An Evidence-Based Study on Automated Exploratory Testing

ABSTRACT: Exploratory Testing is a growing approach in the industrial scenario due to the emerging use of agile practices in the software development process to satisfy the main market requirements such as short-term and high-quality deliveries of products or services. In this context, a literature review was carried out aiming to identify mainly the positive factors and gaps in the applicability of the automated exploratory test, as well as to observe other factors that can directly influence the software life cycle. The results show at least 11 positive factors (benefits) and 15 gaps (challenges) that are interrelated to using this agile testing approach in an automated way, in addition to several other factors that can also be investigated in future research.

Keywords: Automated Exploratory Testing, Exploratory Testing Tools, Automated Ad hoc Testing, Software Testing.

Um Estudo Baseado em Evidências Sobre Teste Exploratório Automatizado

RESUMO: O Teste Exploratório é uma abordagem crescente no cenário industrial em virtude da emergente utilização de práticas ágeis no processo de desenvolvimento de *software* para satisfazer as principais exigências de mercado, tais como entregas de produtos ou serviços em curto prazo e com alta qualidade. Neste contexto, uma revisão da literatura foi realizada objetivando identificar principalmente os fatores positivos e lacunas da aplicabilidade do teste exploratório automatizado, bem como observar outros fatores que podem influenciar diretamente no ciclo de vida do *software*. Os resultados evidenciam pelo menos 11 fatores positivos (benefícios) e 15 lacunas (desafios) que estão inter-relacionadas ao uso dessa abordagem de teste ágil de forma automatizada, além de vários outros fatores que podem também ser investigados em pesquisas futuras.

Palavras-chave: Teste Exploratório Automatizado, Ferramentas de Teste Exploratório, Teste Ad Hoc Automatizado, Teste de Software.

Agradecimentos: Os autores agradecem a CAPES pela concessão de bolsa de Doutorado institucional ao Programa de Pós-Graduação em Ciência da Computação - PPGCC/UFPA. Este trabalho pertence ao projeto SPIDER/UFPA (*http://www.spider.ufpa.br*).

1. INTRODUÇÃO

A importância da existência de um processo de teste de software eficaz e eficiente tem se tornado cada vez evidente no ciclo de desenvolvimento do software, cujo objetivo é permitir a melhoria e a avaliação da qualidade dos produtos de software. Para isso, o testador pode utilizar várias abordagens (técnicas e estratégias de teste) para detectar os defeitos (ROPER, 2019). É notável que o mercado exige que os produtos sejam entregues em curto prazo com alta qualidade, dessa forma tem ocorrido um crescimento pela automação de várias atividades do processo de desenvolvimento de software e de teste, principalmente em ambientes que utilizam um processo ágil. A proposta de automação de teste propõe um conjunto de benefícios, os quais são a redução de custo, tempo e esforço, eficiência em tarefas que são complexas de serem realizadas manualmente, capacidade de realizar uma grande quantidade de atividades repetidas de maneira intensa, entre outros. Contudo, há também vários fatores a serem resolvidos que surgiram naturalmente no contexto de automação de teste (KLAMMER e RAMLER, 2017).

Dentre as várias abordagens de teste ágil (GREGORY e CRISPIN, 2014), alguns estudos centralizaram os esforços em propor ferramentas para automatizar o teste exploratório, visando colocar em software os fatores positivos oferecidos pela abordagem manual (AHO e VOS, 2018). O teste exploratório, também conhecido por teste *Ad hoc*, enfatiza a liberdade e a responsabilidade do testador para explorar qualquer parte do sistema, no qual o testador trata o processo de aprendizado relacionado ao teste, *design* de teste, execução de teste e interpretação de resultados como atividades de apoio executadas em paralelo durante todo o projeto (KANER, 2008). Em (BACH, 2004) e (SWEBOK, 2014) consta a definição que o teste exploratório é um processo interativo que proporciona o aprendizado rápido do software, sendo o *design* de teste e a execução de teste mutuamente aplicados, isto é, os cenários de testes não são definidos previamente em um plano de teste, mas são projetados, executados e modificados simultaneamente.

Em relação à utilização de teste exploratório automatizado, a literatura descreve vários benefícios, por exemplo: a) permite sistematizar os esforços de exploração evitando testes em áreas já cobertas; b) capacidade de executar atividades repetitivas em curto tempo; c) capacidade de gerar muitos casos de teste; d) redução do esforço, tempo e custo; entre outros. Destaca-se que alguns desses fatores positivos são soluções para lacunas existentes na abordagem manual, sendo o caso dos itens “a)” e “b)”. Tais itens permitem organizar a estratégia de teste direcionando em que regiões podem ser focadas a exploração e intensificar a realização de teste em partes complexas do sistema, respectivamente (AHO e VOS, 2018).

De modo geral, a automação de teste pode ocorrer em várias fases do processo de teste e direcionados a alguns contextos de teste, sendo que o teste exploratório por sua característica natural pode ser flexível, no caso permite ser aplicado em qualquer momento do ciclo de vida do software. Nessa circunstância, consideram-se as fases do processo de teste sendo o planejamento, a especificação de casos de teste (projeto), a execução e a análise, e em relação ao contexto de teste é considerado o teste de sistema, teste de regressão, teste unidade, teste de integração, teste de aceitação e outros. Diante disso, o presente trabalho visa identificar os benefícios, as lacunas (desafios) da aplicabilidade do teste exploratório automatizado e outros fatores que podem influenciar no ciclo de vida do software. Para isso, foi conduzida uma revisão da literatura para obter resultados baseando-se em evidências acadêmicas ou industriais inseridas no contexto de engenharia de software.

Sendo assim, o restante deste estudo está estruturado da seguinte maneira: a Seção 2 apresenta alguns trabalhos relacionados; a Seção 3 descreve o método de pesquisa

conduzido; na Seção 4 encontram-se os resultados e as discussões desta pesquisa; a Seção 5 discute sobre as limitações do estudo; e a Seção 6 apresenta as conclusões e os trabalhos futuros.

2. TRABALHOS RELACIONADOS

No estudo de (AHO e VOS, 2018) há apenas a apresentação de vários desafios enfrentados para automatizar teste em GUI (*Graphic User Interface*). O referido estudo apresentou esses desafios organizados em três categorias ou níveis de automação de teste, as quais foram: a) teste baseado em *scripts*; b) baseado em modelo; e c) os desafios na automatização com o teste exploratório. Os resultados apresentados sobre teste exploratório automatizado indicam benefício na redução de tempo, custo e esforço, porém um dos maiores desafios é a definição de propriedades que são invisíveis e visíveis ao olhar humano no oráculo de teste automatizado, bem como seleção dos caminhos a serem percorridos, pois geralmente as ferramentas extraem uma lista de ações na GUI e, a partir disso, selecionam aleatoriamente a próxima ação. Contudo, extrair uma quantidade suficiente de dados que proporcione uma ótima acurácia do modelo do sistema requer um significativo tempo de execução e, geralmente, esse tempo é limitado. A estratégia comum é alcançar todas as partes da GUI com uma quantidade mínima de interações. Além disso, esse estudo não revela qual metodologia foi utilizada para encontrar as evidências (estudos) que atendessem tal proposta do seu trabalho.

Em (PFAHL *et al.*, 2014) houve a aplicação de um *survey* voltado aos engenheiros de software dos países da Estônia e Finlândia. O *survey* objetivou identificar qual a experiência, quais os fatores que influenciam na aplicabilidade do teste exploratório, assim como observar o que os engenheiros de software pensam sobre a referida abordagem de teste. Nesse primeiro questionamento identificou-se detalhadamente qual contexto o teste exploratório automatizado tem sido mais aplicado. Os resultados evidenciaram que o uso mais constante havia sido no contexto de teste de sistema, teste de regressão, teste de usabilidade e *smoke testing*. Nesse contexto, o referido autor sugere a realização de mais estudos focados na identificação de como o teste exploratório com abordagem automática tem sido utilizado. Nota-se que o estudo não abrange a identificação de lacunas e fatores positivos ao uso de teste exploratório automatizado.

No trabalho (GAROUSI e MÄNTYLÄ, 2016) foi conduzida uma revisão da literatura multivocal, onde investigaram-se trabalhos acadêmicos, *blogs* e *web pages* para observar em quais fases do processo de teste a automação de teste pode ser aplicada. Os autores apresentam uma lista de atividades de teste que podem ser automatizadas, dentre essas é mencionado que o uso de teste exploratório automatizado é adequado na fase de especificação de teste (projeto de casos de teste), permitindo o projeto de uma lista de casos de teste ou requisitos de teste para satisfazer principalmente o critério de cobertura integrado. O autores citam também alguns fatores importantes que devem ser considerados ou pensados quando se deseja automatizar teste, os quais são relevantes para definir o que automatizar. De modo geral, o estudo discute atividades de teste que podem ser automatizados, sem descrever quais seriam os fatores positivos e os desafios, por exemplo do uso de teste exploratório automatizado.

Diante disso, ressalta-se que os trabalhos supracitados descrevem alguns desafios que envolvem o teste exploratório automatizado, buscam identificar qual contexto tem sido mais aplicado e as fases do processo de teste que a referida abordagem de teste pode ser automatizada. Dentre os três trabalhos, apenas (GAROUSI e MÄNTYLÄ, 2016) utiliza um procedimento estruturado de investigação das evidências, contudo considera relatos em blogs, *web pages* e outras fontes que eleva o risco da validade de alguns resultados.

Portanto, o presente estudo diferencia-se por focar na discussão sobre os pontos positivos, as lacunas e outros fatores relevantes na aplicabilidade do teste exploratório automatizado. Para isso, foram aplicados procedimentos estruturados e fundamentados na coleta, extração e análise dos dados, buscando evidências em fontes seguras, evitando agregar alto risco de validade aos resultados para que possam fundamentar pesquisas futuras.

