GERENCIAMENTO E AUTOMAÇÃO DE TESTES PARA ALCANCE DA QUALIDADE DE *SOFTWARE*

Durvalino Batista Godoi Neto
Discente do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas
Faculdades Integrada de Três Lagoas (AEMS)

Gabriel Henrique Silva dos Santos
Discente do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas
Faculdades Integrada de Três Lagoas (AEMS)

Samuel Fernandes Pires Siqueira
Discente do Curso de Tecnologia em Análise e Desenvolvimento de Sistema
Faculdades Integrada de Três Lagoas (AEMS)

Alan Pinheiro de Souza
Docente do Curso Tecnologia em Análise e Desenvolvimento de Sistemas
Faculdades Integrada de Três Lagoas (AEMS)

RESUMO
O teste de *software* é uma atividade importante para o desenvolvimento de sistemas de *software* com qualidade. O gerenciamento e a automação de testes são considerados medidas necessárias para melhorar a eficiência dessa atividade. Todavia, para uma aplicação bem sucedida dessas medidas, deve-se adotar uma abordagem sistemática de testes. Este artigo apresenta um levantamento bibliográfico visando reconhecer os objetivos e a importância do planejamento de teste, assim como identificar de que modo gerenciamento pode servir de suporte às atividades de teste. A pesquisa discute ainda de que forma a automação de teste pode contribuir para o alcance de *software* de qualidade ao longo dos processos de desenvolvimento.

PALAVRAS-CHAVE: Gerenciamento de Teste; Automação; Qualidade de *Software;* Processos de *Software*.

INTRODUÇÃO

Quando fala-se em qualidade de *software*, é comum pensar em uma empresa, a qual não pode ficar parada perdendo dinheiro por causa de uma falha no sistema. Assim, deve-se pensar no futuro do ambiente corporativo identificando empresas especializadas no acompanhamento, elaboração, desenvolvimento, teste e homologação de sistemas. Para evitar futuras falhas, deve-se, desde o início do projeto, proceder com seu planejamento, pois grande parte do desenvolvimento com

qualidade de um sistema está associado aos testes, acertos, validações e homologações para entrar em produção (GANDARA, 2012, p.11-12).

A etapa de teste de *software* é destinada a mostrar que um programa faz o que é proposto realizar, além de descobrir os defeitos do programa antes do uso. Durante os procedimentos de teste do *software*, casos são executados usando dados fictícios. Feito isso, os resultados são verificados à procura de erros, anomalias ou informações não esperadas sobre os atributos não funcionais do programa (SOMMERVILE, 2011, p.144).

Este artigo propõe-se a apresentar um levantamento bibliográfico embasado em livros e artigos científicos de modo a alcançar uma visão genérica da adoção de gerenciamento e automação de testes no âmbito de desenvolvimento de *software*. Além da seção introdutória que apresenta uma contextualização e motivação no assunto a ser abordado, o trabalho está dividido em quatro seções. A primeira seção discute aplicação de testes de *software* ao longo do ciclo de vida. A segunda seção caracteriza qualidade de *software*. Na terceira seção do artigo são discutidas práticas de gerenciamento de testes e na quarta seção abordam-se medidas para automação de teste. As seções finais possuem as considerações finais e as referências bibliográficas adotadas na pesquisa.

