NATHALIA REZENDE SILVA

UM MAPEAMENTO SISTEMÁTICO DA LITERATURA SOBRE FERRAMENTAS PARA VALIDAÇÃO DE DADOS EM TESTES DE REGRAS DE NEGÓCIO

LAVRAS-MG

2019

NATHALIA REZENDE SILVA

UM MAPEAMENTO SISTEMÁTICO DA LITERATURA SOBRE FERRAMENTAS PARA VALIDAÇÃO DE DADOS EM TESTES DE REGRAS DE NEGÓCIO

Artigo de Graduação apresentado ao Departamento de Ciência da Computação para a obtenção do título de Bacharel em Sistemas de Informação.

Prof. Dr. Paulo Afonso Parreira Júnior
Orientador

LAVRAS-MG

2019

NATHALIA REZENDE SILVA

UM MAPEAMENTO SISTEMÁTICO DA LITERATURA SOBRE FERRAMENTAS PARA VALIDAÇÃO DE DADOS EM TESTES DE REGRAS DE NEGÓCIO

Artigo de Graduação apresentado ao Departamento de Ciência da Computação para a obtenção do título de Bacharel em Sistemas de Informação.

APROVADA em 12 de Junho de 2019.

Prof. Dr. Paulo Afonso Parreira Júnior UFLA
Prof. Dr. Dilson Lucas Pereira UFLA
Prof. Dra. Renata Teles Moreira UFLA

Prof. Dr. Paulo Afonso Parreira Júnior
Orientador

LAVRAS-MG

2019

Aos meus pais que, com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa de minha vida. E às minhas irmãs que sempre me inspiraram e me impulsionaram a ir além.
DEDICO

AGRADECIMENTOS

Aos meus pais, Delfina e Ronaldo, por fazerem dos meus sonhos suas lutas. Obrigada por todo esforço despendido em todos esses anos em prol da minha educação.

Às minhas irmãs, Nayara e Taynara, pela amizade, carinho e cuidado. Obrigada por me darem asas e por me mostrarem que eu posso voar tão longe quanto eu quiser. Vocês são meu guia!

Ao meu namorado, Matheus Henrique, por toda paciência, amor e companheirismo. Obrigada por fazer o teu melhor por nós e por sempre estar ao meu lado, mesmo nos momentos mais difíceis.

À Equals, por me oferecer um ambiente propício para crescer e me desenvolver profissionalmente.

Ao meu orientador, Paulo Afonso, por todo apoio e direcionamento durante a execução deste trabalho.

Aos professores, Dilson e Renata, pelas disciplinas lecionadas e por aceitarem o convite de participar da banca. Renata, obrigada pelos conselhos "*extraclasse"* e histórias compartilhadas.

À todos que de alguma forma contribuíram para a realização deste trabalho e para a concretização desse sonho.

Muito obrigada!

RESUMO

Com o aumento no uso de Sistemas de Informação (SI), cada vez mais complexos e robustos, tornou-se proeminente a preocupação com os testes realizados durante seu desenvolvimento. A validação de dados, parte dos testes de regras de negócio, tem como enfoque detectar defeitos e inconsistências nos dados do sistema e se trata de um processo de suma importância para a empresa mantenedora do SI, visto que dados incorretos podem acarretar em problemas econômicos e até mesmo legais. No entanto, a execução manual é um processo moroso, difícil de replicar e limita-se o tamanho da amostra. Portanto, o objetivo deste trabalho foi realizar um Mapeamento Sistemático da Literatura (MSL), com o propósito de identificar, classificar e catalogar os apoios computacionais existentes na literatura, que permitem a automação do processo de validação de dados. Com a execução do MSL foram aceitos e analisados 6 estudos e, após a leitura na íntegra de cada um deles, foram identificadas 5 possíveis ferramentas para automação da validação de dados. Por fim, foi verificado que diferentes abordagens são utilizadas em cada ferramenta e a escolha sobre cada uma delas depende primordialmente do cenário e características da empresa e projeto em desenvolvimento. Além disso, percebeu-se uma escassez de estudos sobre a temática, tornando-se assim uma área a ser explorada.

Palavras-chave: Testes de regras de negócio. Validação de dados. Ferramentas computacionais.

ABSTRACT

With the increasing use of Information Systems (IS), increasingly complex and robust, the concern with the tests carried out during its development became prominent. Data validation, part of the business rules tests, focuses on detecting defects and inconsistencies in the system data and is a process of paramount importance for the company that maintains the IS, since incorrect data can lead to economic and even legal problems. However, manual execution is a lengthy and difficult to replicate process, of limited sample size. Therefore, the objective of this work was to perform a Systematic Literature Mapping (SLM), with the purpose of identifying, classifying and cataloging existing computational supports in the literature, which allow the automation of the data validation process. With the implementation of the SLM, 6 studies were accepted and analyzed, and after reading in full of each of them, 5 possible tools for automation of data validation were identified. Finally, it was verified that different approaches are used in each tool and the choice of each one depends primarily on the scenario and characteristics of the company and project under development. In addition, there is a shortage of studies on the subject, making it an area to be explored.

Keywords: Business rules tests. Validation of data. Computational tools.

SUMÁRIO

1 INTRODUÇÃO

O mercado de Tecnologia da Informação e Comunicação tem se expandido notavelmente nos últimos anos e, analogamente, percebe-se um aumento significativo no uso de Sistemas de Informação - SI (LAUDON & LAUDON, 2014). Tais sistemas têm se tornado cada vez mais complexos e robustos, visando atender, com qualidade, as necessidades dos clientes. Uma das principais preocupações durante o desenvolvimento de um SI é com os testes realizados sobre o mesmo, cujo intuito é identificar defeitos e aprimorar, portanto, sua qualidade.

Um processo de testes bem definido e eficiente procura garantir que o código desenvolvido esteja de acordo com o projeto e que faça apenas o que é intencional, ou seja, o que é definido para ser feito (MYERS, 2004). Para a realização desse processo, diversos tipos de testes podem ser aplicados, dependendo da estratégia adotada e levando em consideração fatores como o porte do software, seus requisitos, os riscos do negócio com os quais ele está envolvido, sua arquitetura, entre outros.

Um tipo de arquitetura de software bastante utilizada, principalmente em SI projetados para utilização por meio da Internet (também conhecidos como *aplicações web*), é a *arquitetura de três camadas*. Segundo Myers (2004), essa arquitetura se divide em três camadas: *apresentação*, *regras de negócio* e *armazenamento de dados*. A primeira, como o próprio nome sugere, é a camada responsável por apresentar a interface gráfica ao usuário final. A camada de regras de negócios, por sua vez, é responsável pelo gerenciamento das regras de negócio, bem como da lógica da aplicação. Por fim, a última camada, utilizada pela camada de regras de negócios, é responsável por manter (armazenar, recuperar, atualizar e remover) dados a partir da base de dados do SI.

Um dos tipos de teste que podem ser aplicados em um SI é o *teste de regras de negócios*, o qual possui como enfoque detectar erros na lógica de negócios da aplicação, mais especificamente, na aquisição dos dados e no processamento de transações. Por exemplo, uma das atribuições desse teste é validar os dados informados pelo usuário, a fim de detectar defeitos ou inconsistências nos mesmos (MYERS, 2004). Isso é fundamental, visto que se o SI é executado com informações inválidas, problemas significativos poderão ocorrer. Considerando um SI financeiro, por exemplo, se o cliente utilizar esse software com dados

corrompidos e tiver que passar por uma auditoria, consequências legais e econômicas podem ocorrer, tanto para o cliente quanto para a empresa responsável pelo SI. Além disso, esta última passaria a ter uma notoriedade negativa diante dos outros clientes e futuros interessados no produto. Perante a isso, o enfoque deste trabalho é principalmente na validação de dados como parte do teste de regras de negócio, visto sua importância e, ao mesmo tempo, a escassez de estudos na área. Muito se fala sobre testes unitários e de interface, por exemplo, mas é notável a escassez de estudos sobre teste de regras de negócio e, mais especificamente, sobre a validação de dados, conforme pode ser constatado com a realização deste trabalho.

