UNIDADE ITUIUTABA | UEMG
Intercursos Revista Científica
Ciências Exatas

ANÁLISE, VERIFICAÇÃO E VALIDAÇÃO DE SOFTWARE PARA ESTUDOS DE REPRESENTAÇÕES SOCIAIS

ANALYSIS, VERIFICATION AND VALIDATION OF SOFTWARE FOR SOCIAL REPRESENTATIONS STUDIES

ERISON BRITO CABRAL, HÉLIO OLIVEIRA FERRARI, MAURO HEMERLY GAZZANI, KÁTIA LOPES SILVA, OTÁVIO LAFAIETE GUIMARÃES

RESUMO

Comum em pesquisas que adotam o Núcleo Central como referencial teórico o uso do "quadro das quatro casas" que em geral fornece uma visualização do núcleo central e do sistema periférico, os pesquisadores obtêm os dados através, por exemplo, da técnica de associação livre de palavras e montam um quadro que permite conceber a estrutura da Representação Social identificando o seu núcleo e a periferia, com isso é possível realizar o estudo e obter os resultados pretendidos da pesquisa. Este trabalho tem como foco a verificação e validação do software StrixRS desenvolvido em linguagem Java, elaborado para a construção do quadro das 4 casas para pesquisas de representações sociais de abordagem estrutural. Os tipos de testes utilizados foram: Testes de unidade, Testes de integração, testes funcionais e os testes exploratórios. A técnicas empregadas incluem as os testes de caixa preta e de caixa branca. Como considerações finais, tem-se que os testes exploratórios mostraram-se adequados para cenários de teste específicos, como quando alguém precisa aprender sobre um produto ou aplicativo com rapidez e fornecer feedback rápido e que foi a estratégia adotada na verificação e validação do software deste projeto. Eles ajudam a avaliar a qualidade de um produto desde uma perspectiva do usuário. Porém, com relação ao método das quatro casas implementado, ainda restam alguns ajustes e correções no software.

Palavras chave: Representações Sociais. Evocações. Testes de Software. Testes Exploratórios.

ABSTRACT

Common in researches that adopt the Central Nucleus as a theoretical reference to use the "four houses chart" which in general provides a visualization of the central nucleus and the peripheral system, researchers obtain data through, for example, the technique of free association of words and assemble a framework that allows to conceive the structure of the Social Representation identifying its core and periphery, with this it is possible to carry out the study and obtain the intended results of the research. This work focuses on the verification and validation of the StrixRS software developed in Java language, designed for the construction of the 4 houses framework for research on social representations with a structural

approach. The types of tests used were:Unit tests, Integration tests, Functional tests and Exploratory tests. The techniques employed include black-box and white-box testing. As final considerations, exploratory tests have proved to be adequate for specific test scenarios, such as when someone needs to learn about a product or application quickly and provide quick feedback and that was the strategy adopted in the verification and validation of this software. project. They help to assess the quality of a product from a user's perspective. However, regarding the implemented four-house method, there are still some adjustments and corrections in the software.

Keywords: Social Representations. Evocations. Software Testing. Exploratory Tests.

INTRODUÇÃO

O Segundo Jodelet (2001) as representações sociais são reconhecidas como sistemas de interpretação que dirigem nossa relação com o mundo e com os outros. Elas norteiam e estabelecem as condutas e as comunicações sociais.

Da mesma forma, elas intervêm em processos variados, tais como difusão e a assimilação dos conhecimentos, o desenvolvimento individual e coletivo, a definição das identidades pessoais e sociais, a expressão de grupos e as transformações sociais.

A teoria da representação social desdobra-se em três abordagens: culturalista, liderada por Denise Jodelet (2001); a societal liderada por Willen Doise e a abordagem estrutural liderada estrutural liderada por Jean Claude Abric, conforme De Sá (1998).

Machado e Aniceto (2010, p. 352-353) explicam a ideia por trás desta abordagem: O autor sustenta a hipótese de que toda Representação Social está organizada em torno de um núcleo central e um sistema periférico. O núcleo central está relacionado à memória coletiva dando significação, consistência e permanência à representação sendo, portanto, estável e resistente a mudanças. Esse núcleo é composto pelos elementos estáveis ou mais permanentes da representação social, sendo estes de natureza normativa e funcional. Os aspectos funcionais estão ligados à natureza do objeto representado e os normativos dizem respeito aos valores e normas sociais pertencentes ao meio social do grupo.

O sistema periférico é responsável pela atualização e contextualização da representação. A Teoria do Núcleo Central permite aos pesquisadores visualizarem e

entenderem determinada representação de uma maneira estruturada, clara e objetiva. É comum em pesquisas que adotam o Núcleo Central como referencial teórico o uso do “quadro das quatro casas” que em geral fornece uma visualização do núcleo central e do sistema periférico, os pesquisadores obtêm os dados através, por exemplo, da técnica de associação livre de palavras e montam um quadro que permite conceber a estrutura da Representação Social identificando o seu núcleo e a periferia, com isso é possível realizar o estudo e obter os resultados pretendidos da pesquisa.

O foco desta pesquisa se encontra na abordagem estrutural. A Abordagem Estrutural das Representações proposta por Jean Claude Abric é um desdobramento da Teoria das Representações Sociais, nela é proposta a Teoria do Núcleo Central e tem como objeto de estudo a verificação e validação de um software elaborado para a construção do quadro das 4 casas para pesquisas de representações sociais de abordagem estrutural.

Sant’Anna (2012) identificou uma oportunidade de contribuir no auxílio a pesquisadores, ele desenvolveu uma ferramenta computacional chamada openEvoc com o foco nas representações sociais que permite gerar questionários e ou encaminhar para o público-alvo através de um link ou o próprio pesquisador fazer a transcrição da pesquisa. O programa gera o quadro das quatro casas e possibilita também a análise através de gráficos e outros métodos estatísticos. Ao analisar o openEvoc, foram identificados alguns pontos, é uma aplicação web e, portanto, só pode ser executada com a conexão com a internet e um navegador disponível, o armazenamento e processamento é feito inteiramente na nuvem o que pode a vir a ser um problema já que, há uma limitação de armazenamento e a segurança dos dados junto com o desempenho devem ser garantidos pelo servidor.