3. REVISÃO DA LITERATURA

No presente trabalho foi conduzida uma revisão da literatura avaliando a existência de evidências para obter uma visão holística dos pontos fortes e das lacunas sobre a automação da abordagem de teste exploratório. Esta pesquisa restringe-se a trabalhos publicados no período entre 01/2001 até 04/2020, pois é compreensível que a primeira vez que o termo Teste Exploratório tem sido consolidado foi a partir da publicação do livro *Lessons Learned in Software Testing: A Context-Driven Approach* em 2001 pelos autores Cem Kaner, James Bach, Bret Pettichord (principais pesquisadores na área). Estes autores apresentam os primeiros relatos de experiência da aplicabilidade de Teste Exploratório. A seguir os principais elementos desse processo de revisão da literatura são apresentados.

3.1 Questões de pesquisa

O principal objetivo da revisão é identificar de forma geral quais os pontos fortes e as lacunas são apresentados em trabalhos que aplicam o teste exploratório automatizado ou que envolvam utilizam alguma ferramenta (*framework* não automatizado) visando melhorar a efetividade da aplicabilidade dessa abordagem de teste. Para nortear a identificação desses aspectos, tem sido estabelecido também um conjunto de questões secundárias (QS) inter-relacionadas com a questão principal (QP). Tais questões secundárias objetivam esclarecer outros fatores importantes na aplicabilidade de Teste Exploratório automatizado. São elas

- **QP – Quais os pontos fortes e as lacunas sobre a aplicabilidade do Teste Exploratório automatizado?**
  A resposta é apresentada na forma de uma listagem dos vários fatores que foram positivos na aplicação de teste exploratório automatizado, bem como uma listagem das lacunas de cada estudo sobre essa abordagem de teste.
- **QS1 – Quais os contextos de teste são envolvidos na aplicabilidade do Teste Exploratório automatizado?**
  A resposta é classificada conforme a fase do processo de teste onde cada trabalho é analisado, envolvendo Documentação, Teste de Unidade, Teste de Integração, Teste de Sistema, Teste de Aceitação e Teste de Regressão.
- **QS2 – Quais as atividades de desenvolvimento de projeto de software são envolvidas na aplicabilidade do Teste Exploratório automatizado?**
  A resposta é classificada conforme a atividade de desenvolvimento de projeto de software, onde cada trabalho é analisado se envolve o Planejamento, o Projeto, a Execução, o Gerenciamento e a Medição ou Métricas.
- **QS3 – Quais as abordagens de automação são utilizadas para aplicabilidade do Teste Exploratório?**
  A resposta é classificada conforme as abordagens de automação, envolvendo interface gráfica do usuário (GUI) ou código do software.
- **QS4 – Quais as técnicas de automação são utilizadas para aplicabilidade do Teste Exploratório?**

A resposta é classificada conforme as técnicas de automação utilizadas, por exemplo pode ser processamento de linguagem natural, aprendizado de máquina, algoritmos genéticos, inteligência artificial, entre outras.

- **QS5 – Quais os cenários envolvidos na aplicação do Teste Exploratório automatizado?**
  A resposta é classificada no contexto onde tem sido aplicado o teste exploratório, por exemplo acadêmico ou industrial.

3.2 *String* de Busca

Após a definição das questões de pesquisa, ocorreu a seleção das fontes para realizar as buscas de dados de forma automática, sendo definidas três fontes de buscas relevantes ao meio científico, por serem as principais bases de dados internacionais que reúnem diversos eventos sobre Engenharia de Software no mundo: IEEE Xplore Digital Library, ACM Digital Library e Scopus (Science Direct). Como já mencionado, o período considerado relevante contemplou de 01/2001 a 04/2020.

Posterior à definição das bases, uma pesquisa *ad hoc* foi realizada nas bases selecionadas, utilizando as duas *strings* (str), uma de cada vez, considerando a detecção dessas palavras-chave em todo o conteúdo do texto dos estudos a serem filtrados: str1 – **(("exploratory test*") AND (automat*));** str2 – **(("exploratory test*") AND (tool*))**.

Destaca-se que houve a necessidade da utilização das duas *strings*, pois alguns trabalhos que estavam aderentes ao objetivo deste presente estudo foram detectados somente quando se utilizou especificamente cada uma delas.

3.3 Estratégia de Seleção dos Estudos Primários

Para garantir informações mais confiáveis e relevantes ao presente estudo, houve a definição de alguns critérios de inclusão e exclusão, conforme Quadro 1 e Quadro 2, respectivamente.

Quadro 1 – Critérios de Inclusão.

ID	Critérios
CI1	Estudos que apresentem primária ou secundariamente pontos fortes e/ou lacunas no contexto de aplicação do teste exploratório automatizado.
CI2	Estudos que apresentem relatos de experiência da indústria, ou pesquisas de caráter experimental ou teórico, contanto que apresentem exemplos de aplicação, descrição de experimentos ou casos reais, de pontos fortes e/ou lacunas observadas na aplicação de teste exploratório automatizado.

Fonte: Elaboração própria (2020).

Quadro 2 – Critérios de Exclusão.

ID	Critérios
CE1	Estudos que claramente não atendam a questão de pesquisa.
CE2	Estudos repetidos (em mais de uma fonte de busca) terão apenas sua primeira ocorrência considerada.
CE3	Estudos enquadrados como resumos, *keynote speeches*, cursos, tutoriais,

	workshops e afins.
CE4	Estudo que não estiver inserido no contexto de Projetos de Software, Indústria de Software ou Engenharia de Software.
CE5	Trabalhos que não contêm as *strings* de busca da pesquisa em seu título, resumo ou nas palavras-chave.

Fonte: Elaboração própria (2020).

Após a realização da busca dos trabalhos nas fontes pré-definidas, os estudos primários foram selecionados por um pesquisador conforme os procedimentos: a) Primeira seleção a partir da leitura do título, palavras-chave e resumo; b) Segunda seleção a partir da leitura dos tópicos restantes apresentados, como introdução, metodologia, resultados e conclusão. Apesar de ser apenas um pesquisador realizando a seleção dos trabalhos, ressalta-se que todos os procedimentos e documentos do início ao fim deste estudo foram analisados por um especialista na área de engenharia de *software*, cuja pessoa é professor doutor, coordenador do projeto SPIDER e pesquisador na correspondente área em questão na Universidade Federal do Pará (UFPA). Além disso, menciona-se que a ferramenta JabRef, a qual é voltada para o gerenciamento de referências, foi utilizada para eliminar artigos duplicados.

Apenas foram considerados trabalhos escritos em inglês, haja vista a relevância da análise dos estudos desenvolvidos internacionalmente e também a necessidade de expandir a abrangência da pesquisa, uma vez que é a língua definida como padrão na grande maioria dos periódicos e das conferências internacionais. Além disso, foram desconsiderados os trabalhos que estavam indisponíveis para consulta pela internet, mesmo utilizando o acesso pelo *login* institucional da UFPA.

3.4 Extração e Síntese dos Dados

Para a extração e síntese dos dados, as orientações propostas por Cruzes e Dyba foram seguidas (CRUZES e DYBA, 2011). Mediante a grande quantidade de trabalhos selecionados, utilizou-se uma planilha eletrônica para a coleta dos dados de publicação, contexto e evidências. Além disso, a técnica de síntese e análise temática foi utilizada para apresentar os resultados.

Após aplicar os critérios de seleção e eliminar os trabalhos duplicados, conforme ilustrado na Tabela 1, restou um total de 20 estudos primários relevantes ao tópico desta pesquisa. Esses estudos primários estão listados por meio de identificadores (ID) para uma melhor referência ao longo do texto, onde esta planilha eletrônica pode ser visualizada acessando a URL *https://goo.gl/vfWFZ5*.

Tabela 1 – Seleção dos estudos primários.

Fonte	**Retornados**	**Excluídos**		**Incluídos**	
		Duplicados	**CE**	**Quantidade**	**Perc. (%)**
IEEE	41	7	15	19	91,3%
ACM	28	1	26	1	8,7%
SCOPUS	33	2	31	0	0%
Total	**102**	**10**	**72**	**20**	**100%**

Fonte: Elaboração própria (2020).

4. RESULTADOS E DISCUSSÃO

A partir dos 20 estudos primários selecionados e analisados, considera-se que as questões de pesquisa de interesse deste estudo foram respondidas de forma satisfatória. Os resultados estão organizados por questão de pesquisa sendo sintetizados e analisados seguindo a proposta por Cruzes e Dybå (2011). As Tabelas 2 e 3 (referente à QP) contêm os principais pontos positivos e as lacunas observadas nos trabalhos inclusos, respectivamente.

Inicialmente, destaca-se que os trabalhos primários inclusos abordavam tanto a proposta de desenvolver uma ferramenta (ou *framework*) que aplica o teste exploratório automatizado, quanto ferramentas que auxiliam o processo de teste exploratório manual com o intuito de aperfeiçoar tais testes. No primeiro caso, algumas ferramentas abordavam as fases de projeto, execução e análise de teste de forma automatizada, isto é, basicamente a ferramenta baseava-se em um modelo do sistema para gerar casos de teste e analisar os resultados. No segundo caso, as ferramentas geram automaticamente casos de teste a partir da análise de dados (*logs*) da execução manual de teste exploratório a fim de aprimorar as estratégias de teste por meio da descoberta de áreas e funcionalidades pouco exploradas. Outro fator relevante observado é que 8 estudos evidenciaram o uso dessas propostas automatizadas em processos ágeis, até mesmo pelo fato do teste exploratório ser uma abordagem ágil.

Destaca-se que a estratégia de construção do modelo do sistema em questão foi principalmente baseada no domínio (contexto) da aplicação, eventos de interface gráfica do usuário (GUI) e nos requisitos funcionais do sistema. A partir do modelo do sistema as ferramentas geravam automaticamente os casos de teste, sendo que tais testes foram focados sob a perspectiva do usuário final realizando interações com a GUI ou a partir de uso de API (interface de programa do programa de aplicação) para aplicações sem interface gráfica. Observou-se também que as aplicações mais utilizadas nos experimentos para a validação dessas ferramentas foram sistemas web e aplicações para dispositivos móveis.