1. TESTES NO CICLO DE VIDA DO *SOFTWARE*

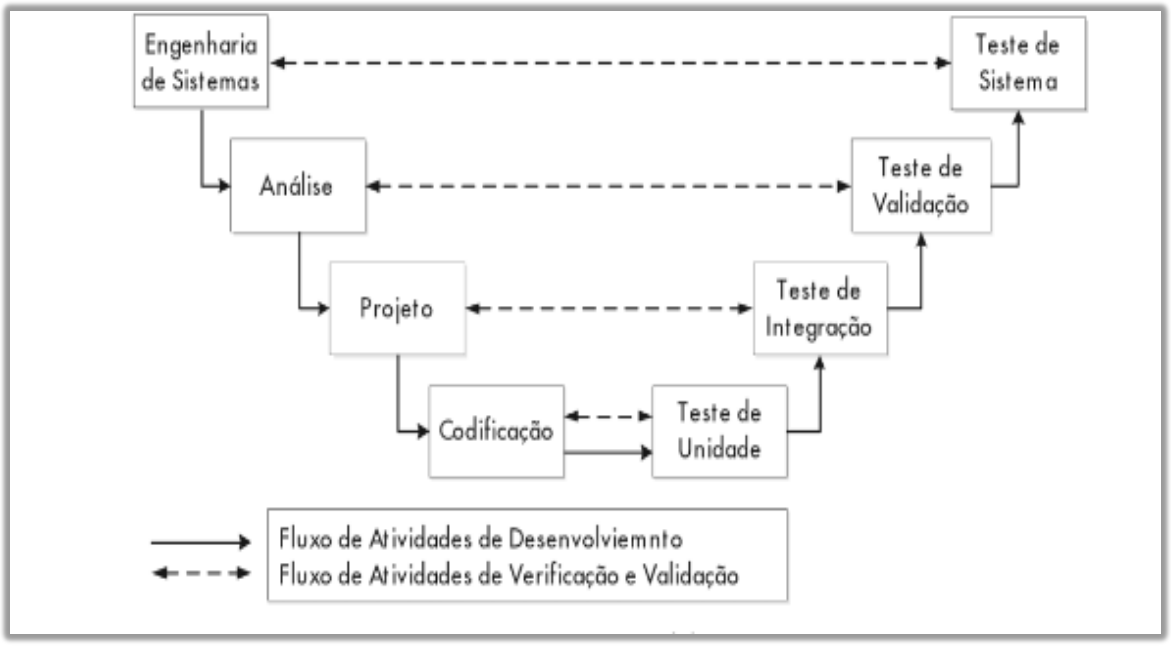
A composição de teste é uma etapa muito importante no ciclo de vida de um projeto. Não será possível uma aplicação correta do processo de teste caso o seu ambiente de teste não esteja adequado e coerente com os tipos de testes que serão realizados. Um esquema bastante interessante para análise é o Modelo V, pois cada etapa é seguida conforme o plano de teste correspondente. Na seção seguinte será discutida essa abordagem de aplicação de testes.

1.1 Modelo V

O modelo V é parecido com um processo de desenvolvimento sequencial, com a diferença de estar preocupado com o planejamento dos testes nas fases de construção do *software*. De forma geral, o Modelo V não é tão popular quanto ao modelo cascata, entretanto, em razão da sua simplicidade e melhor organização, se

torna um processo bastante atraente para disseminar a importância dos testes no processo de desenvolvimento *software* (KECHI, 2012, p.36). Na Figura 1, o lado esquerdo destaca o detalhamento e transformação de requisitos em código ao passo que o lado direito indica diferentes abordagens de testes, verificação e validação adequadas para cada fluxo de desenvolvimento.

Figura 1: Testes no ciclo de vida de *software* modelo V, fluxo de verificação e validação.



Fonte: Retirado de Kechi (2012, p.36).

1.1.1 Teste de Unidade

O teste de unidade concentra-se em cada unidade do *software*, conforme implementado no programa-fonte (classe ou componente). Possui como objetivo explorar a menor unidade do projeto, procurando falhas ocasionadas por defeitos de lógica, seu principal foco são os métodos dos objetos e até mesmo os pequenos trechos de código do sistema (MARTINS, 2007, p.15). Na visão de Pressman e Maxim (2016, p.474):

> O teste de unidade normalmente é considerado um auxiliar para etapa de codificação. O projeto dos testes de unidade pode ocorrer antes de a codificação começar ou depois que o código-fonte tiver sido gerado. Um exame das informações de projeto fornece instruções para estabelecer casos de testes que provavelmente mostrarão os erros.

