UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI
Bacharelado em Sistemas de Informação
Raissa Oliveira Rodrigues

Testes de *Software*: levantamento de dados e informações sobre o trabalho envolvido no dia-adia dos profissionais da área

Diamantina
2019

Raissa Oliveira Rodrigues

Testes de *Software*: levantamento de dados e informações sobre o trabalho envolvido no dia-adia dos profissionais da área

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação, como parte dos requisitos exigidos para a conclusão do curso.

Orientador: Profª. Dra. Caroline Queiroz Santos

Diamantina
2019

Raissa Oliveira Rodrigues

Testes de *Software*: uma avaliação sobre a importância e aplicabilidade no cenário atual dos profissionais

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação, como parte dos requisitos exigidos para a conclusão do curso.

Orientador: Profª. Dra. Caroline Queiroz Santos

Data de aprovação 18/07/2019

Profª. Dra. Caroline Queiroz Santos
Departamento de Computação - UFVJM

Prof. Me. Luis Henrique Rodrigues
Departamento de Computação - UFVJM

Prof. Me. Marcelo Ferreira Rego
Departamento de Computação - UFVJM

Diamantina
2019

Ao meu noivo Gustavo pelo apoio incondicional.

AGRADECIMENTOS

Aos meus pais, Silvanio e Vânia, pelo amor e pelo suporte fornecido durante todo o curso de graduação.

Ao meu noivo Gustavo, por todo amor, amizade, companheirismo e incentivo, sem os quais eu não teria conseguido chegar até aqui.

À minha orientadora, a Prof.ª Drª. Caroline Queiroz Santos, pela disponibilidade, atenção e valioso incentivo durante a etapa final da graduação.

Aos professores do curso, principalmente o Professor Erinaldo e a Professora Geruza, pela disponibilidade, compreensão e pelo apoio em momentos difíceis.

RESUMO

Garantir a qualidade do produto de *software* é a diretriz principal para os profissionais envolvidos na construção desse produto. Os testes, nesse contexto, são imprescindíveis para se obter um *software* de qualidade com o menor número de erros possível. Cada vez mais cresce a demanda por sistemas de *software* mais complexos, distribuídos, envolvendo tecnologias emergentes e conexões com outros serviços disponíveis na Internet. Com isso, procurar erros nos artefatos e no produto de *software* tem sido igualmente complexo, desencadeando uma maior demanda de profissionais qualificados nesta área. A partir da percepção dos avanços da área de testes apresentados na literatura, surgiu o interesse em descobrir como os profissionais têm trabalhado com testes e quais habilidades são necessárias para realizar esse trabalho. Para isso, profissionais de teste foram convidados a responder um questionário online com questões fechadas e abertas relacionadas ao seu perfil pessoal e ao seu trabalho. O objetivo foi identificar os principais desafios encontrados por estes profissionais no exercício da sua função, em relação às técnicas, metodologias, ferramentas e linguagens de programação. Nossos resultados apontam que os participantes encontram limitações para exercício do seu trabalho relacionadas a restrições de tempo, outros disseram ter dificuldades em automatizar os testes, programar, estudar novas linguagens de programação, entre outros.

Palavras-chave: Qualidade de *Software*. Testes de *Software*. Perfil dos profissionais.

ABSTRACT

Ensuring quality is a main goal for professionals involved in building software. Testing, in this context, is essential to achieve high-quality products with as few errors as possible. There is a growing demand for more complex, distributed software systems based on emerging technologies and connections to internet services. The expanding complexity of software has made errors even more complex, triggering a greater need for good testing professionals. Thus, finding errors in artifacts and software products has been equally complex, triggering a greater need for qualified professionals in this field. As we realize the advances in software testing, as seen in the literature, there was an interest to acknowledge how professionals have been working as testers and what skills are needed to perform those tasks. To this end, test professionals were asked to answer an online questionnaire with closed and open questions concerning their personal profile and their working habits. Our goal was to identify the main challenges encountered by these professionals on the practise of their functions ans their relationship with techniques, methodologies, tools and programming languages. Our results indicate that some participants find work limitations related to time constraints, others reported difficultties when automating testing, programming and studying new programming languages, among other obstacles.

Keywords: Software Testing. Software Quality. Work Profile.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API - *Application Programming Interface.*

BDD - *Behavior Driven Development.*

GUI - *Graphical User Interface.*

IEEE – *Institute of Electrical and Electronic Engineers.*

REST - *Representational State Transfer.*

SDK - *Software Development Kit.*

SQL - *Structured Query Language.*

TDD – *Test Driven Development.*

TI - Tecnologia da informação.

XP - *Extreme Programming.*

SUMÁRIO

1 INTRODUÇÃO

Por muitos anos, existia uma única maneira de defesa contra os erros de programação: criar um projeto cuidadoso, contando com a inteligência do programador. Nos tempos atuais, as modernas técnicas de projeto e revisões técnicas estão contribuindo para diminuir a quantidade de erros encontrados, por meio da realização de testes e atividades de garantia da qualidade de *software* (PRESSMAN, 2011). Com isso, diferentes métodos de teste estão sendo agregados às várias metodologias e processos existentes.

A realização de testes de software tem a finalidade de diminuir os riscos causados por defeitos que vêm desde antes do desenvolvimento (codificação). Ao realizar testes em todo o sistema, adquire-se um produto com melhor qualidade e maior segurança (BASTOS *et al.*, 2012). Ao associar processos rigorosos e boa documentação ao teste, pode-se contribuir para a redução de erros, especialmente quando detectados e corrigidos em seguida (OLSE *et al.*, 2018).

Na década de 70, a atividade de teste de *software* era vista como uma tarefa complementar, que era realizada apenas pelos próprios desenvolvedores (SOUZA; GASPAROTTO, 2013). A falta de investimento e importância atribuída aos testes geravam uma grande ocorrência de defeitos, principalmente quando os sistemas já estavam em produção. De acordo com Bastos *et al.* (2012), os problemas atingiram grandes proporções quando começaram a surgir sistemas para internet, pois, com isso, a imagem das empresas ficou mais exposta ao seu público. Além disso, segundo os autores, as aplicações tornam-se mais complexas com os avanços das novas tecnologias (BASTOS *et al.*, 2012). Rios e Moreira (2013) argumentam que um teste mal realizado pode gerar graves problemas, como fraudes, incorreções , bloqueios de *sites*, entre outros. Um *software* com defeitos e problemas de desempenho na execução pode provocar frustração nos usuários, comprometendo a imagem e a confiança da empresa que o desenvolveu.

No entanto, as ações realizadas pelos profissionais que trabalham com a garantia de qualidade de *software* e, naturalmente, com testes, revelam muitos erros, mas não são suficientes. Por isso, é necessário executar o *software* diversas vezes antes que o cliente encontre os erros, o que fortalece a indicação de que os testes sejam executados sistematicamente, envolvendo especialistas na medida em que o processo avançar (PRESSMAN, 2011). Afinal, quanto mais cedo os erros forem identificados e corrigidos, menores serão os custos totais para o desenvolvimento do produto.

Outro papel importante da realização de testes de *software* é a busca pela melhoria na manutenção dos sistemas. Uma grande parte do orçamento de TI das organizações é direcionada à manutenção dos sistemas de *software* após o início da produção. A maioria dos testes realizados durante a manutenção é realizada, também, durante o desenvolvimento. Isso mostra que quanto melhores os testes realizados durante o desenvolvimento do *software*, menores serão os custos de manutenção (RIOS; MOREIRA, 2013).

Na graduação, aprendemos nas disciplinas da área de Engenharia de *Software* que

a atividade de teste é essencial no desenvolvimento de um *software*, pois mesmo com todas as técnicas, métodos e ferramentas utilizados, o produto ainda podem conter erros. Com isso, após conhecer a teoria sobre este assunto, surgiu o interesse em investigar como tem sido o trabalho dos profissionais de teste, o que eles fazem e suas percepções sobre a realização de testes.

1.1 Motivação

Com o advento dos métodos ágeis para desenvolvimento de *software*, muitas atividades presentes nos processos tradicionais foram adaptadas para o contexto ágil. No entanto, a necessidade de as empresas produzirem softwares de qualidade, no prazo acordado e atendendo as necessidades dos clientes é o que norteia todo esse processo. Essa mesma necessidade tem feito com que o interesse das empresas e/ou dos times de desenvolvimento de *software* por métodos ágeis aumente. Com isso, muitas empresas têm adotado o desenvolvimento ágil de *software*.

A motivação para a realização deste trabalho surgiu do interesse e da curiosidade em saber como os testes tem sido realizados no desenvolvimento de *software*, quais habilidades os profissionais de teste precisam ter, quais tecnologias são usadas nessas atividades. Como a qualidade é o objetivo principal ao se construir um produto de *software* e, com isso, a realização de testes é uma atividade fundamental na verificação da qualidade, nos interessamos em obter a visão dos profissionais da área sobre a importância atribuída aos testes no seu contexto de trabalho.

Além disso, acredita-se que o aumento da complexidade dos *softwares* aliado à exigência das empresas por sua qualidade impulsiona a área de testes e, consequentemente, aumenta a necessidade de novas técnicas, metodologias e ferramentas para facilitar e otimizar os processos. Diante disso, surgiu o interesse em pesquisar os desafios encontrados por profissionais de testes de *software* no desempenho do seu trabalho. Esta temática revela-se ainda mais relevante no contexto do constante crescimento da área com novas tecnologias surgindo a todo instante.

1.2 Objetivos

1.2.1 Objetivo Geral

Identificar os principais desafios enfrentados pelos profissionais de teste de *software*, no exercício da sua função, em relação às técnicas, metodologias, ferramentas e linguagens de programação utilizadas por eles.

1.2.2 Objetivos Específicos

Os objetivos específicos desse trabalho são:

- Aprofundar os estudos e conhecimento acerca do conceito de testes de *software*;

- Investigar como os profissionais de teste trabalham e as suas percepções acerca do trabalho que realizam;
- Avaliar os contextos de teste pela perspectiva do testador (ou do profissional da área de teste);
- Analisar e discutir os resultados obtidos, relacionado-os com os propósitos da garantia de qualidade de software encontrados na literatura.

1.3 Organização do trabalho

Este trabalho é composto por 6 (seis) capítulos, sendo este primeiro a introdução. O segundo capítulo apresenta os principais conceitos referentes ao tema proposto, começando por Teste de *Software*, Ambientes para Teste e Metodologias Ágeis. O terceiro capítulo exibe alguns trabalhos relacionados ao tema proposto, seguido pelo quarto capítulo, que apresenta a metodologia de pesquisa utilizada para atingir os objetivos com a descrição das atividades realizadas. No quinto capítulo são apresentados os resultados obtidos com a pesquisa e a análise destes. Por fim, o sexto capítulo traz as conclusões e sugestões de possíveis trabalhos que poderão ser realizados no futuro.

2 REFERENCIAL TEÓRICO

Este capítulo aborda os principais conceitos referentes ao Tesde de *Software*, Tipos de Teste, Ferramentas de Teste, *Frameworks* de Teste, Linguagens de Programação, Tipos de Aplicações e Metodologias de Desenvolvimento Ágil. Esses conceitos fundamentam a base teórica que sustenta a proposta deste trabalho.

2.1 Princípios e Conceitos de Testes

Nesta seção, serão abordados alguns conceitos relacionados ao teste de *software* como Níveis de Teste, Técnicas de Teste e Documentação de Teste.

2.1.1 Teste de *Software*

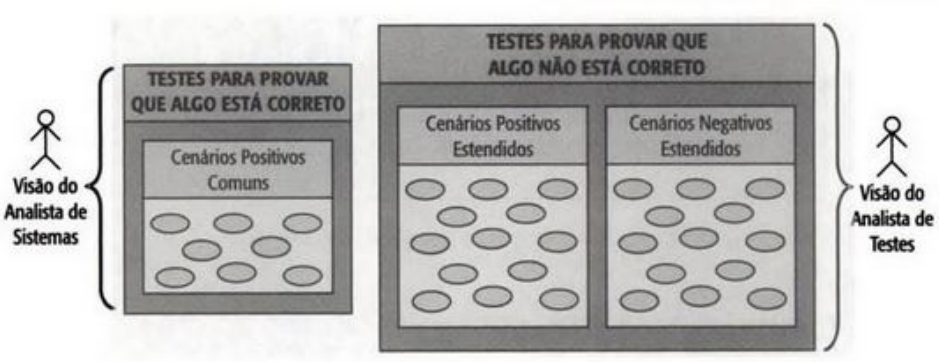
Durante o desenvolvimento de um *software*, são adotados muitos métodos, técnicas, ferramentas e processos. A utilização desses mecanismos é necessária para produzir *software* de alta qualidade. Apesar da grande estrutura empregada na produção, ainda é possível que defeitos permaneçam no produto. Por esse motivo, durante o desenvolvimento são realizados vários testes para avaliar e eliminar defeitos. De acordo com Pereira (2017), a atividade de testes adiciona algum valor ao *software*, aumentando a qualidade e confiabilidade do programa, ao encontrar e remover erros. Deve-se começar com a suposição de que o programa contém erros e, em seguida, testar o programa para encontrá-los.

O processo de teste é visto como uma atividade de Validação e Verificação. Atividades de Verificação buscam garantir que a implementação e/ou especificação do sistema estejam corretos, ou seja, referem-se ao conjunto de atividades que procura garantir que o sistema implementa corretamente uma determinada função. Validação refere-se ao conjunto de atividades que procura garantir que o sistema esteja de acordo com os requisitos do usuário (PRESSMAN, 2011).

Os testes são divididos em níveis para serem utilizados em diferentes atividades, de acordo com cada objetivo. A maioria dos objetivos de teste estão relacionada à verificação. Em geral, apenas o teste de aceitação é efetuado visando a validação do produto (WAZLAWICK, 2013). De acordo com Poppendieck e Poppendieck (2011), para os princípios do desenvolvimento lean de *software*, é um mito afirmar que a função dos testes é encontrar defeitos:

> ”O trabalho de testagem, e das pessoas que desenvolvem e rodam testes, é previnir defeitos, e não encontrá-los. Uma organização que busca garantir a qualidade deveria promover processos que constroem qualidade no código desde o início, em vez de testar a qualidade no final[...] encontrar defeitos deveria ser a exceção, não a regra...”(POPPENDIECK; POPPENDIECK, 2011, p.51).

Figura 1 – Visões dos testes por perspectivas diferentes.



Fonte: (BARTIé, 2002)

A figura 1 mostra que a visão do Analista de Sistema é bem diferente da visão do Analista de Testes. Enquanto um quer testar para provar que algo está correto, o outro quer testar para provar que algo não está correto. Por isso, a visão do Analista de Testes é mais ampla, pois, de acordo com Bartié (2002), é muito mais fácil provar que ”algo funciona”do que provar que ”algo não funciona”. A figura mostra também a importância das diferentes visões do sistema para a busca de erros.

Os testes podem ser manuais ou automatizados. A execução manual de um caso de teste é rápida e efetiva, mas a execução e repetição de um grande conjunto de testes manuais acaba se tornando uma tarefa muito trabalhosa e cansativa (BERNARDO; KON, 2008). Por esse motivo, muitos profissionais preferem automatizar a maior parte de seus testes.