Com o intuito de tornar o processo de validação de dados no teste de regras de negócio (denominado daqui adiante de "validação de dados", por razão de simplicidade) **mais abrangente e eficiente**[1], faz-se necessário utilizar ferramentas computacionais que possam automatizar parte desse processo. Com isso, o testador terá mais tempo para desenvolver bons casos de testes, ao invés de dedicar esforços em tarefas repetitivas e facilmente automatizáveis. Além disso, com auxílio de ferramentas computacionais, o testador poderá construir os casos de testes apenas uma vez e os utilizar quantas vezes for necessário, aprimorando a replicabilidade do processo de teste. Entretanto, poucos estudos apresentam uma compilação de trabalhos a respeito desse assunto, tornando difícil para membros da academia e da indústria localizar ferramentas que mais atendem às suas necessidades.

Diante disso, o objetivo deste trabalho é realizar um Mapeamento Sistemático da Literatura (MSL), com o propósito de *identificar*, *classificar* e *catalogar* os apoios computacionais existentes na literatura, que possam permitir a automação do processo de validação de dados. Este artigo está organizado como segue: na Seção 2, são abordados, sucintamente, alguns conceitos importantes à respeito de testes de software e sua utilização em sistemas de software para Internet. Na Seção 3, é apresentado o planejamento do MSL e na, Seção 4, os resultados e discussões a partir do mapeamento realizado. Por fim, na Seção 5, apresenta-se as considerações finais deste trabalho e as propostas de trabalhos futuros.

[1] Neste trabalho, entende-se por *abrangência*, a quantidade de casos de teste utilizada, do universo de todos os casos de teste possíveis. Quanto à *eficiência*, esta pode ser definida como a capacidade de se atingir um objetivo, por exemplo a detecção de defeitos, com o menor consumo de recursos possível, como o tempo para execução dos testes.

2 TESTES DE REGRAS DE NEGÓCIO

Cada nível de uma *arquitetura de três camadas* conta com diferentes tipos de testes a serem realizados nela, a fim de garantir a qualidade do software como um todo. Segundo Myers (2004), o teste para a camada de regras de negócios pode avaliar diferentes aspectos, tais como performance, validação de dados e transações. Bastos (2012), por outro lado, define esse tipo de teste como *teste de controle* e explica que entre os controles estão a validação de dados, a integridade de arquivos e outros aspectos do software relacionados à integridade. Ele aborda um cenário mais amplo, onde o teste de controle tem por objetivo garantir, além da corretude e completude dos dados, a autorização de transações, manutenção de informações e a eficiência do processamento dessas informações. Pressman (2011) já utiliza outra terminologia, a saber *teste de base de dados*, que, segundo ele, avalia a exatidão e a integridade dos dados que estão armazenados na base de dados do software com o objetivo de encontrar os defeitos.

Independentemente da terminologia utilizada, é nítida a preocupação com a validação dos dados utilizados no software. Para isso, diferentes apoios computacionais podem ser utilizados. Na Seção 4 deste trabalho são apresentados mais detalhes sobre esse tipo de teste.

3 PLANEJAMENTO DO MAPEAMENTO SISTEMÁTICO DA LITERATURA

Mapeamento Sistemático da Literatura (MSL) é um processo de revisão mais amplo da literatura que analisa estudos primários já existentes, descrevendo sua metodologia e resultados (PETERSEN et al., 2008). Ele é realizado por meio das etapas de planejamento, condução e documentação A etapa de planejamento é dividida entre a definição de questões de pesquisa (o que pretende-se responder a partir dos resultados do MSL), método de busca (onde e como realizar a busca pelos estudos) e critérios de inclusão/exclusão (quais são os parâmetros para se considerar um estudo como relevante ou não). Devido ao enfoque dado ao planejamento e à documentação, o MSL se torna um processo de revisão da literatura sistemático, confiável e reprodutível e, portanto, mais apropriado do que uma revisão realizada de forma *ad-hoc*.

3.1 Questões de Pesquisa

A primeira atividade do planejamento de um MSL visa definir uma ou mais questões de pesquisa, que têm como objetivo direcionar o restante do processo de execução do mapeamento. A seguir, são apresentadas as questões de pesquisa elaboradas para este trabalho, cujo enfoque está na descoberta e classificação de ferramentas para automação do processo de validação de dados.

Q1*: Quais são as ferramentas computacionais existentes na literatura para validação de dados e quais são seus principais atributos, tais como nome, tipo de licença, versão, link para download ou compra, data da última atualização, plataforma de desenvolvimento, entre outros?* **Justificativa**: antes de se tomar a decisão de utilizar uma ou outra ferramenta em um ambiente de trabalho/negócios, é importante conhecer quais ferramentas existem, bem como quais são suas principais características.

Q2*: Quais tipos de integração com outras ferramentas estão disponíveis, tais como ferramentas para rastreamento de bugs, SGBDs, entre outros?* **Justificativa**: uma vez que o processo de teste de um software faz parte de um processo maior de desenvolvimento e manutenção, faz-se necessário conhecer com quais outras tecnologias essas ferramentas se integram. Por exemplo, seria interessante que uma ferramenta de apoio à validação de dados se integrasse com ferramentas de rastreamento de *bugs*.

Q3*: Quais são as formas de elaboração de casos de teste contemplados por essas ferramentas?* **Justificativa**: a maneira requerida pelas ferramentas para confecção de casos de testes pode impactar significativamente na produtividade da equipe, bem como na flexibilidade e aplicabilidade da ferramenta para diversos tipos de domínio. Portanto, faz-se necessário conhecer a catalogar esse tipo de informação.

Q4*: Quais são os principais benefícios e desafios ressaltados nos estudos, quanto à utilização dessas ferramentas?* **Justificativa**: a disseminação adequada desse tipo de conhecimento pode favorecer à tomada de decisões mais precisas, por parte de pesquisadores e profissionais da indústria, quanto à utilização ou não de determinada ferramenta em seu ambiente de trabalho/negócio.

3.2 *String* de busca

Conforme citado na Seção 2 deste trabalho, a terminologia utilizada para descrever o processo de testes da camada de regra de negócio pode variar de autor para autor. Assim sendo, com base nos termos apresentados por Myers (2004), Bastos (2012) e Pressman (2011), as seguintes palavras-chave foram utilizadas para formar a *string* de busca para obtenção de estudos deste MSL: *business layer testing*, *control test* e *database test*. Além dessas, as palavras-chave *tool*, *computational support* e *data validity* foram utilizadas, pois estão diretamente relacionadas com o objetivo deste trabalho. Com base neste conjunto de palavras-chaves (e suas variações), a seguinte *string* de busca foi definida: *("data validity" OR "data validation") AND ("business layer testing" OR "control test" OR "control testing" OR "database test" OR "database testing") AND (tool OR tools OR "computational support")*.

3.3 Fontes de busca

Em geral, MSLs delimitam suas buscas apenas em pesquisas publicadas em eventos e periódicos científicos (denominada daqui adiante de "literatura formal"), ignorando trabalhos produzidos por profissionais da área, mas que são publicados em outros veículos de comunicação, tais como *blogs*, fóruns, *websites*, entre outros. Esse tipo de estudo faz parte do que convencionou-se denominar de "literatura cinza" (GAROUSI; KÜÇÜK, 2018). Ainda que os estudos da literatura cinza não tenham passado por um crivo de revisão mais acurado, como ocorre na literatura formal, muitas vezes eles podem apresentar o "estado da prática", ou seja, indicam o que está sendo utilizado na atualidade por empresas e profissionais de determinada área. Esse fato torna esses tipo de estudo relevante em áreas como a Engenharia de Software, em que a prática se renova constantemente. Neste MSL foi realizada uma busca tanto na literatura formal quanto na cinza, a fim de encontrar o maior número de ferramentas disponíveis e relevantes para a automação do processo de validação de dados.