1.1.2 Teste de Integração

Na etapa de teste de integração, o propósito é encontrar falhas provenientes da conexão interna dos componentes de um sistema. Geralmente, a maioria dos erros encontrados ocorre em razão de problemas na transferência de dados. Por exemplo, o componente A pode estar esperando retorno de um valor X ao executar um método do componente B; porém, esse retorna um valor Y, gerando um erro. Não faz parte do escopo desta etapa de teste o tratamento de *interfaces* com outros sistemas. Essas verificações são atentadas na fase de teste do sistema, apesar que, ao critério do gerente do projeto, essas *interfaces* podem ser testadas mesmo antes do sistema estar plenamente construído. Para Reisswitz (2009, p.53), é relevante verificar se o sistema se comporta conforme especificação do *design* arquitetural. No entendimento de Wazlawick (2011, p.309):

> O objetivo do teste de integração é verificar se os objetos se comunicam adequadamente. Isso pode ser feito de forma modular e sistemática. Estando os métodos básicos resolvidos, cabe verificar se os métodos delegados e operações de sistema funcionam conforme o esperado.

1.1.3 Teste de Validação

Ao ser realizado um teste de validação, deve-se verificar se as expectativas do cliente, descritas na especificação dos requisitos de *software*, elaborados durante a etapa de análise, estão sendo atendidas. Segundo Kechi (2012, p.37), esse teste fornece a garantia final de que o *software* realmente satisfaz a todos os requisitos funcionais e comportamentais desejados pelos agentes humanos envolvidos no projeto. Para Pezzè e Young (2009, p. 40):

> As atividades de validação procuram medir o quanto o sistema realmente atinge seus propósitos. As atividades de validação referem-se principalmente a especificação geral do sistema e ao código final. Com respeito à especificação geral do sistema, a validação busca discrepâncias entre as necessidades reais e a especificação do sistema tal como elaborada pela análise a fim de garantir que a especificação é um guia adequado para construir um produto que atingirá os objetivos.

1.1.4 Teste de Sistema

Nessa etapa de teste, a finalidade é executar o sistema com relação ao ponto de vista do seu usuário final, fazendo uma varredura em busca de falhas associadas aos objetivos originais. Para Reisswitz (2009, p.53), os testes de sistema são executados em condições similares de ambiente, *interfaces* sistêmicas e massas de dados àquelas que um usuário utilizará no seu dia-dia de manipulação do sistema. Na visão de Sommerville (2011, p.153), esse teste examina se os componentes são compatíveis, se interagem corretamente e se transferem os dados certos na hora certa.

Entretanto, existem duas diferenças importantes. A primeira é que durante o teste de sistema, os componentes reusáveis são desenvolvidos separadamente e os sistemas de prateleira podem ser integrados com módulos recém-desenvolvidos. Dessa maneira, o sistema completo é testado. Já a segunda diferença é que os componentes desenvolvidos por diferentes grupos ou pessoas da equipe podem ser integrados. Na verdade, o teste de sistema é um processo coletivo, não individual. Em algumas empresas, o teste de sistema pode envolver uma equipe independente, sem participação de projetistas e programadores. Quando você integra partes para criar um sistema, um novo comportamento é obtido. Portanto, esse teste deve focar em verificar interações entre componentes e objetos que formam um sistema.

