https://eventos.utfpr.edu.br//sicite/sicite2020

Notação gráfica baseada em grafos para testes funcionais de software em GUIs

Graph-based graphical notation for functional testing of software in GUIs

Leonardo Norio Yamasaki Cruz
leonardocruz@alunos.utfpr.edu.br
Universidade Tecnológica Federal do Paraná, Cornélio Procópio, Paraná, Brasil

Cléber Gimenez Corrêa
clebergimenez@utfpr.edu.br
Universidade Tecnológica Federal do Paraná, Cornélio Procópio, Paraná, Brasil

RESUMO

Teste de *software* tem como objetivo avaliar a qualidade. Dependendo da finalidade, diferentes abordagens de teste pode ser utilizadas, como por exemplo: o teste funcional, que verifica se as funcionalidades do software atendem aos requisitos especificados; e o teste estrutural, que é uma técnica que fornece critérios para testar os caminhos lógicos de uma aplicação. Este trabalho propõe uma notação que, por meio da interface gráfica do usuário (*Graphical User Interfaces* ou GUIs) e da documentação (por exemplo, requisitos), torne possível criar e visualizar um grafo de fluxo de dados para auxiliar o testador na criação de casos de teste. O software para criação dos grafos foi implementado na ferramenta *Creately*. O artigo além de utilizar definições da literatura propõe outras representações como complementos. Também apresenta componentes e propriedades para auxiliar na criação e visualização dos grafos. Foi criado um grafo de exemplo para mostrar o resultado baseado em uma interface.

PALAVRAS-CHAVE: Software - Testes. Teoria dos grafos. Interface gráfica com o usuário (Sistemas de computação).

Recebido:
Aprovado:

ABSTRACT

Software testing aims to assess quality. Depending on the purpose, different testing approaches can be used, such as: the functional test, which verifies that if the software features meet the specified requirements; and the structural test, which is a technique and the structural test is a technique that provides criteria for testing the logical paths in the application. This paper proposes a notation that, based on the Graphical User Interface (GUIs) and documentation (for example, requirements), allows to create and visualize a data flow graph to assist the tester in the creation of test cases. The software for graphs creation was implemented in the Creately tool. In addition to using definitions from the literature, the article proposes other representations as complements. It also features components and properties to assist in the creation and visualization of graphs. An example graph was created to show the result based on an interface.

KEYWORDS: Software - Tests. Theory of graphs. Graphical user interface (Computer systems).

INTRODUÇÃO

O teste de *software* tem como objetivo a avaliação da qualidade, possibilitando melhorias ao revelar falhas, erros ou defeitos no *software* (PEZZÈ; YOUNG, 2008). A atividade de teste pode ser classificada conforme o tipo de teste, sendo por exemplo, teste funcional ou teste estrutural.

De acordo com Delamaro, Maldonado e Jino (2007) um teste funcional é composto por critérios para criar casos de teste em que são especificadas as entradas e saídas. Para testar se as saídas do programa em teste a partir das entradas usadas condizem com os objetivos especificados ou saídas dos casos de teste. No teste funcional não há conhecimento sobre a estrutura ou o código-fonte do programa. As entradas e saídas são especificadas conforme documentação, como requisitos. Por outro lado, o teste estrutural é uma técnica que fornece critérios para testar os caminhos lógicos de uma aplicação, que tem como objetivo provar, tanto o uso de variáveis, como os conjuntos de condições, estruturas de decisão e repetição (DELAMARO; MALDONADO; JINO, 2007).

No contexto de teste de *software*, grafos podem ser utilizados em determinados critérios para geração de casos de teste. Maldonado (1991) apresenta que um grafo de fluxo é um grafo dirigido que apresenta caminhos de execução de um programa em teste (do início ao fim), onde cada vértice ou nó equivale a uma sequência de comandos que são executados como blocos. Cada aresta representa a transição entre os blocos e ao percorrer os nós em sequência equivale a seguir o fluxo do programa. Esses grafos são adotados em testes estruturais, utilizando o código fonte do programa.

