HAPPYPATH: UMA PROPOSTA DE AUTOMAÇÃO DE TESTES UTILIZANDO INTELIGÊNCIA ARTIFICIAL

HAPPYPATH: A PROPOSAL FOR TEST AUTOMATION USING ARTIFICIAL INTELLIGENCE

Fernanda Pereira dos Santos Sousa
https://orcid.org/0000-0002-4604-7336
Centro Paula Souza – Fatec Sorocaba/SP
fernanda.pss@live.com

Orientador: profa. Dra. Maria das Graças J. M. Tomazela
https://orcid.org/0000-0002-5471-2658
Centro Paula Souza – Fatec Sorocaba/SP
graca.tomazela@fatec.sp.gov.br

RESUMO: As novas metodologias de desenvolvimento de *software, como as metodologias ágeis,* vêm ganhando cada dia mais força no meio empresarial, uma vez que são altamente gerenciáveis, proporcionam uma visibilidade maior do progresso realizado, possibilitando uma identificação e resolução precoce de problemas e por serem capazes de agregar maior valor ao produto a cada entrega, trazendo sempre novas funcionalidades ao sistema do cliente. Entretanto, isso exige um maior esforço de teste, pois para garantir o correto funcionamento das novas funcionalidades entregues, a correta integração entre os módulos e o pleno funcionamento da aplicação, além dos testes unitários e testes de funcionalidade, testes de regressão são necessários em cada entrega, tornando o teste repetitivo e mais extenso em cada iteração. Assim, o objetivo deste trabalho foi propor uma nova metodologia de teste que utiliza mecanismos inteligentes para auxiliar a geração de casos de teste automatizados. Por meio de uma proposta teórica, elaborou-se uma metodologia capaz de unir fragmentos de código de testes automatizados, criados na linguagem de programação da preferência do usuário, e empregar algoritmos inteligentes que sejam capazes de traçar passos implícitos no Caso de Teste ao analisar o conteúdo já existente em uma base de conhecimento. Com a metodologia elaborada neste trabalho será aberta a possibilidade de desenvolvimento de um *software* que implemente um mecanismo de teste automatizado, baseado em Inteligência Artificial para assim auxiliar o testador e sua equipe na criação e utilização de Casos de testes automatizados, economizando o tempo despendido nas tarefas de teste e, consequentemente, o dinheiro gasto nesta etapa do desenvolvimento.

ABSTRACT: New software development methodologies, such as agile methodologies, are increasingly gaining strength in the business environment, since they are highly manageable, provide greater visibility of the progress made, enabling early identification and resolution of problems and for being able to add greater value to the product with each delivery, always bringing new features to the customer system. However, this requires a greater testing effort, as to ensure the correct functioning of the new functionalities delivered, the correct integration between modules and full application operation, in addition to unit tests and functionality tests, regression tests are required for each delivery, making the test repetitive and more extensive each iteration. Thus, the goal of this undergraduate thesis was to propose a new test methodology that uses intelligent mechanisms to assist the generation of automated test cases. Through a theoretical proposal, a methodology was

developed, capable of assembling automated test code fragments, created in the user's preference programming language and applying intelligent algorithms that can trace implicit steps in the Test Case by analyzing existing content in a knowledge base. With the methodology created in this undergraduate thesis, it will be possible to develop software for the implementation of an automated test mechanism, based on Artificial Intelligence to assist the tester and her/his team in the creation and use of automated test cases, saving the time spent on testing tasks and, consequently, the money spent at this stage of development.

PALAVRAS-CHAVE: Testes automatizados. Inteligência artificial. Casos de teste. Regras de produção.

KEYWORDS: Artificial intelligence. Automated testing. Test cases. Production rules.

1 INTRODUÇÃO

O cenário empresarial e tecnológico atual mostra uma grande quantidade de aplicativos e aplicações web sendo requisitadas e liberadas para consumo diariamente, isso gera uma grande carga de trabalho, não só para desenvolvedores como para testadores, uma vez que as atividades de testes costumam ser mais morosas que o próprio desenvolvimento.

Ao observar as metodologias usadas no desenvolvimento, percebe-se, por exemplo, que na metodologia Cascata a maior parte do teste é executada ao final de todo o período de desenvolvimento, gerando uma altíssima carga de testes. Já na metodologia Ágil o teste é feito em pequenos blocos, assim como o desenvolvimento, diminuindo sua carga e aumentando sua frequência, exigindo um maior número de testes de integração, repetindo várias e várias vezes o mesmo tipo de teste (SOMMERVILLE, 2011).

Com essa repetição, surgiram os mecanismos de automação de testes, que, ao utilizar frameworks próprios e linguagens de programação já usadas por desenvolvedores, são capazes de executar diversas vezes e automaticamente um mesmo Caso de Teste (CT). Porém, essa automação deve ser inteiramente construída pelo testador para cada CT criado, o que pode levar muito tempo e exigir bastante esforço.