2. QUALIDADE DE *SOFTWARE*

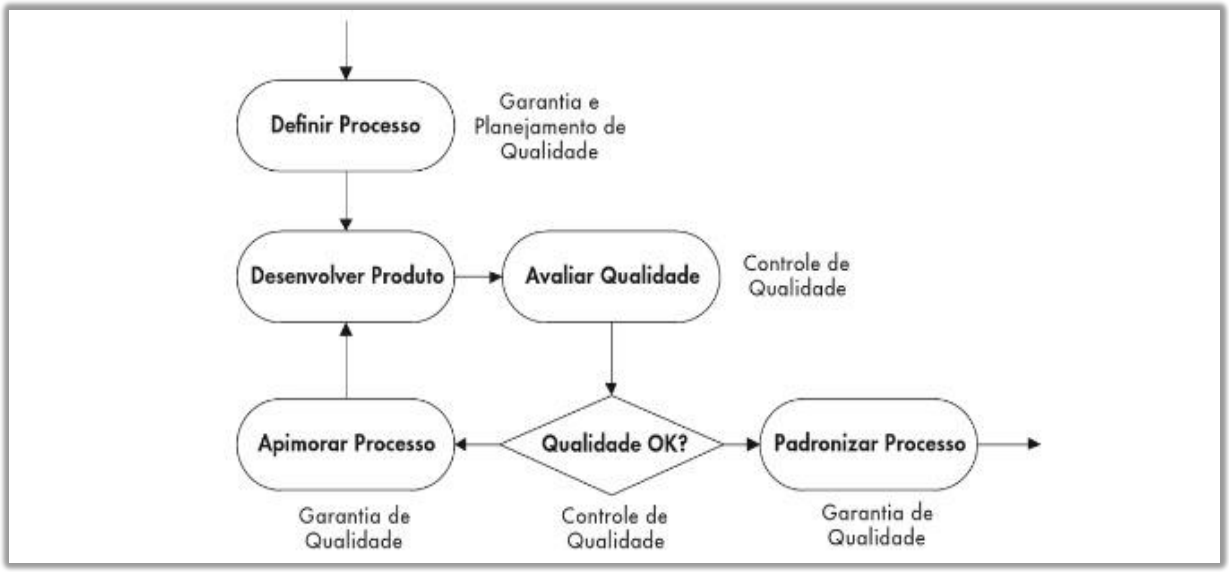
Qualidade é algo difícil de definir. Ainda mais complicado é garantir a qualidade de um produto, considerando que cada pessoa tem uma preocupação diferente para definir se um *software* realmente tem qualidade, essa questão é influenciada pelas expectativas de cada um sobre um produto ou serviço. Se os desejos são atendidos de alguma maneira, é possível dizer que esse produto ou serviço tem qualidade. A qualidade de *software* está ligada em entregar ao cliente o produto final que satisfaça suas expectativas, levando em consideração aquilo que foi acordado inicialmente por meio dos requisitos do projeto (KECHI, 2012, p.140).

Segundo a ISO 9000 (*International Organization For Standardization*), qualidade é o nível que um conjunto de características inerentes a um produto,

processo ou sistema cumpre com as características inicialmente estipuladas, podendo ser vista como a conformidade aos requisitos do projeto. Já garantia da qualidade do *software* é algo que deve ser analisado em todo o ciclo de evolução, com a finalidade de atingir padrões de qualidade acessíveis para processo e produto (ENGHOLM, 2010, p.275).

Em outras palavras, qualidade de *software* pode ser definida como conformidade aos requisitos funcionais e de desempenho explicitamente declarados, aos padrões de desenvolvimento claramente documentados e às características implícitas que são esperadas de todo *software* profissionalmente desenvolvido. Essa qualidade pode ser entendida como uma combinação complexa de fatores que variam de acordo com diferentes contextos, aplicações e clientes. Os fatores que afetam a qualidade do *software* podem ser categorizados em dois grandes grupos: o primeiro fator é aquele a ser medido diretamente (por exemplo, erros) e o segundo fator é o que pode ser medidos apenas indiretamente (por exemplo, usabilidade ou manutenibilidade). Segundo Pressman (1995, p.724), em cada caso deve ocorrer medição, é preciso comparar o *software* (programas, documentos e dados) com alguma referência e chegar a uma indicação de qualidade. A Figura 2 apresenta um possível processo de gerenciamento, controle e garantia de qualidade, retirado de (KECHI, 2012, p.148).

Figura 2: Gerenciamento de qualidade adaptado.



Fonte: Retirado de Kechi (2012, p.148).