Delamaro, Maldonado e Jino (2007) mostram que o critério funcional de grafo de causa-efeito utiliza também permitem explorar ambiguidades e incompletude nas especificações. Cada nó dentro do grafo representa uma causa ou um efeito, ambos presentes na especificação do programa, para determinar entradas e saídas.

Diversas aplicações computacionais possuem interfaces gráficas. De acordo com Bonsiepe (1997), as interfaces gráficas são constituídas por componentes, tais como janelas, ícones, menus, botões, caixas de diálogo e teclas. Os componentes das interfaces gráficas envolvem eventos e determinadas ações, como por exemplo o clique de um botão para ser redirecionado para outra página, ou o preenchimento de um campo e envio de formulário.

Para embasar o presente trabalho, foram identificados três artigos. O primeiro é de Wen-Jing e Sheng-Hong (2008), que mostra a possibilidade da utilização de grafos de fluxo para realizar testes funcionais em *software* com interfaces gráficas e apresenta determinados componentes para a criação do grafo. O segundo também dos mesmos autores Wen-Jing e Sheng-Hong (2009) adiciona novos complementos para a montagem dos grafos de fluxo. O terceiro é o artigo de Yunming e Wei (2008) que apresenta sobre o método de caminho de definição e uso para balancear os testes funcional e estrutural.

Com base nas abordagens citadas , o presente trabalho propõe uma notação para a criação e visualização de grafos de fluxo de dados de *software* com interface gráfica baseando-se em documentação (por exemplo, requisitos) e da própria interface. Os grafos devem auxiliar na geração de casos de testes por meio da visualização de informações relevantes (componentes e propriedades).

MATERIAIS E MÉTODOS

Para a realizar o trabalho foi utilizada a ferramenta online *Creately* para criação manual dos grafos (CREATELY, 2020). Esta é uma ferramenta que permite a criação de diversos tipos de representações gráficas. Também foi usado o navegador *Google Chrome*, onde foi aberta a interface gráfica do sistema adotado de Alcântara (2019).

RESULTADOS E DISCUSSÕES

Para realizar a especificação dos grafos criados neste trabalho, foram utilizados definições e componentes de dois trabalhos anteriores (WEN-JING e SHENG-HONG, 2008; WEN-JIN e SHENG-HONG, 2009). Adicionalmente, diferentes componentes e definições foram criadas para enriquecimento dos grafos, permitindo que o testador insira informações relevantes para geração dos casos de testes.

Os elementos criados por Wen-Jin e Sheng-Hong (2008), que devem ser usados para criação dos grafos, são:

- Componente de comportamento – um menu ou botão, que ao ser acionado por um usuário dispara uma funcionalidade do programa;

- Componente de dados – entradas e saídas de dados do programa, como, por exemplo, uma mensagem para o usuário ou a submissão de um formulário em um sistema;

- Armazenamento de dados – usado para representar o armazenamento de dados, deve ser nomeado alinhado com os dados, representado com um símbolo de banco de dados genérico.

As definições utilizadas por Wen-Jing e Sheng-Hong (2009) foram:

- Fonte e final de dados – origem e destino de dados. Normalmente este complemento é representado pelo usuário;

- Processamento de dados – operação nos dados, que converte dados de entrada para dados de saída;

- Fluxo de dados – série de dados em sequência formando uma linha direcionada que contém o nome do fluxo e uma condição que o dado deve cumprir.

Os novos elementos de notação propostos neste trabalho são os seguintes:

- Componente expansível – possibilita criar um nó que pode expandir ou retrair, criando subgrafos dentro de um grafo. Tem a finalidade de melhorar a visibilidade do grafo para o testador sem deixar de cobrir os caminhos possíveis, permitindo um equilíbrio entre quantidade de informações que podem ser adicionadas e a quantidade de informações que podem ser visualizadas, representado com o símbolo de adição quando é possível expandi-lo ou com o símbolo de subtração quando é possível retraí-lo;