Tabela 2 – Mapeamento dos dados referentes à questão principal desta pesquisa.

QP – Quais os pontos fortes e as lacunas sobre a aplicabilidade do Teste Exploratório automatizado?		
ID	Descrição dos Fatores Positivos	Estudos
1	Diminui o custo para executar as tarefas manuais;	A4, A6, A7, A11, A15, A17, A20
2	Diminui o tempo para execução de tarefas manuais;	A2, A6, A9, A11, A13, A17, A18, A19, A20
3	Diminui o esforço e retrabalho de tarefas manuais;	A2, A3, A4, A5, A6, A11, A17, A18, A20
4	Eficiente como abordagem complementar a outras técnicas de teste (teste baseado a modelo, guiado a eventos na GUI, baseados em requisitos funcionais e não funcionais, teste de usabilidade);	A1, A2, A3, A6, A10, A11, A13, A14, A18, A19, A20
5	Ferramentas geram uma variedade grande de casos de teste;	A3, A4, A6, A10, A11, A15, A16, A18, A19
6	Oráculo automatizado oferece resultados rápidos de uma	A3, A6, A8, A11,

QP – Quais os pontos fortes e as lacunas sobre a aplicabilidade do Teste Exploratório automatizado?		
ID	Descrição dos Fatores Positivos	Estudos
	grande quantidade de dados/resultados a serem analisados;	A18, A19
7	Permite aplicar estratégias de exploração sistemáticas;	A1, A12, A15, A19
8	Permite melhorar o modelo do sistema ou requisitos;	A2, A3, A8, A10, A11, A13, A18, A19, A20
9	As ferramentas geram casos de teste com alta cobertura das funcionalidades;	A6, A11, A13, A20
10	Permite a verificação minuciosa das funcionalidades de difícil análise e observação humana;	A5, A6, A10, A11, A15, A16, A18, A19, A20
11	Casos de teste gerados automaticamente podem ser editáveis manualmente	A6, A11, A13, A16, A18

Fonte: Elaboração própria (2020).

Enfatiza-se que os principais benefícios providos nas estratégias utilizadas para automatizar ou auxiliar o teste exploratório foi a redução do esforço manual, do tempo e o do custo (vide Tabela 2). Esses três benefícios supracitados são alcançados, principalmente, na atividade de avaliação dos resultados pelo oráculo de teste, por exemplo, como as ferramentas geram e executam automaticamente uma quantidade bastante expressiva de casos de teste, surge então a necessidade de avaliar muitos resultados para definir o que seria defeito ou não. Nesse contexto, essa avaliação torna-se uma tarefa muita onerosa em tempo, esforço e custo para realizar manualmente.

As ferramentas de software apresentaram um ponto bastante importante em relação às estratégias de exploração e que também tem ligação direta com economia de esforço, o qual é o caso de aperfeiçoar (otimizar) a exploração evitando a repetição de teste em áreas ou funcionalidades que já tenham sido exploradas. Para exemplificar tal benefício, na aplicação de teste exploratório não há um guia de apoio instantâneo ao testador durante a exploração que o conceda a possibilidade de observar quais áreas ou funcionalidades estão sendo alcançadas. Dessa forma, há uma alta probabilidade de ocorrer a repetição de testes em áreas já cobertas durante a sessão de teste, pois geralmente a análise de cobertura é realizada somente após o fim das sessões de teste. Nesse caso, a ferramenta otimizou a exploração em tempo real, conforme o tempo da sessão, proporcionando a aplicação de testes mais sistemáticos, sejam automáticos ou manuais a partir da análise dos caminhos percorridos. Essa aplicabilidade de teste mais sistemática gerou resultados que possibilitaram melhorar, principalmente, o modelo e os requisitos do sistema, permitindo até mesmo desvendar requisitos omissos no momento da elicitação.

As estratégias automatizadas proporcionam também uma quantidade de verificações (analise de resultados e exploração minuciosa) considerada quase impossível de ser realizada manualmente em um curto período de tempo e de difícil percepção ao olhar humano. Contudo, ainda houve a necessidade de intervenção humana para certificar que os caminhos percorridos não estavam mapeados no modelo ou que os resultados obtidos são válidos para definir se era de fato um defeito.

A atividade manual em conjunto com a automática foi importante também para melhorar *scripts* de teste a fim de alcançar o máximo de efetividade. Diante dessas circunstâncias, foi observado que alguns estudos, como (SCHAEFER e DO, 2014),

(HELMANN e MAURER, 2011) e (GEBIZLI e SOZER, 2014), citam que o teste exploratório automatizado é mais eficaz utilizando para complementar a outra técnica de teste, pois a referida abordagem de teste ágil conseguiu detectar diferentes tipos de defeitos que a outra técnica não detectou e vice-versa.

A geração automática de muitos casos de teste com alta cobertura das áreas do sistema tem também fatores negativos, por exemplo, algumas áreas do sistema não foram cobertas, principalmente, quando se tratava de interação com a GUI. Segundo fator negativo condiz ao ônus de custo e tempo para executá-los (automático e manual), sendo que é essencial evitá-los, pois as empresas geralmente utilizam processos ágeis. Além disso, os estudos relataram que os testes gerados automaticamente cobriam áreas do sistema de duas formas, sendo muito abstrato (genérico) ou em propriedades muito específicas, não havendo uma moderação.

Em outras palavras, algumas soluções automáticas utilizam a estratégia de cobrir o máximo de área possível do sistema sob teste a partir do modelo ou eventos de GUI gerando casos de teste mais genéricos, enquanto outros estudos utilizam a estratégia de avaliar minuciosamente os detalhes, sendo considerados todos os caminhos, as transições e as propriedades da GUI (*widgets*, funções, atributos, eventos, etc.), seja visível ou não ao usuário final. Em contrapartida, nessa segunda estratégia exige-se mais tempo para executar vários testes, pois é necessário analisar várias propriedades, seja de código ou de GUI. Esse ônus de tempo é elevado em virtude de ocorrer naturalmente frequentes mudanças de GUI e de requisitos no processo de desenvolvimento havendo também a necessidade de manutenção constante desses testes para evitar erros por desatualização.

Nesse contexto, o trabalho A15 utiliza *crawler* (emula e compara vários algoritmos de rastreamento) para usar como estratégia de exploração em *web pages*, propondo ser uma alternativa eficaz em relação a frequentes mudanças de GUI e também não mais específica ao contexto. Contudo, o autor do referido trabalho relata ter sido difícil automatizar a GUI, pois sempre há muitos *widgets* anônimos (sem identificador ou nome de campo), sendo que muitos não são visíveis aos usuários. Dessa forma, torna-se difícil também ser útil para outras aplicações, pois a solução mencionada seria padronizar o processo de desenvolvimento de *web pages*. Além disso, outra dificuldade foi que sistemas de grandes instituições conseguem detectar *crawler* e automaticamente bloqueiam (HALLÉ *et al.*, 2014).

Em relação à manutenção dos casos de teste, é uma atividade que se torna também frequente, pois geralmente o código torna-se fortemente acoplado ao sistema a ser testado. Mesmo esse acoplamento não sendo interessante, tem sido uma alternativa utilizada na tentativa de minimizar as mudanças constantes na GUI e nos requisitos. Tais mudanças dificultam também garantir uma suíte de teste que possa ser utilizada em teste de regressão, dessa forma gerando um custo financeiro significativo à empresa.

No contexto de estratégias utilizadas para gerar automaticamente casos de teste, observou-se que é oneroso em tempo para processar uma variedade de casos de teste que não seja muito genérico nem muito específico, pois para isso é necessário definir quais propriedades (de GUI ou de código) e perspectiva de teste (visão do testador, desenvolvedor, usuário final) devem ser utilizadas. Em virtude disso, algumas ferramentas oferecem a possibilidade de intervenção humana, seja para editar *scripts* de teste ou para avaliar se o defeito não é falso positivo. Nesse contexto, essa tarefa de definição deve ser bem planejada e realizada com cautela, pois pode afetar diretamente na eficiência de casos de teste gerados automaticamente e também do oráculo de teste.

Destaca-se que a intervenção humana, principalmente, no refinamento do modelo do sistema ou *scripts* de teste e avaliar os resultados está condicionada à necessidade de

serem executadas por pessoas qualificadas, isto é, colaboradores que possuem vasta experiência em teste, conhecimento de domínio (regras de negócio) e conhecimento do próprio sistema a ser testado. Dessa forma, observa-se que há um *tradeoff*, pois as empresas atualmente buscam utilizar ferramenta de software que possa ser mais eficiente possível, ocasionando o mínimo de ônus possível em tempo, custo e esforço. Além disso, há também a necessidade de intervenção humana em algumas situações por condições da própria capacidade tecnológica oferecida atualmente para tratar as referidas lacunas existentes na aplicabilidade de teste exploratório automatizado. Na Tabela 3 é possível observar todas as lacunas encontradas.

Tabela 3 – Mapeamento dos dados referente a questão principal desta pesquisa.