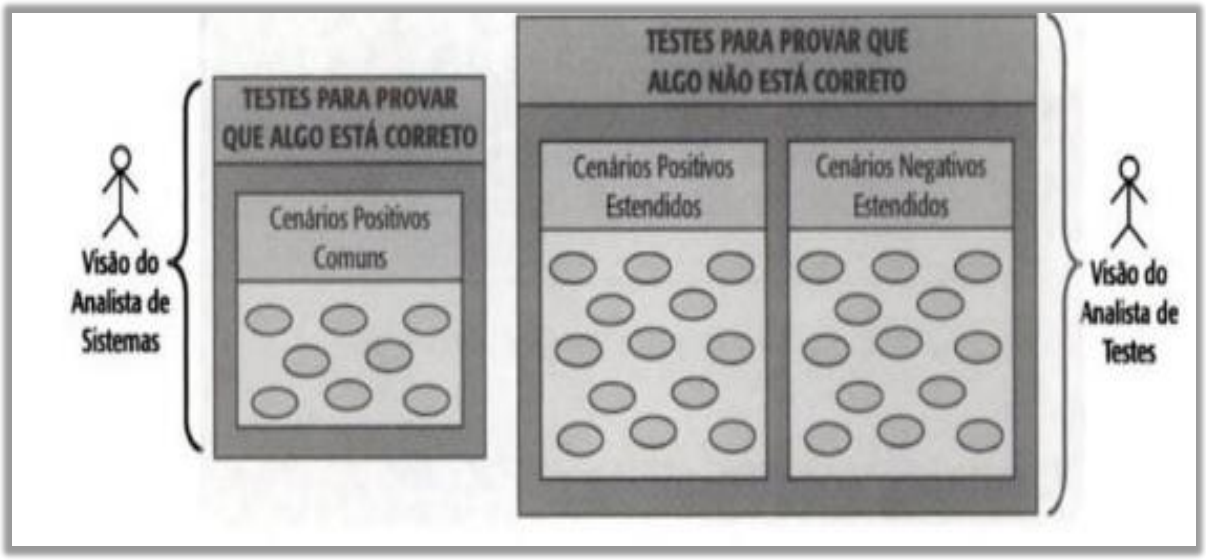
3. GERENCIAMENTO DE TESTES

O gerenciamento de testes está totalmente associado a aspectos de qualidade de um *software* e às versões que acomodam correções e/ou alterações do sistema. Durante o gerenciamento, pode ocorrer decisão pela reutilização de testes, definindo estratégias econômicas para tempo e custo, além do aumento da produtividade. O gerenciamento de testes está presente em diversas etapas, por exemplo, desenvolvimento, testes e produção (BARTIÉ, 2002, p.1). Uma das vantagens de um gerenciamento bem definido é obter um nível maior de qualidade nos trabalhos desenvolvidos por uma equipe que presta serviços para determinada organização. Com isso, podem-se avaliar vários benefícios: antever falhas, controlar processo de testes, reduzir custos associados ao ciclo de vida da aplicação, alcançar maior produtividade para atender prazos, obter clientes mais satisfeitos, desenvolver sistemas com maior vantagem competitiva, evitar desperdício de tempo e minimizar ônus com manutenção, reduzir custo entre desenvolvimento e teste antes de colocar o sistema em funcionamento no cliente, além de construir melhor *scripts* para a automação dos testes (GANDARA, 2012, p.23).

> A estratégia de teste deve ser planejada sob medida para cada projeto, considerando o tempo que será investido neste trabalho, a disponibilidade de recursos e a tecnologia utilizada na construção do *software.* Para planejar o teste do *software* de forma adequada, devem ser coletadas métricas durante o teste e fazer uso de modelos existentes de confiabilidade de *software*, de modo que possam ser estabelecidas diretrizes significativas para que a equipe saiba quando parar de testar (MARTINS, 2007, p.15).

Para obter sucesso em uma identificação de erros, no gerenciamento de testes deve-se levar em consideração que o teste não é feito para provar que tudo está correto e funcionando bem em um *software*, pois nesse caso está sendo envolvido somente um cenário favorável para a sua utilização, ou seja, um cenário comum de uso. Por outro lado, quando se tem o objetivo de provar que há algo de errado ou inadequado, devem-se analisar cenários desfavoráveis, investigando além dos cenários comuns e positivos (BARTIÉ, 2002, p.21). Na Figura 3 pode-se ver a representação e a comparação entre diferentes perspectivas de testes. Analista de Sistemas e Analista de Testes possuem diferentes visões sobre a realização dos procedimentos de testes, podendo desejar provar se algo está correto ou se algo não está correto nos cenários sendo analisados.

Figura 3: Visões dos testes por perspectivas diferentes.



Fonte: Retirado de Bartié (2002, p.21).