Os testes automatizados são programas ou *scripts*[1] simples que exercitam funcionalidades do sistema testado e fazem verificações automáticas nos efeitos colaterais obtidos. Essa abordagem possui uma grande vantagem; todos os casos de teste podem ser facilmente e rapidamente repetidos a qualquer momento e sem muito esforço (BERNARDO; KON, 2008).

Alguns testes, como o de carga em ambiente *web*, necessitam de auxílio de ferramentas de automação de testes para que possam ser realizados, pois o ambiente real, nesse caso, é muito difícil de ser simulado manualmente (BASTOS *et al.*, 2012).

Automatizar os testes nada mais é do que repassar para o computador as atividades que geralmente são processadas de forma manual. A automação de testes deve começar a partir de uma atividade manual de teste já estabelecida e consolidada. No mercado existem várias ferramentas pagas ou gratuitas disponíveis.(BARBOSA; TORRES, 2011 apud LAGES, 2010).

A automação pode ser a solução de muitos problemas, porém é ilusão achar que a automação deixará a vida dos testadores mais fácil, seu trabalho mais simples e que os prazos serão sempre cumpridos. Automação de testes vai muito além de adquirir uma ferramenta pois não adianta querer automatizar se a organização não possuir um processo consolidado de testes. Contudo não há dúvida de que seja um apoio indispensável em casos específicos de testes especialmente quando realizado por pessoas qualificadas. (RIOS; MOREIRA, 2013).

[1] *Script* é um conjunto de instruções em código, escritas em linguagem de computador.

2.1.2 Estágios (ou Níveis) de Teste

Os níveis de testes ou estágio de testes, referem-se às medidas do teste que caracterizam em qual fase do desenvolvimento um determinado teste deve ser aplicado. De acordo com Bastos *et al.* (2012), os níveis de teste estão divididos em:

- Testes de Unidade: são testes aplicados aos menores componentes de código criados. Geralmente são realizados pelo programador e verificam o funcionamento de fragmentos do sistema ou do *software* de forma individual: funções, objetos e componentes.
- Testes de Integração: são realizados ao término de cada interação, dentro de um ambiente operacional controlado e executados em uma combinação de componentes para verificar se eles funcionam corretamente ao serem integrados.
- Testes de Sistema: realizados dentro de um ambiente controlado, testando suas funções e apurando se o *software* está executando corretamente o que foi definido em seus requisitos.
- Testes de Aceitação: são testes realizados pelos usuários do programa para validá-lo e liberá-lo antes de sua utilização no ambiente de produção. Esses testes são realizados com suporte da equipe de testes e da equipe do projeto.

2.1.3 Técnicas de Teste

Existem muitas maneiras de testar um *software*, todas elas com o objetivo de encontrar e corrigir erros. Técnica é o processo que vai assegurar o correto funcionamento de alguns aspectos de *software* ou de sua unidade. As principais técnicas de testes estão separadas em dois grupos: estrutural e funcional (SILVA *et al.*, 2016).

Para detecção de um maior número de erros inseridos durante a codificação do programa ou na codificação das funcionalidades que foram definidas através das especificações, é necessário escolher qual a técnica mais adequada à situação. Segue, abaixo, a definição de técnicas que podem ser adotadas para a detecção de erros, de acordo com Bastos *et al.* (2012):

2.1.3.1 Técnica Estrutural

A Técnica Estrutural, também conhecida por Testes de Caixa Branca, visa verificar o código, a lógica interna do componente que foi codificado, as configurações e outros elementos técnicos, com o objetivo de garantir que o produto tenha uma estrutura sólida e funcione de acordo com as funcionalidades que foram definidas.

- Teste de Estresse: avalia o comportamento do *software* ao submetê-lo a condições extremas como, por exemplo, restrições de memória ou grandes volumes de dados acima das médias esperadas.

- Teste de Execução: avalia o comportamento do sistema, determinando se o sistema pode atingir critérios de desempenho específicos, como o tempo de resposta, tempo de desempenho e a performance.
- Teste de Recuperação: estima se o *software* é capaz de dar continuidade à suas operações após forçá-lo a falhar, verificando se sua recuperação é executada da maneira correta.
- Teste de Operação: é realizado quando ocorre a integração da aplicação com o ambiente operacional, verificando se o sistema é executável no decorrer da operação normal.
- Teste de Conformidade: é realizado para garantir que a aplicação está de acordo com as metodologias.
- Teste de Segurança: este teste é realizado para assegurar a confidencialidade das informações, resguardando os dados contra acessos inapropriados realizados por terceiros.

2.1.3.2 Técnica Funcional

A Técnica Funcional, também conhecida como Testes de Caixa Preta, possui a finalidade de garantir que os requisitos e especificações do sistema estão de acordo com o que foi especificado. Esta técnica não se baseia em qualquer conhecimento do código e da lógica interna do componente.

- Testes de Requisitos: verificam se o sistema executa as funcionalidades corretamente por um período determinado sem a presença de falhas.
- Testes de Regressão: são realizados após a implementação de uma modificação em segmentos do *software* que, devido à alteração, precisam ser testados novamente.
- Testes de Tratamento de Erros: observam a capacidade do sistema para tratar erros, quando sujeito a transações incorretas.
- Testes de Interconexão: conferem se a comunicação de dados entre os *software* de aplicação está de acordo com o previsto.
- Testes de Controle: utilizam controladores. Entre eles estão a validação de dados e a integridade de arquivos.
- Testes Paralelo: são técnicas que comparam o processamento da versão antiga de um sistema com uma versão nova, verificando se há consistência entre elas.

2.1.4 Outros Tipos de Testes

Nesta seção serão citados outros tipos de testes, de acordo com Rios e Moreira (2013).

- Testes de Configuração: esses tipos de testes são utilizados para verificar se o *software* está apto para ser executado em diferentes versões ou configurações de ambientes como, por exemplo diferentes *browsers*.
- Testes de Instalação: analisam o processo de instalação ou atualização do sistema.
- Testes de Usabilidade: são utilizados para verificar o nível de facilidade e dificuldade do usuário para utilização do produto. É um teste subjetivo, baseado em opiniões de usuários, coletadas através de entrevistas ou outras pesquisas.
- Testes de Volume: submetem o sistema a grandes quantidades de dados para determinar quais os limites para a causa de uma falha (BASTOS *et al.*, 2012).

2.1.5 Documentação de Teste

Atualmente, para tratar a documentação que deve ser usada em projetos de teste de *software*, é utilizada a norma IEEE 829:2008. Até o momento, este documento é o único padrão existente no mercado para definir os documentos básicos que devem ser produzidos nos projetos de teste de *software* (RIOS; MOREIRA, 2013).

2.1.5.1 Plano Master de Teste

De acordo com Rios e Moreira (2013), a última versão da norma estabelece que deveria existir um Plano de Teste para cada nível de teste, sendo eles: teste unitário ou de componente, teste de integração, teste de sistema e teste de aceitação. O Plano Master de Teste deve ser considerado como o documento estratégico e tático.

2.1.5.2 Plano de Teste

O documento Plano de Teste é uma maneira de documentar o projeto de teste junto com outros documentos, permitindo que estes possam ser repetidos e controlados (BASTOS *et al.*, 2012).

É utilizado na tarefa de planejamento para execução do teste no qual serão identificadas as funcionalidades a serem testadas, permitindo definir o nível de cobertura de acordo com as prioridades estabelecidas. Esse documento é de responsabilidade da equipe de testes (IEEE, 2008).

2.1.5.3 Especificação do Projeto de Teste

Especifica os detalhes da abordagem do teste e identifica as características a serem testadas. A tarefa de especificação de testes é composta pelos documentos Especificação do Projeto de Teste, Especificação de Caso de Teste e Especificação de Procedimento de Teste (IEEE, 2008).

2.1.5.4 Especificação de Caso de Teste

Identifica e define o conjunto de casos de teste a serem realizados com os dados de entrada, resultados esperados, ações e condições gerais para a execução do teste (IEEE, 2008).

2.1.5.5 Especificação de Procedimento de Teste

A Especificação de Procedimento de Teste identifica quais serão os passos para executar os casos de teste. A criação desse documento é de responsabilidade dos líderes de projetos, da equipe de teste e dos usuários (IEEE, 2008).

2.1.5.6 Relatórios de Teste

De acordo com IEEE (2008), os relatórios de teste são compostos por quatro documentos: Diário de Teste, Relatório de Incidente de Teste, Relatório-Resumo de Teste e Relatório de Encaminhamento de Item de Teste.

- Diário de Teste: também conhecido por Log de Teste, exibe os registros cronológicos dos dados relevantes relacionados com a execução dos testes.
- Relatório de Incidente de Teste: também conhecido por Relatório de Defeitos, documenta qualquer evento que ocorra durante a atividade de teste e que necessite de análise posterior. No qual são relatados os erros que podem ocorrer durante da execução do teste. Esses erros podem ser causados tanto pela falha na construção do teste como por erros de programação. Após o registro do erro o relatório deverá ser encaminhado ao responsável pela correção.
- Relatório Resumo de Teste: mostra um resumo dos resultados das atividades de teste associadas com uma ou mais especificações de projeto de teste e provê avaliações baseadas nesses resultados.
- Relatório de Encaminhamento de Item de Teste: identifica os itens encaminhados para teste no caso de equipes distintas serem responsáveis pelas tarefas de desenvolvimento e de teste.

2.2 Ambientes para Teste

É necessário planejar o ambiente para teste de acordo com a aplicação a ser testada. Essa fase em questão deve ser executada antes da elaboração dos cenários de teste. O ambiente não é apenas uma configuração de *hardware*, mas também a estrutura utilizada para executar o teste. (BASTOS *et al.*, 2012). Alguns elementos do ambiente de teste serão citados nas subseções seguintes.

2.2.1 *Frameworks*

Um *framework* é um conjunto de componentes e regras para se fazer algo.

- *Calabash* - É uma tecnologia de teste automatizada para aplicativos nativos e híbridos para *Android e iOS*. Além disso, é um projeto de código aberto gratuito que foi desenvolvido e mantido pela *Xamarin*.[2]

- *Capybara* - é um *framework* de testes agnóstica de navegadores que roda seus *drivers* através do *Selenium*, utilizando a linguagem de programação *Ruby*. [3]

- *Eclipse* - *Eclipse Modeling Framework* (EMF) é uma estrutura de modelagem e um recurso de geração de código para construir ferramentas e outros aplicativos com base em um modelo de dados estruturados. [4]

- *Espresso* - Trata-se de um *framework* de testes automatizados para aplicativos aplicativos *Android*. É utilizado para escrever testes de interface do usuário.[5]

- *HttpParty* - É um *framework* para realizar requisições de *web services* e examinar as saídas resultantes dessas requisições. O uso do *HTTParty* pode ser para pequenas consultas formatadas no formato *curl* ou estruturado para automatizar testes de regressão em uma *API*.[6]

- *Jasmine* - É uma estrutura de desenvolvimento baseada na metodologia BDD. É utilizada para testar código em *JavaScript* e não depende de outro *framework* para ser utilizada. [7]

- *JUnit* - Trata-se de um *framework* simples para escrever testes repetitivos. É uma instância da arquitetura *xUnit* para estruturas de teste de unidade.[8]

- *Mocha* - *Mocha* é um *framework JavaScript* que roda no *NodeJS* e também no próprio *browser* que permite realizar suites de teste de forma rápida e fácil em suas aplicações *JavaScript*.[9]

- *NUnit* - Estrutura de teste de unidade para todas as linguagens *.Net*.[10]

- *Protractor* - É um *framework*de teste *end-to-end* para aplicações *Angular* e *AngularJS*.[11]

[2] https://github.com/calabash
[3] https://github.com/teamcapybara/capybara
[4] https://www.eclipse.org/modeling/emf
[5] https://www.infoq.com/br/news/2013/11/google-espresso
[6] https://medium.com/qaninja/apresentando-o-famoso-httparty-1c3c8df74519
[7] https://jasmine.github.io
[8] https://junit.org/junit4
[9] https://mochajs.org
[10] https://nunit.org
[11] https://www.protractortest.org

- *REST-assured* - Trata-se de um *framework* para testar e validar serviços *REST*, utilizando a linguagem *Java* para a escrita dos testes. [12]
- *Robot Framework* - É conhecida por ser uma estrutura de automação e possuir código aberto. Implementa a abordagem orientada por palavras-chave para o teste de aceitação e o desenvolvimento orientado a testes de aceitação *(ATDD)*.[13]
- *Rspec* - Trata-se de um *framework* para Desenvolvimento Orientado por Comportamento *(TDD)* para *Ruby*.[14]
- *SpecFlow* - É um *framework* utilizado para definir, gerenciar e automatizar testes em projetos *.NET*.[15]
- *TestNG* - Inspirado no *JUnit e no NUnitO*, é um *framework* de testes. Foi elaborado para abranger todas as categorias de testes: unidade, funcional, fim a fim, integração, etc.[16]
- *WebdriverIO* - Trata-se de um *framework* de código aberto para automação em *Node.js*, utilizando *Selenium*.[17]

2.2.2 Ferramentas

Há aproximadamente trinta anos, muitos esforços vem sendo criados no processo de desenvolvimento, relegando o teste, a princípio, a uma função manual. Novos métodos estão sendo desenvolvidos para melhorar a eficiência da área de teste como, por exemplo, as ferramentas de testes. (BASTOS *et al.*, 2012).

Atualmente existem inúmeras ferramentas de teste. Cada uma delas é direcionada para algum tipo de teste, com um objetivo diferente. Elas geralmente são separadas não só por seu objetivo, mas também por seu custo. De acordo com Rios e Moreira (2013), as ferramentas de teste podem ser utilizadas para realizar as tarefas de desenvolvimento e execução dos testes, além de examinar informações de resultado. O uso adequado dessas ferramentas de testes aumenta a eficiência e a eficácia dos processos de testes.