QP – Quais os pontos fortes e lacunas sobre a aplicabilidade do Teste Exploratório automatizado?		
ID	Descrição das Lacunas	Estudos
1	Casos de teste gerados são muito abrangentes ou muito específicos;	A3, A4, A5, A6, A8, A9, A11, A13, A18, A19
2	Difícil definir quais propriedades devem ser analisadas pelo oráculo para serem eficientes na definição se é defeito ou não;	A1, A3, A4, A5, A8, A14, A18, A19, A20
3	Automatização de modelos do sistema tem encontrado dificuldades na própria tecnologia;	A1, A3, A4, A5, A6, A8, A11, A12, A14, A15, A16, A18, A19, A20
4	Dificuldade na transferência de aprendizado do testador para a ferramenta automatizada e vice-versa;	A3, A6, A8, A11, A14, A15, A18, A20
5	Difícil identificar o impacto da personalidade e do conhecimento do testador quando são necessárias atividades manuais em conjunto com atividades automatizadas;	A2, A3, A7, A9, A10, A13, A18
6	A construção de um modelo do sistema depende muito do especialista (experiência, conhecimento do sistema, conhecimento de domínio);	A2, A3, A6, A7, A10, A11, A13, A18, A19
7	Muitos resultados são incoerentes ou sem sentido com os requisitos do sistema	A1, A3, A5, A14, A18
8	Difícil garantir uma suíte de teste que possa ser reutilizada em teste de regressão em virtude da ocorrência de mudanças no sistema sob teste (especificações e propriedades específicas de código);	A1, A3, A18
9	Muito custoso, em esforço e financeiramente, realizar a manutenção de casos de teste ou do modelo que sofreu impacto das mudanças ocorridas no sistema sob teste (especificações ou propriedades específicas de código);	A2, A3, A5, A6, A11, A13, A14, A18
10	Muito custoso, em tempo e financeiramente, executar todos os casos de teste gerados automaticamente;	A3, A6, A10, A11, A13, A14, A18
11	Pouca precisão da classificação, robustez e confiabilidade	A4, A5, A14, A16
12	As ferramentas automatizadas não exploram diferentes perspectivas (visão de desenvolvedor, visão de testador e de usuário)	A5, A18, A19

QP – Quais os pontos fortes e lacunas sobre a aplicabilidade do Teste Exploratório automatizado?		
ID	Descrição das Lacunas	Estudos
13	A ferramenta automatizada não tem uma forma de medir o tempo ocioso do testador na exploração quando há atividades manuais para melhorar as atividades automatizadas;	A17, A19
14	A ferramenta automatizada não classifica severidade nem tipo de defeitos;	A6, A11, A14
15	Algumas áreas do sistema não são cobertas pela geração automática de casos de teste;	A1, A2, A3, A6, A7, A11, A16, A18, A19

Fonte: Elaboração própria (2020).

Apesar de alguns trabalhos relatarem que relativamente as atividades manuais são fundamentais para melhorar a efetividade dos testes e da validação dos resultados, contudo ainda tem sido dificultoso dimensionar qual o impacto dos fatores subjetivos aos seres humanos, por exemplo, a personalidade e o conhecimento do testador. Além disso, é árduo definir uma abordagem eficiente e segura na transferência de conhecimento do ser humano para uma ferramenta de software e vice-versa. Em virtude disso, alguns estudos visam propor ferramentas de software para auxiliar na efetividade do processo de exploração, onde se observou uma delimitação em identificar áreas pouca exploradas para direcionar o teste exploratório manual. Em outras palavras, a ferramenta identifica "o que" tem sido mais ou menos explorado para otimizar o processo de teste, contudo não aborda "como" explorar, isto é, os estudos não mencionam quais procedimentos podem ser utilizados para explorar, se são sistemáticos ou não para efetividade do teste.

Nesse contexto, apenas o trabalho A19 (BURES; FRAJTAK e AHMED, 2018) menciona ter implementado na ferramenta a técnica de classe de equivalência para direcionar a exploração conforme as áreas menos exploradas. Ressalta-se que esses dois aspectos supracitados ("O que" e "Como") tornam-se triviais para os testadores alcançarem a maior eficiência do teste, pois tendo o conhecimento de quais áreas do sistema precisam ser mais exploradas e como proceder com *expertise*, proporciona uma exploração sistematizada obtendo conhecimento tácito mais estruturado, com isso permite proceder com mais efetividade na exploração (BACH, 2015).

Observou-se também que as ferramentas não classificavam automaticamente o nível de severidade dos defeitos, essa referida atividade tem sido realizada por um especialista (testador), pois houve relatos que a ferramenta precisava melhorar a classificação, robustez e confiabilidade. Apesar de não haver registro nos estudos que a ferramenta também não definia qual o tipo de defeito foi detectado, quase sempre se tratava de defeitos funcionais ou de interface de usuário, pois focaram bastante na estratégia de realizar o teste na perspectiva do usuário final (teste de aceitação) interagindo em nível de GUI ou a partir de API.

Por fim, conforme já mencionado em determinados estudos, utiliza-se ferramenta de *Capture & Replay* para capturar as ações realizadas nas atividades de testes manuais, por conseguinte serem analisadas pela ferramenta proposta para então serem gerados automaticamente os testes ou apresentar as áreas menos exploradas. Nesse processo de importação de dados, para a realização de análise de *logs* a ferramenta em questão não conseguiu identificar qual a quantidade de tempo que o testador ficou ocioso, por exemplo, não identificou quanto o testador permanecia períodos de tempo apenas observando

atentamente um carregamento de página, comportamento após transação de banco de dados, etc. Esse problema influencia também na precisão da análise de resultados, pois condiciona a obtenção de muitos resultados imprecisos ou sem coerência com os requisitos. Portanto, todos esses fatores são relevantes para uma ferramenta que objetiva apoiar o processo de teste exploratório.

Na Figura 1 (referente à QS1), observa-se que a maioria dos trabalhos centraliza-se na proposta de construção de ferramentas ou *frameworks* que sejam úteis em teste de sistema, aceitação e regressão. Essa convergência da aplicação do teste exploratório na fase final do ciclo de desenvolvimento pode ser justificada pelo fato de focarem em automatizar à perspectiva do usuário final a partir da interação com o sistema através da GUI ou API para verificar em sua maioria requisitos funcionais. Diante desses dados, menciona-se a importância de uma abordagem mais transversal atingindo todos os contextos de teste para que o uso do teste exploratório seja mais equilibrado. Tendo em vista que o teste exploratório é flexível, com isso considera-se bastante útil também como estratégia inicial no desenvolvimento do sistema para prevenir a detecção de defeitos mais severos somente no final, amenizando até mesmo a quantidade de retrabalho.

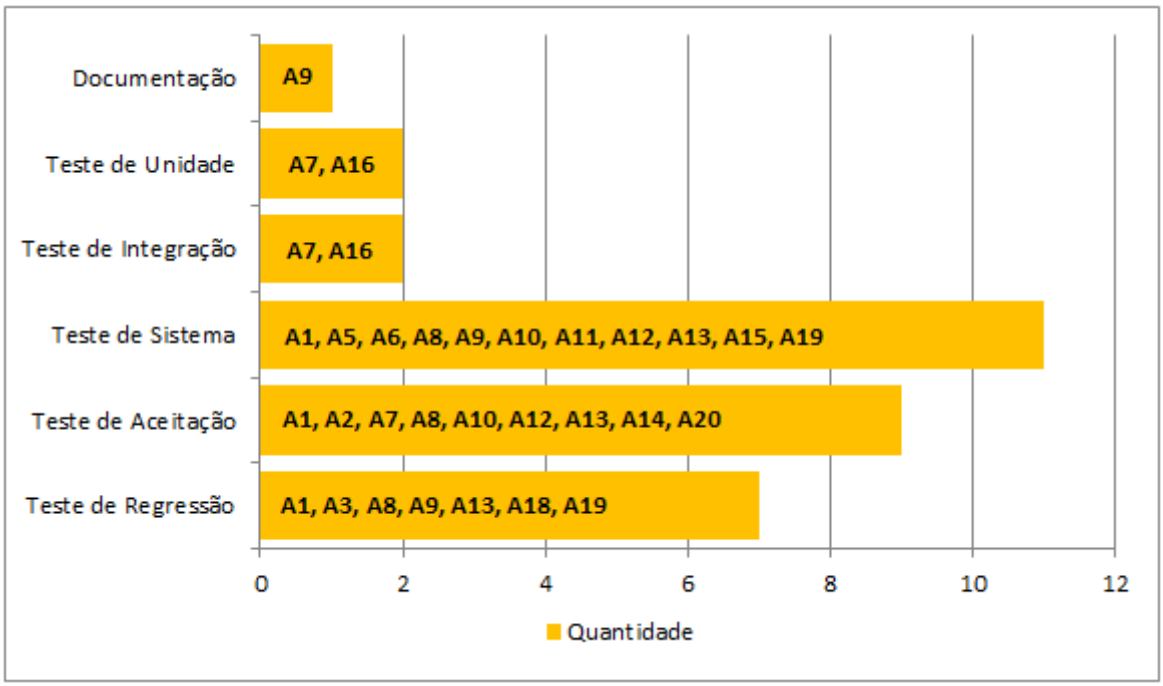

Figura 1 – Contextos de teste envolvidos no teste exploratório automatizado.
Fonte: Elaboração própria (2020).

Na Figura 2 (referente à QS2), observa-se que o teste exploratório automatizado é mais frequentemente utilizado em fase de execução do projeto de *software*, justamente quando há código funcional aplicando testes dinâmicos. Alguns estudos apresentam uma proposta que envolve a fase de projeto e gerenciamento, por exemplo, realiza teste em protótipos, utiliza o diagrama de caso de uso, especificação de requisitos e outros artefatos para definir estratégias de aplicação do teste exploratório. Esses artefatos geralmente são utilizados tanto para apoiar um processo de teste manual quanto à ferramenta ter como base para gerar automaticamente os casos de teste, bem como refinar o modelo do sistema.

Destaca-se que nenhum trabalho evidenciou uma proposta de haver uma ferramenta que envolvesse a realização de planejamento, métricas e medição de teste exploratório.

Esses fatores são essenciais tanto ao processo de teste quanto para um projeto de desenvolvimento de *software* como um todo, pois sem planejamento o projeto de software contém muitas incertezas sobre o que e como fazer, estando sujeito a altos riscos de insucesso. Nesse contexto, a medição e as métricas também são fundamentais, pois concedem suporte às atividades de planejamento objetivando minimizar as incertezas e evitando a alta probabilidade de insucesso. Esses fatores supracitados são triviais também na aplicação de teste exploratório pelo fato de ser uma abordagem de teste ágil, sendo assim requer a elaboração de ótimos planos e estratégias de aplicação de teste, sem sacrificar a coleta de dados necessária à compreensão de todo o processo aplicado (incluindo tempo, custo, esforço, recursos, etc.) e para projeções de projetos futuros.