- Propriedade do componente – ao criar um componente vão existir propriedades que podem ser visualizadas por meio de um menu lateral, sendo essas características: o tipo de dispositivo, tipo de evento, se existir, e nome do nó, enriquecendo o grafo com informações e evitando a poluição visual do grafo;

- Componentes com interrogação – um nó com interrogação para representar o desconhecimento da origem do dado ou o desinteresse na origem no dado. Por exemplo, uma informação do banco de dados é buscada pelo sistema, porém não é interessante colocar o fluxo de busca;

- Componente de terceiros – um nó para representar os dados de origem e destino de programas ou módulos de programas de terceiros;

- Propriedade de evento – pode ser uma ação ou uma sequência de ações do usuário, por exemplo, um clique usando o dispositivo *mouse*;

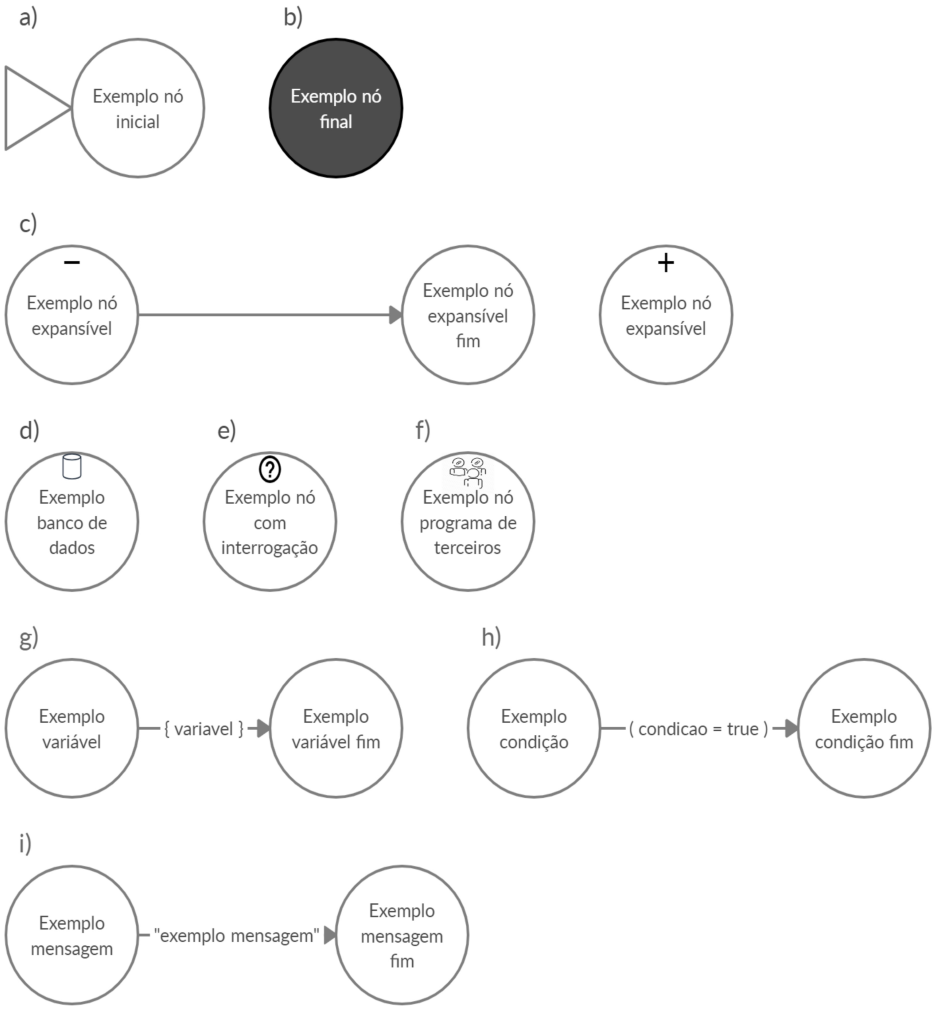
- Componente inicial e componentes finais – o componente inicial é o início do grafo e os componentes finais são o término do grafo, podendo ter mais de um final diferente. Esses componentes são representados por uma seta (componente inicial) e por cores escuras (componentes finais).