Quanto à fonte de busca para a literatura formal, foi decidido pelo uso do *Google Scholar*[2], uma vez que é conhecido que os resultados desta fonte inclui outras grandes fontes de estudos científicos, tais como a *ACM Digital Library* e a *IEEE Xplore Digital Library*, por

[2] https://scholar.google.com.br/

exemplo (Neuhaus *et al*., 2006 *apud* Garousi e Küçük, 2018). Para a busca na literatura cinza, optou-se pelo uso do mecanismo de busca regular do *Google*[3]. Há diversos estudos recentes que sugerem o uso do Google para busca de estudos na literatura cinza (Garousi e Küçük, 2018; Godin *et al.,* 2015; Mahood, van Eerd e Irvin, 2014; Adams *et al.,* 2016).

É sabido que os mecanismos de busca do Google, seja regular ou acadêmico, pode apresentar numerosos resultados, a partir de uma busca. Sabe-se ainda que esses resultados são separados em várias páginas e que os mais relevantes são exibidos nas páginas iniciais. Com base nisso, foi adotada a heurística proposta por Garousi e Küçük (2018), que utiliza-se do algoritmo *PageRank* do Google para auxiliar no processo de escolha dos resultados mais relevantes para a pesquisa. O funcionamento da heurística é como segue: a partir da primeira página retornada pelo mecanismo de busca, analisa-se cada estudo exibido na página atual, avançando para a próxima página, enquanto houver estudos que sejam relevantes para a pesquisa em questão. Assim, a busca é interrompida quando determinada página não apresentar resultados relevantes para a pesquisa. A relevância dos estudos é verificada com base nos critérios de inclusão/exclusão propostos para o MSL (Seção 3.4).

3.4 Critérios de inclusão/exclusão

Os critérios de inclusão/exclusão são definidos antes da execução das buscas e visam nortear a aceitação ou não de estudos relevantes para análise no MSL. Para este MSL, foram definidos critérios para estudos advindos da literatura formal e para estudos advindos da literatura cinza.

Quanto à literatura formal, os seguintes *critérios de inclusão* foram definidos: (i) o texto completo do estudo está disponível para acesso via web; (ii) o texto completo do estudo foi escrito no idioma inglês ou português; (iii) o estudo não é uma versão duplicada ou mais antiga de outro estudo; (iv) o estudo diz respeito ao processo de validação de dados e apresenta uma ferramenta computacional de apoio a esse processo. Como *critério de exclusão*, tem-se que qualquer estudo que não satisfaça um ou mais critérios de inclusão deve ser automaticamente excluído.

Quanto à literatura cinza, critérios específicos para esse tipo de fonte foram utilizados para garantir a relevância e a qualidade dos estudos selecionados. Esses critérios,

[3] https://google.com.br

apresentados na Tabela 1, foram propostos nos trabalhos de Garousi, Felderer e Mäntylä (2017) e Tyndall (2010) e têm sido utilizados em mapeamentos sistemáticos recentemente publicados, tais como Garousi e Küçük (2018). Como *critério de exclusão*, seguindo a compreensão dos autores dos critérios da Tabela 1, tem-se que qualquer estudo da literatura cinza que não satisfaça um ou mais critérios dessa tabela deve ser automaticamente excluído da análise do MSL.

Tabela 1 – Critérios de inclusão para estudos da literatura cinza.

Aspecto	Critérios
Autoridade	● É um autor individual associado a uma organização respeitável? ● O autor publicou outros trabalhos no campo?
Precisão	● A fonte tem um objetivo claramente declarado? ● A fonte tem uma metodologia declarada? ● A fonte é apoiada por referências autorizadas e documentadas?
Cobertura	● Existem limites claramente definidos? ● O trabalho cobre uma questão específica? ● O trabalho se refere a uma população em particular?
Objetividade	● O trabalho parece ser equilibrado na apresentação? ● A declaração é uma opinião subjetiva? ● As conclusões são tendenciosas?
Data	● Verifique a bibliografia: o material contemporâneo chave foi incluído? ● Se nenhuma data é dada, mas pode ser verificada de perto, existe uma razão válida para a sua ausência? ● O item tem uma data claramente declarada relacionada ao conteúdo?
Significância	● Enriquece ou acrescenta algo único à pesquisa? ● Fortalece ou refuta a posição atual?

Fonte: Garousi, Felderer e Mäntylä (2017); Tyndall (2010) *apud* Garousi e Küçük (2018).

Tanto na busca formal quanto na informal não foi definido nenhum período de datas, ou seja, independente da data de publicação do estudo, se cumprisse os critérios de inclusão, ele seria aceito para a pesquisa. Já em relação ao idioma, ainda que a *string* de busca definida seja composta apenas por palavras em inglês, entende-se que muitos artigos são publicados com *abstract*, ou seja, com o resumo no idioma inglês, o que tornaria possível encontrar estudos escritos no idioma português considerando apenas seu *abstract*.

3.5 Execução do MSL

Após a definição das questões de pesquisa, da *string* e das fontes de busca, além dos critérios de inclusão/exclusão, passa-se à etapa de execução do MSL, a qual ocorreu de acordo com as atividades descritas nesta seção.

***Atividade 1 - Busca inicial*:** consiste em realizar pesquisas nas fontes de busca definidas no protocolo deste MSL, utilizando-se a *string* de busca. Foram retornados **742** resultados no Google Acadêmico e **39.200** resultados no Google regular. Esses números foram coletados antes da aplicação da heurística e dos critérios de inclusão/exclusão apresentados nas Seções 3.3 e 3.4.

***Atividade 2 - Refinamento da busca inicial*:** a partir dos resultados da atividade 1, aplicou-se a heurística proposta por Garousi e Küçük (2018). Todas as páginas que continham pelo menos um estudo relevante para a pesquisa em questão foram consideradas. Os critérios de inclusão/exclusão foram aplicados a cada estudo, a partir da leitura inicial do seu título e do seu resumo (*abstract*). Após esse refinamento, chegou-se a um número de **7** estudos aceitos no total, sendo 5 estudos obtidos do Google Acadêmico e 2 do Google Regular. No caso do Google Acadêmico, a busca foi interrompida na página **4**, totalizando 40 artigos verificados, enquanto que no Google Regular, a mesma foi interrompida na página **5**, com 50 artigos verificados ao todo. Ou seja, em relação ao número inicial apresentado na atividade 1, teve-se uma queda visível no número total de estudos verificados, por conta da heurística adotada. Já em relação ao total de estudos verificados (90 no total), grande parte foi rejeitada por não apresentarem uma ferramenta para automação de validação de dados ou porquê não apresentavam autoridade e precisão sobre o assunto.

***Atividade 3 - Refinamento final e extração de dados*:** a partir dos estudos aceitos na atividade 2, foi realizada a leitura integral de cada um deles, aplicando-se novamente os critérios de inclusão/exclusão. Nesta etapa, **6** estudos foram aceitos e apenas **1** foi rejeitado, sendo **4** estudos do Google Acadêmico e **2** estudos do Google Regular, por não apresentar ferramentas para automação de validação de dados, não atendendendo assim ao critério de inclusão/exclusão. A partir dos 6 (seis) estudos selecionados, extraiu-se as informações necessárias para se responder às questões de pesquisa.

4 RESULTADOS E DISCUSSÃO

Nesta seção são apresentados os resultados obtidos a partir da execução do MSL, respondendo-se às questões de pesquisa definidas no planejamento do MSL. Na Tabela 2 estão os estudos aceitos após a execução de todas as etapas do MSL (seção 3.5). A primeira coluna da tabela apresenta um código de identificação do estudo, o qual será utilizado ao longo do texto teste trabalho. A segunda coluna apresenta o título do estudo e a terceira, a referência para o trabalho.