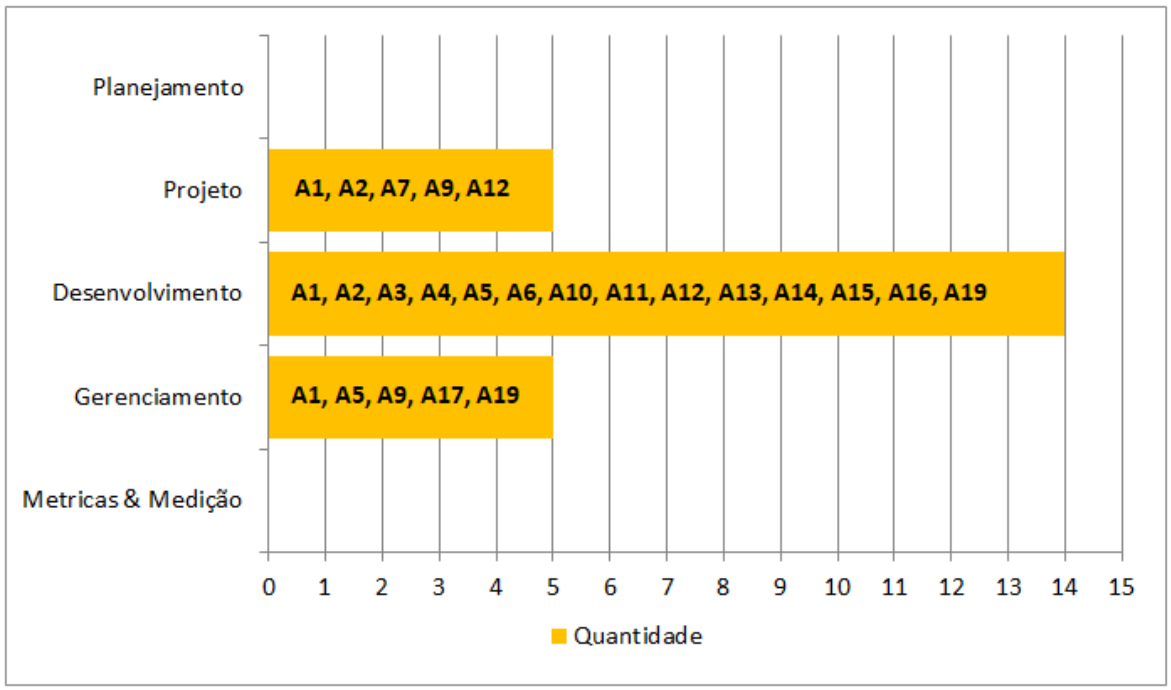

Figura 2 – As fases de desenvolvimento de projeto de software envolvidas no teste exploratório automatizado.
Fonte: Elaboração própria (2020).

Observa-se na Figura 3 (referente à QS3) que os estudos relataram o uso da estratégia de automatizar o teste exploratório em nível de GUI, onde relatam que o foco principal foi realizar testes funcionais para verificar a aderência aos requisitos funcionais. Em relação a utilizar teste exploratório em nível de código não houve registo, apenas os trabalhos A7 (HELLMANN; HOSSEINI-KHAYAT e MAURER, 2010) e o A16 (KHUN, 2013) que utilizaram os registros de teste exploratório em protótipos de baixa fidelidade para gerar teste de unidade, beneficiando posteriores testes na GUI ao testar elementos antes de serem implementados. Além disso, os trabalhos relacionados a teste funcional são apenas para destacar que vários estudos envolveram o referido objetivo de teste quando aplicado teste na GUI. Destaca-se também que houve cinco trabalhos que não envolveram o uso de GUI, os quais são: o trabalho A4 (ROPER, 2019), cujo foco é na realização de análises automáticas dos resultados gerados a partir da exploração manual; o trabalho A9 (LIMA *et al.*, 2018), tendo seu foco na construção de uma ferramenta que realiza uma análise de relatórios de defeitos e *Release Notes* da atual versão do software em questão

para mostrar potenciais áreas com alta probabilidade de detecção de defeitos à equipe de teste exploratório.

Além desses, há os trabalhos A7 e A16, conforme já mencionados, os quais propõem uma ferramenta para extrair casos de teste de unidade com base nas entradas e chamadas de funções a partir de registros de *log* do teste exploratório manual em protótipos de baixa fidelidade. Em especial, no trabalho A16 a ferramenta analisa o *log* e gera automaticamente casos de teste de unidade que podem ser ajustáveis manualmente a fim de permitir o refinamento dos testes. Por fim, o trabalho A17 (SVIRIDOVA; STAKHOVA e MARIKUTSA, 2013) apresenta uma ferramenta apenas para gerenciamento do processo de teste, onde a ideia é centralizar dados relacionados ao processo de teste exploratório, desde o lançamento de um defeito até a sua correção, bem como manter um registro de fácil acesso por quem necessitar consultar.

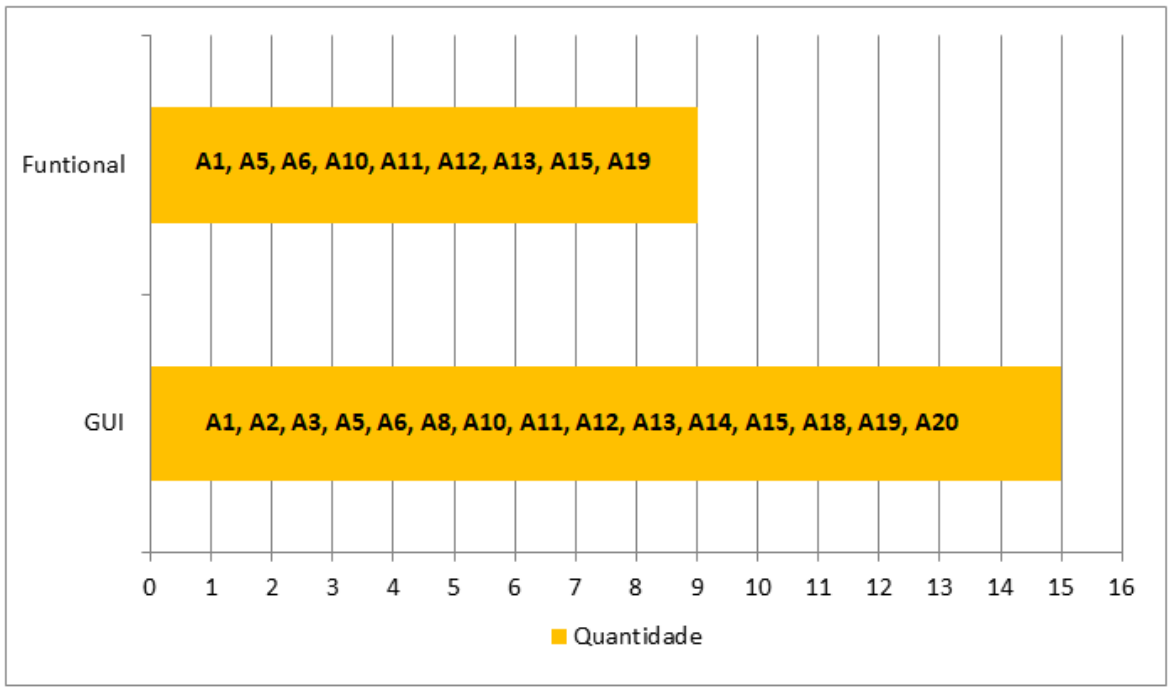

Figura 3 – Estudos que envolveram automatização de teste ema nível de GUI e Funcional.
Fonte: Elaboração própria (2020).

Relacionado à estratégia de automatização de teste exploratório, destaca-se que a maioria dos estudos relataram sobre o uso de uma ferramenta que seja semi-automatizada (vide Figura 4, referente à QS3), pois algumas atividades ainda são muito onerosas em tempo para executar automaticamente. Outro fato é a dificuldade em garantir que uma ferramenta totalmente automatizada seja confiável sem comprometer a eficácia e a eficiência dessa abordagem de teste ágil. A maioria dos entraves já mencionados ainda são difíceis de resolver quando envolve automatização de teste em nível de GUI.

Apesar das atuais técnicas de desenvolvimento já evoluírem, os fundamentos propostos por Mike Cohn (COHN, 2009) ainda prevalecem, o qual relata que as dificuldades de implementar testes automatizados crescem no sentido da base ao topo da pirâmide (vide Figura 5). Com isso, também se torna oneroso em esforço manter os casos de teste sempre aderentes às modificações e exigindo sempre colaboradores com vasta experiência, conhecimento do sistema e das tecnologias utilizadas. Enfatiza-se que a importância de colaboradores qualificados faz-se necessário, pois as tecnologias utilizadas

para automatizar o teste exploratório são focadas no uso de inteligência artificial, aprendizado de máquina, algoritmos genéticos, etc. (AHO e VOS, 2018). Destaca-se em especial o trabalho A8 por não propor uma ferramenta para minimizar as lacunas de teste exploratório automatizado, mas por apresentar vários desafios no processo de automatização total de teste em nível GUI, descrevendo em detalhes tais desafios enfrentados quando se utiliza teste exploratório, conforme as evidências na literatura.