O programador (desenvolvedor) possui suas responsabilidades na tarefa de teste do *software*, assumindo as atribuições de testar os componentes do *software* e garantir que suas funções estão realmente implementadas e funcionando conforme esperado. Os testes do sistema podem ser iniciados logo após a finalização da arquitetura do *software*, com isso é feito um teste abrangente de modo a analisar se todos os requisitos foram atendidos. Outra questão que deve ser gerenciada são os testes de validação. Nesse momento, o *software* já está totalmente montado e pronto para entrar em funcionamento no cliente, todas as saídas do usuário devem ser verificadas, devendo estas atender aos requisitos do sistema. Segundo Martins (2007, p.15), testes são executados para forçar falhas no sistema de modo a identificar erros novos e/ou não previstos pela equipe.

> Existem ferramentas de planejamento da qualidade que podem programar atividades de geração e execução de testes consistentes com as dependências entre as atividades de qualidade e entre as atividades de qualidade e desenvolvimento. Também podem reconhecer a entrega de um dado artefato, programar automaticamente a execução de um conjunto de testes, notificar um projetista de *software* sobre os resultados, registrar o tempo real de execução da atividade e sinalizar desvios do cronograma para os gerentes de qualidade (PEZZÈ; YOUNG, 2008, p.469).

4. AUTOMAÇÃO DE TESTES

Não se pode dizer que um *software* está livre de erros, mas este deve chegar a um ponto em que seja aceitável e atenda aos requisitos solicitados pelo cliente. Todavia, para que isso ocorra, ele passa por infinitas etapas de testes antes de ser implantado. A realização dos testes não é algo simples, pois exige etapas exaustivas que devem ser repetidas inúmeras vezes.

Por muito tempo, os testes eram feitos de forma manual, sendo realizados a partir de hipóteses e questões fictícias, gerados pelo testador. Com a existência de poucas ferramentas, deixava cada vez mais esse processo problemático e demorado, provendo limitações para execução desses procedimentos. Em razão da evolução tecnológica, pode-se afirmar que houve aumento na complexidade dos sistemas, tornando as tarefas de testes mais complexas (GANDARA, 2012, p. 27).

A decisão de adotar a automação dos testes depende de uma verificação das etapas manuais, básicas e fundamentais, estarem incorporadas adequadamente ao processo, do contrário, a automação poderá não fornecer uma solução eficaz. A automação pode ser uma ótima solução para os testadores, mas não torna tudo mais fácil, nem o trabalho mais simples. Por exemplo, o custo é elevado, pois além de agregar complexidade exige esforço de implantação. É necessário cuidado e organização, aceitando as regras impostas pela ferramenta, para que se possa ter um teste bem executado. A automação busca simular usuários e/ou operações efetuadas por seres humanos, não são executados procedimentos manuais durante esses testes. A correta aplicação da automação depende da existência de procedimentos manuais bem definidos e efetuados (RIOS; MOREIRA, 2013, p.162).

Para Gandara (2012, p.26), a etapa de teste não é um momento simples de ser executado, pois não se resume em colocar pessoas aleatórias apertando botões para tentar encontrar problemas no sistema, é necessário seguir uma metodologia, um roteiro sistemático para esse processo. Uma abordagem disciplinada permite ganho de tempo, controle e confiabilidade aos procedimentos de teste.

> Pode diminuir os riscos da entrada em operação dos sistemas, [...] um sistema de testes automatizados permitirá a diminuição do tempo, um melhor acompanhamento, o planejamento dos testes e, quando o sistema sofrer uma manutenção, você poderá acompanhar rapidamente a sua atualização, por estar com os testes anteriormente realizados, arquivados para serem novamente executados, validando as funcionalidades antes testadas (GANDARA, 2012, p. 27).

A Tabela 1 apresenta um comparativo entre testes manuais e testes automatizados. Pode-se observar que, geralmente, os processos baseados em ferramentas automatizadas apresentam melhorias em relação aos testes manuais, especialmente, durante a sua execução. A etapa de planejamento é uma exceção, pois exige mais horas para sua realização na abordagem apoiada por computador.

Tabela 1: Comparativo entre testes manuais e automatizados.