O contexto explicitado serviu de motivação para o desenvolvimento de uma nova aplicação que busque cobrir pontos que podem ser melhorados nas aplicações existentes e, com isso, contribuir com uma nova solução para os pesquisadores que utilizam a Abordagem Estruturada das Representações Sociais.

Esta nova aplicação, o StrixRS (GUIMARÃES, 2021) foi desenvolvida tendo como requisitos fundamentais a) Compatibilidade com os principais sistemas operacionais utilizados atualmente; b) Possibilidade de utilização mesmo na ausência de internet; c) Criação e edição ilimitada de questionários; d) Métodos para encaminhar os questionários para o público alvo; e) Produção de gráficos e métodos estatísticos para auxiliar na visualização dos dados; f) Geração o quadro das quatro casas para auxiliar na identificação da representação em análise; g) Ter compatibilidade para importação e exportação de arquivos de dados; h) Ser simples, intuitiva e objetiva, com isso, tendo uma baixa curva de aprendizado e facilitando seu uso. Tendo, portanto, o protótipo pronto, a pesquisa se insere no campo da verificação e validação do software para ser ofertando à comunidade de pesquisadores.

Uma vez que a nova aplicação, o StrixRS (GUIMARÃES, 2021) foi desenvolvida, é importante testar, verificar e validar a mesma. Desta forma, um processo com atividades de testes de software é necessário.

Neste contexto, é importante definir o tipo de teste, que é de forma geral, um processo de avaliação que o sistema específico atende aos seus requisitos originalmente especificados ou não. É principalmente um processo que abrange validação e processo de verificação se o sistema desenvolvido atende aos requisitos definidos pelo usuário. Portanto, essa atividade resulta em uma diferença entre o resultado real e o esperado. Teste de software refere-se a encontrar *bugs*, erros ou requisitos ausentes no sistema ou software desenvolvido.

Na indústria de software, para fins de design, desenvolvimento e testes ao mesmo tempo, para que a qualidade do software seja melhorada, todos esses processos são realizados usando um método denominado ciclo de vida de desenvolvimento de software. A verificação e teste de validação de software são definidas pela IEEE (IEEE-Std-610.12,1990) como:

> um conjunto de padrões de engenharia de software, que define o teste de verificação como um teste de um sistema para provar que ele

> atende a todos os seus requisitos especificados em uma fase específica de seu desenvolvimento (IEEE-Std-610.12,1990).

A definição do glossário padrão IEEE citada acima estabelece que o teste de verificação é o processo de examinar todas as especificações de software predeterminadas - documentos, código, design e programa - para garantir que o software atenda a essas especificações. Alguns documentos importantes a serem revisados nesta fase são especificação de requisitos, projetos de projeto, diagramas ER, design de tabela de banco de dados, casos de uso e cenários de teste etc. A verificação precoce e frequente reduz o número de falhas (*bugs*) e defeitos que podem aparecer em estágios posteriores.

Além disso, o Glossário Padrão IEEE de Terminologia de Engenharia de Software define também:

> [...] que o teste de validação como uma atividade que garante que as verdadeiras necessidades e expectativas de uma parte interessada do produto final sejam atendidas (IEEE-Std-610.12,1990).

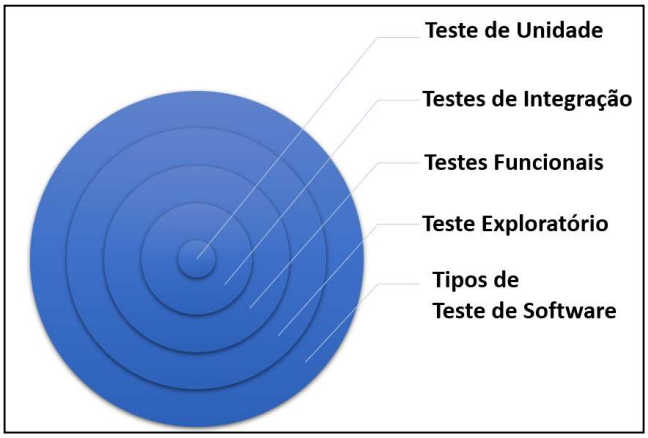
Ao contrário do teste de verificação, que ocorre em todas as etapas do desenvolvimento, o teste de validação ocorre no final de um determinado módulo ou mesmo depois que o software foi completamente construído. Seu principal objetivo é verificar se o produto corresponde aos requisitos das partes interessadas e do cliente.

Conforme os autores Anwar e Kar (2020) da *Bangladesh University of Business and Technology*:

> Nunca podemos dizer que um produto é "Perfeito". O teste é um processo sem fim. O teste mostra apenas a presença de erros, não a ausência. É um processo demorado e intensivo, portanto, técnicas atualizadas e metodologias inovadoras são necessárias para manter a qualidade do software. (ANWAR; KAR, 2019, p.9, tradução nossa)

Os tipos de testes de software são mostrados na figura 1.

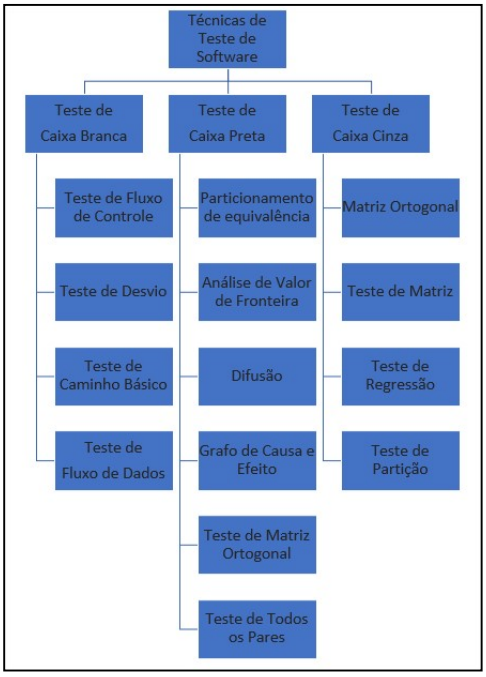
Figura 1 - Tipos de testes de Software



Fonte: Autores

As técnicas de teste de software que podem ser utilizadas para cada tipo são mostradas na figura 2. Pode-se notar que os casos de teste são desenvolvidos usando várias técnicas de teste, para o teste eficaz e preciso. As principais técnicas de teste são: testes de caixa preta, caixa branca e caixa cinza.