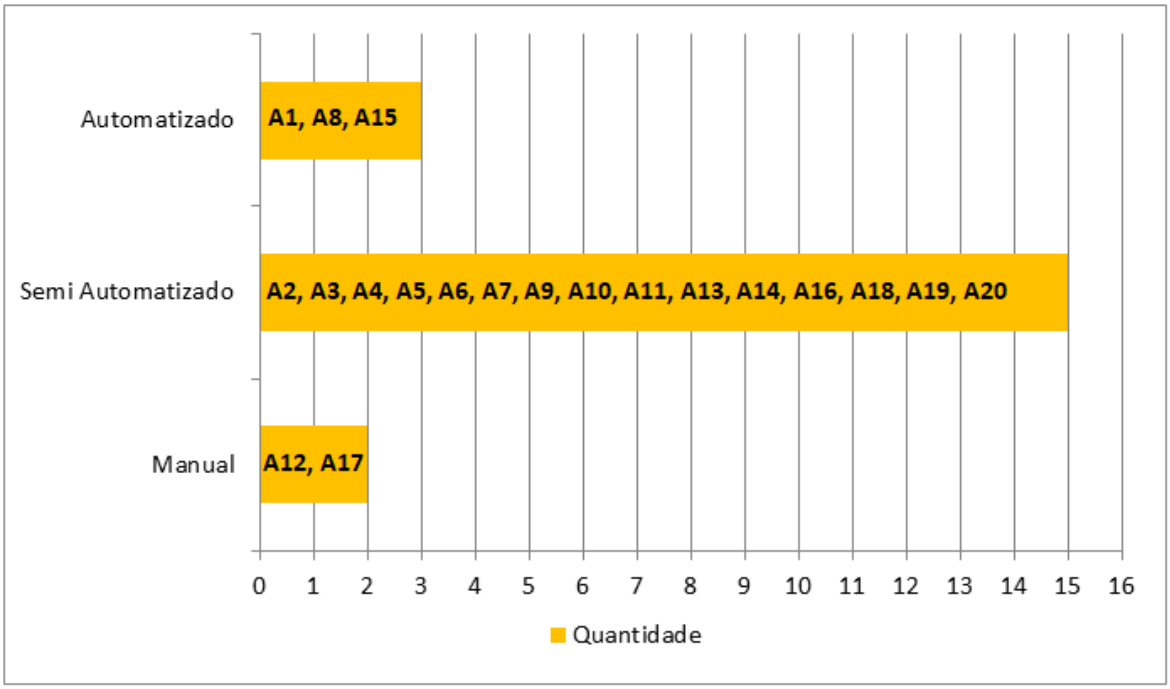

Figura 4 – Estudos que usam processo de teste automatiza, semi automatizado e manual.
Fonte: Elaboração própria (2020).

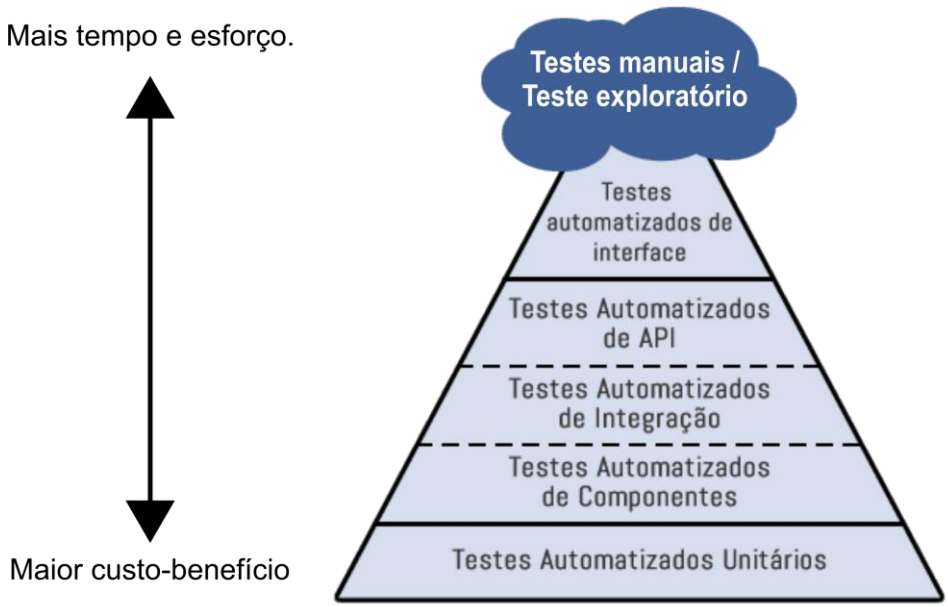

Figura 5 – Pirâmide para Automação de teste.
Fonte: Adaptado de Cohn (2009).

Em relação à QS4, que diz respeito às técnicas utilizadas para automação de teste exploratório, foi evidenciado que a maioria tem aplicado autômatos ou máquinas de estados justamente para construir o modelo do sistema (vide Figura 6). Em relação à exploração (percorrer os caminhos), tendo como base o modelo do sistema, houve um equilíbrio entre o uso de quatro técnicas, as quais foram: a) processamento de arquivo XML em virtude da ferramenta em questão ser voltado a sistema web; b) inteligência artificial aplicando a estratégia de aprendizado de máquina, processamento de linguagem natural, busca em grafo e rede neural artificial; c) *script* convencional; e d) algoritmo genético. Dentre as técnicas destaca-se que alguns trabalhos envolveram mais de uma subárea ds inteligência artificial.

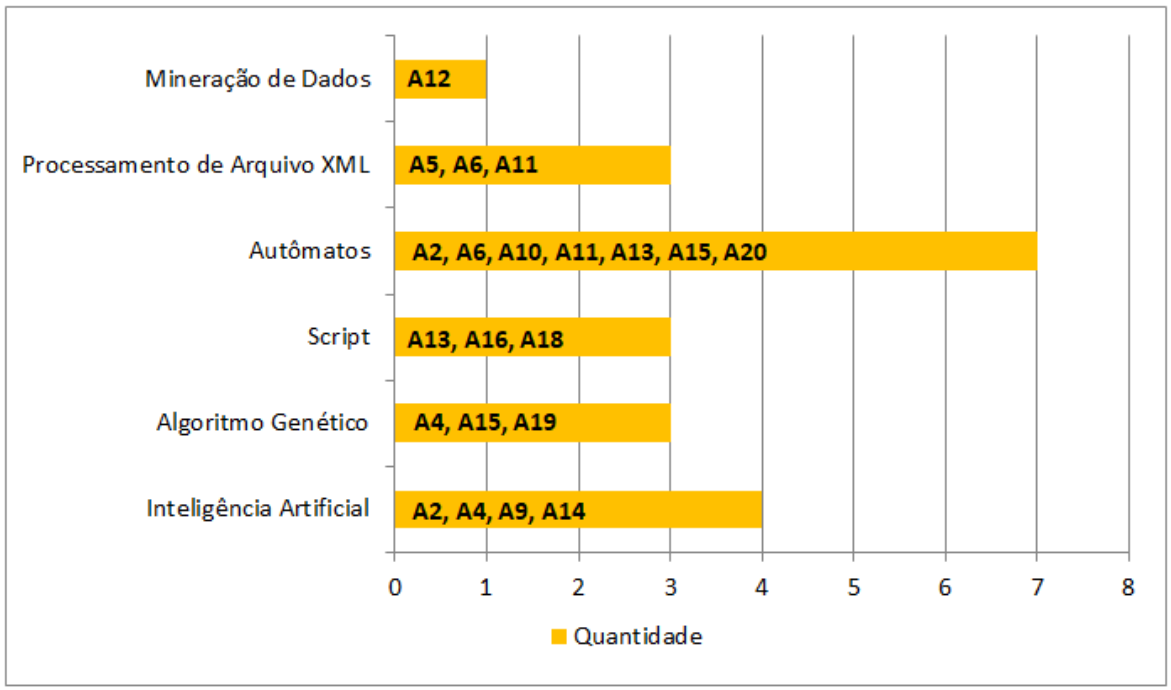

Figura 6 – Estudos que usam processo de teste automatiza, semi automatizado e manual.
Fonte: Elaboração própria (2020).

Na Figura 7 é apresentada a quantidade de trabalhos que foram aplicados no cenário industrial e acadêmico, onde é perceptível um equilíbrio de estudos entre tais cenários. Essa situação pode ter ocorrido, pois já tem sido apresentados em vários trabalhos os benefícios do teste exploratório, porém por ser uma abordagem de teste ágil, torna-se uma necessidade compreender quais benefícios e lacunas dessa abordagem automatizada. Dessa forma, isso se torna importante tanto na indústria a qual tem investido bastante em atividades de teste automáticas, sendo até mesmo pela própria imposição do mercado para entregar o produto ou serviço em curto tempo, quanto no contexto acadêmico que busca compreender minuciosamente os fatores que podem influenciar na efetividade do teste exploratório automatizado.

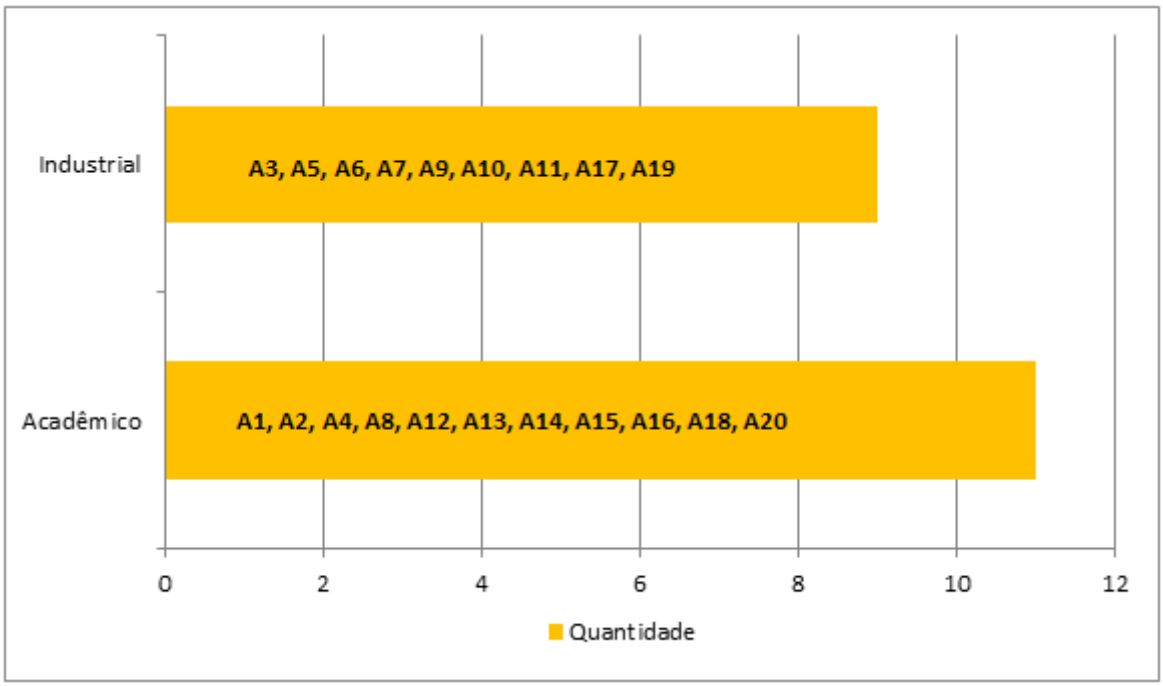

Figura 7 – Relação entre os contextos de aplicação do teste exploratório automatizado.
Fonte: Elaboração própria (2020).