- *Appium* - É baseado na ideia de que para testar aplicativos nativos não é necessário exigir a inclusão de um *SDK* ou a recompilação de seu aplicativo, permitindo ao usuário a utilização de práticas, estruturas e ferramentas de teste preferidas. O *Appium* é um projeto de código aberto com objetivo de automatizar qualquer aplicativo móvel, de qualquer idioma, e qualquer estrutura de teste, com acesso total a APIs de *back-end* e bancos de dados do código de teste.[18]

[12] http://rest-assured.io
[13] https://robotframework.org
[14] https://rspec.info
[15] https://specflow.org/
[16] https://testng.org/doc/
[17] https://webdriver.io/
[18] http://appium.io

- *Cypress* - É uma ferramenta utilizada em um navegador, sendo o *Javascript* a linguagem de programação empregada para criação de testes. Possui excelente documentação, facilidade de uso e possibilita testes de *APIs REST*.[19]

- *Chai* - É uma biblioteca de assertiva *BDD/TDD* para *NodeJS* e também para *browser* que pode ser combinada com qualquer *framework* de testes *JavaScript*. Ela fornece ao desenvolvedor diversos estilos para aplicar *BDD e TDD* às suas suites de teste. [20] A combinação da *Mocha e Chai*, hoje, é muito conhecida para a realização de testes em *API Rest* desenvolvidas em *NodeJS*.[21]

- *Cucumber* - É uma ferramenta de testes *open source* desenvolvido com o conceito *BDD* (desenvolvimento orientado por comportamento). Esta ferramenta suporta várias linguagens como *Ruby, Java, NET, Scala e Groovy*.[22]

- *JMeter* - É um *software* de código aberto, um aplicativo totalmente *Java* puro, projetado para carregar o comportamento funcional dos testes e medir o desempenho. Ele foi originalmente projetado para testar aplicativos da *web*, mas, desde então, expandiu para outras funções de teste. [23]

- *Katalon* - *Katalon Studio* é é uma solução gratuita para automação de testes funcionais para aplicações *web* e móveis.[24]

- *Postman* - Trata-se de um ambiente de desenvolvimento de *API*, oferecendo as opções de projetar, simular, depurar, testar, documentar, monitorar e publicar *APIs*. [25]

- *RFT* - O *Rational Functional Tester* - *RFT* é uma ferramenta para teste funcional automatizado, teste de regressão, teste da interface com o usuário e teste dirigido a dados.[26]

- *Selenium* - É uma ferramenta de testes funcionais para aplicações *Web*. Os *scripts* podem ser escritos em várias linguagens de programação como *Java*, *Perl*, *JavaScript*, *PHP*, *Python*, *C*, *Ruby* e *Groovy*.[27]

- *Sikuli* - Ferramenta de automação para *desktop*, que pode ser executada no *Windows, Mac* ou *Linux/Unix*.[28]

[19] https://www.cypress.io
[20] https://www.chaijs.com
[21] https://medium.com/@rafaelvicio/testando-api-rest-com-mocha-e-chai-bf3764ac2797
[22] https://github.com/cucumber/cucumber
[23] https://jmeter.apache.org
[24] https://www.katalon.com
[25] https://www.getpostman.com/about-postman
[26] https://www.ibm.com/developerworks/rational/library/test-automation-framework-rational-functional-tester/index.html
[27] https://www.seleniumhq.org
[28] http://sikulix.com/

- *SoapUI* - Ferramenta para teste funcional, utilizada para testes de *SOAP e REST*. Essa ferramenta permite a criação e execução de testes automatizados, como testes funcionais, de regressão e de carga.[29]

- *TestComplete* - Ferramenta paga, utilizada para automação de testes de uso comercial. Pode ser empregada para automatizar testes do *desktop*, celular e aplicações *web*, além de permitir também a utilização de diversas linguagens, como *JavaScript, VBScript e Python*.[30]

- *UFT* - O *Unified Functional Testing (UFT)* é uma ferramenta de teste comercial, utilizada em testes funcionais. Ele fornece automação para *API*, aplicativos *Web*, móveis, híbridos *RPA* e aplicativos corporativos.[31]

- *VScode - O Visual Studio Code - VScode* é um editor de código-fonte leve, que é executado na área de trabalho. É disponível para Windows, *macOS* e /Linux. Essa ferramenta vem com suporte embutido para *JavaScript, TypeScript e Node.js*. Possui um rico ecossistema de extensões para outras linguagens.[32]

2.2.3 Aplicações

As aplicações podem ser divididas em *Desktop*, *Web* e *Mobile*.

2.2.3.1 *Desktop*

Por definição, uma aplicação *desktop* é qualquer *software* que pode ser instalado em um computador e usado para executar tarefas específicas. Algumas aplicações *desktop* também podem ser usadas por vários usuários em um ambiente com rede.[33]

2.2.3.2 *Web*

Aplicações web são programas focados em rede, abrangendo um grande número de *software* acessados por um navegador a partir de computadores e dispositivos móveis.[34]

2.2.3.3 *Mobile*

- *Android* - É um sistema operacional baseado em *Linux* e desenvolvido pelo *Google* em parceria tecnológica com empresas da *Open Handset Alliance*. É voltado para os dispositi-

[29] https://www.soapui.org/open-source.html
[30] https://smartbear.com/product/testcomplete/overview/
[31] https://www.microfocus.com/en-us/products/unified-functional-automated-testing/overview
[32] https://code.visualstudio.com/docs
[33] http://blog.academiadocodigo.com.br/2015/04/o-que-e-programacao-web-e-programacao-desktop/
[34] https://universidadedatecnologia.com.br/tipos-de-software-e-suas-classificacoes

vos móveis modernos com maior capacidade de memória e processamento, conhecidos como *Smartphones e Tablets*.[35]

- *IOS* - É um sistema operacional criado pela empresa de tecnologia *Apple* para os seus dispositivos móveis.[36]

2.2.3.4 API

O significado de *API* é *Application Programming Interface* ou, em português, Interface de Programação de Aplicativos. Esta interface de programação é um conjunto de padrões de programação que permitem a construção de aplicativos e a sua utilização.[37]

2.2.3.5 Sistemas Embarcados

Sistemas Embarcados é o nome dado a programas e sistemas embutidos em microprocessadores, que executam tarefas específicas em um aparelho como, por exemplo, impressoras e semáforos. [38]

2.2.4 Linguagens de Programação

A seguir, serão citadas algumas das linguagens de programação mais utilizadas na atualidade em automação de testes de *software*.

- *C#*: C Sharp é uma linguagem de programação elegante, orientada a objeto e fortemente tipada, que permite que os desenvolvedores criem uma variedade de aplicativos robustos e seguros executados no .NET Framework.[39]

- *Go*: também conhecida como GoLang, é uma linguagem criada em 2007 pela Google, possui desempenho baseado na linguagem C, é multiplataforma e *Open Source*. É uma linguagem projetada para construir *software* simples, confiável e eficiente.[40]

- *Groovy*: é uma linguagem de programação orientada a objetos, desenvolvida para a plataforma Java.[41]

- *Java*: é uma linguagem de programação e plataforma computacional lançada pela primeira vez pela Sun Microsystems em 1995. Atualmente existem mais de 9 milhões de desenvolvedores Java no mundo. [42]

[35] https://www.devmedia.com.br/conhecendo-o-android-revista-mobile-magazine-42/24688
[36] https://www.techtudo.com.br/tudo-sobre/ios.html
[37] https://www.devmedia.com.br/application-programming-interface-desenvolvendo-apis-de-software/30548
[38] https://guiadoestudante.abril.com.br/profissoes/sistemas-embarcados/
[39] https://docs.microsoft.com/pt-br/dotnet/csharp/
[40] https://golang.org/
[41] https://groovy-lang.org/
[42] https://www.java.com/pt$_BR$/about/whatis$_j$ava.jsp

- *JavaScript*: é uma linguagem de programação criada em 1995 e hoje é uma das linguagens de programação mais populares e usadas no mundo. Com ela, é possível construir páginas dinâmicas, desenvolver aplicativos para smartphones e também jogos eletrônicos. [43]

- *kotlin*: trata-se de uma linguagem de programação criada pela JetBrains em 2010 e que foi adicionada à lista de linguagens para o desenvolvimento do *Android*. Esse fato ocorreu em meados de 2017, quando, até então, só faziam parte da lista o *Java* e o C++.[44]

- *Php*: é uma linguagem de script popular de uso geral que é especialmente adequada para desenvolvimento *web*. Considerada uma linguagem rápida, flexível e pragmática.[45]

- *Python*: foi criada em 1991, possui características que possibilitam escrever o mesmo requisito em menos linhas de código que o necessário em outras linguagens de programação. Hoje, além de adotado na construção de soluções *web*, também está sendo muito utilizado em aplicações que lidam com processamento de texto, *machine learning* e recomendação de conteúdo.[46]

- *Ruby*: linguagem dinâmica, *open source* com foco na simplicidade e na produtividade. Tem uma sintaxe elegante de leitura natural e fácil escrita.[47]

2.3 Metodologias Ágeis

Por muitos anos, a Engenharia de *Software* utilizou como exemplo processos de manufatura para a estabilização de seus métodos de trabalho. Uma de suas inspirações foi o campo automobilístico que estava em ampla ascensão industrial quando ocorreu a constituição da nova indústria de TI. Utilizando como modelo a forma de produção em série de Henry Ford, a ciência do desenvolvimento de *software* desenrolou-se com intenso foco na padronização de processos e componentes e na automatização do movimento. Por volta dos anos 90 começaram a surgir processos alternativos ao desenvolvimento de *software*, como mostra Prikladnicki, Willi e Milani (2014):

> "Já em meados dos anos 90, começaram a surgir processos alternativos de desenvolvimento de *software*, em resposta àqueles tradicionais, considerados excessivamente regrados, lentos, burocráticos e inadequados à natureza da atividade. Esses novos processos foram apelidados de "leves (lightweight), em oposição os anteriores, "pesados"(heavyweight)."(PRIKLADNICKI; WILLI; MILANI, 2014, p.3).

Ainda de acordo com Prikladnicki, Willi e Milani (2014), ambos os processos que surgiram são baseados em desenvolvimento interativos, em que requisitos e soluções evoluem

[43] https://www.javascript.com/
[44] https://kotlinlang.org/
[45] https://php.iis.net/
[46] https://python.org.br/
[47] https://www.ruby-lang.org/pt/

com a colaboração entre equipes auto gerenciáveis e *cross-funcional* (pessoas com diferentes expertises). Estimulam frequente inspeção e adaptação, filosofia de liderança, alinhamento entre o desenvolvimento e os objetivos das empresas ou dos clientes, e um conjunto de boas práticas de engenharia que admitia entregas rápidas e de alta qualidade.

As Metodologias Ágeis tornaram-se populares em 2001 (dois mil e um) quando 17 (dezessete) especialistas em desenvolvimento de *software* reuniram-se na cidade de *Snowbird* para discutirem diversos assuntos e, por consequência dessa discussão, assinaram o manifesto contendo valores e princípios[48] do Manifesto Ágil (MASSARI, 2014). De acordo com Prikladnicki, Willi e Milani (2014), diversos profissionais mencionam que o ”pensamento ágil”já era adotado no Brasil antes da formalização do Manifesto Ágil, porém só não tinha um nome e nem era formalizado.

Para Wazlawick (2013), apesar de os métodos ágeis serem usualmente mais leves, não é correto entendê-los como modelos de processos menos complexos ou simples. Tratam-se apenas de focar mais nos resultados do que no processo.

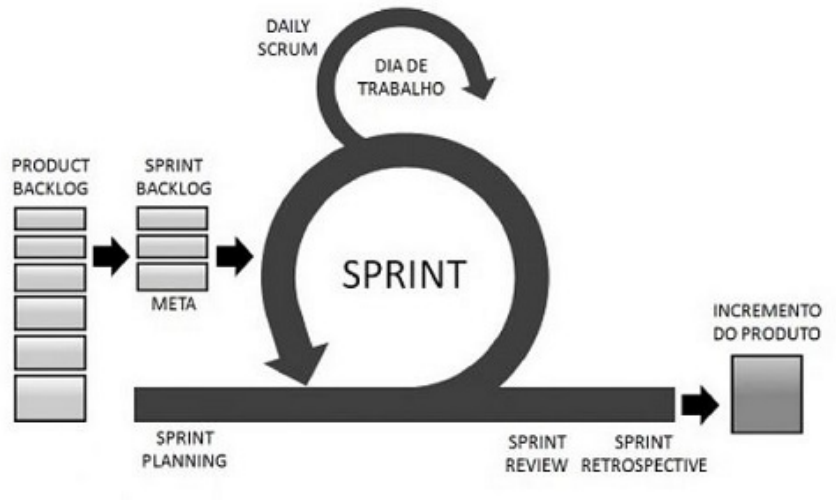
Em ciclos de vida Ágil, é gerado a cada iteração, um *software* que oferece funcionalidades de valor para as partes interessadas.(BOARD, 2014 apud ALEXANDRE, 2016). Em equipes ágeis, o teste se torna parte e prática central do processo de desenvolvimento em vez de ser uma atividade realizada apenas no final.(COHN, 2011 apud ALEXANDRE, 2016). Dentre os processos existentes, vamos falar dos mais populares no mercado de desenvolvimento de *software*.

2.3.1 *Scrum*

O *Scrum*[49] é um *framework* ágil utilizado para auxiliar no gerenciamento de projetos complexos e desenvolvimento de produtos. É conhecido por ser um *framework* que prescreve um conjunto de práticas leves e objetivas que são muito utilizadas na área de desenvolvimento de *software*. (PRIKLADNICKI; WILLI; MILANI, 2014).

[48] http://www.metodoagil.com/manifesto-agil/
[49] https://www.scrum.org/

Figura 2 – O *Framework* do *Scrum*.



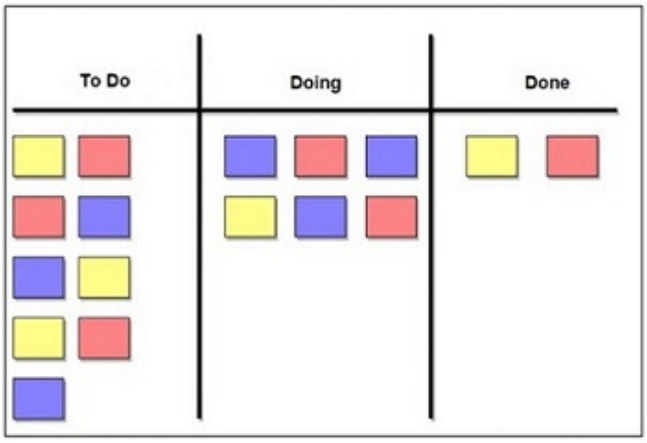
Fonte: Autociência

A figura 2 mostra as iterações entre as atividades no processo utilizando o *Scrum*. O *Product Owner* (o proprietário do produto) tem uma visão do que ele quer criar (o grande cubo). Como o cubo pode ser grande, por meio de uma atividade chamada *Grooming*, ele é dividido em um conjunto de funcionalidades que são compiladas em uma única lista priorizada chamada de *Product Backlog*. A primeira reunião de Planejamento de *Sprint* é realizada para definir o *Sprint Backlog*, que contém todo o trabalho que será executado durante o *Sprint*. O *Sprint* tem duração média de 2 a 4 semanas e são feitas reuniões diárias de acompanhamento (*Daily Scrum*) do trabalho que duram em média 15 minutos.

2.3.2 *Kanban*

Kanban é um sistema japonês criado após a segunda guerra mundial, que nada mais é do que um sistema ágil e visual para controle de produção ou gestão de tarefas. Ele não é prescritivo ou impõe regras para que o trabalho seja feito corretamente, apenas possibilita que a equipe execute suas tarefas com mais clareza e colaboração. De acordo com Silva e Anastácio (2019), por ser bastante simples, barato e ágil, esse método é muito útil para empresas que querem controlar a produção, diminuir custos, garantindo o bom desempenho da empresa e sua equipe.

Figura 3 – Quadro *Kanban*.



Fonte: (Clarify)

A figura 3 mostra um exemplo de quadro utilizando o *Kanban*. As colunas podem variar de acordo com o uso mas, no geral, há no mínimo 3 (três) colunas importantes que possibilitam o gerenciamento do projeto:

- A Fazer (*To Do*): define as atividades que têm prioridade para serem executadas ou atividades que ainda não foram iniciadas.
- Fazendo (*Doing*): tarefas que já se encontram em execução.
- Conlcuído (*Done*): todas as tarefas que já foram realizadas e aguardam a póxima ação.