Figura 2 - Técnicas de teste de software estruturados



Fonte: Modificado de Jamil et al. (2016)

Os tipos e técnicas de testes trabalham com uma abordagem estruturada. Deste modo, os casos de teste normalmente são estabelecidos utilizando as descrições de usuário e os dados de teste são estruturados com base nos casos de teste definidos. Porém, é comum faltar os casos extremos, que em geral são descobertos durante o Teste de Aceitação do Usuário (UAT).

Neste contexto, os testes exploratórios (TEs) são uma abordagem para testes de software que, com frequência, é descrita como aprendizagem simultânea, *design* de teste e execução. O termo "testes exploratórios" foi introduzido de maneira formal pelo autor Cem Kaner em seu livro: *Testing Computer Software* (Kaner, 1988).

Importante ressaltar que o foco está na descoberta e depende da habilidade do testador individual para descobrir defeitos que não são abrangidos com facilidade no escopo de outros testes na fase de desenvolvimento. Os testes exploratórios existem há algum tempo, mas muitas vezes eram chamados de “testes *ad-hoc*.

TEs são um meio de continuar testando software depois de executar os testes com script e evitar investir em esforço para projetar e documentar cuidadosamente os

testes quando o software está em um estado instável. A literatura entende que uma das principais razões para usar TEs é poder ser capaz de utilizar a criatividade e a experiência dos testadores durante o teste execução, no lugar de gastar muito tempo na concepção casos de teste antes da execução do teste.

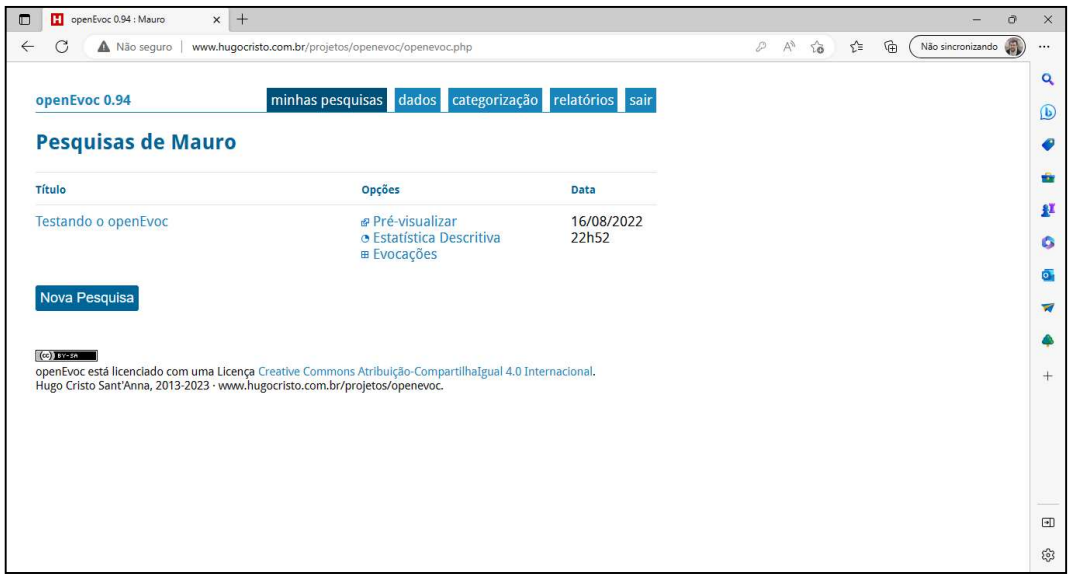
Este trabalho tem como objetivo verificar e validar o software StrixRS (GUIMARÃES, 2021) desenvolvido em linguagem Java, elaborado para a construção do quadro das 4 casas para pesquisas de representações sociais de abordagem estrutural. Os tipos de testes utilizados foram: Testes exploratórios com testes de integração, testes funcionais. A técnicas empregadas incluem as os testes de caixa preta e de caixa branca.

MATERIAL E MÉTODOS

Alguns trabalhos estão fortemente relacionados com este projeto, como exemplo tem-se o artigo chamado *openEvoc*: Um Programa de Apoio à Pesquisas em Representações Sociais foi apresentado por Sant'Anna (2012) no intuito de demonstrar a aplicação por ele desenvolvida, explicando os motivos por trás de sua criação e seu funcionamento. (GUIMARÃES, 2021)

Essa aplicação é bem constituída e possui funcionalidades voltadas para pesquisas em Representações Sociais com sua abordagem estruturada. O acesso a aplicação é feito de forma online, o pesquisador interessado cria sua conta utilizando um endereço de e-mail para identificação e escolhendo uma senha para garantir a segurança, a figura 3 a seguir mostra a tela inicial da aplicação.