Na montagem dos grafos a formatação dos dados (por exemplo, verificar a formatação de datas, validar campos de números, validar campo sem caracteres especiais, entre outros), será representada por um único nó em vez de diversos nós, um para cada tipo de formato exigido, pois assim o grafo se torna reduzido, facilitando a visualização dos caminhos lógicos do sistema em teste.

Os grafos gerados a partir do sistema tem comandos condicionais, entrada e saída de dados presentes nas arestas, e os comandos lógicos podem ser representados nos nós. Os subgrafos são partes do grafo principal, e podem ser criadas variações dos subgrafos de uma mesma parte, que serão denominadas dimensões. Por exemplo, ao criar um subgrafo que trata do tipo de um usuário para login, ao acessar como administrador as funções que podem ser executadas são diferentes das de um usuário que acessa o sistema como usuário comum.

Para facilitar o entendimento dos grafos serão criados nós com ícones diferentes para determinadas necessidades, ícone para o nó inicial (Figura 1 (a)), ícone para os nós finais (Figura 1 (b)), ícone para nó expansível (Figura 1 (c)), ícone para nó de banco de dados (Figura 1 (d)), ícone para nó de interrogação (Figura 1 (e)) e ícone para nó de programa de terceiros (Figura 1 (f)). Ainda para facilitar o entendimento dos grafos, os dados que estão presentes nas arestas poderão ter três representações diferentes. A primeira é o dado entre chaves ({ e }), que representa uma variável do sistema (Figura 1 (g)). A segunda é o dado entre parênteses, que mostra que esse dado é uma condição verdadeira (Figura 1 (h)). Por fim, a última representação é o dado entre aspas (“ e ”), que informa que é uma mensagem para o usuário (Figura 1 (i)).

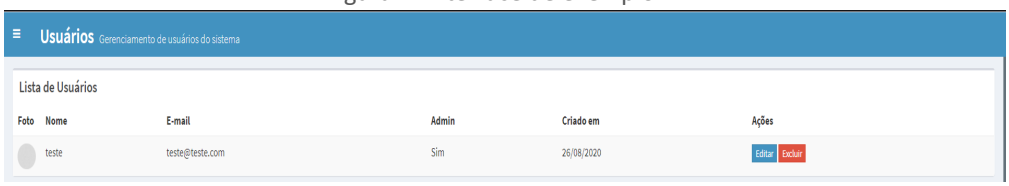
Figura 1: Exemplo de ícones e de tipos de dados.



Fonte: Autoria própria.

Foi utilizado um sistema de cadastro de usuário de Alcântara (2019) para exemplificar a criação dos grafos. No sistema é possível cadastrar um usuário, visualizar os usuários, excluir os usuários e editar usuário. Na Figura 2 é mostrada uma parte da interface gráfica na tela de visualização de usuários.

Figura 2: Interface de exemplo.



Fonte: Autoria própria.

Para criar o grafo de exemplo (Figura 3) foi usada a ferramenta *Creately*. O grafo foi gerado manualmente com base na análise da interface gráfica e documentação, contudo a documentação que estava disponível no *site* era

apenas uma breve descrição do programa. Com esse grafo é possível analisar caminhos ao definir entradas para atingir determinadas saídas, gerando os casos de teste, tais como: teste de edição de usuário, exclusão de usuário e cancelamento de edição de usuário.

Na Figura 3 é apresentado o grafo que representa o fluxo para editar ou excluir usuários previamente cadastrados. Na Figura 2 (interface gráfica), o fluxo que o grafo representa são os dois botões da parte direita. O primeiro é o botão de editar usuário, enquanto que o segundo é para excluir.

Com a criação dos grafos para o teste funcional espera-se que o testador tenha uma melhor visualização dos caminhos que deve percorrer para uma melhor geração dos casos de teste, ressaltando que não há acesso ao código fonte e procura-se inferir esses caminhos. Contudo a criação dos grafos em primeira instância dependerá inteiramente do testador, e com isso sua experiência tem influência direta no uso da notação e aplicação dos testes. Os requisitos funcionais do *software* são indispensáveis para que se possa construir o grafo que representa os casos de teste.