2.3.3 *XP – eXtreme Programming*

Programação Extrema[50], ou *XP* (*eXtreme Programming*, é uma metodologia adequada para equipes pequenas e médias, baseada em uma série de valores, princípios e regras. O *XP* surgiu nos Estados Unidos no final da década de 90 (noventa). Wazlawick (2013) cita algumas características:

> ”Então, o *XP* preconiza mudanças incrementais e *feedback* rápido, além de considerar a mudança algo positivo, que deve ser entendido como parte do processo. Além disso, o *XP* valoriza o aspecto da qualidade, pois considera que pequenos ganhos a curto prazo pelo sacrifício da qualidade não são compensados pelas perdas a médio e a longo prazo. A esses princípios pode-se adicionar ainda a priorização de funcionalidades mais importantes, de forma que, se o trabalho não puder ser todo concluído, pelo menos as partes mais importantes terão sido.”(WAZLAWICK, 2013, p.3).

Segundo Teles (2014), *XP* é um processo de desenvolvimento que tem como objetivo garantir que o cliente tenha um máximo de retorno a cada dia, acrescentaando também, que a filosofia do *XP* se baseia fortemente no *feedback* fornecido pelos clientes e desenvolvedores.

[50] http://www.extremeprogramming.org/

O *XP* inclui uma abordagem de testes para reduzir as possibilidades de erros desconhecidos. De acordo com Sommerville (2013), as principais características dos testes em *XP* são:

- Desenvolvimento *test-first*: utiliza a abordagem de escrever os testes para um código antes de escrever o código, possibilitando a execução do teste enquanto o código é escrito facilitando também a identificação de problemas durante o desenvolvimento.
- Desenvolvimento de teste incremental utilizando cenários: o papel do cliente no processo de testes é ajudar a desenvolver testes de aceitação. No método *XP* o teste de aceitação é incremental. O cliente faz parte da equipe, escrevendo testes enquanto o desenvolvimento avança. Muitas vezes é difícil contar com o apoio do cliente no processo de testes pois geralmente clientes têm pouco tempo disponível, tornando isso uma grande dificuldade no processo de testes em *XP*.
- Envolvimento dos usuários no desenvolvimento de testes e validação: os requisitos do usuário são representados como cenários ou estórias. O usuário escolhe quais estórias e cenários são prioridade e a equipe de desenvolvimento avalia os escolhidos, dividindo a tarefa.
- Uso de *frameworks* de testes automatizados: a automação dos testes é essencial para o desenvolvimento *test-first*. Inicialmente, antes que a tarefa seja implementada, os testes são escritos como componentes executáveis. Um *framework* de testes automatizados facilita a escrita de testes executáveis e a submissão de um conjunto de testes para execução.

2.3.4 *Lean*

O *Lean* é um método altamente eficaz, conhecido mundialmente por proporcionar a entrega de cada vez mais valor, com a aplicação de cada vez menos esforço. Foi criado em meados da década de 40, pela *Toyota*, para competir com os fabricantes de carros americanos. O *Lean* é fundamentado por todos os valores e princípios descendentes da cultura japonesa como, por exemplo, responsabilidade e disciplina. (PRIKLADNICKI; WILLI; MILANI, 2014).

Filosofia *Lean* é uma metodologia de gestão que otimiza custos e reduz o tempo e os desperdícios de uma empresa, partindo do princípio de que toda iniciativa precisa ser baseada no consumidor final. O propósito é criar valor para este público.

3 TRABALHOS RELACIONADOS

Neste capítulo descrevemos alguns trabalhos relacionados a Testes de *Software* e que contribuíram para a realização do trabalho proposto.

Em seu trabalho, Neto *et al.* (2006) realizaram um estudo em um cenário de desenvolvimento de *software* específico, utilizando um conjunto de práticas de teste de *software*, extraídas da literatura, para a avaliação de como algumas dessas práticas estão sendo utilizadas. A partir da aplicação de questionário, avaliaram a aplicabilidade dessas práticas nas organizações de *software* e o grau de importância dessas práticas na visão dos profissionais dessas organizações.

A pesquisa realizada por Neto *et al.* (2006) permitiu observar que, no cenário avaliado por eles, as organizações estão utilizando práticas de teste em seus projetos, porém com limitações. Essa maior preocupação em relação à organização dos testes, no entanto, não tem sido observada nas atividades de planejamento, controle, medição e análise, que são pouco aplicadas na indústria de *software*. Além disso, foi observada uma grande quantidade de práticas consideradas não-importantes e não-aplicadas, independentemente do tamanho das organizações. O trabalho citado se difere do proposto, por ter aplicado um questionário para avaliar práticas de testes de *software* relacionadas à empresas específicas, enquanto este tem como objetivo avaliar as práticas realizadas por profissionais em um contexto geral, sem aplicação específica em alguma empresa.

Crespo *et al.* (2004) desenvolveram, em seu trabalho, uma metodologia para implantação ou melhoria do processo de teste em empresas desenvolvedoras de *software* a partir do Centro de Pesquisas Renato Archer – CenPRA e pelo seu grupo de teste da Divisão de Melhoria de Processos de *Software* - DMPS. O trabalho teve como objetivo criar uma metodologia para a introdução ou melhoria do processo de teste de *software* em empresas produtoras de *software*, englobando técnicas, procedimentos e ferramentas a partir da norma IEEE 829-1998. A experimentação e validação da metodologia de teste foi realizada em uma micro empresa desenvolvedora de *software*.

O trabalho de Crespo *et al.* (2004) mostrou que a metodologia desenvolvida ajuda a estruturar as atividades de teste desde o início de um projeto de desenvolvimento de *software*, tornando claro para todas as partes envolvidas o relacionamento entre o processo de desenvolvimento de software e as atividades de teste.

Bruneli (2006) apresentou um processo de teste de *software* onde a atividade de teste começa junto com o desenvolvimento e caminha em paralelo com o ciclo tradicional de desenvolvimento. Seu trabalho mostrou que, com o uso dessa abordagem, é possível detectar e prevenir erros através do processo de desenvolvimento, conduzindo assim para uma maior confiança e qualidade do *software*. Também é apresentado um estudo de caso para ilustrar a aplicabilidade do método e os resultados obtidos. Os levantamentos de dados contaram com a participação da equipe de teste a partir da fase de levantamento de dados e análise de requisitos.

Foi possível constatar que a utilização da metodologia proposta por Bruneli (2006) proporcionou aumento na qualidade dos objetos produzidos, menor número de erros implantados em produção e identificação de erros na definição de requisitos, logo na fase inicial do projeto, que refletiram na satisfação e confiança dos clientes internos. Vale destacar também que, através da pesquisa, pode-se perceber que a elaboração de casos de testes mostrou-se um processo frágil, pois dependia da experiência, conhecimento sistêmico e até mesmo da imaginação dos envolvidos para a elaboração de testes eficazes.

Salomão (2016) descreveu, em seu trabalho, quatro abordagens para a realização de teste de regressão são utilizadas em desenvolvimento de *software*. Foram apresentados resultados de uma pesquisa exploratória e quantitativa que foi realizada com 17 empresas de desenvolvimento de *software* da região do Triângulo Mineiro para identificar a relevância dos testes de regressão e o volume de investimento que pode ser esperado nesta área. Os resultados da pesquisa mostraram que apesar de as empresas considerarem os testes de regressão relevantes, elas não investem suficientemente na área devido a falsas crenças oriundas do não conhecimento total da teoria relacionada à qualidade de *software*. Na sua conclusão, o artigo apresenta e discute alguns problemas em aberto, bem como direções para trabalhos futuros.

4 METODOLOGIA

A metodologia utilizada para o desenvolvimento deste trabalho foi baseada na revisão bibliográfica de materiais publicados em livros, revistas, Internet (*sites* de repositório de artigos), periódicos especializados, monografias e dissertações, como fonte de coleta de dados sobre a influência do teste de *software* no mercado de trabalho, provendo, assim, fundamentos teóricos à pesquisa. Foi realizada uma uma análise descritiva e pesquisa qualitativa.

Considerando que o objetivo da pesquisa é analisar o contexto do trabalho dos profissionais de testes e os principais desafios encontrados por eles, após a realização do levantamento bibliográfico, foi aplicação um questionário *online* a pessoas que trabalham com *testes* de software. Essas etapas são detalhadas nas seções seguintes.

4.1 Levantamento Bibliográfico

O levantamento bibliográfico foi realizado por meio da leitura de trabalhos relacionados a Engenharia de *Software* e Testes de *Software*, publicados em livros, artigos, teses, revistas e páginas *web*. A leitura desses trabalhos possibilitou o aprofundamento do conhecimento acerca do assunto, bem como a compreensão dos diferentes contextos e momentos em que os testes podem ser realizados.

4.2 Questionário *Online*

A aplicação do questionário foi realizada nos meses de abril e maio de 2019 e contou com 171 participantes que afirmaram trabalhar na área de Testes de *Software*. O questionário foi elaborado com o objetivo de levantar o perfil dos participantes bem como de levantar as principais atividades e técnicas utilizadas por eles em seus ambientes de trabalho. A divulgação e distribuição do questionário aconteceu por meio da internet, em grupos específicos para divulgação e discussão sobre testes de *software* em redes sociais como o *Facebook* e *WhatsApp*.

O questionário foi composto por dezoito perguntas, sendo quatro perguntas de múltipla escolha, sete com caixas de seleção e seis perguntas totalmente abertas. Das sete perguntas com caixa de seleção, seis possibilitavam a inserção de uma opção diferente das oferecidas, por meio da escolha da opção “Outros”.

Algumas perguntas do questionário foram baseadas no questionário aplicado por Luft (2012) que em seu trabalho, realizou um estudo de caso com a aplicação de questionário à 14 empresas na área de TI. O objetivo da aplicação de questionário realizado por Luft (2012), era verificar e comparar os métodos de testes *software* utilizados pelas empresas participantes.

Após a obtenção das respostas dos participantes, foi realizada uma Análise Descritiva e Análise Qualitativa dos dados coletados. De acordo com Gerhardt e Silveira (2009) a Pesquisa Qualitativa se preocupa com a investigação e aspectos da realidade que não podem ser

quantificados. Reis e Reis (2002) afirmam que a Análise Descritiva é utilizada para organizar, descrever e resumir aspectos importantes de um conjunto ou comparar tais características entre outros conjuntos possibilitando a identificação de alguma anomalia.

No capítulo 5 serão apresentados os resultados obtidos por meio do questionário, bem como a análise desses resultados.

5 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Neste capítulo são apresentados os resultados obtidos com a aplicação do questionário *online*.

5.1 Contexto do trabalho dos participantes

Nesta seção, serão apresentados os dados obtidos com a aplicação do questionário *online*. Inicialmente, os participantes responderam a perguntas para possibilitar a definição de um perfil. Em seguida, foram questionados quanto aos conhecimentos e práticas que utilizam em seu trabalho de testes.

5.1.1 Perfil dos Participantes

As perguntas que puderam caracterizar o perfil dos participantes têm relação com gênero, idade, estado onde moram, nível de escolaridade e tempo de trabalho na área de testes. Na figura 4 podemos verificar que a maior parte dos participantes é do gênero masculino.

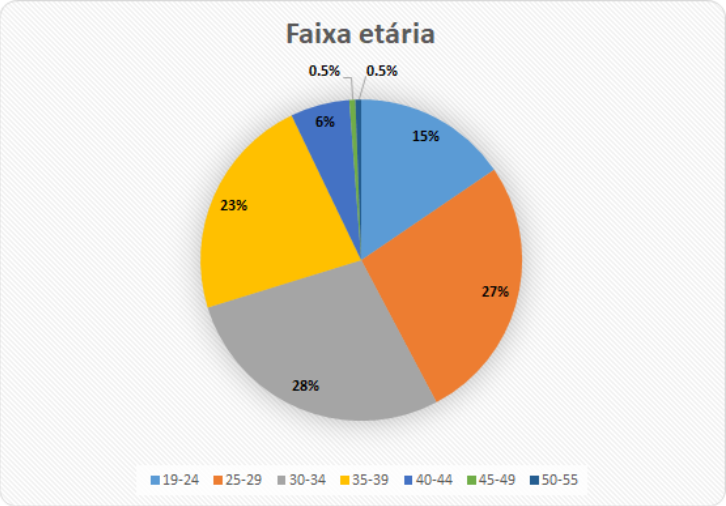
Figura 4 – Gêneros dos participantes da pesquisa.

Fonte: Dados do autor.

Dos 171 participantes, 168 informaram a idade. Na figura 5 podemos verificar que a faixa etária predominante é a de pessoas com idade entre 30 e 34 anos (28%), seguida por 25 e 29 anos (27%). Em terceiro lugar, ficaram os participantes com idade entre 35 e 39 anos de idade (23%). Isso mostra que a maior parte dos participantes (78%) afirma possuir de 25 a 39 anos de idade.

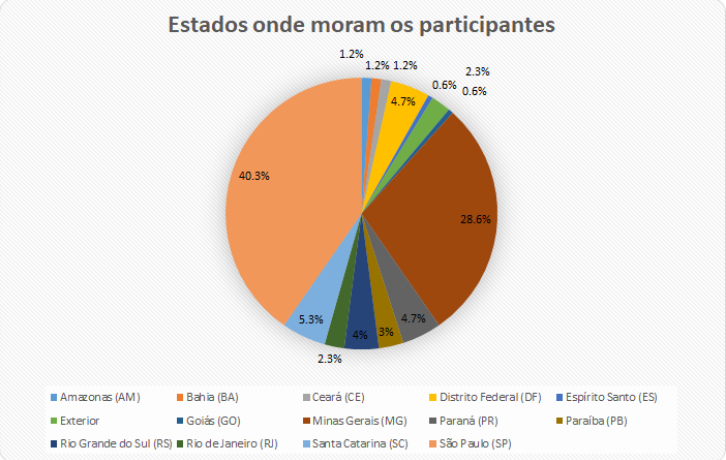
De acordo com as respostas informadas pelos participantes, ao perguntá-los sobre qual estado eles moram, 68.9% deles afirmaram morar nos estados de São Paulo (SP) (40,3%) e Minas Gerais (MG) (28,6%), como mostra a figura 6.

Figura 5 – Faixa etária dos participantes.



Fonte: Dados do autor.

Figura 6 – Estados em que residem os participantes.



Fonte: Dados do autor.

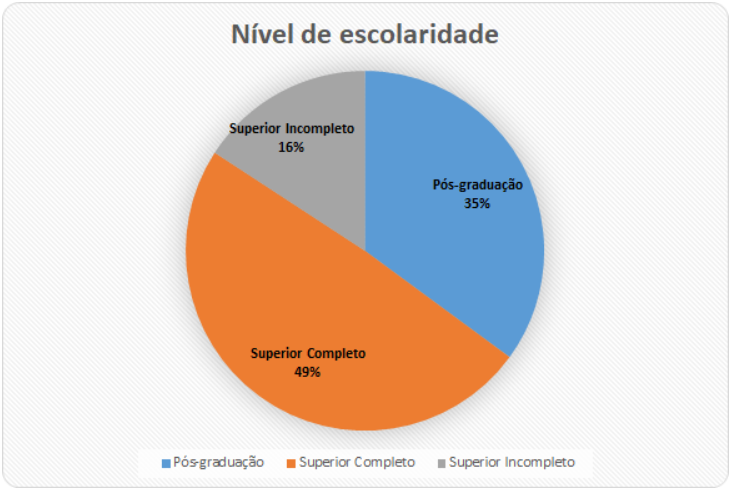
Foi perguntado aos participantes qual o seu nível de escolaridade. A pesquisa foi restrita a profissionais que estão cursando, pelo menos, alguma graduação na área de TI. A maior parte dos profissionais afirmou que possui Nível Superior Completo (49,12%), sem nenhuma Pós-graduação (figura 7.