Figura 3 - Tela Inicial do *openEvoc*

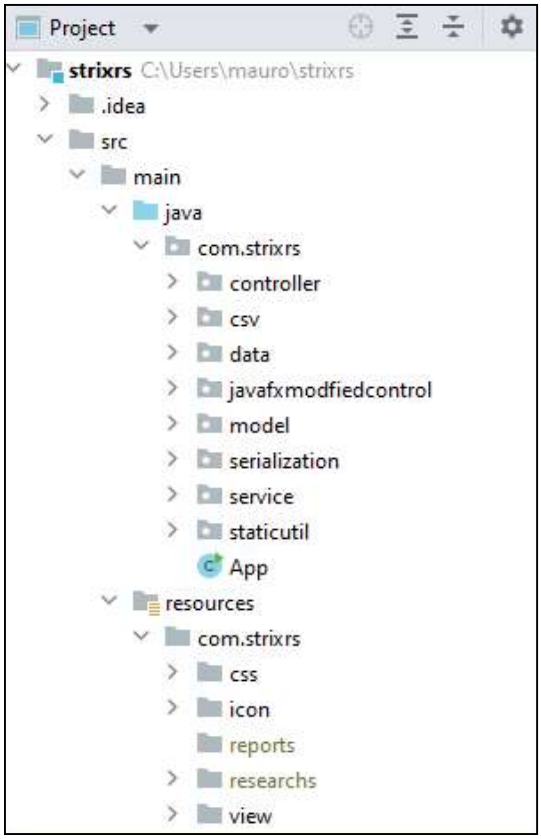


Fonte: Autores

A Figura 4 mostra a organização dos arquivos do projeto. O padrão de arquitetura de software MVC (*Model-View-Controller*) foi seguido e que tem por objetivo separar os arquivos de acordo com suas funções, facilitando assim a sua manutenção e reuso.

No pacote Java está contido as classes referentes a lógica da aplicação e suas regras de negócio, o pacote *model* contém as classes modelo que serão manipuladas pelos controladores que, por sua vez, fazem o papel de intermediadores da interface gráfica com essas classes modelo. Toda a parte gráfica, conhecida como a camada *View* no MVC está contida no pacote *resources* (GUIMARÃES, 2021)

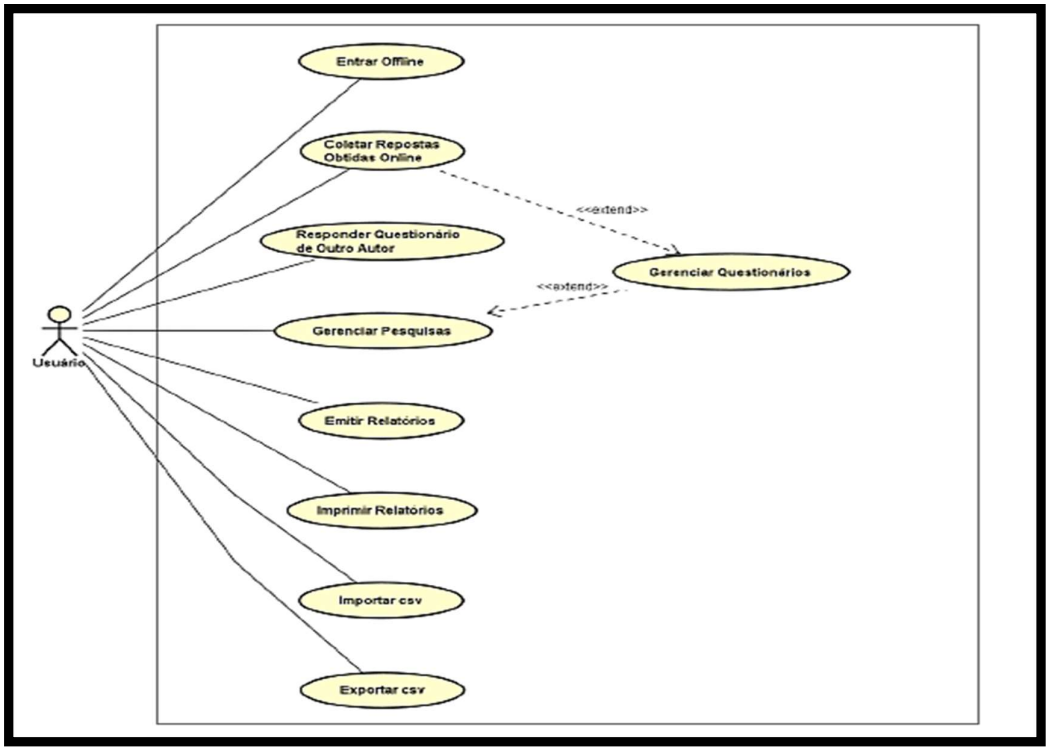
Figura 4 - Estrutura do Projeto



Fonte: Modificado de Guimarães, 2021

O software contempla a verificação da integração adequada dos componentes; a análise se todos os requisitos foram implementados corretamente e a garantia da qualidade do mesmo com relação aos seus requisitos funcionais e casos de uso implementados. A figura 5 mostra o diagrama dos casos de uso que são relacionados com requisitos funcionais da aplicação.

Figura 5 - Diagrama de Casos de Uso



Fonte: Guimarães, 2021

Com relação aos testes exploratórios, a metodologia consiste nas seguintes etapas:

1) **Definir o escopo:** Planeje o que deve ser tratado durante a sessão de teste exploratório, por exemplo, uma área específica de funcionalidade em seu produto, um recurso específico, um conjunto de diálogos do usuário final, um fluxo de trabalho específico etc.
2) **Definir esquema de tempo**: Definir quanto tempo cada participante da sessão deve gastar para a sessão de teste. Isso mitiga o risco de os participantes se perderem no aplicativo e perderem tempo brincando com áreas ou aspectos sem importância. Além disso, a concentração dos

participantes diminuirá quanto mais tempo demorar uma sessão. Portanto, limitar a duração da sessão por *timeboxing* é um fator de sucesso crucial.