4.1 Resposta à Q1

Com a resposta para a questão de pesquisa 01, buscou-se identificar as ferramentas para validação de dados existentes e presentes na literatura, além de indicar determinadas características de cada uma delas. Nota-se, a partir da Tabela 3, que 5 (cinco) ferramentas foram identificadas nos estudos obtidos e que a maioria delas (três) se trata de ferramentas proprietárias.

Tabela 2 – Estudos analisado para responder às questões de pesquisa do MSL.

#	Título	Referência
01	Regression Testing of A Relational Database	Chopra, Kapoor e Kapoor (2009)
02	Regression Testing for Data-Driven Applications	Sharma e Agrawal (2013)
03	Database Testing using Selenium Web Driver – A Case Study	V.neethidevan e G.chandrasekaran (2018)
04	Data warehouse testing	Elgamal, Elbastawissy e Galal-edeen (2013)
05	Deliver Trusted Data by Leveraging ETL Testing	Kumar (2014)
06	Automation in ETL Testing	Kulkarni (2010)

Fonte: Do autor (2019).

É importante salientar que os estudos não apresentaram os dados esperados para responder essa questão de pesquisa por completo, já que todos eles apenas citaram as ferramentas e não necessariamente discorreram sobre a utilização das mesmas. Por conta disso, foi necessária a realização de uma pesquisa à parte no *website* oficial de cada

ferramenta, a fim de identificar as características indicadas na Tabela 3 e entender um pouco mais sobre cada uma delas.

Pode-se notar que as ferramentas proprietárias não informam sobre suas plataformas de desenvolvimento e, obviamente, não permitem acesso ao código fonte, limitando seus usuários às funcionalidades pré-existentes na ferramenta. Quanto à periodicidade de atualização, considerando as ferramentas que constavam essa informação, observa-se datas bem próximas, sendo a última data de atualização ocorrida a menos de um ano até o momento de escrita deste trabalho (Maio/2019). No que diz respeito à versão, não foram encontradas informações nos *websites* das ferramentas *QuerySurge* e *iCEDQ*. Em contrapartida, entre as ferramentas que apresentam essa informação, percebe-se uma semelhança nas versões do *Selenium WebDriver* e *Datagaps*.

Outro ponto interessante, que pode ser verificado por meio da Tabela 3, refere-se à quantidade de citações. Nota-se que 3 estudos indicaram a ferramenta *DBUnit* e 2 estudos citaram o *Selenium WebDriver* e *QuerySurge*, enquanto apenas 1 estudo citou em seu texto ambas as ferramentas *Datagaps* e *iCEDQ*. Isso mostra que as ferramentas proprietárias são menos citadas em relação às ferramentas livres. No caso do *DBUnit,* que foi a mais citada, tem-se como hipótese que esta seja uma ferramenta mais conhecida e consequentemente mais utilizada. Além disso, ter um código aberto que permite personalizações pode ser um fator relevante considerado por seus usuários, tornando mais notório seu uso e disseminação.

Tabela 3 – Ferramentas para testes automatizados de validação de dados.

Nome	Licença	Versão	Download em	Última atualização	Plataforma de desenvolvimento	Estudos
DBUnit	GNU	2.6.0	https://search.maven.org/search?q=g:org.dbunit	Nov/2018	Java; JUnit	[01] [02] [04]
Selenium WebDriver	Apache 2.0	3.14.0	https://www.seleniumhq.org/download/	Nov/2018	Java; Python	[03] [05]
QuerySurge	Proprietária	-	https://www.querysurge.com/compare-trial-options	-	-	[04] [06]
Datagaps	Proprietária	3.4.7	http://www.datagaps.com/etl-testing-tools/etl-validator-download	Julho/2018	-	[06]
iCEDQ	Proprietária	-	https://icedq.com/download-icedq-trial	-	-	[06]

Fonte: Do autor (2019).

4.2 Resposta à Q2

Com a questão de pesquisa 02, buscou-se levantar os tipos de integração que cada ferramenta analisada oferecia, seja em relação a rastreamento de *bugs*, controle de versão ou SGBD (Sistemas Gerenciadores de Base de Dados). Diante disso, foram verificados nos *websites* de cada ferramenta as informações necessárias para responder a esta questão. Quanto às ferramentas de rastreamento de *bugs*, responsáveis pelos relatórios e gerenciamento de *bugs* detectados durante o processo de teste de software, tal como o *Mantis*[4], nenhuma das ferramentas analisadas neste trabalho provê integração com tais tipos de tecnologia. Isso aponta para um interessante *gap* de pesquisa a ser explorado.

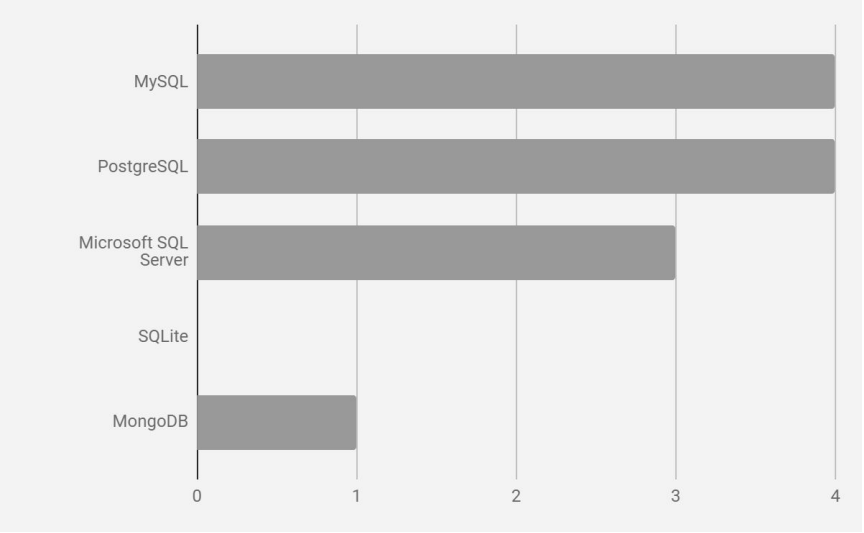
Em relação às ferramentas de controle de versão, incumbidas de gerenciar versões distintas no desenvolvimento de um software, tais como *Git*[5] e *Apache Subversion*[6], as ferramentas *DBUnit* e *Selenium WebDriver* apresentaram integração com o *Git*, que é um sistema de controle de versão amplamente utilizado por desenvolvedores de software. De

[4] https://www.mantisbt.org/
[5] https://git-scm.com/downloads
[6] https://subversion.apache.org/

acordo com o relatório *Octoverse*[7], publicado pela plataforma *GitHub*, há mais de 31 milhões de desenvolvedores que usam a ferramenta *Git*, totalizando mais de 96 milhões de repositórios, sendo que os números têm crescido cada vez mais. Já o restante das ferramentas não informaram qualquer tipo de integração com esse tipo de sistema.

Em contrapartida, ao verificar os SGBDs suportados por cada uma delas, verificou-se uma grande variedade de possibilidades de integração, exceto pelo *Selenium WebDriver*, para o qual não foi encontrada informações a respeito. De acordo com o *Developer Survey Results*, uma pesquisa realizada anualmente pelo fórum de desenvolvimento *Stack Overflow* (2019), os cinco primeiros SGBDs preferidos em 2018 foram: *MySQL*, *PostgreSQL*, *Microsoft SQL Server*, *SQLite* e *MongoDB*. Considerando isso, verificou-se quais ferramentas possuem integração com esses bancos de dados, conforme pode ser visto no gráfico da Figura 1. Das quatro ferramentas que indicavam permitir integração com SGBDs, todas elas possuem integração com o *PostgreSQL* e *MySQL*. Em contrapartida, nenhuma delas possuem integração com o *SQLite* e apenas o *QuerySurge* integra-se ao *MongoDB*. Já o *Microsoft SQL Server* pode ser integrado ao *QuerySurge*, *Datagaps* e *iCEDQ*.

Figura 1 – Integração das ferramentas com os *top* 5 bancos de dados do *Developer Survey Results*.