4.1 Questões Bibliométricas

Nesta seção são apresentados os gráficos gerados a partir dos dados bibliométricos, ou seja, será apresentada uma análise das informações sobre os trabalhos primários que foram selecionados. Na Figura 8 é apresentada a quantidade de estudos por instituição, onde é observado que não houve uma instituição com uma quantidade tão expressiva, mas destacam-se a Universidade Técnica Tcheca, Universidade de Calgary do Canadá e do Centro de Competência de Software Hagenberg GmbH da Áustria, as quais obtiveram dois trabalhos selecionados. Nesse contexto, a Figura 9 apresenta a quantidade de estudos por país, onde se observa uma expressividade maior no Canadá, com 4 trabalhos, e no Estados Unidos, tendo 3 trabalhos.

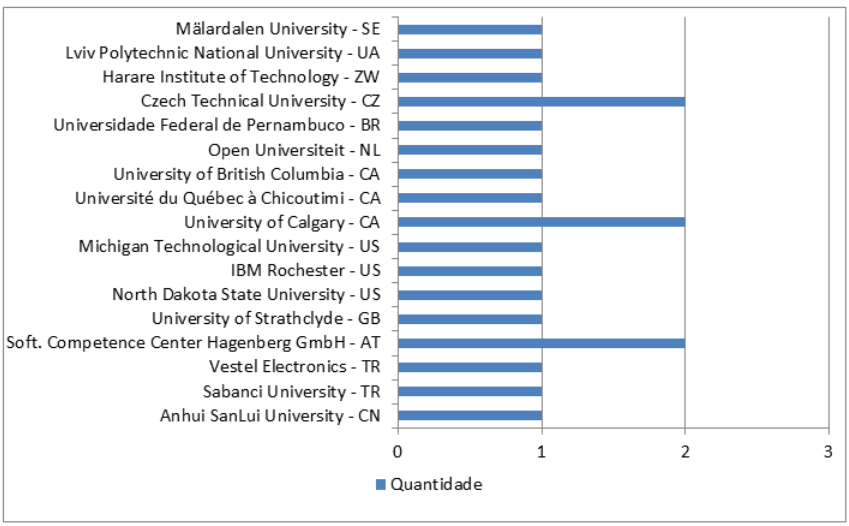

Figura 8 – Gráfico dos estudos selecionados por instituição.
Fonte: Elaboração própria (2020).

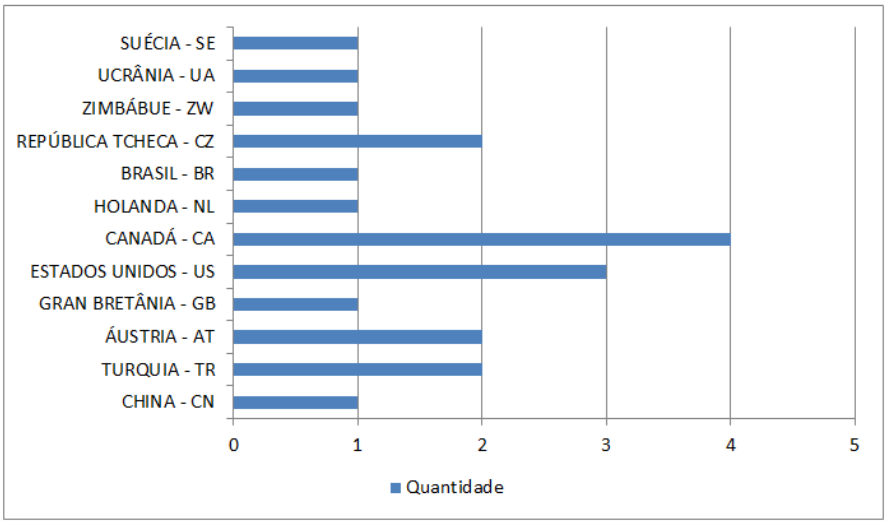

Figura 9 – Gráfico dos estudos selecionados por país.
Fonte: Elaboração própria (2020).

Na Figura 10 é apresentada a quantidade de estudos por ano de publicação, sendo notável que os estudos sobre teste exploratório ainda são considerados recentes mesmo já tendo registro desde 2008, onde se intensificou a partir de 2013. Apesar de haver poucos trabalhos em alguns anos intercalados, nota-se uma tendência de crescimento na quantidade de estudos a cada ano, sendo perceptível a partir de 2016. Portanto, é importante que prossiga esta tendência atualmente até chegar ao ponto de exaurir o máximo a identificação de fatores positivos e estratégias que minimizam as lacunas a fim de garantir uma aplicabilidade de teste exploratório automatizado eficaz e eficiente. Apesar de não constar estudos em 2020, ressalta-se que foram considerados apenas os quatro primeiros meses do referido ano.

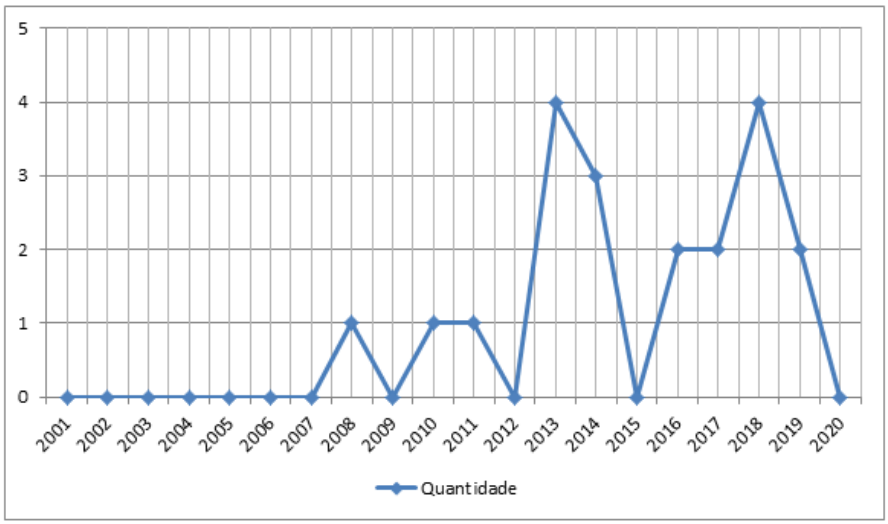

Figura 10 – Relação entre os contextos de aplicação do teste exploratório automatizado.
Fonte: Elaboração própria (2020).

Além disso, destaca-se que dentre 20 trabalhos selecionados, 19 foram publicados em conferências e apenas 1 em periódico (revista). Essa situação pode ter ocorrido pelo

fato que nas conferências exige-se menos tempo de revisão, a partir da submissão até a publicação, e também por haver inúmeras conferências, se comparada com a quantidade de periódicos. Diante disso, é importante destacar a necessidade de haver mais trabalhos em revistas para possibilitar a compreensão mais minuciosa das abordagens dos estudos.

5. AMEAÇAS À VALIDADE

O viés de seleção dos estudos primários é uma das principais ameaças que estão inerentes a uma revisão da literatura. Para fins de minimizar tal problema, foi realizado um teste-piloto com uma amostra de estudos primários para verificar a aderência da *string* de busca com o assunto a ser pesquisado. Além disso, todas as etapas do processo da revisão foram realizadas por apenas um pesquisador, porém quaisquer dúvidas e/ou situação que comprometesse o desenvolvimento deste presente estudo a estratégia foi realizar reuniões com o especialista, o qual se manteve disponível acompanhando a todo instante o trabalho, justamente para que se obtivesse um consenso das ações que foram realizadas.

Outra dificuldade foi a categorização dos processos em automatizado e semi-automatizado, pois grande parte dos trabalhos não mencionam uma maneira formal de aplicação, mas utilizaram algumas estratégias específicas a depender do contexto. Diante dessas circunstâncias, classificou-se como automatizado o processo descrito em que era totalmente realizado pela ferramenta de software proposta e semi-automatizado foram classificados os processos descritos que em algum momento havia uma intervenção humana.

6. CONCLUSÕES

Neste presente estudo tem sido conduzida uma revisão da literatura visando identificar fatores positivos e lacunas da aplicabilidade do teste exploratório automatizado, bem como outros fatores que podem influenciar na aplicação desta abordagem. Tais aspectos investigados referem-se às fases de teste (processo de Verificação e Validação) que têm sido aplicadas, quais as fases de projeto de desenvolvimento de software, quais as abordagens, as técnicas e os cenários mais utilizados. Considera-se que os resultados obtidos atenderam satisfatoriamente as questões desta pesquisa, com as quais foi possível identificar vários benefícios e lacunas que confirmam a necessidade de mais pesquisas sobre a efetividade do teste exploratório, bem como explorar alternativas para minimizar algumas lacunas que estão sendo difíceis de resolver com a tecnologia atual.