3) **Selecionar a equipe**: É melhor convidar e incluir pessoas com experiências e funções diferentes: não apenas especialistas em teste, mas também, por exemplo, gerentes de projeto, desenvolvedores, proprietários de produtos, analistas de negócios ou, é claro, seus clientes ou usuários finais. Além disso, pessoas de fora do projeto podem fornecer pistas interessantes, pois abordam o sistema desconhecido com uma mente mais imparcial.
4) **Investigar:** Iniciar a sessão e (junto ou em paralelo com todos os participantes da sessão) e começar a explorar! Executar a história do usuário ou recurso em questão e observe o comportamento do sistema. Registrar brevemente suas descobertas e perguntas que surgem – em movimento – em paralelo às suas atividades de teste – o mínimo possível, tanto quanto necessário.
5) **Avaliar os resultados:** Após cada sessão de teste exploratório, deve-se avaliar as descobertas e os resultados da sessão. Revisar e discutir os resultados de todos os participantes e os prós/contras do comportamento atual do produto. Se necessário, preparar e enviar relatórios de defeitos ao seu sistema de gerenciamento de defeitos. Decidir sobre as próximas etapas, por exemplo, adicionar ou atualizar itens de requisitos ou derivar casos de teste que podem ser usados posteriormente para testes de regressão (automatizados).

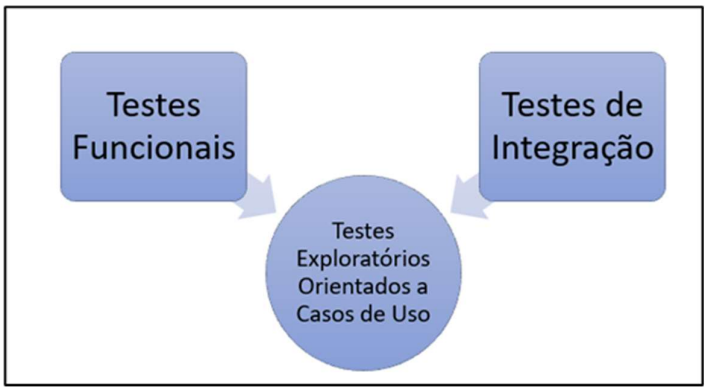
Conforme Itkonen e Rautiainen (2005) as técnicas utilizadas para testes exploratórios podem ser as mesmas utilizadas em testes estruturados (conforme a figura 2). Porém, existem as seguintes formas mais comuns de se utilizar testes exploratórios: TEs baseados em sessão, teste exploratório de fumaça, TEs de regressão exploratória, TEs subcontratado e TEs f*reestyle*.

Vários motivos levaram a ser utilizados TEs nesta pesquisa, dentre eles tem-se: 1) O software pode ser usado de muitas maneiras ou há tantas combinações entre características diferentes que escrever casos de teste detalhados para tudo é difícil,

trabalhoso e mesmo impossível. 2) TEs adaptam-se bem ao teste do ponto de vista do usuário. 3) TEs enfatizam a utilização da experiência e criatividade dos testadores para encontrar defeitos. 4) TEs ajudam a fornecer *feedback* rápido sobre novos recursos de testadores a desenvolvedores. 5) TEs adaptam-se bem a situações em que os requisitos e as características testadas mudam frequentemente, e as especificações são vagas ou incompletas. 6) TEs oferecem uma forma de aprender sobre o sistema, cujos resultados podem ser utilizados em outras tarefas, como suporte ao cliente e treinamento.

Neste trabalho, os testes exploratórios foram orientados a casos de uso, utilizando testes funcionais e de integração. A figura 6 mostra uma visão geral dos tipos e técnicas de testes utilizados

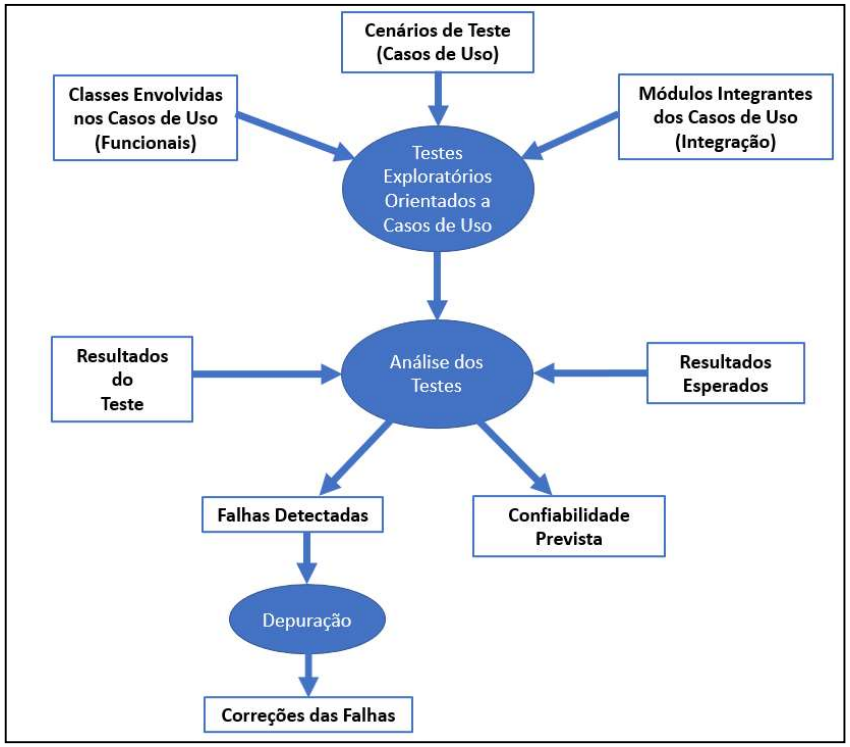
Figura 6 - Visão geral dos tipos e técnicas de testes utilizadas neste trabalho



Fonte: Autores

O processo para executar os TEs seguiram o fluxo de informações do processo de teste mostrado na figura 7.

Figura 7 – Fluxo de informações do processo de teste



Fonte: Autores

RESULTADOS E DISCUSSÃO

Os testes foram desenvolvidos baseados nos casos de uso, usando as técnicas descritas na sessão anterior, para obter testes mais eficazes. Para isto, a integridade do software é fornecida e as condições de testes que obtêm a maior probabilidade de encontrar erros são escolhidos. Portanto, os testadores não adivinham qual casos de teste devem escolher, e as técnicas de teste permitem que eles projetarem condições de teste de forma sistemática.