Fonte: Do autor (2019).

Além disso, a ferramenta *iCEDQ* permite integração com os servidores de integração contínua *Jenkins* e *Bamboo*, enquanto o *Datagaps* permite integração apenas com o *Jenkins*.

[7] https://octoverse.github.com/

O servidor de integração contínua é responsável, basicamente, pela integração automática de todas as partes de um trabalho, detectando erros e retornando *feedbacks* rápidos.

A ferramenta *QuerySurge* também permite integração com alguns sistemas de gerenciamento de testes como o *Micro Focus Quality Center*, *Microsoft Team Foundation Server*, e *IBM Rational Quality Manager*. O sistema de gerenciamento de testes auxilia a gestão de atividades do teste de software, permitindo a consolidação de um plano de teste e seu controle.

A Tabela 4 relaciona as ferramentas para validação de dados apresentadas neste trabalho e as possíveis integrações com outras ferramentas de apoio.

Tabela 4 – Ferramentas de teste e possíveis integrações com outras ferramentas.

		DBUnit	*Selenium WebDriver*	*QuerySurge*	*Datagaps*	*iCEDQ*
Controle de Versão	*Git*	**X**	**X**			
Gerenciamento de Banco de Dados	*MySQL*	**X**		**X**	**X**	**X**
	PostgreSQL	**X**		**X**	**X**	**X**
	Microsoft SQL Server			**X**	**X**	**X**
	MongoDB			**X**		
Integração Contínua	*Jenkins*				**X**	**X**
	Bamboo					**X**
Gerenciamento de Testes	*Micro Focus Quality Center*					**X**
	Microsoft Team Foundation Server					**X**
	IBM Rational Quality Manager					**X**

Fonte: Do autor (2019).

Como pode-se ver, a ferramenta *iCEDQ* se integra com praticamente todas as ferramentas de apoio citadas, não apresentando integração apenas com o *Git*. Isso permite um bom gerenciamento e escalonamento dos testes, visto que, com o uso de integração contínua, os testes se tornam mais rápidos e ágeis, ao passo que o gerenciamento dos caso de testes se torna também facilitado com o auxílio de uma ferramenta com este fim. Em contrapartida, foi

verificado que o *Selenium WebDriver* apresenta integração apenas com o *Git*, no entanto, por se tratar de uma ferramenta de código aberto, o usuário tem a possibilidade de implementar integrações com outras ferramentas conforme sua necessidade e preferência. Por fim, verifica-se que o *QuerySurge* é o que apresenta integração com o maior número dos bancos de dados citados, oferecendo assim maior flexibilidade para sua utilização.

4.3 Resposta à Q3

Em relação às formas de elaboração dos casos de testes, foram localizados poucos conteúdos sobre cada uma das ferramentas. *DBUnit* se trata de uma extensão do *JUnit* e o desenvolvimento de um caso de teste utilizando-o segue basicamente a estrutura de um projeto Java. Deve-se criar uma classe de teste que estende a classe *DBTestCase*, conforme a primeira linha da Listagem 1, que, por sua vez, estende a classe *JUnit TestCase*.

Listagem 1 – Exemplo de implementação do *DBUnit*.

```
1  public class SampleTest extends DBTestCase {
2     public SampleTest(String name) {
3         super(name);
4         System.setProperty(PropertiesBasedJdbcDatabaseTester.DBUNIT_DRIVER_CLASS,
5             "org.hsqldb.jdbcDriver" );
6         System.setProperty( PropertiesBasedJdbcDatabaseTester.DBUNIT_CONNECTION_URL,
7             "jdbc:hsqldb:sample" );
8         System.setProperty( PropertiesBasedJdbcDatabaseTester.DBUNIT_USERNAME, "sa" );
9         System.setProperty( PropertiesBasedJdbcDatabaseTester.DBUNIT_PASSWORD, "" );
10    }
11    public void testMe() throws Exception {
12        // Busca dados do banco de dados
13        IDataSet databaseDataSet = getConnection().createDataSet();
14        ITable actualTable = databaseDataSet.getTable("TABLE_NAME");
15
16        // Carrega dados esperados de um conjunto de dados XML
17        IDataSet expectedDataSet = new FlatXmlDataSetBuilder().build(new
18            File("expectedDataSet.xml"));
19        ITable expectedTable = expectedDataSet.getTable("TABLE_NAME");
20
21        // Compara os dados da tabela do banco de dados com os dados do arquivo XML
22        Assertion.assertEquals(expectedTable, actualTable);
23    }
24 }
```

Fonte: *Getting Started DBUnit* (2019).

O caso de teste apresentado na Listagem 1, se trata de um código simples que compara os valores de uma tabela do banco de dados com valores recuperados de um arquivo XML (*Extensible Markup Language*). Entre as linhas 2 e 10, tem-se a configuração com o banco de dados que será utilizado nos testes. Nas linhas 13 e 14 é feita a busca de dados no banco de dados, enquanto nas linhas 17, 18 e 19 são carregados os dados do arquivo em XML. Por fim,

na linha 22, é feita a comparação entre os dados recuperados do banco de dados e aqueles recuperados do arquivo.

Quanto ao *Selenium WebDriver*, ele oferece uma maior gama de formas de uso, podendo ter seu desenvolvimento feito por meio das linguagens *Java*, *C#*, *Python*, *Ruby, PHP*, *Perl* ou *Javascript*. No exemplo apresentado na Listagem 2, utiliza-se a linguagem *Java* e, após realizar conexão com o banco de dados, alguns dados são recuperados para serem utilizados na interface. No caso do e-mail, por exemplo, ao utilizar o dado recuperado do banco de dados, é possível verificar se é realmente um e-mail válido, conforme exemplo mostrado na Listagem 2.

Listagem 2 – Exemplo de implementação do *Selenium WebDriver* para validação de dados.

```
1   // Carrega o driver JDBC do Microsoft SQL Server
2   Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
3
4   // Prepara a URL de conexão
5   String url = "jdbc:sqlserver://192.168.1.180:1433;DatabaseName=TEST_DB";
6
7   // Obtém conexão com o banco de dados.
8   public static Connection con =
9   DriverManager.getConnection(url, "username", "password");
10
11  // Cria o objeto de instrução que será usado para escrever a instrução DDL e DML SQL.
12  public static Statement stmt = con.createStatement();
13
14  // Envia instruções SQL SELECT ao banco de dados por meio do método
15  // Statement.executeQuery, que retorna as informações solicitadas como linhas de dados em // um
16  objeto ResultSet
17
18  ResultSet result =  stmt.executeQuery
19  ("select top 1 email_address from user_register_table");
20
21  // Move o cursor da posição padrão para a primeira linha do conjunto de resultados
22  result.next();
23
24  // Busca o valor de "email_address" do objeto "result".
25  String emailaddress = result.getString("email_address");
26
27  // Usa o valor emailAddress para efetuar login no aplicativo
28  driver.findElement(By.id, "userID").sendKeys(emailaddress);
29  driver.findElement(By.id, "password").sendKeys(secretPassword);
30  driver.findElement(By.id, "loginButton").click();
31  WebElement element = driver.findElement(By.xpath, "//*[contains(.,'Welcome back ')]");
32  Assert.assertTrue(element.getText().contains(emailaddress), "Unable to log in for user" +
33  emailaddress)
```

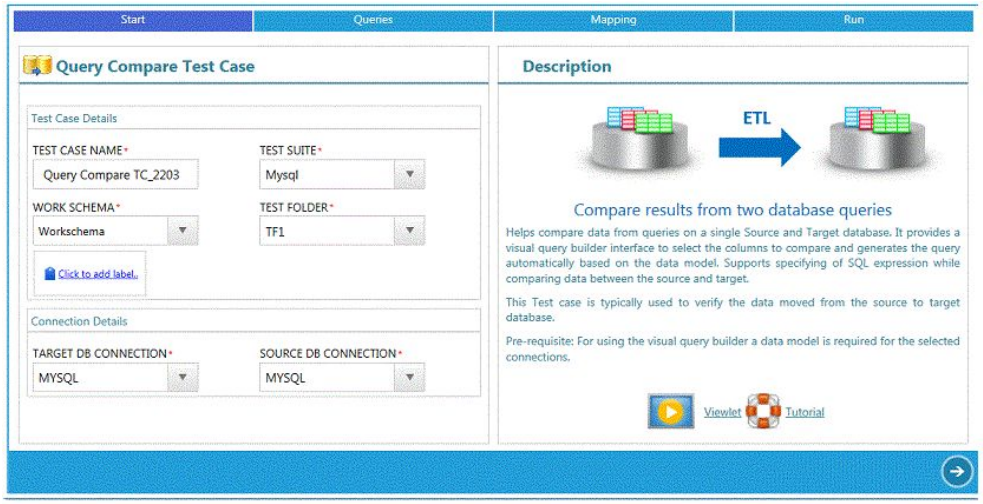
Fonte: *Database Validation, Selenium WebDriver Documentation* (2019).