Conforme mostra o gráfico da figura 8, a maioria dos participantes afirmou possuir de 1 a 3 anos de atuação na área (22,81%), seguido por 3 a 5 anos (20,47%). Em terceiro lugar ficaram os profissionais com mais de 10 anos de atuação (18,71%).

5.1.2 Perfil do Trabalho

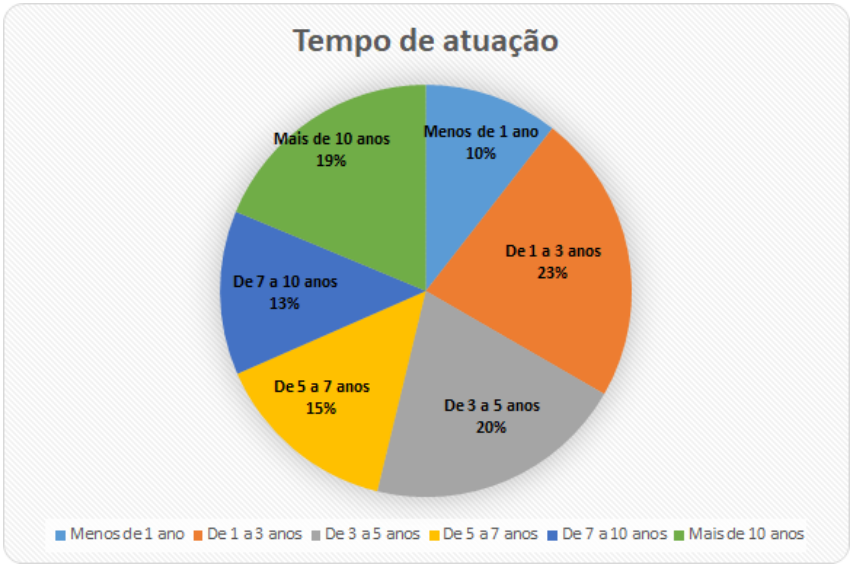
A segunda seção do questionário foi aplicada com o objetivo de verificar quais conhecimentos e técnicas relacionadas aos testes de *software* os participantes possuem e praticam em

Figura 7 – Nível de escolaridade dos participantes.



Fonte: Dados do autor.

Figura 8 – Tempo de atuação na área.



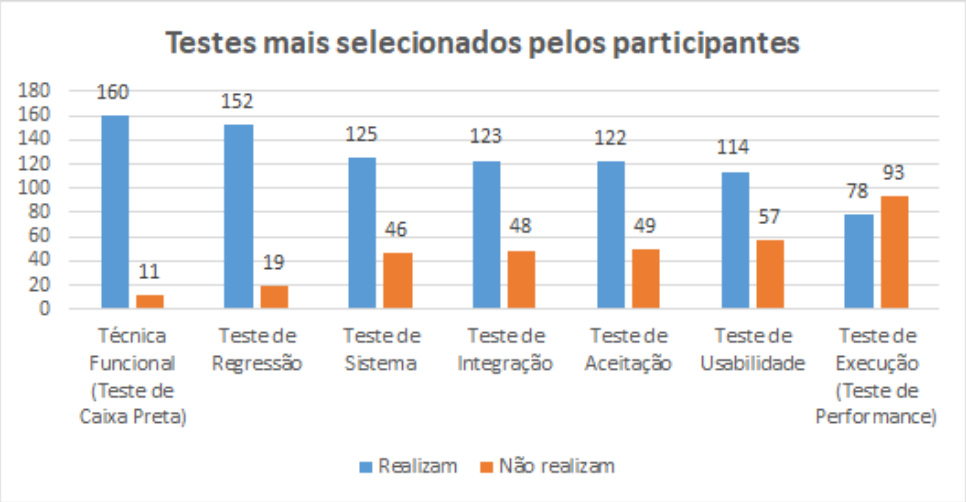
Fonte: Dados do autor.

seu ambiente de trabalho, possibilitando observar, também, os principais desafios relacionados à execução e automação de testes, bem como as documentações que fazem parte de suas atividades. As perguntas foram elaboradas para oferecerem um panorama de como os participantes exercem a sua profissão.

Foi perguntado aos participantes, sobre quais testes de *software* eles realizam e/ou já realizaram. A pergunta era composta por 16 alternativas, sendo que, os participantes puderam selecionar mais de uma opção além da possibilidade de inserir outros tipos de teste como resposta através da opção "Outro". A figura 9 mostra as opções de resposta mais selecionadas pelos participantes: Técnica Funcional (Teste de Caixa Preta), Teste de Regressão com 152 respostas (88,89% do total), vale lembrar que o Teste de Regressão faz parte dos testes que

compõem a Técnica Funcional, Teste de Sistema, Teste de Integração, Teste de Aceitação, Teste de Usabilidade e Teste de Execução (Teste de Performance). Cada alternativa possuem uma barra azul e uma barra vermelha. As barras vermelhas mostram o número de participantes que afirmaram não realizar o tipo de teste descrito e as barras azuis mostram o número de participantes que afirmaram realizar o tipo de teste descrito. Cada opção descrita no gráfico, possui um total de 171 respostas divididas em ”Realizam”e ”Não realizam”.

Figura 9 – Testes mais usados pelos participantes.

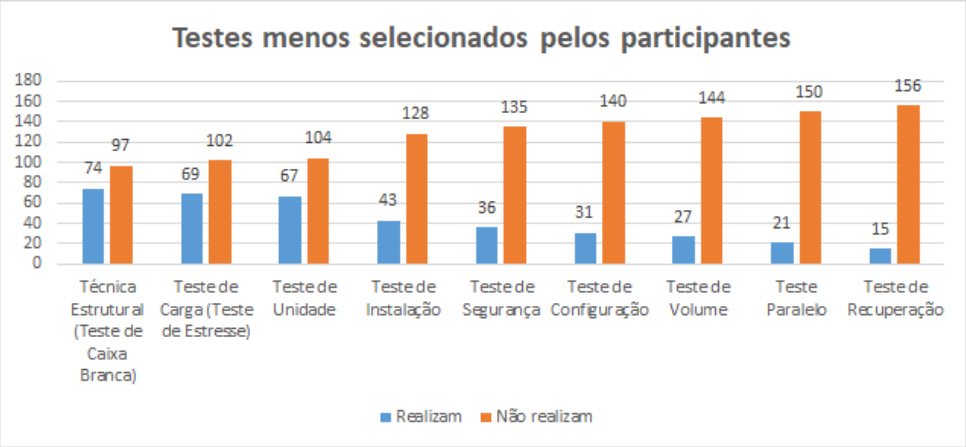


Fonte: Dados do autor.

A figura 10 mostra as opções de testes que foram menos selecionados pelos participantes: Técnica Estrutural (Teste de Caixa Branca), Teste de Carga (Teste de Estresse), Teste de Unidade, Teste de Instalação, Teste de Segurança, Teste de Configuração, Teste de Volume, Teste Paralelo e Teste de Recuperação. Na alternativa “Outro” da pergunta 6, sobre os testes de *software* realizados, 5 participantes citaram Teste de Automação e 4 participantes citaram Teste Exploratório. Também foram citados por 1 participante cada: Teste de Serviço, Teste de Confiabilidade e Teste de Consumo de Memória.

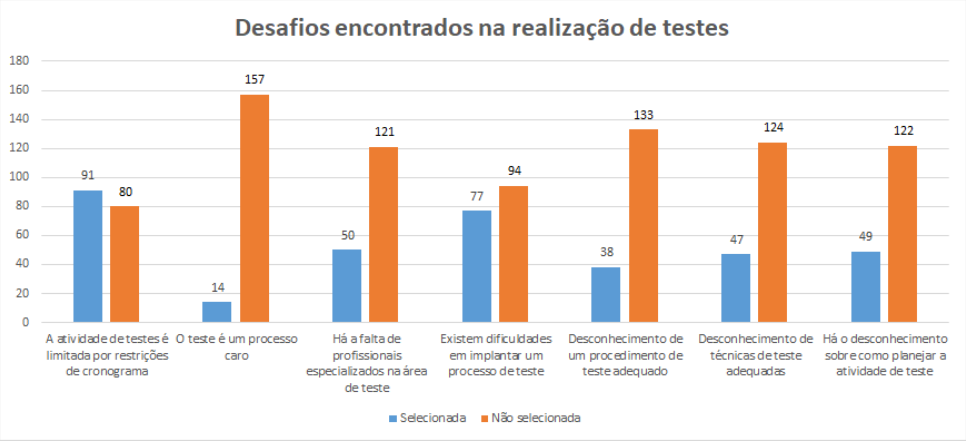
Sobre os desafios encontrados pelos participantes na realização de testes de *software*, 91 deles selecionaram a opção “A atividade de teste é limitada por restrições de cronograma”, 77 selecionaram a opção “Existem dificuldades em implantar um processo de teste” e 50 participantes Selecionaram a opção “Há falta de profissionais especializados na área de teste”, conforme apresentado na figura 11. Cada alternativa possui uma barra azul e uma barra vermelha. As barras vermelhas mostram o número de participantes que não selecionaram a opção descrita e as barras azuis mostram o número de participantes que selecionaram a opção descrita. Cada opção descrita no gráfico, possui um total de 171 respostas divididas em “Selecionada” e “Não selecionada”. A pergunta possuía a alternativa “Outro”. As opções citadas por alguns participantes foram: Falta de Requisitos/Documentação/Informação (citado por 9), Nenhuma dificuldade/desafio (citado por 3), Ausência de Estratégia/Cultura de Teste (citado por 3), Aumento de escopo com o mesmo prazo (citado por 2) e Falta de Estrutura/ambiente (citado por 2).

Figura 10 – Testes menos usados pelos participantes.



Fonte: Dados do autor.

Figura 11 – Desafios encontrados na realização de testes.

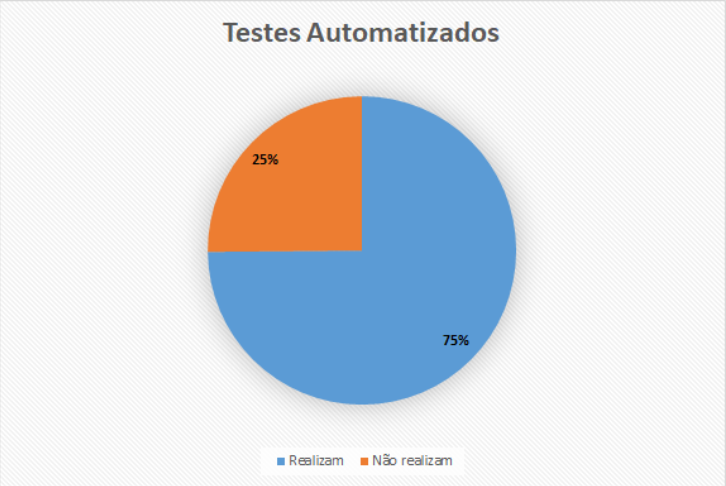


Fonte: Dados do autor.

Ao questioná-los sobre realização de testes manuais e de automação, 118 participantes (94%) afirmaram realizar testes manuais e 10 participantes (6%) afirmaram não realizar testes manuais. A figuras 12 e 13 mostram que 128 participantes (75%) afirmaram realizar automação de testes e 43 participantes (25%) afirmaram não realizar automação de testes, respectivamente.

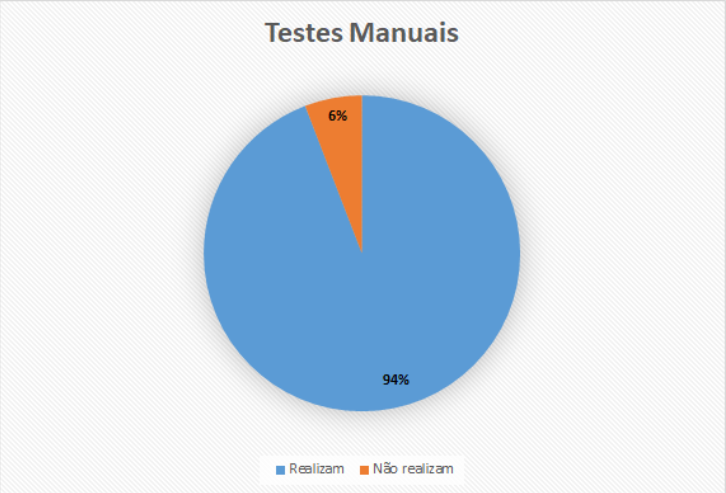
O gráfico da figura 14 mostra que, dos 128 participantes que afirmaram realizar automação, 82 (64%) deles são do gênero masculino e 46 (36%) são do gênero feminino. Na figura 15 podemos visualizar a distribuição de gênero em relação à realização de automação de testes. As amostras utilizadas para a composição do gráfico foram o total de participantes do gênero masculino e do gênero feminino que afirmaram realizar automação de testes, sendo 46 do gênero feminino e 82 masculino. Ao realizar uma simples leitura das tabelas é possível verificar que existe uma pequena diferença entre a proporção de mulheres que realizam automação em relação a proporção de homens que automatizam. A automação é comum em homens e mulheres mas a partir dos resultados foi possível perceber que a automação é mais comum nos homens.

Figura 12 – Realização de Testes Automatizados.



Fonte: Dados do autor.

Figura 13 – Realização de Testes Manuais.

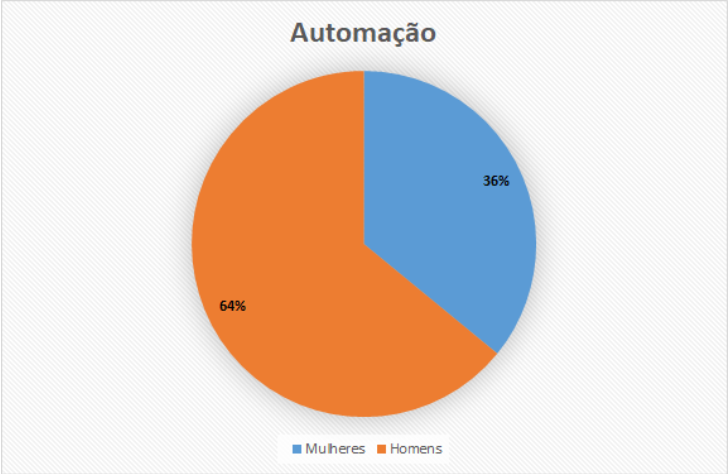


Fonte: Dados do autor.

A partir das variáveis tempo de escolaridade, tempo de atuação, idade, gênero e região (São Paulo e Minas Gerais) e com as proporções de usuários que informaram realizar automação de testes foi possível obter alguns dados importantes. Para uma análise mais direcionada, limitamos a análise aos estados de Minas Gerais (MG) e São Paulo (SP), que contaram com o maior número de participantes. As figuras de número 16 até a figura 20 mostram os resultados dessa comparação.