Foram realizados os testes e coletadas as avaliações dos benefícios e deficiências do StrixRS segundo os cenários de testes. Um cenário de teste é basicamente uma documentação de um caso de uso. Em outras palavras, descreve uma ação que o usuário pode realizar com um site ou aplicativo. Também pode descrever uma situação em que o usuário pode se encontrar enquanto usa esse software.

Cenários de teste são criados para garantir que todas as funcionalidades oferecidas por um site ou aplicativo estejam funcionando como esperado. Para criar cenários de teste precisos, é importante reunir informações de clientes, partes interessadas e desenvolvedores. Isso ajuda a cobrir efetivamente todos os possíveis cenários de usuário e permite testes abrangentes de todos os fluxos de negócios do software em questão.

Quadro 1 - Resultados dos testes Realizados

Cenário de Teste	Especificação dos Cenários de Teste	Classes Java Envolvidas	Falhas Detectadas	Descrição das Falhas
Casos de Uso	Coletar Respostas Obtidas On-line	OnlinePaneService CollectAnswerOnlineController CollectAnswerOnlineService MainController AbstractController Question Answer	Não	Coleta respostas com sucesso
	Responder Questionário de Outro Autor	AddAnswerOnlineService AddAnswerOnlineController OnlinePaneService Question Answer MainController AbstractController	Sim	Código não validado
	Entrar Off-line	AbstractController LoginController	Não	Inicia sem conexão internet
	Gerenciar Pesquisas	AddResearchController AddResearchService ResearchPaneService AbstractController MainController	Não	Gerencia a pesquisa conforme requisitos
	Gerenciar Questionários	AddQuestionController AddQuestionService QuestionPaneService AddAnswerController AddAnswerService MainController AbstractController Research Question Answer	Sim	Valida questionários com mesmo nome
	Emitir Relatórios	AddReportController AddReportService AddSpecificReportController ReportPaneService SpecificReportPaneService SpecificReportService MainController AbstractController Report Research Answer Question	Não	Emite relatórios com respostas
	Importar CSV	ExportPaneService AbstractController MainController ImportCSV	Não	Gerenciador de arquivos reconhece arquivos com extensão CSV

Cenário de Teste	Especificação dos Cenários de Teste	Classes Java Envolvidas	Falhas Detectadas	Descrição das Falhas
	Exportar CSV	ExportPaneService AbstractController MainController ExportCSV	Não	Exporta relatórios com respostas

Fonte: Autores

Os resultados apresentados no quadro 1 mostram que o software StrixRS apresenta algumas falhas que devem ser corrigidas, principalmente nos casos de testes: Gerenciar Pesquisas e Questionários. Pode-se notar que o que os testes, na verdade são um processo sem fim. O teste mostra apenas a presença de erros, não a ausência.

Com relação os testes de integração o quadro 2 mostra os resultados dos testes, novamente orientados a caso de uso.

Quadro 2 – Cenários de Testes de Integração – Nível de Pacotes Java

Cenário de Teste	Especificação dos Cenários de Teste	Módulos	Falha na Integração	Módulos Envolvidos
Casos de Uso	Coletar Respostas Obtidas On-line	com.strixrs.controller com.strixrs.data com.strixrs.model com.strixrs.service com.strixrs.staticutil java.io java.sql java.util javafx.event javafx.fxml javafx.scene javafx.scene.control javafx.scene.image javafx.scene.input javafx.scene.layout javafx.stage	Não	N/A
	Responder Questionário de Outro Autor	com.strixrs.controller com.strixrs.model com.strixrs.service com.strixrs.staticutil java.io java.sql java.util javafx.event javafx.fxml javafx.scene.control	Sim	com.strixrs.model com.strixrs.service