No entanto, outras abordagens podem ser adotadas. Os dados recuperados do banco de dados podem, por exemplo, ser utilizados para comparar com os dados exibidos em tela. Outra alternativa, ainda, seria criar uma página web simples com uma tabela de dados válidos e compará-los com os dados recuperados do banco de dados.

Já a ferramenta *Datagaps* oferece uma abordagem diferente, pois a elaboração de casos de testes é baseada em uma interface gráfica, não sendo necessário a escrita de códigos. As interfaces são orientadas por assistente com recursos de arrastar e soltar, tanto para criar quanto para executar os casos de testes. A proposta da empresa é oferecer uma interface intuitiva que permita que as equipes de testes sejam mais produtivas e realizem testes mais abrangentes com facilidade.

A *Datagaps* permite comparação dos dados de uma única base de origem ou entre dados de diferentes bases. Na Figura 2 é exibida a primeira etapa de construção do caso de teste, na qual o testador irá nomear o caso de teste e selecionar as conexões que serão utilizadas no teste.

Figura 2 – Primeira etapa do caso de teste de comparação no *Datagaps*.

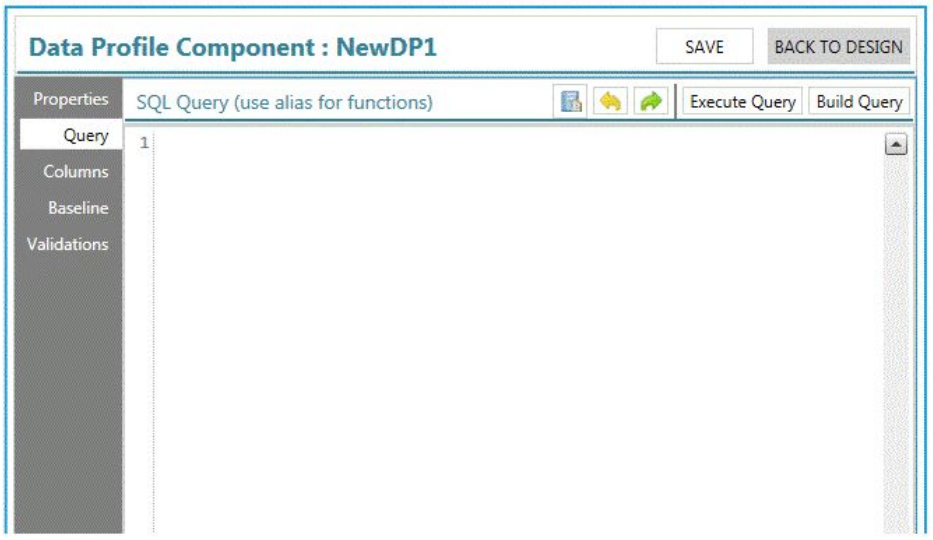


Fonte: *ETL Validator User Guide Datagaps* (2019).

Posteriormente, o testador deverá criar as *querys* que serão utilizadas na execução do teste. Para isso, um componente será apresentado ao usuário, conforme exibido na Figura 3, para que seja incluída a *query* em questão.

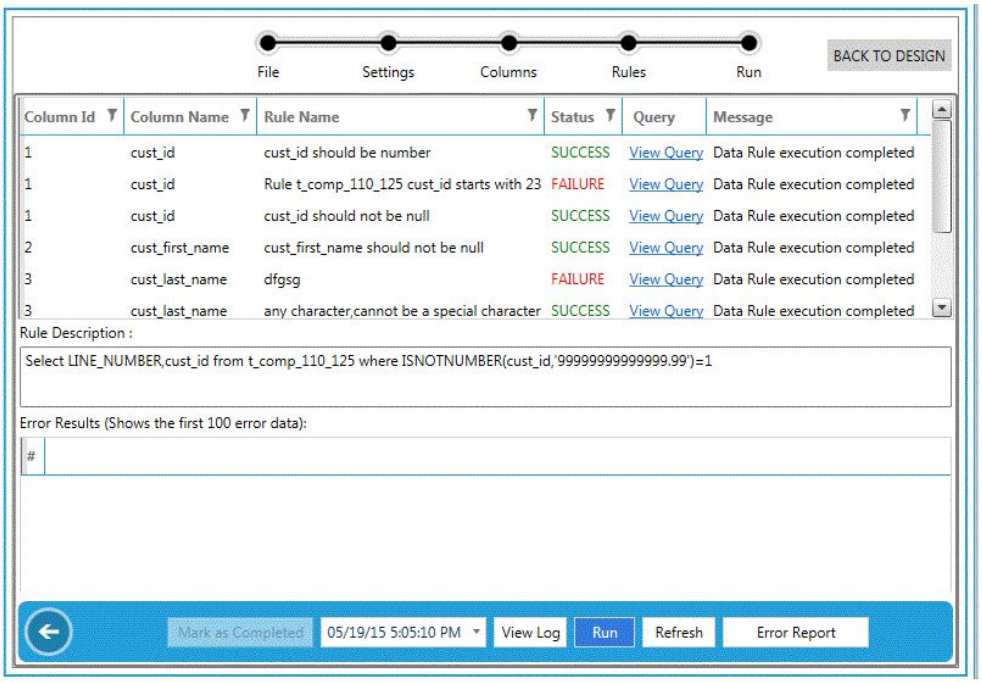
Após realizar a execução dos casos de testes, o *Datagaps* apresenta um relatório com informações sobre a execução, facilitando assim a visualização dos casos que tiveram sucesso ou falharam. Na Figura 4, é exibido um exemplo de um relatório com informações da execução de um teste.

Figura 3 – Componente para inserir *query*.



Fonte: *ETL Validator User Guide Datagaps*.