De acordo com os resultados, foi possível perceber inúmeros fatores positivos e de grande relevância na aplicabilidade do teste exploratório automatizado, por exemplo, redução de tempo, custo, esforço, bem como na melhoria das estratégias de aplicação do teste exploratório. Além disso, observa-se também várias lacunas que são onerosas em tempo e custo para serem resolvidas ou que ainda encontram dificuldades na própria tecnologia para minimizá-los. Alguns exemplos são: como transmitir o conhecimento do ser humano para uma ferramenta de software? como executar uma quantidade imensa de teste se o tempo é curto? como minimizar o esforço de manutenção de código em virtude da existência de mudanças frequentes de GUI e de requisitos? como definir as propriedades que podem ser relevantes para garantir que o oráculo de teste consiga ser relativamente robusto (eficiente) ao avaliar uma imensidão de resultados gerados e que tal ferramenta de software seja confiável na definição do que seria defeito ou não classificando com eficácia quanto ao seu tipo e grau de severidade. De modo geral, os desafios enfrentados podem ser classificados em dois contextos: a) dificuldade em implementar uma ferramenta que tenha alta eficiência e seja confiável em seus resultados;

b) dificuldades ocasionadas em decorrência do próprio processo de teste, considerando principalmente quando os teste são direcionados à GUI.

Dentre as inúmeras lacunas observadas, destaca-se o caso do trabalho A19, o qual implementou a técnica de teste de classes de equivalência como estratégia sistemática para proceder na exploração automática. Esse trabalho é um exemplo de estudo focando no "o que" e "como" agir, pois visou a identificação de potencias áreas de teste e seguindo procedimentos sistemáticos para detectar defeitos. Ressalta-se que o A19 é um exemplo de estudo que tenta contornar as dificuldades descritas no contexto do item "a" anteriormente citado.

Em relação aos demais trabalhos, basicamente mencionam que a ferramenta de software proposta explora seguindo todos os caminhos ou nós de acordo com o modelo do sistema previamente elaborado. Contudo, esses trabalhos relataram que foi difícil definir uma condição de parada para uma sequência de teste. Uma estratégia para minimizar tal problema foi estabelecer um tempo fixo, porém em alguns casos ocorreu do teste terminar inesperadamente antes do tempo pré-determinado, geralmente, em virtude de erros críticos relacionados à necessidade de parâmetros para prosseguir a exploração. Essa é uma lacuna complexa de automatizar, pois envolve outras lacunas, por exemplo, a transferência de conhecimento, a implementação de regras e procedimentos sistemáticos de exploração (classe de equivalência, teste de fronteira, etc.), a análise e classificação confiável dos defeitos, a robustez e outros entraves que limitam a capacidade tecnológica atualmente para garantir que a ferramenta de software seja eficaz e eficiente sem qualquer intervenção humana.

Diante disso, observa-se várias oportunidades de trabalhos futuros, nesse sentido surge a proposta de realizar um estudo sobre uma abordagem de ensino que foque no aprendizado de técnicas sistemáticas de exploração para que o testador consiga desenvolver melhor o conhecimento tácito, possibilitando-o aplicar de forma mais eficiente a exploração seja em qualquer sistema a ser testado, bem como envolva projeto e execução, mas também atividades de planejamento, gerenciamento e estimativas (métricas e medições) do processo de teste exploratório. Conforme supracitado, essa abordagem de ensino visa atender todas as áreas de projeto de teste e também procedimentos sistemáticos para a exploração, sendo que todas essas atividade envolvidas devem ser estruturadas, estando aderentes às práticas da área de programa de treinamento organizacional do TMMi (*Test Maturity Model Integration*). Além disso, destaca-se que o conteúdo a ser ministrado sobre as técnicas sistemáticas e fundamentos básicos sobre teste deve ser baseado na apostila *Syllabus* referente ao nível mais básico (*Foundation Level*) do *International Software Testing Qualifications Board* (ISTQB).

A ideia de envolver atividades aderentes a práticas de treinamento organizacional do TMMi objetiva-se obter uma estruturação organizada e formal da abordagem, proporcionando aos participantes uma melhor compreensão das atividades do processo de teste. Em relação ao uso do *Syllabus*, justifica-se por ser o material mais difundido mundialmente e utilizado pelas pessoas que buscam a certificação internacional na área de teste.

Neste sentido, BACH (2015) corrobora ao relatar que a eficiência da abordagem de teste exploratório está fundamentalmente relacionada com as estratégias (técnicas) de exploração e não somente com o conhecimento (experiência) do testador. O referido autor relata também que as ferramentas de software tornam-se importantes de serem utilizadas como recurso de apoio, pois, como tem sido evidenciado, é complexo implementar tais técnicas de teste exploratório em um software, isto é, colocar em bits computáveis o conhecimento tácito. Para exemplificar tal complexidade, BACH (2015) cita que um

algoritmo segue “ordens” caminhos pré-definidos e, mesmo com pontos de inicio e fim, é difícil definir o momento de parada do processo de teste, enquanto um ser humano consegue desviar um caminho ao observar um evento anormal, mesmo se pré-estabelecido mentalmente.

REFERÊNCIAS BIBLIOGRÁFICAS

AHO, P., VOS, T., (2018). “Challenges in Automated Testing through Graphical User Interface”. IEEE International Conference on Software Testing, Verification and Validation Workshops. DOI 10.1109/ICSTW.2018.00038.

BACH, J., (2004). Exploratory Testing. In: The Testing Software engineer, 2nd ed., E. van Veenendaal (Ed.) Den Bosch: UTN Publishers, pp. 253-265.

BACH, J., (2015). Exploratory Testing. SATISFACE, Software Testing for Serious People. Disponível em: <https://www.satisfice.com/exploratory-testing>

BURES, M.; FRAJTAK, K.; AHMED, B., (2018). Tapir: Automation Support of Exploratory Testing Using Model Reconstruction of the System Under Test. IEEE TRANSACTIONS ON RELIABILITY, VOL. 67, NO. 2, JUNE 2018. DOI 10.1109/TR.2018.2799957.

CRUZES, D. S., DYBA, T. (2011): Recommended Steps for Thematic Synthesis in Software Engineering. EPEM 275 – 284.

COHN, M., (2009). “Succeeding with agile: software development using Scrum,” Pearson Education. Addison-Wesley Professional; 1 edition.

GAROUSI, V.; MÄNTYLÄ, M., (2016). When and what to automate in software testing? A multi-vocal literature review. Information and Software Technology, 76, 92-117. https://doi.org/10.1016/j.infsof.2016.04.015

GEBIZLI, C.; SOZER, H., (2014). Improving Models for Model-based Testing based on Exploratory Testing. IEEE 38th Annual International Computers, Software and Applications Conference Workshops. DOI 10.1109/COMPSACW.2014.110.

GREGORY J, CRISPIN L., (2014). Chapter 23-testing and DevOps: In more agile testing: Learning journeys for the whole team. Addison-Wesley Professional; 2014.

HALLÉ, S.; BRETON, G.; MARONNAUD, F.; MASSÉ, A.; GABOURY, S., (2014). Exhaustive Exploration of Ajax Web Applications With Selective Jumping. IEEE International Conference on Software Testing, Verification, and Validation Workshops. DOI 10.1109/ICSTW.2014.26.

HELMANN, T.; MAURER, F., (2011). Rule-Based Exploratory Testing of Graphical User Interfaces. 2011 Agile Conference. DOI 10.1109/AGILE.2011.23.

HELLMANN, T.; HOSSEINI-KHAYAT, A.; MAURER, F., (2010). Supporting Test-Driven Development of Graphical User Interfaces Using Agile

Interaction Design. Third International Conference on Software Testing, Verification, and Validation Workshops. DOI 10.1109/ICSTW.2010.35.

KANER, C., (2008). A Tutorial in Exploratory Testing. QUEPT 2008. Disponível em: http://www.kaner.com/pdfs/QAIExploring.pdf.

KHUN, A., (2013). On Extracting Unit Tests from Interactive Live Programming Sessions. ICSE 2013, San Francisco, CA, USA New Ideas and Emerging Results. DOI 978-1-4673-3076-3/13/$31.00@IEEE.

KLAMMER, C.; RAMLER, R., (2017). A Journey from Manual Testing to Automated Test Generation in an Industry Project. International Conference on Software Quality, Reliability and Security (Companion Volume). DOI 10.1109/QRS-C.2017.108

LIMA, C.; SANTOS, I.; BARROS, F.; MOTA, A., (2018). SPt: a text mining process to extract relevant areas from SW documents to exploratory tests. 7th Brazilian Conference on Intelligent Systems. DOI 10.1109/BRACIS.2018.00051.

PFAHL, D. et al., (2014). How is Exploratory Testing Used?: A state of the Practice Survey. EPEM'14, September 18-19, Torino, Italy. Copyright 2014 ACM 978-1-4503-2774-9/14/09.

ROPER, M., (2019). Using Machine Learning to Classify Test Outcomes. IEEE International Conference on Artificial Intelligence Testing (AITest). DOI 10.1109/AITest.2019.00009.

SWEBOK, (2014). Guide to the Software Engineering Body of Knowledge. Ed.V3.0. Disponível em: <http://www.computer.org/portal/web/swebok/>.

SCHAEFER, C.; DO, H., (2014). Model-Based Exploratory Testing: A Controlled Experiment. IEEE International Conference on Software Testing, Verification, and Validation Workshops. DOI 10.1109/ICSTW.2014.31

SVIRIDOVA, T.; STAKHOVA, D.; MARIKUTSA, U., (2013). Exploratory Testing: Management Solution. CADSM 2013, 19-23 February, 2013, Polyana-Svalyava (Zakarpattya), UKRAINE.