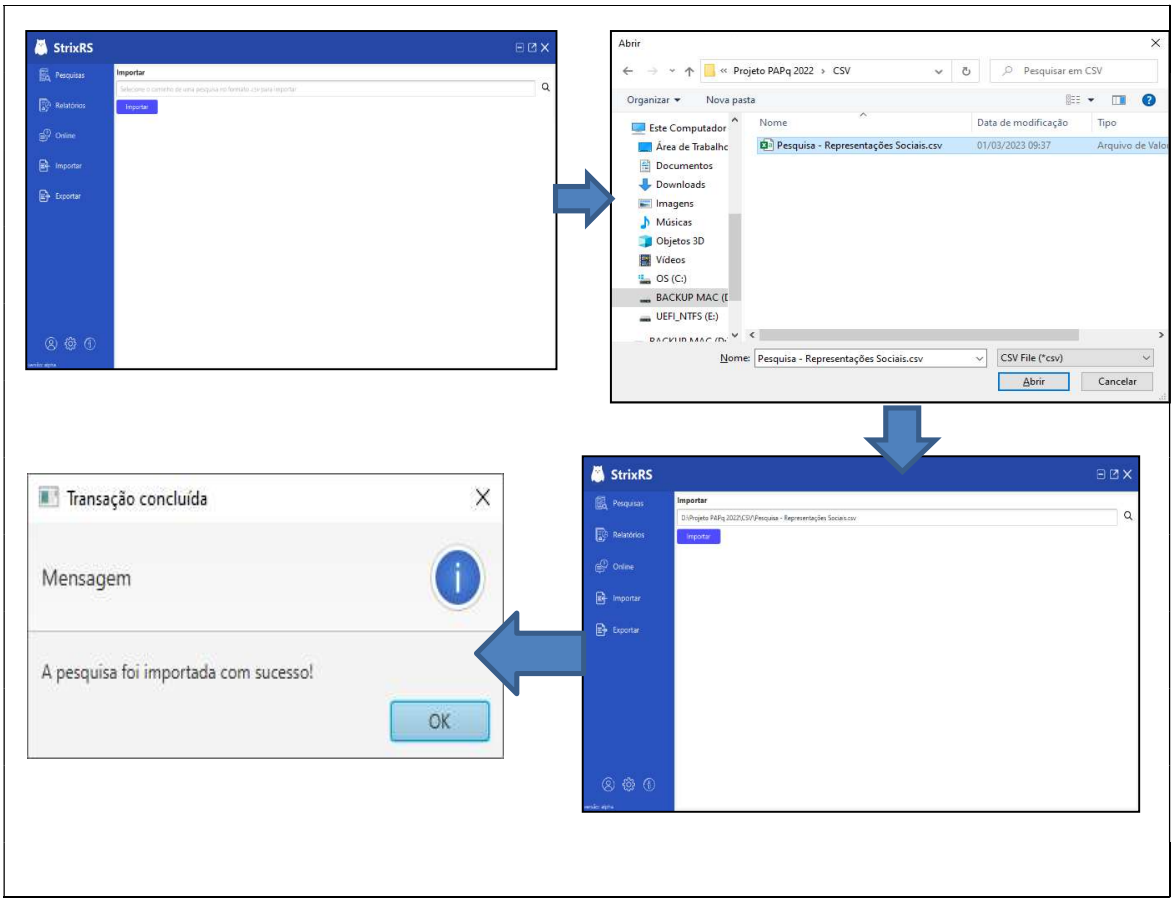
Cenário de Teste	Especificação dos Cenários de Teste	Módulos	Falha na Integração	Módulos Envolvidos
		javafx.scene.control.cell javafx.scene.image javafx.scene.input javafx.scene.layout javafx.stage		
	Entrar Off-line	com.strixrs.controller com.strixrs.staticutil java.io javafx.event javafx.fxml javafx.scene javafx.scene.control javafx.scene.image javafx.stage javafx.stage	Não	N/A
	Gerenciar Pesquisas	com.strixrs.controller com.strixrs.data com.strixrs.javafxmodfiedcontrol com.strixrs.model com.strixrs.service com.strixrs.staticutil java.io java.sql javafx.event javafx.fxml javafx.scene.control javafx.scene.control.cell javafx.scene.image javafx.scene.input javafx.scene.layout javafx.stage	Não	N/A
	Gerenciar Questionários	com.strixrs.controller com.strixrs.data com.strixrs.javafxmodfiedcontrol com.strixrs.model com.strixrs.service com.strixrs.staticutil java.io java.sql java.util javafx.event javafx.event javafx.fxml javafx.scene javafx.scene.control javafx.scene.image javafx.scene.input javafx.scene.layout javafx.stage	Sim	com.strixrs.data com.strixrs.model
	Emitir Relatórios	com.strixrs.controller com.strixrs.data com.strixrs.javafxmodfiedcontrol com.strixrs.model com.strixrs.service com.strixrs.staticutil java.io	Não	N/A

Cenário de Teste	Especificação dos Cenários de Teste	Módulos	Falha na Integração	Módulos Envolvidos
		java.util javafx.event javafx.fxml javafx.scene.control javafx.scene.image javafx.scene.input javafx.scene.layout javafx.scene.paint javafx.stage		
	Importar CSV	com.strixrs.controller com.strixrs.csv com.strixrs.data com.strixrs.model com.strixrs.service com.strixrs.staticutil java.io java.nio.charset java.nio.file java.sql java.util javafx.event javafx.fxml javafx.scene.control javafx.scene.image javafx.scene.input javafx.scene.layout javafx.stage	Não	N/A
	Exportar CSV	com.strixrs.csv java.io java.nio	Não	N/A

Fonte: Autores

A figuras 8 mostra a execução do caso de uso: IMPORTAR CSV – Importar pesquisa em formato CSV. Pode-se notar que este caso de uso foi executado com sucesso.

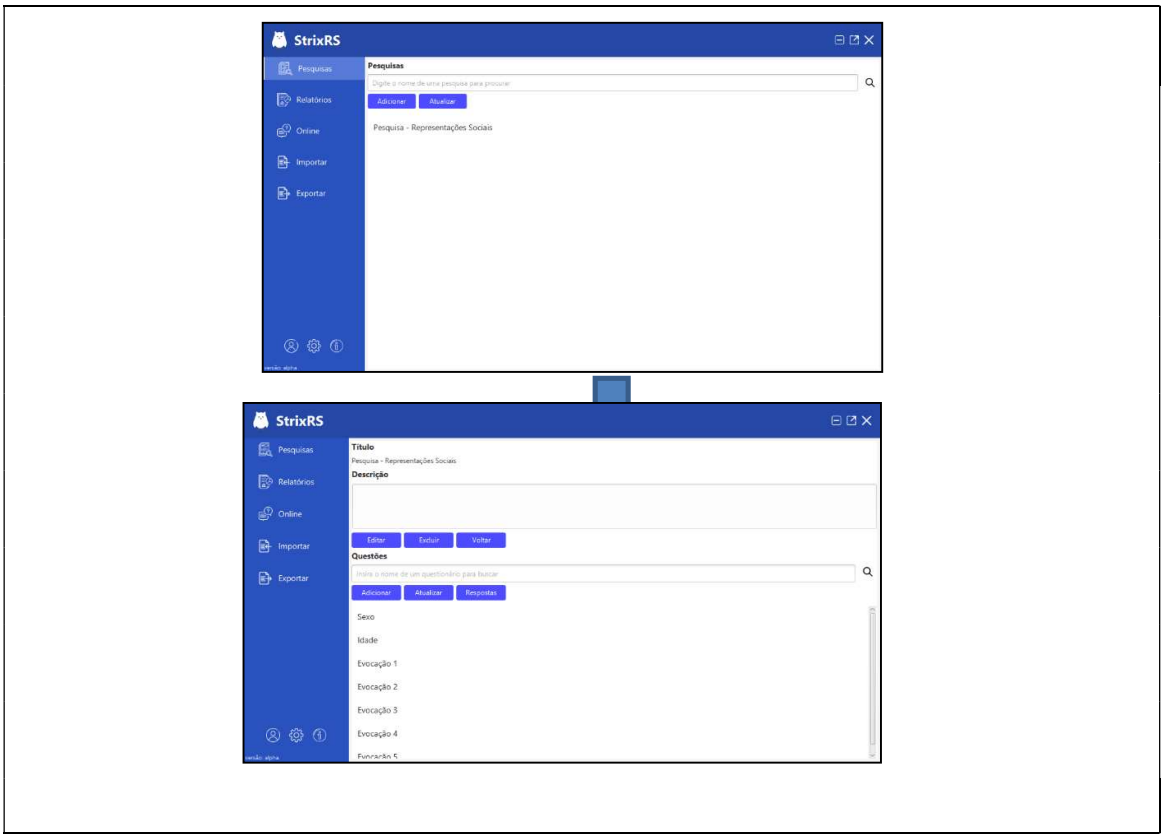
Figura 8 - Execução do Caso de Uso: IMPORTAR CSV – Importar pesquisa em formato CSV