Por fim, recomenda-se que o testador deve ter consciência no uso da notação e na medida do possível, sejam elaborados grafos com reduzida poluição visual. Para isso, a notação oferece o recurso de “nó expansível”. Esse é um componente que permite o testador expandir e retrair uma parte do grafo, representados pelos ícones ”-” e “+”, respectivamente.

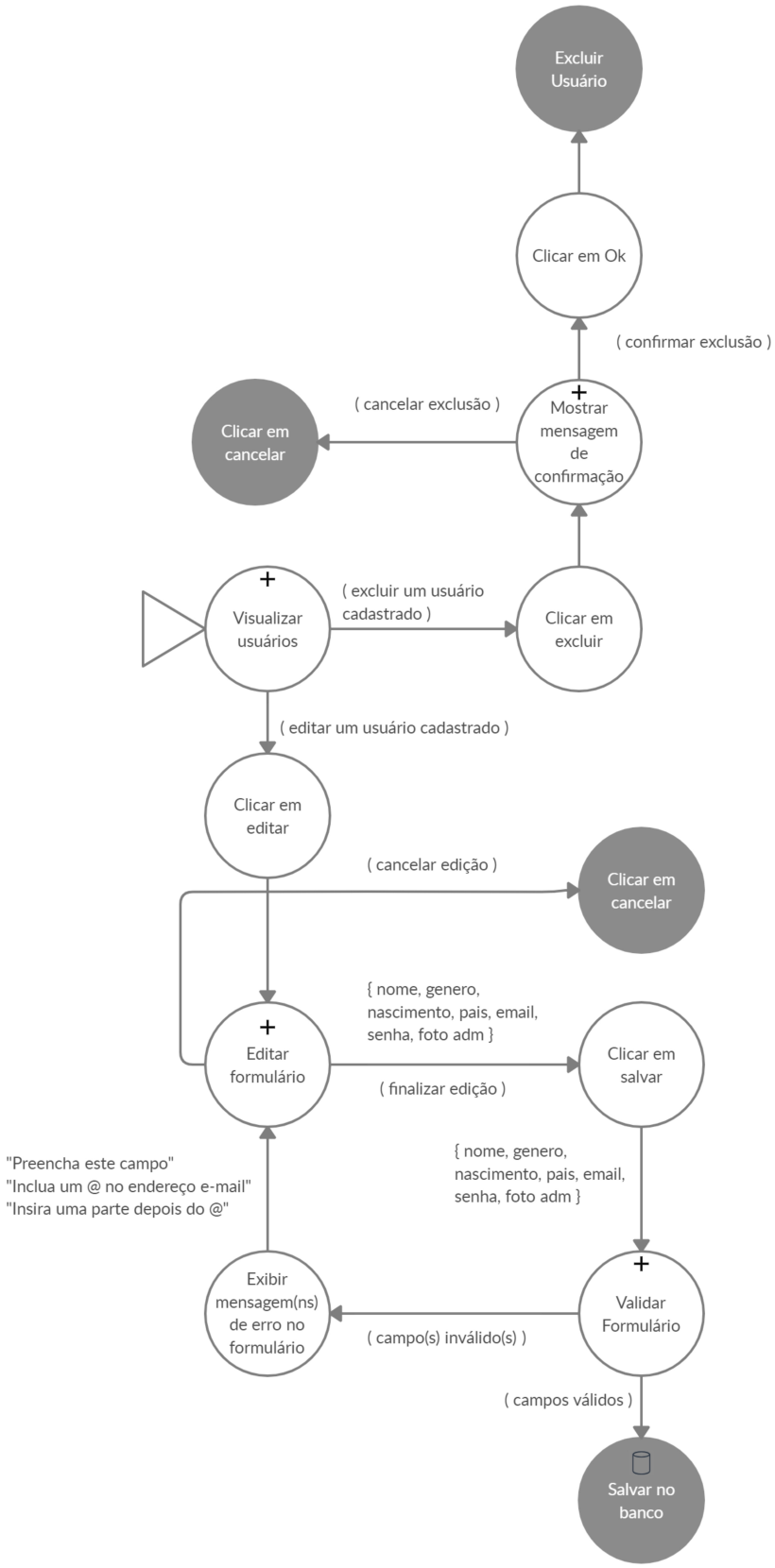
CONCLUSÕES

Este trabalho propôs uma notação para auxiliar o testador na criação e visualização de grafos de fluxos de dados para geração de casos de testes. A notação deve facilitar a criação manual dos grafos, arrastando e criando componentes, bem como as propriedades desses componentes. Para tornar mais completo o projeto, além de componentes e definições utilizadas da literatura, foram criados outros componentes e definições. A notação pode ser útil para complementar a execução de testes funcionais. Trabalhos futuros envolvem a implementação de uma ferramenta para uso da notação, incluindo a participação de testadores, para comparação com outras abordagens funcionais, como o particionamento por classes de equivalência.

AGRADECIMENTO

Os autores agradecem à Universidade Tecnológica Federal do Paraná, pela participação no Programa de Voluntariado em Iniciação Científica e Tecnológica (Edital PROPPG 02/2019 - PIVIC) do acadêmico Leonardo Norio Yamasaki Cruz.

Figura 3: Grafo Exemplo.



Fonte: Autoria própria.

REFERÊNCIAS

ALCÂNTARA, Bruno. **Cadastro-usuarios-javascript.** 2019. Disponível em: https://github.com/allcantara/cadastro-usuarios-javascript. Acesso em: 15 agosto 2020.

AMMANN, Paul; OFFUTT, Jeff. **Introduction to Software Testing.** United States of America: Cambridge University Press, New York, 2008.