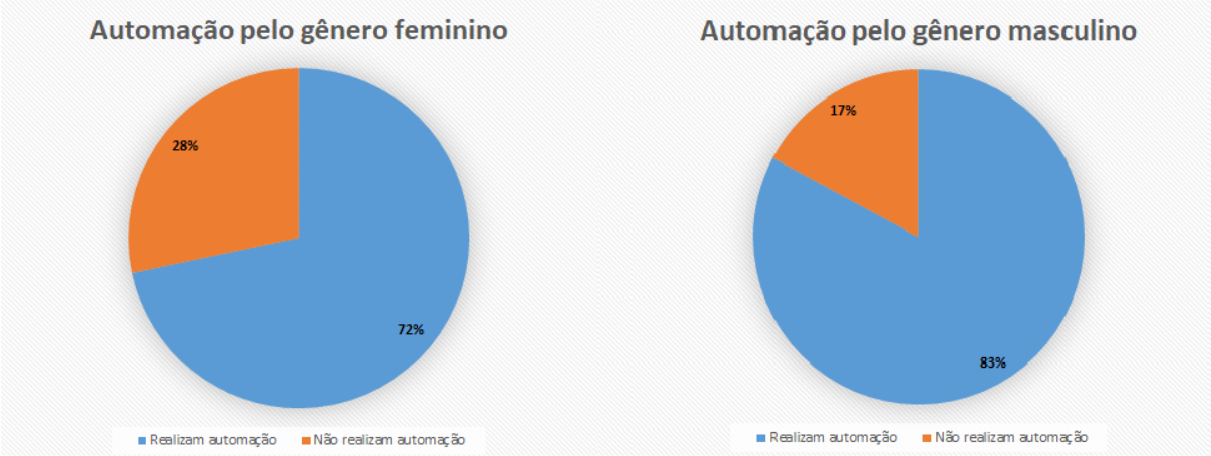
A análise da distribuição dos resultados sobre a realização de automação em relação à faixa etária, foi realizada separando a faixa etária dos participantes em duas, abaixo de 30 anos e a partir de 30 anos. A diferença entre as faixas etárias é de 16%, os resultados mostraram que não existe uma evidência estatística de que esses grupos são diferentes quanto a proporção de usuários que realizam a automação mas, é possível observar uma maior proporção de automação

Figura 14 – Realização de automação em relação ao gênero dos participantes.



Fonte: Dados do autor.

Figura 15 – Comparação entre gêneros em relação à automação de testes.



Fonte: Dados do autor.

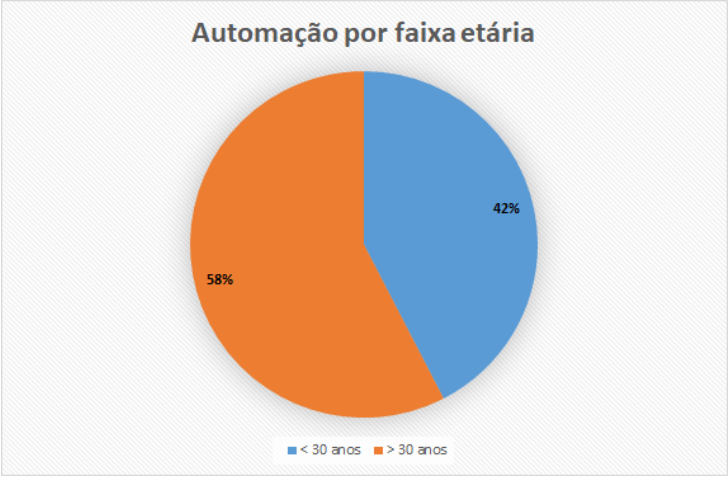
em participantes com idade igual ou superior à 30 anos.

A figura 17 mostra a distribuição dos resultados sobre a realização de automação em relação à região. Também foi realizada uma comparação para identificar essa distribuição de dados sobre a realização de automação em relação a região. Apenas os estados de São Paulo (SP) e Minas Gerais (MG) foram selecionados para a análise, devido a maior concentração de participantes, possibilitando resultados mais próximos a realidade.

A comparação entre a variável localização (dividida em SP e MG) e a proporção de usuários que realizam automação, apresentou uma proporção menor de automação por Minas Gerais do que em São Paulo mas não houveram evidências suficientes para dizer que essas pequenas diferenças são estatisticamente significativas. Através dos resultados é possível constatar uma diferença de 16% entre os estados citados.

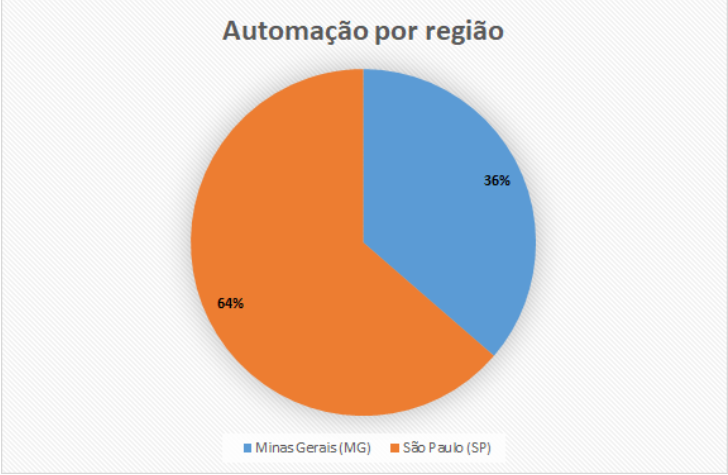
A comparação entre a variável tempo de atuação e a proporção de usuários que

Figura 16 – Realização de automação em relação à faixa etária.



Fonte: Dados do autor.

Figura 17 – Realização de automação em relação à região.



Fonte: Dados do autor.

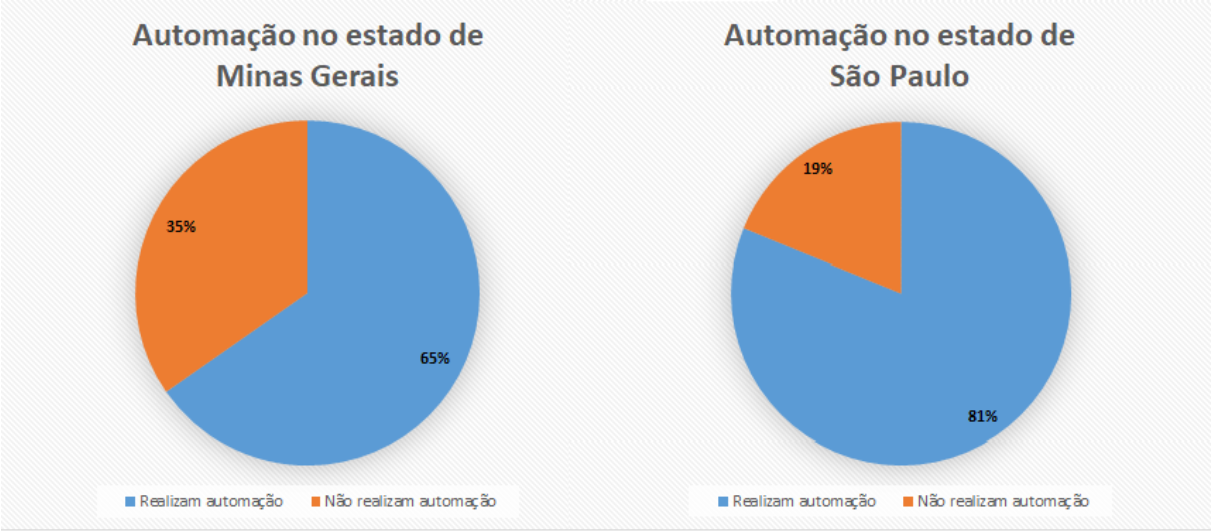
realizam automação, apresentou uma proporção menor de automação por profissionais que possuem de 0 a 5 anos de atuação na área de testes do que profissionais que mas também não houveram evidências suficientes para dizer que essas pequenas diferenças são estatisticamente significativas.

Ao realizar a comparação entre a variável escolaridade e a proporção de usuários que realizam automação, os resultados não apresentaram nenhuma diferença significativa entre os níveis de escolaridade em relação realização de automação.

Ao analisar a realização de automação dentre os participantes que possuem pós-graduação, foi possível perceber que existe probabilidade de um profissional de testes, que possui pós-graduação, realizar testes automatizados.

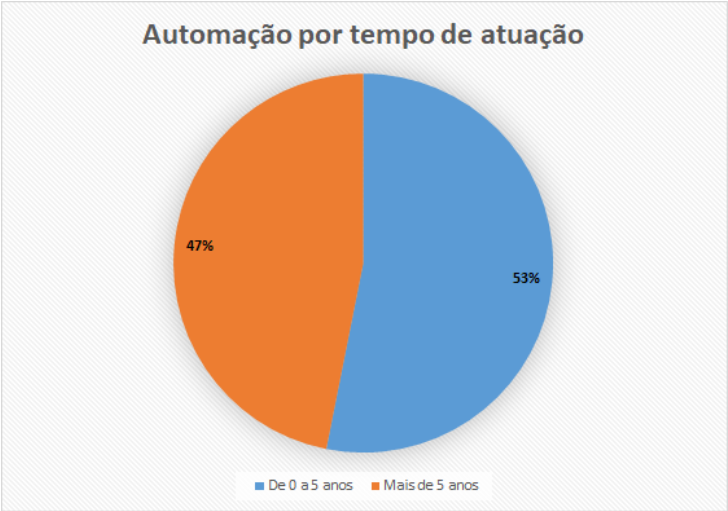
Após a análise do percentual de automação dentre os participantes que possuem nível superior completo, não houve nenhum resultado que mostre diretamente uma relação entre

Figura 18 – Comparação entre regiões em relação à automação de testes.



Fonte: Dados do autor.

Figura 19 – Realização de automação em relação ao tempo de atuação.



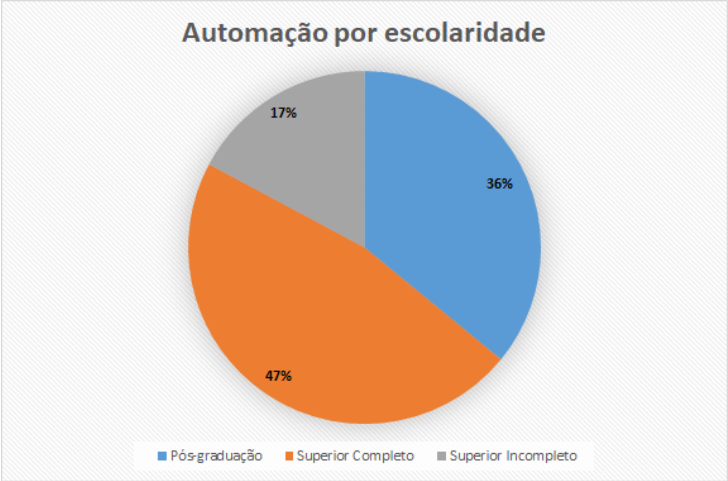
Fonte: Dados do autor.

profissionais que possuem curso superior completo e realização de testes automatizados.

Ao analisar a realização de automação dentre os participantes que possuem ensino superior incompleto, foi possível perceber que existe uma menor ocorrência de prática de automação de testes por profissionais com este nível de escolaridade.

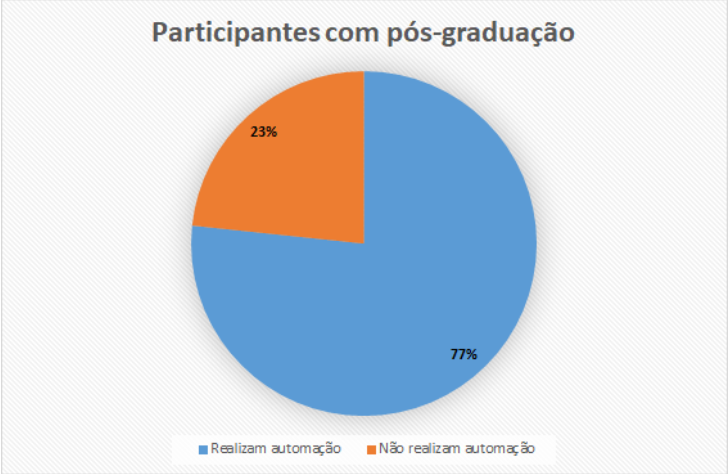
Ao questionar os participantes sobre quais as vantagens dos testes automatizados em relação aos manuais, a maioria selecionou a alternativa ”Eliminação do trabalho repetitivo de inserção de dados e observação dos resultados”seguida por ”Diminuição do tempo gasto com a execução dos testes”. Vale ressaltar que nesta pergunta os participantes poderiam selecionar mais de uma opção como resposta. Nenhum participante selecionou a opção ”Não há vantagens”,

Figura 20 – Realização de automação em relação ao nível de escolaridade.



Fonte: Dados do autor.

Figura 21 – Realização de automação entre os participantes que possuem pós-graduação.



Fonte: Dados do autor.

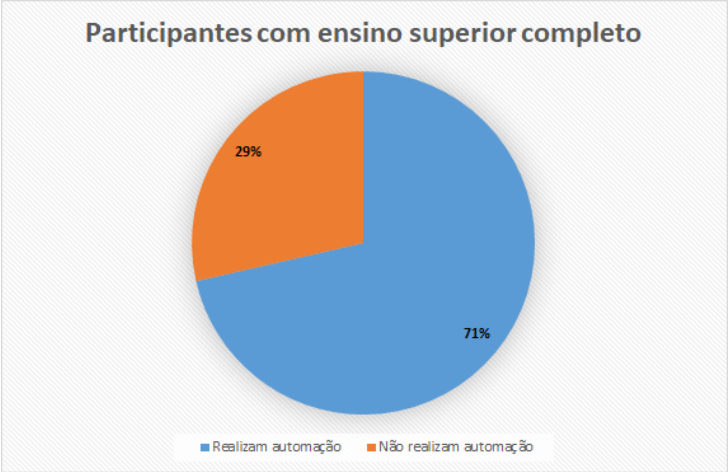
mostrando que todos eles estão cientes da importância dos testes automatizados.

Os participantes foram perguntados sobre quais documentos fazem parte da atividade de teste realizada por eles. Lembrando que nesta pergunta os participantes poderiam selecionar mais de uma opção como resposta. A figura 25 mostra os resultados obtidos. As opções de respostas com maior número de escolha foram Plano de Teste e Especificação de Caso de Teste. O documento menos selecionado por eles foi Relatório de Encaminhamento de Itens de Teste.

Os participantes foram questionados sobre quais aplicações para as quais eles realizam os testes. A pergunta foi de múltipla seleção, permitindo ao participante selecionar mais de uma resposta bem como inserir outra alternativa no campo “Outro”. A figura 26 mostra os resultados obtidos após a análise. A opção mais selecionada foi *Web*, seguido por *Android*. As opções citadas pelos participantes foram *APIs* e Sistemas embarcados.

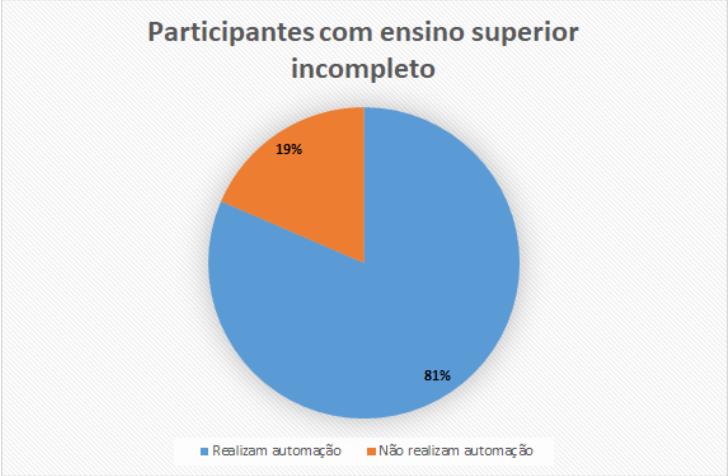
Também foi perguntado aos participantes, quais Ferramentas e *Frameworks* utilizadas

Figura 22 – Realização de automação dentre os participantes com superior completo.