Figura 4 – Detalhes da execução de teste no *Datagaps*

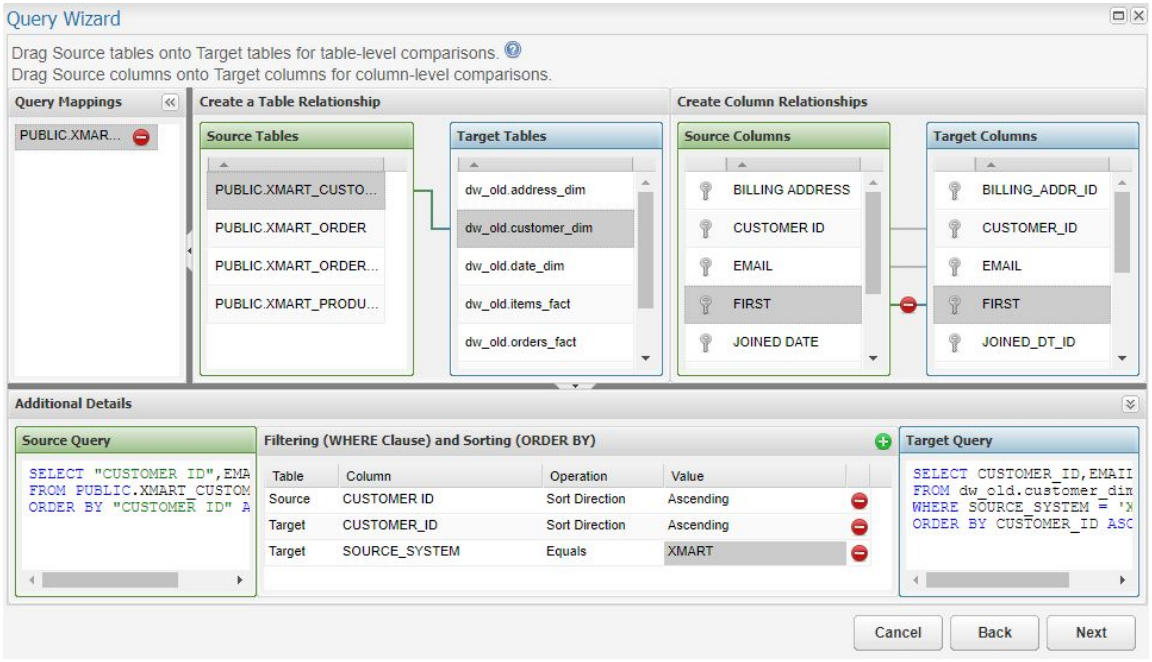


Fonte: ETL Validator User Guide Datagaps.

Analogamente à *Datagaps*, a ferramenta *QuerySurge* também oferece uma interface ao usuário para criação e execução dos casos de testes. Ela permite a comparação entre dados de um arquivo e banco de dados ou entre dois bancos de dados. Além disso, os níveis de comparação são entre colunas, tabelas ou contagem de linhas. Na Figura 5, por exemplo,

tem-se um mapeamento de tabela e colunas para comparação em nível de colunas. Dessa forma, é realizada uma ligação entre a tabela que será utilizada e outras ligações entre as colunas do arquivo (ou banco de dados) de origem e as colunas do banco de dados alvo (que contêm os dados a serem validados). A partir desse mapeamento, as *querys* vão sendo construídas automaticamente.

Figura 5 – Mapeamento de tabela e colunas no *QuerySurge*.

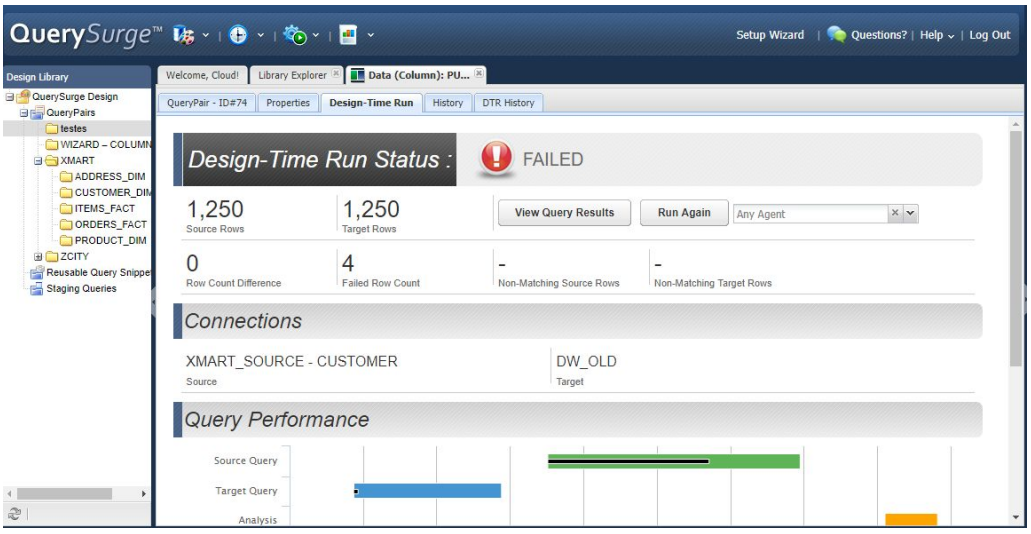


Fonte: Do autor (2019).

Após o mapeamento das colunas e a construção das *querys*, o caso de teste pode ser salvo e executado posteriormente. Ao executar um caso de teste, o *QuerySurge* apresenta um relatório com informações sobre a execução. A Figura 6 apresenta o relatório da execução do teste criado e exibido anteriormente na Figura 5. É possível verificar que o teste falhou e que, mais especificamente, 4 linhas comparadas entre as bases não obtiveram sucesso na comparação.

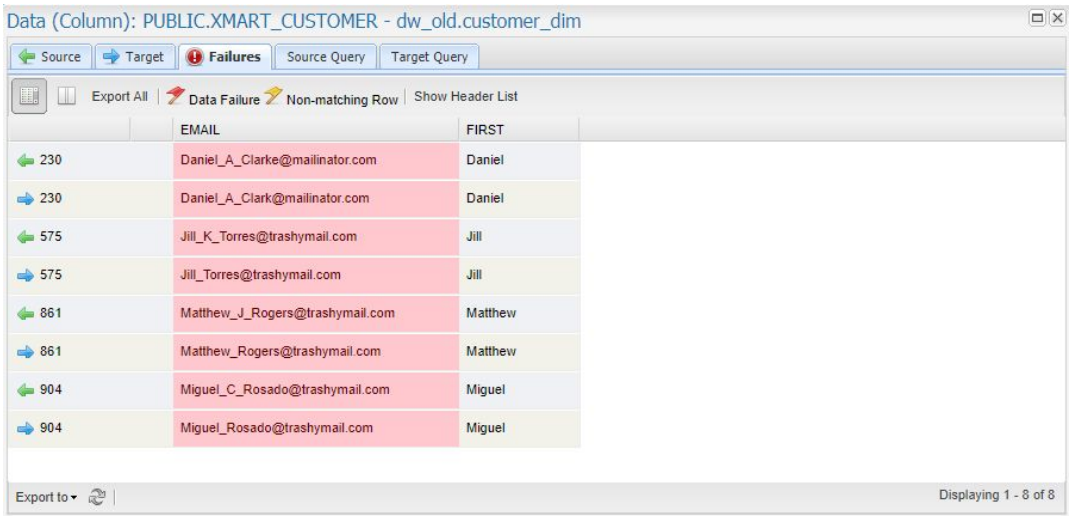
Ao visualizar os resultados das *querys* executadas, conforme exibido na Figura 7, verifica-se quais linhas falharam, comparando os dados que são apresentados em cada uma das bases. No exemplo, percebe-se que os e-mails possuem algumas diferenças, como no primeiro em que se tem o nome “Clark” em um e-mail e “Clarke” em outro, se diferenciando então pela letra “e”.

Figura 6 – Relatório de execução de teste do QuerySurge.



Fonte: Do autor (2019).

Figura 7 – Resultados que falharam no teste executado.



Fonte: Do autor (2019).

Por fim, em relação à ferramenta *iCEDQ*, não foram encontradas informações à respeito da elaboração de teste, bem como não foi possível o acesso à ferramenta, visto que para obter a versão de teste é necessário realizar um cadastro no site da ferramenta, informando alguns dados pessoais e da empresa em que atua, para somente então ser contatado posteriormente por um consultor da ferramenta que irá liberar o acesso. No entanto, ainda que o cadastro tenha sido realizado no site, nenhum retorno foi recebido por e-mail.

4.4 Resposta à Q4

Por fim, a questão de pesquisa 04 tinha como objetivo levantar os benefícios e desafios apresentados nos estudos a respeito da utilização das ferramentas citadas por eles. Entretanto, como dito anteriormente, os artigos apenas citaram as ferramentas e não exploraram as minúcias da utilização de cada uma delas, visto que o foco desses artigos foram principalmente os conceitos sobre teste de regras de negócio e não necessariamente a utilização de ferramentas em si. As ferramenta foram citadas como recomendações para execução desse tipo de teste, tornando possível o levantamento realizado neste trabalho.