BONSIEPE, Gui. **Design: do material ao digital.** Trad. Cláudio Dutra. – Florianópolis: FIESC/IEL, 1997.

CREATELY. **Creately.** 2020. Disponível em: https://creately.com/. Acesso em: 05 junho 2020.

DELAMARO, Márcio Eduardo; MALDONADO, José Carlos; JINO, Mario. **Introdução ao Teste de Software.** Rio de Janeiro: Campus, 2007.

MALDONADO, José Carlos. **Critérios potenciais usos: uma contribuição ao teste estrutural de Software.** 1991. [261]f. Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica, Campinas, SP. Disponível em: http://www.repositorio.unicamp.br/handle/REPOSIP/260444. Acesso em: 1 junho 2019.

MYERS, Glenford J.; BADGETT, Tom; SANDLER, Corey. **The Art of Software Testing.** Nova Jersey: John Wiley & Sons, Inc., 3. ed.,2012.

PEZZÈ, Mauro; YOUNG, Michal. **Teste e Análise de Software: Processos, Princípios e Técnicas.** Porto Alegre: Bookman, 2008.

WEN-JING, Cao; SHENG-HONG, Xu. **A Software Function Testing Method Based on Data Flow Graph.** IEEE, 2008. In: INTERNATIONAL SYMPOSIUM ON INFORMATION SCIENCE AND ENGINEERING, 2008, Shanghai, China. Disponível em: https://ieeexplore.ieee.org/document/4732336. Acesso em: 20 maio 2019.

WEN-JING, Cao; SHENG-HONG, Xu. **A Function Testing Method for Interactive Software.** IEEE, 2010. In: Second International Workshop on Computer Science and Engineering, 2009, Qingdao, China. Disponível em: https://ieeexplore.ieee.org/document/5403186. Acesso em: 18 abril 2020.

YUNMING, Pu; WEI, Wang. **A Balancing Model between Structural Testing and Functional Testing.** 2008. In: International Conference on Computer Science and Software Engineering, 2008, Hubei, China. Disponível em: https://ieeexplore.ieee.org/document/4722151. Acesso em: 28 abril 2020.