Fonte: Dados do autor.

Figura 23 – Realização de automação entre os participantes com superior incompleto.



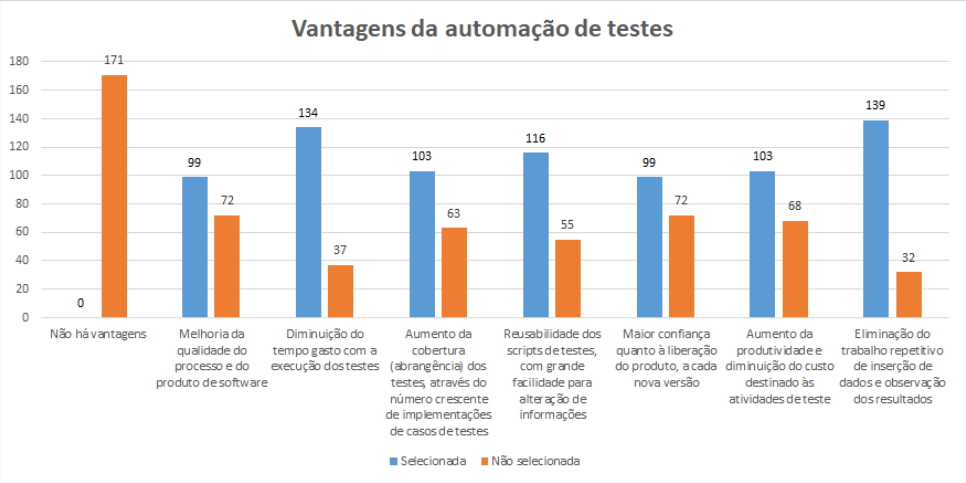
Fonte: Dados do autor.

por eles no processo de automação. Vale salientar que essa questão era aberta e os participantes poderiam responder o que quisessem. A tabela 1 contém os nomes das ferramentas e o número de participantes que as citaram. As ferramentas mais citadas foram *Selenium*, *Cucumber* e *Appium*.

Na pergunta sobre ferramentas utilizadas pelos participantes, muitos deles citaram também alguns *frameworks* que utilizam no processo de automação. A tabela 2 contém os nomes dos *frameworks* mais citados por eles e o número de participantes que os citaram. Os *frameworks* mais citados com maior número de citações foram *Capybara*, *JUnit*, *Robot Framework* e *Protractor*.

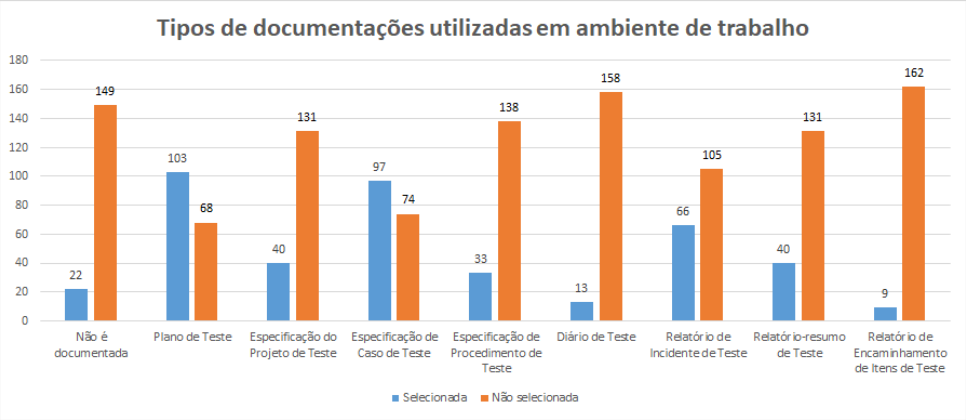
Os participantes foram questionados quanto à linguagem de programação utilizada para automatizar os testes. Esta foi uma pergunta aberta, permitindo citarem livremente as linguagens utilizadas por eles. As mais citadas foram: *Java* (70 citações), seguida por *Ruby* (38 citações) e *JavaScript* (24 citações). Na nuvem de palavras da figura 27 podemos perceber

Figura 24 – Vantagens da automação de testes.



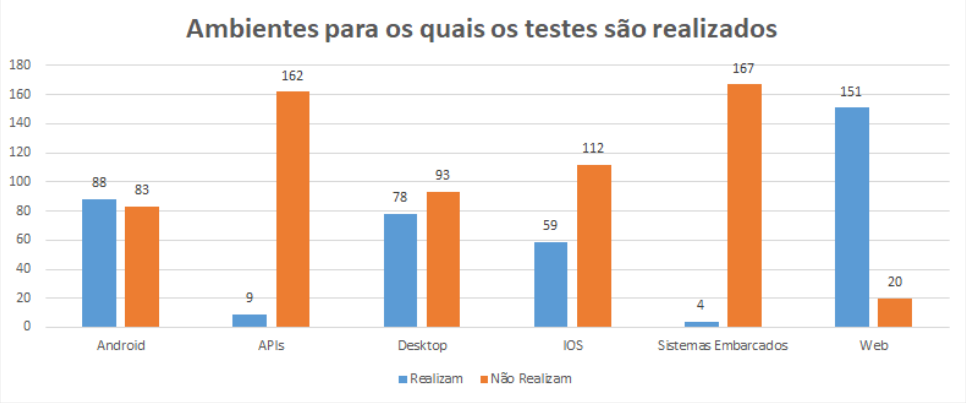
Fonte: Dados do autor.

Figura 25 – Documentos que fazem parte das atividades de testes.



Fonte: Dados do autor.

Figura 26 – Ambientes para os quais os testes são realizados.



Fonte: Dados do autor.

Ferramenta	Quantidade de pessoas que usam
Selenium	74
Cucumber	35
Appium	23
Jmeter	9
Vscode	8
Postman	6
SoapUI	6
Chai	4
Cypress	3
Katalon	3
RFT	3
TestComplete	3
Sikuli	2
UFT	2

Tabela 1 – Ferramentas citadas pelos participantes.

Framework	Quantidade de pessoas que usam
Capybara	12
Junit	12
Robot Framework	11
Protractor	10
Eclipse	6
HttpParty	5
Mocha	4
TestNG	4
Espresso	3
Rest-assured	3
Rspec	3
Calabash	2
Jasmine	2
Nunit	2
SpecFlow	2
WebdriverIO	2

Tabela 2 – *Frameworks* citados pelos participantes.

as linguagens mais frequentes nas respostas. O tamanho da fonte, na figura, simboliza o quão frequente o termo apareceu.

Foi perguntado aos participantes se eles realizam desenvolvimento ágil em seu ambiente de trabalho e se realizassem, deveriam informar qual metodologia era utilizada. A pergunta foi aberta e permitiu a livre escolha das respostas por parte dos participantes. A figura 28 mostra o percentual das respostas, apenas 28 (16%) participantes afirmaram não realizar desenvolvimento ágil, enquanto 143 (84%) responderam que realizam.

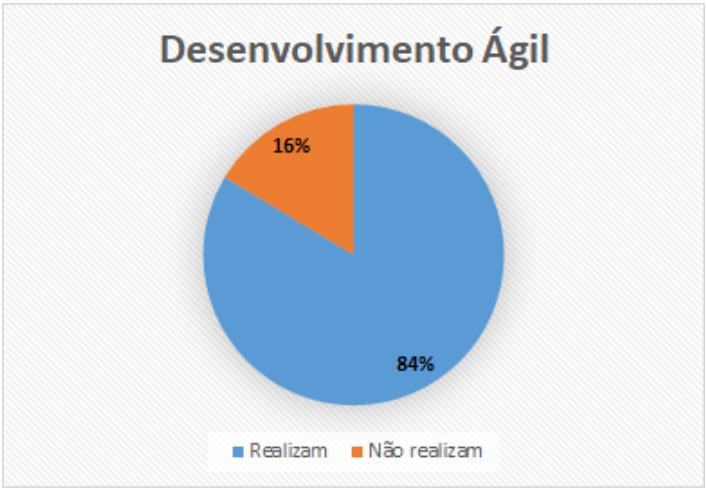
A tabela 3 mostra as principais metodologias ágeis mencionadas pelos participantes. Muitos deles citaram mais de uma metodologia e a mais mencionada delas foi o *Scrum* com 128 citações. Esse resultado mostra que o *Scrum* é uma metodologia amplamente utilizada por empresas de TI no mercado atual.

O questionário abordou a opinião dos participantes em relação aos conhecimentos

Figura 27 – Linguagens de programação mais usadas pelos participantes.

Fonte: Dados do autor.

Figura 28 – Realização de desenvolvimento ágil em ambiente de trabalho.



Fonte: Dados do autor.

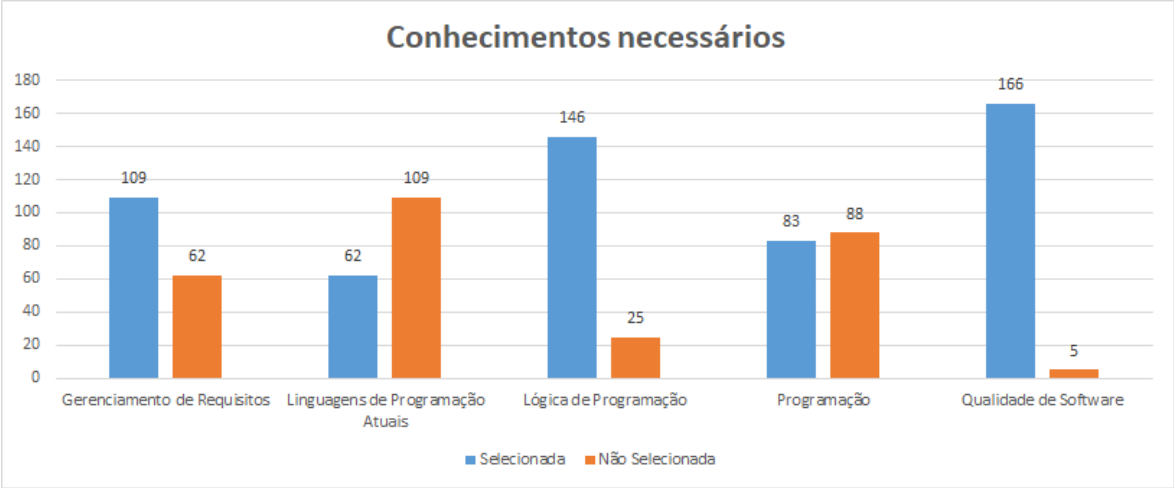
Metodologia Ágil	Quantidade de pessoas que usam
Scrum	128
Kanban	28
Lean	4
XP - eXtreme Programming	4

Tabela 3 – Metodologias Ágeis citadas pelos participantes.

que eles julgavam necessários para o desempenho do trabalho de um profissional de teste. A pergunta era composta por 5 alternativas de caixa de seleção além do campo “Outro” que possibilitou a inserção de outras alternativas de respostas. Foram disponibilizadas as opções Gerenciamento de Requisitos, Linguagens de Programação Atuais, Lógica de Programação, Programação e Qualidade de Software. Dentre as opções a mais selecionada foi Engenharia de Software seguida por Lógica de Programação.

Alguns participantes citaram outras opções de resposta no campo “Outro”. Os

Figura 29 – Conhecimentos necessários aos profissionais de teste de *software*.



Fonte: Dados do autor.

conhecimentos que de acordo com eles também são necessários são: Banco de Dados/SQL (citado por 5), Conhecimento em APIs (citado por 3), Regras de Negócio (citado por 4), Atualização sobre novas tecnologias (citado por 2) e Processos e metodologias ágeis (citado por 2).

Em pergunta aberta, os participantes foram questionados sobre os desafios encontrados por eles na automação de testes. A análise utilizada para a análise das respostas desta pergunta foi a análise qualitativa. Os principais desafios citados por eles foram:

- Falta de tempo
- Pouco conhecimento em programação
- Dificuldade em programar em uma linguagem específica
- Dificuldades em configurar o ambiente
- Pouco material de apoio em português
- Falta de oportunidade para praticar a automação
- Pouco conhecimento/domínio das ferramentas

Ao perguntá-los sobre os desafios enfrentados em seu ambiente de trabalho, o fator tempo foi novamente muito mencionado, como também a dificuldade em automação. Os seguintes desafios também foram citados pelos participantes:

- Cronograma curto
- Pouca documentação
- Difícil comunicação com os desenvolvedores
- Falta de valorização profissional

Os participantes foram questionados sobre como eles acham que os testes deveriam ser realizados em processos de desenvolvimento ágil de *software*. A maioria afirmou que os testes deveriam ser realizados em paralelo ao desenvolvimento. Muitos participantes afirmaram não saber responder a pergunta ou preferiram não respondê-la.

5.2 Discussão dos Resultados

Este estudo possibilitou representar opiniões e impressões de profissionais que atuam na área de testes de *software*. De acordo com os resultados, foi possível verificar que a maioria dos participantes é do sexo masculino e está concentrada nos estados de São Paulo e Minas Gerais. A faixa etária predominante é de 25 a 39 anos. Também foi possível dizer a maioria realiza testes automatizados bem como testes manuais. O nível de escolaridade predominante é superior completo.

Sobre o tempo de atuação na área de testes de *software*, as respostas foram bem variadas, mostrando que os participantes da pesquisa estão inseridos em vários níveis de carreira profissional. Em relação aos testes que eles realizam ou já realizaram, foi possível perceber que a Técnica Funcional é a mais utilizada por eles. Outro detalhe importante é que grande parte dos profissionais ainda não realizou muitos dos tipos de teste de *software* citados no questionário.

Sobre os desafios enfrentados por eles durante o trabalho, a maioria dos participantes afirmou enfrentar limitações por conta de restrições de tempo, como por exemplo, aumento do escopo de testes sem o aumento do prazo de entrega. Também foi possível identificar que alguns participantes afirmaram ter dificuldades em realizar automação. Algumas justificativas citadas por eles foram: pouco tempo disponível, dificuldade em programar, desconhecimento sobre linguagens de programação específicas, poucos conhecimentos para utilização das ferramentas de teste e dificuldade em encontrar materiais de apoio em português para o auxílio do utilização dessas ferramentas.

No que se refere aos desafios relacionados ao trabalho, novamente o fator tempo foi mencionado por grande parte dos participantes como um dos principais empecilhos para a realização de seus trabalhos. Alguns profissionais mencionaram que algumas empresas nem sempre disponibilizam aos seus colaboradores, uma documentação para auxílio na realização dos testes e que algumas vezes, quando disponibilizam uma documentação, nem sempre é suficiente.

Em relação às Ferramentas e *Frameworks* utilizados na automação, foi possível observar que os participantes utilizam principalmente os *frameworks Capybara*, *Junit* e *Robot Framework*. A ferramenta *Selenium* se destacou nesta pesquisa com 74 citações, sendo assim, a mais utilizada pelos participantes. Em relação à linguagens de programação, a linguagem *Java* foi citada por 70 participantes, sendo então, a mais utilizada por eles na automação de testes *software*.

