um *software*, as falhas, de forma geral, costumam se manifestar no processo de transformação dos dados nos diversos níveis do desenvolvimento. Percebe-se que mesmo que se acione os melhores métodos, ferramentas e conceitos bem como de profissionais de renome, os testes são indispensáveis, por mais seguro que se esteja frente a esse software. Já a extensão do projeto e a quantidade de pessoas envolvidas nesse processo de desenvolvimento podem favorecer o aparecimento de falhas, porém, elas não podem ser totalmente impedidas, mesmo que se tome o devido cuidado. Os erros podem aparecer em razão de diversos fatores, podendo eles ser a especificação errada ou incompleta ou a base de dados desse sistema não permitir que sejam distinguidos os tipos de usuário.

Figura 2: Modelo do teste

Fonte: SOMMERVILLE(2011)

Os testes, por sua vez, dividem-se em quatro níveis responsáveis cada qual por definir as fases nas quais serão executados. Revelam, então, o melhor momento para a sua execução em cada um desses níveis.

- a) Teste unitário: as unidades individuais de programa ou classes de objetos são testadas individualmente; valida a menor parte do código-fonte e neste nível de teste é verificado, por exemplo, o método de uma classe;
- b) Teste de componentes: várias unidades individuais são integradas para criar componentes compostos; é validado se a integração entre sistemas, componente ou outras funcionalidades foi feita corretamente;
- c) Teste de sistema: alguns ou todos os componentes de um sistema estão integrados e o sistema é testado como um todo; o *software* é exercitado como um todo a fim de encontrar discrepâncias entre o funcionamento especificado nos requisitos e o construído;
- d) Testes de verificação: devem garantir a qualidade do processo de desenvolvimento do *software*, avaliando a documentação e atividades com objetivo de encontrar erros e quebras dos processos.

As possibilidades de aplicação dos testes são definidas no momento do planejamento. Todavia, para que a sua realização seja possível é preciso refletir, de forma analítica, sobre alguns elementos, tais como: a definição do público que utilizará esse sistema; se este é acessível para deficientes visuais; se ficará disponível na internet; se consegue realizar integração com sistemas outros; se consegue trabalhar com múltiplos registros trabalhados; se é possível o seu acesso por usuários distintos de forma simultânea, etc. Objetiva-se, ainda, a verificação dos riscos desse projeto bem como uma análise acerca do seu nível de tolerância aos possíveis erros.

2.4.1 DESENVOLVIMENTO DIRIGIDO A TESTES (TDD DO INGLÊS *TEST DRIVEN DEVELOPMENT*)

O Desenvolvimento Orientado a Testes ou TDD (*Test Driven Development*) se trata de uma atividade da área de engenharia de softwares que pode ser utilizada como uma ferramenta para a otimização da produção. Assim sendo, a sua configuração consiste em um ciclo curto composto por repetições. Nele, o desenvolvedor fica

responsável pela descrição de casos em que esses testes foram aplicados para a automação. Devem ser realizados tanto para validar um dado ponto quanto para atribuir uma nova função. Entretanto, para que essa prática possa ser feita de maneira eficaz, é necessário o emprego de atividades complementares. Para isso, recomenda-se o uso de ferramentas igualmente eficientes para que algumas precauções sejam tomadas bem como para que essas atividades sejam desempenhadas da melhor forma. Hoje em dia, sabe-se que a maior parte da linguagem possui frameworks próprios para que esses testes sejam realizados. Seguem alguns exemplos dessa linguagem:

- **JUnit:** O JUnit é um framework de teste para Java, que permite a criação de testes unitários. Além disso, está disponível como plug-in para os mais diversos IDE'S como Eclipse, Netbeans etc;
- **TesteNG:** Outra ferramenta de teste unitária, disponível para Java;
- **PHPUnit:** Framework XUnit para teste unitário em PHP, também é possível integrar aos IDE's assim como o JUnit;
- **SimpleTest:** Outra ferramenta para realização de teste para PHP. Além de possibilitar os testes unitários, é possível realizar MOCKS e outros testes;
- **NUnit:** Framework de teste no molde XUnit para a plataforma .NET;
- **Jasmine:** Framework para teste unitário de JavaScript;
- **CUnit:** Ferramenta para os testes unitários disponível para Linguagem C;
- **PyUnit:** Framework Xunit para testes na linguagem Python.

A fim de uma melhor compreensão, o esquema abaixo revela melhor como é feito o processo de desenvolvimento dos testes.

Figura 3: Desenvolvimento dos testes

Figura 3 Desenvolvimento Dirigido a Testes

Fonte: SOMMERVILLE(2011)

O Desenvolvimento Dirigido por Testes implica na responsabilidade, por parte dos desenvolvedores, pela criação de testes de unidades automatizados para a definição dos requisitos na forma de códigos. Isso deve ocorrer antes da escrita desse código da aplicação. De forma geral, desenvolve-se um código a ser incrementado em consonância com um teste para tal incremento. Não é possível avançar para a próxima etapa de incrementação até que esse código passe no teste. Na esfera acadêmicas, muitos defendem que o TDD deve atuar na redução de defeitos de um software, em contrapartida, outros optam por investigar como o TDD pode produzir códigos mais eficientes de forma a garantir a qualidade. Outros, por fim, atentam-se a como aumentar a produtividade. O processo do TDD é de fácil entendimento, conforme pode ser visualizado na figura 4.