Dessa maneira, este trabalho teve como objetivo propor uma metodologia de criação de casos de testes automatizados utilizando um mecanismo de Inteligência Artificial.

2 METODOLOGIA

Com base na definição de Gil (2008), essa pesquisa pode ser classificada, quanto à sua natureza, como uma pesquisa aplicada e, quanto a seu propósito, com uma pesquisa exploratória, dado que sua principal finalidade é esclarecer conceitos e ideias, tendo em

vista a formulação de hipóteses pesquisáveis e aplicáveis de maneira prática em estudos posteriores. Sua abordagem pode ser definida como qualitativa utilizando procedimentos de revisão de literatura especializada, tanto nacional quanto estrangeira, sendo utilizados livros, revistas, artigos científicos, dissertações de mestrado e teses de doutorado.

Este experimento de pesquisa propôs uma metodologia focada na possibilidade de criação de códigos de automação de testes a partir de casos de teste padronizados, atribuindo linhas de código já cadastradas a passos semelhantes de diversos casos de teste. Dessa forma o testador pode cadastrar cada tipo de ação apenas uma vez na base de conhecimento e reutilizá-la em todos os testes subsequentes.

Para o desenvolvimento de tal metodologia é necessário o uso de tecnologias de padronização de escrita de casos de teste, para deixar as informações mais completas e uniformes para análise, ferramentas de inteligência artificial para o reconhecimento de passos necessários para complementar a automação do teste, além de uma base de conhecimento para manter e consultar os passos já criados em cada CT.

O paradigma de IA proposto foi o uso de Regras de Produção utilizando frameworks e linguagens de programação voltadas ou adaptáveis para automação de testes, sendo essas livres para uso como convier ao usuário.

2 DESENVOLVIMENTO

Neste trabalho utilizou-se de conhecimentos da área de teste de software e utilização de regras de produção para sua concepção.

As técnicas de teste de software podem ser definidas de algumas maneiras segundo diferentes autores. Segundo Luo (2001), teste de software é uma área muito abrangente, que envolve várias outras áreas, técnicas e não-técnicas, como especificação, design e implementação, manutenção, processamento e gerência de problemas em engenharia de software. Segundo Miller (1981) o objetivo geral dos testes é afirmar a qualidade dos sistemas de software, exercitando-os sistematicamente em circunstâncias cuidadosamente controladas. Já Myers, Badgett e Sandler (2012) afirmam que o teste deve ter como maior intenção encontrar erros e esse só é bem-sucedido quando os encontra.

Já as regras de produção são uma representação do conhecimento baseada nas propostas do matemático Emil Post (1943), como cita Heinzle (1995), que apresentava os sistemas de produção como um modelo computacional para a solução de problemas. Heinzle (1995) explica que o termo “sistema de produção” pode ser usado para definir

sistemas constituídos de uma coleção de regras para descrever condições e ações, regras essas que são armazenadas como um conjunto de declarações SE-ENTÃO

SE < premissas > ENTÃO <conclusão>

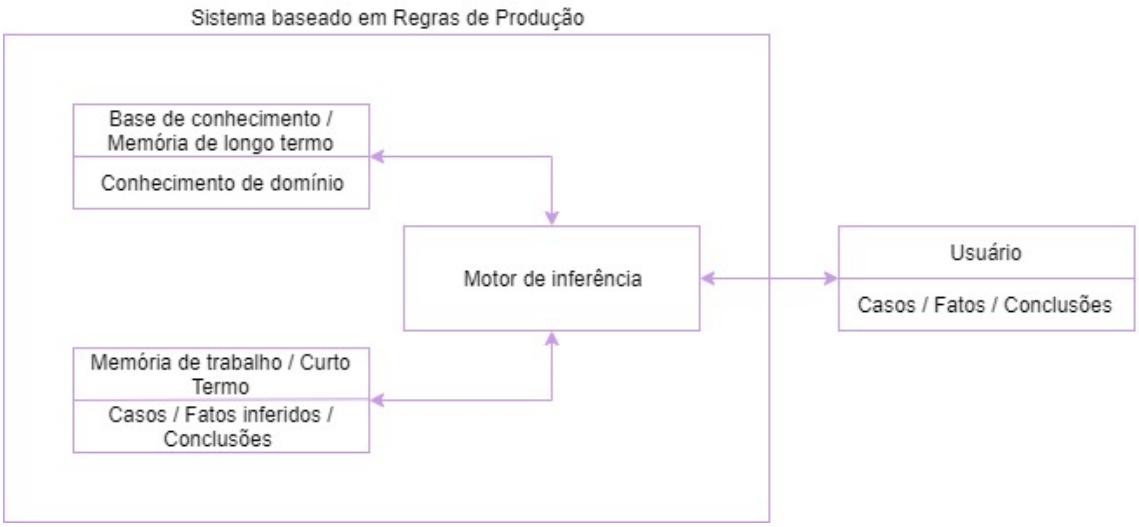
O corpo da regra delimita-se pela palavra SE, localizada à esquerda, antecedendo a premissa ou condição e deve ser avaliada em relação à base de conhecimentos como um todo.