Quanto às aplicações, os participantes afirmam realizar a maior parte de seus testes em aplicações *Web*. Mas também é possível observar que as aplicações *mobile*, como aplicações *Android* e *IOS*, também foram muito mencionadas. Para os participantes, o conhecimento em Qualidade de *Software* é indispensável para que o profissional possa atuar na área de testes de *Software*, bem como o conhecimento de Lógica de Programação.

6 CONSIDERAÇÕES FINAIS

Para acompanhar as constantes mudanças e possibilitar a criação de bons sistemas, muitas técnicas são necessárias para a garantia de qualidade de *software*. Este trabalho se baseou na descrição dos principais desafios enfrentados pelos profissionais de teste de *software*. O levantamento bibliográfico teve como objetivo ampliar a compreensão em relação aos conceitos dos testes de *software*.

Após correlacionar os conceitos estudados com as respostas obtidas através da aplicação de questionário, foi possível ter uma visão mais ampla do atual contexto de testes de *software*, possibilitando identificar possíveis dificuldades enfrentadas pelos profissionais na aplicação das técnicas e conceitos existentes.

Os resultados encontrados mostraram que existem muitas opções de ferramentas e *frameworks* disponíveis no mercado, possibilitando a escolha de acordo com a necessidade. Foi possível observar também que muitos profissionais enfrentam desafios para acompanhar as constantes atualizações e demandas por novas técnicas, linguagens de programação, *frameworks* e ferramentas para automação.

A partir da análise realizada, comparando o gênero dos participantes com o fator automação, evidenciou-se que homens e mulheres têm comportamentos diferentes. Os participantes do gênero feminino automatizam menos do que os participantes do gênero masculino. A pequena diferença de automação pelo gênero feminino para o gênero masculino, foi de aproximadamente 20%, mostrando a importância de incentivar e ensinar mais mulheres a automatizar.

Conclui-se, ao final deste trabalho, que a área de testes de *software* está em constante transformação e crescimento, influenciando cada vez mais, a criação de *software* de qualidade.

Um ponto fraco encontrado durante a realização deste trabalho foi o número reduzido de participantes de algumas regiões do Brasil, além da inexistência de participantes de alguns estados, impossibilitando uma análise mais profunda do contexto nacional. Mesmo diante dessa dificuldade, podemos considerar que este trabalho contribuiu para a compreensão da importância da execução de testes de *software*.

As limitações deste trabalho, descritas anteriormente, restringiram um pouco a análise e o diagnóstico do contexto nacional dos profissionais de testes de *software*. Faz-se necessário, então, realizar novas pesquisas com um maior recorte de localidade, com representação significativa de profissionais de TI de alguma região específica. Apesar da limitação geográfica do estudo proposto, possibilitaria um diagnóstico amplo e refinado de um contexto local, com riqueza de detalhes e fidedigno do contexto geral dos profissionais de testes de *software*.

REFERÊNCIAS

ALEXANDRE, D. M. Qualidade de software e o framework scrum: Uma proposta para melhorar a integração dos testes no processo de desenvolvimento de software da empresa pp ltda. 2016.

BARBOSA, F. B.; TORRES, I. V. O teste de software no mercado de trabalho. **Revista Tecnologias em Projeção**, v. 2, n. 1, p. 49–52, jun. 2011. Disponível em: ⟨http://revista.faculdadeprojecao.edu.br/index.php/Projecao4/article/view/82⟩.

BARTIé, A. **Garantia da Qualidade de Software**. 9. ed. [S.l.: s.n.], 2002. ISBN 978-85-352-1124-5.

BASTOS, A.; RIOS, E.; CRISTALLI, R.; MOREIRA, T. **Base de conhecimento em teste de software**. 3. ed. São Paulo: Evandro Mendonça Martins Fontes, 2012. ISBN 978-85-8063-053-4.

BERNARDO, P. C.; KON, F. A importância dos testes automatizados: Controle ágil, rápido e confiável de qualidade. **Engenharia de Software Magazine,**, v. 3, n. 1, p. 54–57, 2008. Disponível em: ⟨https://www.ime.usp.br/~kon/papers/EngSoftMagazine-IntroducaoTestes.pdf⟩.

BOARD, B. S. T. Q. B. Tecnologia, **Syllabys Agile Testing**. 2014.

BRUNELI, M. V. d. Q. A utilização de uma metodologia de teste no processo da melhoria da qualidade do software. 2006.

COHN, M. **Desenvolvimento de software com Scrum: Aplicando Métodos Ágeis com Sucesso**. 1. ed. [S.l.: s.n.], 2011.

CRESPO, A. N.; SILVA, O. J. d.; BORGES, C. A.; SALVIANO, C. F.; JUNIOR, M. d. T. e. A.; JINO, M. Uma metodologia para teste de software no contexto da melhoria de processo. 2004.

GERHARDT, T. E.; SILVEIRA, D. T. **Métodos de Pesquisa**. 1. ed. [S.l.]: UFRGS, 2009.

IEEE, C. S. **IEEE Standard for Software and System Test Documentation**. Ieee std 829-2008. [S.l.: s.n.], 2008. ISBN 978-0-7381-5746-7.

LAGES, D. S. Tecnologia, **Automação dos Testes: um lobo na pele de cordeiro?** 2010.

LUFT, C. C. Teste de software: uma necessidade das empresas. 2012.

MASSARI, V. L. **Gerenciamento ágil de projetos: uma visão preparatória para a certificação ágil do PMI (PMI-ACP)**. 1. ed. Rio de Janeiro: Brasport, 2014. ISBN 978-85-7452-696-6.

NETO, A. C. D.; NATALI, A. C. C.; ROCHA, A. R.; TRAVASSOS, G. H. Caracterização do estado da prática das atividades de teste em um cenário de desenvolvimento de software brasileiro. 2006.

OLSE, K.; PARVEEN, T.; BLACK, R.; DEBRA, F.; HAMBURGO, M.; MCKAY, J.; POSTHUMA, M.; SCHAEFER, H. Tecnologia, **Certified Tester: Foundation Level Syllabus**. 2018. Disponível em: ⟨https://www.bstqb.org.br/uploads/syllabus/syllabus_ctfl_2018br.pdf⟩.

PEREIRA, A. G. M. Testes funcionais de aplicações web e mobile. maio 2017. Disponível em: ⟨http://repositorio.uportu.pt/xmlui/handle/11328/2068⟩.

POPPENDIECK, M.; POPPENDIECK, T. **Implementando o Desenvolvimento Lean de Software: Do conceito ao dinheiro**. 1. ed. [S.l.: s.n.], 2011. ISBN 9788577807796.

PRESSMAN, R. **Engenharia de Software: uma abordagem profissional**. 7. ed. Porto Alegre: AMGH, 2011. ISBN 978-85-8055-044-3.

PRIKLADNICKI, R.; WILLI, R.; MILANI, F. **Métodos ágeis para desenvolvimento de software**. 1. ed. Porto Alegre: bookman, 2014. ISBN 978-85-8260-207-2.

REIS, E. A.; REIS, I. A. **Análise Descritiva de Dados**. 2002.

RIOS, E.; MOREIRA, T. **Teste de Software: 3ª Edição revisada e atualizada**. 3. ed. Rio de Janeiro: Editora Alta Books, 2013. ISBN 978-85-7608-775-5.

SALOMãO, R. G. Análise da relevância de testes de regressão para o mercado de desenvolvimento de software do triângulo mineiro. 2016.

SILVA, J. B. d.; ANASTáCIO, F. A. d. M. Multidisciplinar e Psicologia, **Método Kanban como Ferramenta de Controle de Gestão**. 2019. 1018–1027 p. Disponível em: ⟨https://idonline.emnuvens.com.br/id/article/view/1575⟩.

SILVA, J. d. S.; VIANA, G. B.; MACHADO, G. B. G.; SILVA, R. O. d. Tecnologia, **O processo de Teste de Software**. 2016. 99–108 p. Disponível em: ⟨http://revista.faculdadeprojecao.edu.br/index.php/Projecao4/article/view/704⟩.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. [S.l.: s.n.], 2013. ISBN 978-85-7936-108-1.

SOUZA, K. P. d.; GASPAROTTO, A. M. S. A importância da atividade de teste no desenvolvimento de software. **XXXIII Encontro Nacional de Engenharia de Produção**, 2013. Disponível em: ⟨http://www.abepro.org.br/biblioteca/enegep2013_TN_STO_177_007_23030.pdf⟩.

TELES, V. M. **Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade**. 2. ed. [S.l.: s.n.], 2014. ISBN 978-85-7522-400-7.

WAZLAWICK, R. S. **Engenharia de Software Conceitos e Práticas**. 1. ed. Rio de Janeiro: Elsevier, 2013. ISBN 978-85-352-6120-2.

APÊNDICE A – QUESTIONÁRIO *ONLINE* APLICADO AOS PROFISSIONAIS QUE REALIZAM ATIVIDADES DE TESTES DE *SOFTWARE*

Questionário sobre Atividades de Testes de Software

Meu nome é Raissa Oliveira Rodrigues e este questionário auxiliará na pesquisa do meu Trabalho de Conclusão de Curso em Sistemas de Informação pela Universidade Federal dos Vales do Jequitinhonha e Mucuri (UFVJM).

O objetivo principal é coletar dados a respeito de profissionais que atuam na área de Testes de Software.

Sua participação é voluntária e os resultados desta pesquisa serão utilizados apenas para fins acadêmicos. Conto com a sua participação e se possível, gostaria que você compartilhasse este questionário com outras pessoas.

Obrigada por sua colaboração!

*Obrigatório

Sobre você

1. **Qual o seu gênero?** *
   Marcar apenas uma oval.

   - ◯ Feminino
   - ◯ Masculino
   - ◯ Prefiro não informar

2. **Qual a sua idade?** *

   _____

3. **Em qual estado você mora?** *

Marcar apenas uma oval.

- Acre (AC)
- Alagoas (AL)
- Amapá (AP)
- Amazonas (AM)
- Bahia (BA)
- Ceará (CE)
- Distrito Federal (DF)
- Espírito Santo (ES)
- Goiás (GO)
- Maranhão (MA)
- Mato Grosso (MT)
- Mato Grosso do Sul (MS)
- Minas Gerais (MG)
- Pará (PA)
- Paraíba (PB)
- Paraná (PR)
- Pernambuco (PE)
- Piauí (PI)
- Rio de Janeiro (RJ)
- Rio Grande do Norte (RN)
- Rio Grande do Sul (RS)
- Rondônia (RO)
- Roraima (RR)
- Santa Catarina (SC)
- São Paulo (SP)
- Sergipe (SE)
- Tocantins (TO)
- Não moro no Brasil

4. **Qual o seu nível de escolaridade?** *

Marcar apenas uma oval.

- Superior Incompleto
- Superior Completo
- Pós-graduação

5. **Você atua na área de Testes de Software há quantos anos?** *
Marcar apenas uma oval.

- ◯ Menos de 1 ano
- ◯ De 1 a 3 anos
- ◯ De 3 a 5 anos
- ◯ De 5 a 7 anos
- ◯ De 7 a 10 anos
- ◯ Mais de 10 anos

Sobre o seu trabalho

6. **Quais testes de softwares você realiza ou já realizou?** *
Marque todas que se aplicam.

- ☐ Técnica Funcional (Teste Caixa Preta)
- ☐ Teste de Regressão
- ☐ Teste Paralelo
- ☐ Técnica Estrutural (Teste Caixa Branca)
- ☐ Teste de Carga (Teste de Estresse)
- ☐ Teste de Execução (Teste de Performance)
- ☐ Teste de Usabilidade
- ☐ Teste de Aceitação
- ☐ Teste de Unidade
- ☐ Teste de Integração
- ☐ Teste de Sistema
- ☐ Teste de Configuração
- ☐ Teste de Instalação
- ☐ Teste de Recuperação
- ☐ Teste de Segurança
- ☐ Teste de Volume
- ☐ Outro: _____

7. **Quais os desafios encontrados por você na realização de testes de software?** *
Marque todas que se aplicam.

- ☐ A atividade de teste é limitada por restrições de cronograma
- ☐ Há a falta de profissionais especializados na área de teste
- ☐ Existem dificuldades em implantar um processo de teste
- ☐ Desconhecimento de um procedimento de teste adequado
- ☐ Desconhecimento de técnicas de teste adequadas
- ☐ Há o desconhecimento sobre como planejar a atividade de teste
- ☐ O teste é um processo caro
- ☐ Outro: _____

8. **Como os testes são realizados?** *
   Marque todas que se aplicam.

   - [ ] Manualmente
   - [ ] Através da automatização

9. **Se automatizado, quais ferramentas são utilizadas?**

   _____

   _____

   _____

   _____

   _____

10. **Quais a(s) linguagem(s) de programação você utiliza para automação dos seus testes?**

    _____

    _____

    _____

    _____

    _____

11. **Na sua opinião, quais as vantagens dos testes automatizados em relação aos testes manuais?** *
    Marque todas que se aplicam.

    - [ ] Maior confiança quanto à liberação do produto, a cada nova versão
    - [ ] Eliminação do trabalho repetitivo de inserção de dados e observação dos resultados
    - [ ] Aumento da produtividade e diminuição do custo destinado às atividades de teste
    - [ ] Não há vantagens
    - [ ] Diminuição do tempo gasto com a execução dos testes
    - [ ] Reusabilidade dos scripts de testes, com grande facilidade para alteração de informações
    - [ ] Aumento da cobertura (abrangência) dos testes, através do número crescente de implementações de casos de testes
    - [ ] Melhoria da qualidade do processo e produto de software
    - [ ] Outro: _____

12. **Marque os documentos abaixo que fazem parte das atividades de testes no seu trabalho:** *
Marque todas que se aplicam.

- [ ] Plano de Teste
- [ ] Especificação do Projeto de Teste
- [ ] Especificação de Caso de Teste
- [ ] Especificação de Procedimento de Teste
- [ ] Diário de Teste
- [ ] Relatório de Incidente de Teste
- [ ] Relatório Resumo de Teste
- [ ] Relatório de Encaminhamento de Item de Teste
- [ ] Não é documentada
- [ ] Outro: _____

13. **A empresa que você trabalha realiza desenvolvimento ágil? Se sim, baseado em qual metodologia?** *

_____

14. **Quais dificuldades encontradas por você na automação de testes?** *

_____

_____

_____

_____

_____

15. **Quais os desafios que você encontra no trabalho com testes?** *

_____

_____

_____

_____

_____

16. **Você realiza testes de software para qual (quais) aplicações?** *
Marque todas que se aplicam.

- [ ] Android
- [ ] IOS
- [ ] Desktop
- [ ] Web
- [ ] Outro: _____

17. **Na sua opinião, como os testes deveriam ser realizados em processos de desenvolvimento ágil de software?** *

18. **Quais conhecimentos você considera necessários para uma pessoa trabalhar com testes de software?** *

Marque todas que se aplicam.

- [ ] Programação
- [ ] Lógica de Programação
- [ ] Linguagens de Programação atuais
- [ ] Gerenciamento de Requisitos
- [ ] Qualidade de Software
- [ ] Outro:

Obrigada!