Fonte: Autores

A figura 9 mostra a execução do caso de uso: Gerenciar Pesquisas. Pode-se notar que este caso de uso também foi executado com sucesso.

Figura 9 - Execução do Caso de Uso: Gerenciar Pesquisas



Fonte: Autores

CONSIDERAÇÕES FINAIS

Os testes exploratórios são adequados para cenários de teste específicos, como quando alguém precisa aprender sobre um produto ou aplicativo com rapidez e fornecer feedback rápido e que foi a estratégia adotada na verificação e validação do software deste projeto. Eles ajudam a avaliar a qualidade de um produto desde uma perspectiva do usuário. Porém, com relação ao método das quatro casas implementado, ainda restam alguns ajustes e correções no software.

REFERÊNCIAS

ANWAR, N.; KAR, S. Review Paper on Various Software Testing Techniques & Strategies. **Global Journal of Computer Science and Technology: C Software & Data Engineering**. Volume 19 Issue 2 Version 1.0 Year 2019 Type: Double Blind Peer Reviewed International Research Journal Publisher: Global Journals Online ISSN: 0975-4172 & Print ISSN: 0975-4350.

DE SÁ, Celso Pereira. **A construção do objeto de pesquisa em representações sociais.** EdUERJ, 1998.

GUIMARÃES, O.L. **StrixRS**: um software para apoio de pesquisas Quali-quantitativas que utilizam a abordagem estrutural da Teoria das representações sociais. 2021. 86p. (Graduação). Engenharia da Computação, Departamento de Engenharia e Sistemas, UEMG, Ituiutaba, 2022.

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS - IEEE. **IEEE Standard Glossary of Software Engineering Terminology**, IEEE Std 610.12-1990. New York, 1990. ISBN:978-0-7381-1165-0. Disponível em: <https://standards.ieee.org/ieee/610.12/855>. Acesso em: 31 de out. 2022.

ITKONEN, J; RAUTIAINEN, K. Exploratory Testing: A Multiple Case Study. **International Symposium on Empirical Software Engineering, 2005.** Noosa Heads, QLD, Australia, Year: 2005, Pages: 10 pp. DOI Bookmark: 10.1109/ISESE.2005.1541817. Disponível em < https://www.computer.org/csdl/proceedings-article/isese/2005/01541817/12OmNAJ4pfR>. Acesso em: 19 de set. 2022.

JAMIL et al. Software Testing Techniques: A Literature Review. **6th International Conference on Information and Communication Technology for The Muslim World (ICT4M)**, Jakarta, Indonesia, 2016, pp. 177-182, doi: 10.1109/ICT4M.2016.045. Disponível em < https://ieeexplore.ieee.org/document/7814898>. Acesso em: 22 de ago. 2022.

JODELET D. **As representações sociais**. Rio de Janeiro: UERJ, 2001.

KANER, C. **Testing computer software**. 1st Edition. New York: Tab Books: 1988. 313 p.

MACHADO, L. B.; ANICETO, R. de A. Núcleo central e periferia das representações sociais de ciclos de aprendizagem entre professores. **Ensaio: avaliação e políticas públicas em educação**, v. 18, n. 67, p. 345-363, 2010.

SANT'ANNA, H. C. **openEvoc**: um programa de apoio à pesquisa em Representações Sociais. In: AVELAR, L. et al. (Org.). *Psicologia Social: desafios contemporâneos*. Vitória: GM Gráfica e Editora, 2012.

AUTORES:
Erison Brito Cabral, *Graduando do Curso de Engenharia da Computação na Universidade do Estado de Minas Gerais – UEMG, Unidade Ituiutaba. E-mail : erison.1592469@discente.uemg.br*

Kátia Lopes Silva, *Docteur en Sciences Appliquées pela Université de Liège (Bélgica). Bacharel em Engenharia Química pela Universidade Federal de Uberlândia. Professor do Curso de Graduação em Engenharia Elétrica, da Universidade do Estado de Minas Gerais – UEMG, Unidade Ituiutaba. E-mail: katia.lopes@uemg.br .*

Mauro Hemerly Gazzani, *Doutor em Engenharia Elétrica pela Universidade Federal de Uberlândia. Bacharel em Engenharia Elétrica pela Universidade Federal de Uberlândia. Professor do Curso de Graduação em Engenharia Elétrica, da Universidade do Estado de Minas Gerais – UEMG, Unidade Ituiutaba. E-mail: mauro.gazzani@uemg.br .*

Hélio Oliveira Ferrari, *Doutor em Engenharia Elétrica pela Universidade Federal de Uberlândia. Bacharel em Engenharia Elétrica pela Universidade Federal de Uberlândia. Professor Adjunto na UFVJM, Unidade Janaúba. gandhiferrari@gmail.com*

Otávio Lafaiete Guimarães, *Graduado em Engenharia da Computação pela Universidade do Estado de Minas Gerais – UEMG, Unidade Ituiutaba. E-mail: otavio.1502481@discente.uemg.br*