Figura 4 - CICLO DO DESENVOLVIMENTO ORIENTADO A TESTE OU TDD

FONTE <https://javacodehouse.com/blog/test-driven-development-tutorial/>

a) Escreva sobre um teste que falhou: é essencial que você averigue, de forma analítica, uma das características do software. Pensar em como a funcionalidade deste pode ser garantida é algo que deve ser considerado pelo teste. Se o bom funcionamento ser detectado pelo teste, sua permanência se torna optativa, pois os resultados apontam que o sistema já detectou o bom funcionamento;

b) Faça o teste passar: Depois de acionado o *baby steps*, é recomendado que seja feita uma quantidade menor de codificação para que se consiga passar no teste, mesmo quando o método retornar um valor esperado para este teste. Objetiva-se, simplesmente, passar no teste;

c) Realize uma nova faturação do seu código de teste: Deve-se refatorar o código de teste por meio do desenvolvimento de novos testes de forma que se obtenha menos erros/falhas. O código deve continuar atendendo ao requisito e passando pelos testes.

O desenvolvimento orientado a testes (TDD) é um processo responsável pela elaboração do software que depende da repetição de um ciclo de desenvolvimento muito curto. Os requisitos são transformados em casos de teste muito específicos e em seguida, o software é aprimorado para passar apenas nos novos testes, ou seja,

o procedimento deve ser refeito até que o código em processo de teste esteja menos favorável a acarretar em possíveis erros. A cada refatoração deste, é recomendado que os testes sejam aplicados novamente para que esses possam ser atualizados com seu código de produção. É algo bastante complexo, pois se implica na necessidade de se aderir a um outro comportamento para a obtenção de outros resultados.

Para que o código se torne mais conciso bem como para que seja capaz de atuar de forma menos dependente, é responsabilidade do programador ter um bom raciocínio. Dessa forma, é preciso que mude, automaticamente, a forma de desenvolvimento, e, assim, ao invés de se projetar a aplicação, deve-se tentar escrever um código objetivando mudanças para o futuro. Nesse contexto, realiza-se um código de teste que, ao mesmo tempo, precisa ser refatorado para que ele fique menos acoplado e mais coeso. Isso faz com que o código possa ser, futuramente, manuseado sem grandes dificuldades para a solução de possíveis problemas. Devido a tais características positivas, o TDD se torna recomendável para o fortalecimento da segurança e da qualidade no processo de desenvolvimento do software.

CONCLUSÃO

O presente artigo possibilitou uma visão da importância da automação de teste em desenvolvimento para divulgar e incentivar a atividade de teste, a fim de, verificar e validar um produto. A introdução de softwares com qualidade no mercado mostra que a realização de testes, é um processo importante e essencial na produção e desenvolvimento de um bom sistema. Testar um software significa verificar através de uma execução controlada, se o seu comportamento ocorre de acordo com o especificado; selecionando e definindo quais os padrões que devem ser aplicados durante o processo de desenvolvimento. Através dos testes automatizados é possível visualizar quais testes passaram e quais falharam em qual passo teve erro e qual o motivo, ou seja, o teste deve mostrar que o programa faz aquilo que foi proposto como também, descobrir eventuais defeitos antes de seu uso.

Os objetivos propostos no artigo têm como relevância a realização da automação de testes, para que não haja frustração em projetos, descobrindo de forma tardia algum tipo de erro/bug, inviabilizando tempo e custo. Como não se pode testar e automatizar o processo em 100 % faz-se necessário, analisar o que é de fato significativo para o usuário. Os questionamentos “o que, como e por que” automatizar, são fundamentais em um planejamento inicial, pois contribuem para um aumento na qualidade do produto final. Os testes podem ser divididos em níveis que definem a fase, ou seja, o melhor momento de se executar determinado teste.

Como resultado do artigo destaca-se que, a falta de um processo de teste planejado e automatizado, como já salientado anteriormente, poderá ocorrer na demora da entrega do produto final ao cliente, encarecimento do custo do produto, frustrações e outros problemas; tanto para a empresa responsável como para o cliente. É importante ressaltar que a área de Tecnologia de Informação está em crescente avanço, assim como as empresas desenvolvedoras de software profissional; que tem a finalidade de atender as necessidades exclusivas de seus clientes. Por essa razão, o estudo e pesquisa devem ser constantes, uma vez que surgem ferramentas e técnicas novas, visando melhorias na entrega do produto final e dentro do prazo estabelecido.

REFERÊNCIAS

BERNARDO, P. C; KON, F. **A Importância dos Testes Automatizados. Artigo publicado na Engenharia de Software Magazine.** 2008. Disponível em: <https://www.ime.usp.br/~kon/papers/EngSoftMagazine-IntroducaoTestes.pdf>. Acesso em: 13. Nov. 2019.

CRISPIN, L; GREGORY, J. **AGILE TESTING.** 1ª ed. Boston: Pearson Education, Inc. 2009.

PRESSMAN, R. S. **Engenharia de Software.** 7ª ed. Porto Alegre: AMGH Editora Ltda, 2011.

RIOS, E; FILHO, T. M. **Teste de Software**. 3ª ed. Rio de Janeiro: Alta Books, 2013.

SILVA, P. C. B; ALVES, T. S; BRUNO, E. A. **Automação de testes funcionais: testes funcionais automatizados de software**. Revista de Ciências Exatas e Tecnologia, v. 6, n. 6, p. 113-133, 2011.

SOMMERVILLE, I. **Engenharia de Software**. 9ª ed. São Paulo: Pearson Education do Brasil, 2011.

Enviado: Outubro, 2019.

Aprovado: Dezembro, 2019.