Entende-se, no entanto, que determinadas características apresentadas como resposta para a questão 01, podem ser tidos como possíveis benefícios e/ou desafios, dependendo do cenário da empresa. No caso do *DBUnit* e *Selenium WebDriver*, por exemplo, tem-se a vantagem de serem ferramentas com licenças de software livre e código aberto, permitindo seu uso gratuito e possibilitando alguma personalização específica, se necessário. Já as outras ferramentas são pagas e proprietárias, não permitindo o acesso ao código fonte, mas oferecem um período de teste gratuito que possibilita um maior conhecimento sobre seu uso e auxilia na tomada de decisão sobre contratar a ferramenta ou não.

Em relação à elaboração dos casos de testes, as características de cada ferramenta também podem ser vistas como pontos positivos ou negativos, de acordo com o ambiente em que serão aplicadas. As ferramentas que oferecem interface gráfica podem demandar menos conhecimento técnico dos testadores, mas não permite que os casos de testes evoluam além das funcionalidades já existentes. Enquanto as ferramentas de código aberto oferecem o contrário, pois permitem a personalização, se necessário, mas também demanda um maior conhecimento técnico por parte dos envolvidos.

5 CONSIDERAÇÕES FINAIS

Este trabalho teve por objetivo identificar, classificar e catalogar os apoios computacionais existentes na literatura que possam auxiliar na automação dos testes de validação de dados, bem como salientar algumas de suas características, apresentar

brevemente sobre as formas de elaboração de casos de testes utilizadas por cada uma delas, além de discutir alguns benefícios e desafios obtidos por meio da utilização desses recursos.

Por meio da execução de um mapeamento sistemático da literatura, tornou-se perceptível a escassez de estudos em torno desta temática, fazendo necessário a busca por informações complementares por meio de outras fontes. Ou seja, já é sabido sobre a importância de manter íntegros os dados de um sistema, a fim de não afetar o funcionamento do sistema como um todo e não prejudicar seus usuários, porém pouco tem se falado sobre as ferramentas que podem auxiliar nesse processo de teste, visando maior eficiência da equipe e abrangência dos testes.

Outro ponto importante se refere à qualidade dos artigos publicados sobre o tema, visto que a maioria dos estudos retornados no Google Regular não atendiam os critérios descritos na Tabela 1 deste trabalho, por conta de falta de objetividade (já que muitos deles são tendenciosos e buscam vender sua própria ferramenta), autoridade (alguns dos estudos nem mesmo apresentavam o nome do autor do texto) ou precisão (a grande maioria dos estudos não apresentam referências bibliográficas), por exemplo. Com isso, poucos estudos puderam ser analisados e, consequentemente, poucas ferramentas foram levantadas. Acredita-se que há, de fato, um *gap* de pesquisa e até mesmo de prática em torno da temática, resultando em insumos diminutos.

É importante ressaltar que a heurística utilizada para interromper as buscas no Google Acadêmico e Regular foi apresentada por outros autores e utilizada em mapeamentos sistemáticos publicados, mas entende-se que ela pode não ser totalmente efetiva e, por conta disso, artigos sobre o assunto podem não ser abordados neste trabalho.

No que se refere às ferramentas apresentadas, foi possível verificar que diferentes abordagens podem ser utilizadas para este tipo de teste, seja utilizando uma interface gráfica ou escrevendo códigos, por exemplo. Contudo, percebeu-se também que a escolha de uma ferramenta em detrimento de outra está fortemente relacionada ao ambiente em que ela será aplicada, dependendo do nível de conhecimento dos testadores, disponibilidade financeira para compra de uma ferramenta e assim por diante.

Como trabalhos futuros, pretende-se levantar dados à respeito da utilização de testes de validação de dados para entender a causa de ainda ser uma temática tão pouco abordada na literatura. Além disso, pretende-se também realizar uma validação das ferramentas

mencionadas para verificar na prática os possíveis benefícios e desafios de cada uma delas em um ambiente organizacional.

REFERÊNCIAS

BASTOS, Aderson *et al*. **Base de conhecimento em teste de software.** 3. ed. São Paulo: Martins, 2012.

CHARLIER, Cédric. **NBi.** 2018. Disponível em: <http://www.nbi.io/>. Acesso em: 03 nov. 2018.

CHOPRA, Rajiv; KAPOOR, Kapila; KAPOOR, Geetika. Regression Testing of A Relational Database. In: COMPUTING FOR NATION DEVELOPMENT, 3., 2009, New Delhi. **Proceedings of the 3 rd National Conference.** New Delhi: Bharati Vidyapeeth's Institute Of Computer Applications And Management, 2009. Disponível em: <http://www.bvicam.ac.in/news/INDIACom%202009%20Proceedings/pdfs/papers/265.pdf>. Acesso em: 15 maio 2019.

ELGAMAL, Neveen; ELBASTAWISSY, Ali; GALAL-EDEEN, Galal. Data warehouse testing. In: EXTENDING DATABASE TECHNOLOGY, 13., 2013, Genoa. **EDBT '13 Proceedings of the Joint EDBT/ICDT 2013 Workshops.** New York: Acm, 2013. p. 1 - 8. Disponível em: <https://dl.acm.org/citation.cfm?id=2457319>. Acesso em: 15 maio 2019.

GAROUSI, Vahid; KÜÇÜK, Barış. Smells in software test code: a survey of knowledge in industry and academia. **Journal Of Systems And Software**. Amsterdam. abr. 2018.

KULKARNI, Omkar Sanjay. Automation in ETL Testing. **International Journal Of Computer Science And Information Technologies**, Sangli, v. 8, n. 6, p.586-589, fev. 2010. Mensal. Disponível em: <http://ijcsit.com/docs/Volume%208/vol8issue6/ijcsit2017080602.pdf>. Acesso em: 15 maio 2019.

KUMAR, Vijay. **Deliver Trusted Data by Leveraging ETL Testing. 2014.** Elaborado por Cognizant. Disponível em: <https://www.cognizant.com/InsightsWhitepapers/deliver-trusted-data-by-leveraging-etl-testing-codex1031.pdf>. Acesso em: 15 maio 2019.

LAUDON, Kenneth C.; LAUDON, Jane P. **Sistemas de informação gerenciais.** 11. ed. São Paulo, SP: Pearson, 2014. xx, 484 p. ISBN 9788543005850 (broch.).

MYERS, Glenford J.; BADGETT, Tom; THOMAS, Todd M.; SANDLER, Corey. **The art of software testing.** 2nd ed. Hoboken, NJ: J. Wiley, 2004.

PETERSEN, Kai et al. Systematic mapping studies in software engineering. **Ease'08 Proceedings Of The 12th International Conference On Evaluation And Assessment In Software Engineering.** Italy, p. 68-77. jun. 2008. Disponível em: <https://dl.acm.org/citation.cfm?id=2227115.2227123>. Acesso em: 21 jun. 2019.

PRESSMAN, Roger S.. **Engenharia de software:** uma abordagem profissional. 7. ed. Porto Alegre: AMGH, 2011.

STACK OVERFLOW (Org.). **Developer Survey Results.** 2019. Disponível em: <https://insights.stackoverflow.com/survey/2019#technology>. Acesso em: 15 maio 2019.

SHARMA, Vandana; AGRAWAL, Arun Prakash. Regression Testing for Data-Driven Applications. **International Journal Of Innovative Technology And Exploring Engineering (ijitee)**, Noida, v. 8, p.209-211, jun. 2013. Mensal. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.687.5220&rep=rep1&type=pdf>. Acesso em: 15 maio 2019.

V.NEETHIDEVAN; G.CHANDRASEKARAN. Database Testing using Selenium Web Driver – A Case Study. **International Journal Of Pure And Applied Mathematics**, v. 118, n.8, p.559-565, 2018. Disponível em: <https://acadpubl.eu/jsi/2018-118-7-9/articles/8/81.pdf>. Acesso em: 15 maio 2019.