Etapas de Testes	Teste Manual	Teste Automatizado	Melhoria (%)
Planejamento	32	40	-25%
Definição dos Casos de Testes	262	117	55%
Execução dos Testes	466	23	95%
Conferência dos Testes	117	58	50%
Gerenciamento do Erro	117	23	80%
Relatórios Finais	96	16	83%
Duração Total (em Horas)	1.090	277	75%

Fonte: Retirado de Bartié (2002, p.64).

Para Rios e Moreira (2013, p.161), os testes das infinitas possibilidades de uso de *interfaces* gráficas e os testes de desempenho e estresse de um sistema cliente/servidor são exemplos de situações difíceis de serem realizadas sem auxílio de uma ferramenta de automação. Alguns sistemas podem alcançar mais de 1000 componentes, o que torna mais complexo a execução do processo de teste como um todo e o seu gerenciamento.

CONSIDERAÇÕES FINAIS

Este estudo permitiu identificar que, cada vez mais, empresas de desenvolvimento de sistemas necessitam adotar processos de teste de *software* eficientes. Isto é importante, pois construir um *software* com qualidade demanda a utilização técnicas de gerenciamento e automação de testes para que o produto seja entregue ao cliente com o máximo de conformidade e confiabilidade. A implantação do teste em um projeto de *software* ajuda a identificar e obter uma visão completa dos erros a serem analisados e corrigidos ao longo do processo de desenvolvimento de *software*. Para isto, devem-se definir etapas de gerenciamento e automação dos testes que contribuem para redução do tempo e custo gastos com testes.

Espera-se que essa pesquisa bibliográfica possa contribuir em futuros trabalhos a serem desenvolvido nesta área de gerenciamento e automação de teste, auxiliando desenvolvedores acadêmicos e comerciais, em seus futuros testes para o desenvolvimento de *software* com qualidade. O projeto buscou mostrar também o porquê da utilização e supostos benefícios da automação e do gerenciamento de testes, relatando ganhos de qualidade e agilidade alcançados no desenvolvimento do *software*. Uma das limitações enfrentadas nos estudos sobre gerenciamento e automação de testes de *software* é a amplitude do assunto, em razão da existência dos diversos tipos, níveis e ferramentas de testes. É possível, em projetos futuros, realizar todas estas etapas de gerenciamento e automação, a fim de colocar em prática cada abordagem para obter conclusões concretas de quais testes são mais adequados a depender do domínio de estudo.

REFERÊNCIAS

BARTIÉ, Alexandre. **Garantia da Qualidade de *Software***. 5ª Ed., Rio de Janeiro: Elsevier, 2002.

ENGHOLM, Hélio. **Engenharia de *Software* na Prática**. Rio de Janeiro: Novatec, 2010.

GANDARA, Fernando. **Qualidade e Teste em *Software***. Joinville: Clube dos Autores, 2012.

KECHI, Hirama. **Engenharia de *Software***. Rio de Janeiro: Elsevier, 2012.

MARTINS, José Carlos Cordeiro. **Técnicas para Gerenciamento de Projetos de *Software***. Rio de Janeiro: Brasport, 2007.

PEZZÈ, Mauro; YOUNG, Michal. **Teste e Análise de *Software***. Porto Alegre: Bookman, 2009.

PRESSMAN, Roger. **Engenharia de *Software.*** São Paulo: Makron Books, 1995.

PRESSMAN, Roger; MAXIM, Bruce. **Engenharia de *Software***. 8ª Ed., São Paulo: McGraw-Hill, 2016

REISSWITZ, Flavia. **Análise e de Sistemas V. 4**. Joinville: Clube de Autores, 2009.

RIOS, Emerson; MOREIRA, Trayahú. **Teste de *Software***. Rio de Janeiro: Alta Books, 2013.

SOMMERVILLE, Ian. **Engenharia de *Software***. 9ª Ed., São Paulo: Addison Wesley, 2011.

WAZLAWICK, Raul Sidnei. **Análise e Projeto de Sistemas da Informação**. 2ª Ed., Rio de Janeiro: Elsivier, 2011.