A base de conhecimentos pode ser definida, segundo Heinzle (1995), como um depósito de conhecimento acerca de um determinado assunto, inseridos por um especialista por meio de um sistema e armazenado por esse de maneira própria, permitindo posterior processamento e inferência.

Para a localização das regras é necessário o uso de um mecanismo de busca, sendo a mais comumente usada a árvore de decisão que, segundo Monard e Baranaukas (2003), é uma estrutura de dados definida recursivamente, sendo um nó folha correspondente a uma classe ou um nó de decisão.

O funcionamento padrão de um sistema que utiliza regras de produção pode ser observado na Figura 1.

Figura 1 - Funcionamento de um Sistema Baseado em Regras de Produção



Fonte: Adaptado de Russell e Norvig (2003)

A metodologia proposta tem como objetivo auxiliar os testadores durante a criação de testes automatizados ao armazenar pedaços de código reutilizáveis e organizá-los de maneira a atender as necessidades de cada CT.

A proposta é composta por cinco elementos: 1 - utilização de CT padronizado; 2 - codificação referente a cada passo desse CT; 3 - utilização de um banco de dados para armazenar a Base de Conhecimento, ou seja, as informações referentes aos CTs e automações inseridas no sistema; 4 - utilização de uma memória de trabalho, ou de curto termo, usada pela IA durante o processo de inferência e; 5 - utilização de um mecanismo de IA capaz de analisar os passos existentes na base de conhecimentos e propor uma codificação para cobrir passos implícitos anteriores aos informados.

Com o intuito de ilustrar o funcionamento da metodologia foi sugerida uma aplicação exemplo em que cada CT deve possuir um “Ponto de partida” – delimitando a tela em que o CT deve iniciar – e um ou mais passos a serem executados. O “Ponto de partida” deve ser selecionado em um elemento *dropdown* e os passos serão adicionados um a um registrando uma “Condição inicial”, uma “Ação” e um “Resultado esperado / Conclusão” selecionando-os também em elementos *dropdown*. Ao selecionar a função desejada (condição inicial, ação ou resultado esperado) um número de caixas de texto igual ao de parâmetros necessários serão exibidos e não devem ser deixados em branco.

Em seu primeiro uso o testador deverá alimentar a aplicação com as funções desejadas incluindo seu Nome, Número de parâmetros e Código da função.

A técnica Regras de Produção será utilizada na criação dos testes automatizados para traçar um caminho partindo da Página Inicial até o Ponto de Partida definido em um CT, para isso serão verificados todos os Casos de Teste observando suas Condições Iniciais e Resultados Esperados. Considerando um cenário em que haja Casos de Teste em todas as páginas da aplicação, uma espécie de mapeamento será produzida, partindo do Ponto de Partida desejado e “fazendo o caminho inverso” até chegar na página inicial.

4 RESULTADOS OBTIDOS

É possível observar que a aplicação poderia usar as Regras de Produção da seguinte forma: Cada passo dos Casos de Testes será considerado como uma regra (Condição Inicial = SE, Ação = ENTÃO, Resultado Esperado = Conclusão / Verdade), ao iniciar no Ponto de Partida a aplicação considera-o como verdade e busca qual Condição Inicial levou até àquele resultado.

Para exemplificar o funcionamento é possível considerar os seguintes Casos de Teste registrados na aplicação: O CT “Abas Principais” em que um dos passos leva da página inicial (Home) à página Administração, o CT “Opções da Página Administração” no qual um dos passos leva da página Administração à página Requisição de acesso e o CT “Criação

de Requisição de Acesso" em que um dos passos tem como Condição Inicial a página Requisição de acesso.

Tendo em vista que o CT escolhido para conversão em código automatizado tem como Ponto de Partida a página "Requisição de acesso" (Caso de Teste Criação de Requisição de Acesso), então "Página [Requisição de acesso] visível" é considerada verdade e procura-se a Condição Inicial que leva a esse resultado, encontrando assim a Condição Inicial "Página [Administração] visível" (Caso de Teste Opções da Página Administração). O código relacionado a esse passo é reservado na memória de trabalho, a Condição Inicial é comparada com a página inicial buscada e não sendo igual o processo continua. Se inicia a busca pela Condição Inicial que tem como resultado esperado "Página [Administração] visível" encontrando assim a Condição Inicial "Página [Home] visível" (Caso de Teste: Abas Principais), o código relacionado a esse passo é reservado acima do anterior e ao comparar a página encontrada com a página inicial buscada se tem um resultado positivo e o processo é encerrado.

Nesse processo, dois trechos de código foram adicionados antes do começo do CT escolhido. Dessa forma, o CT convertido terá a estrutura apresentada no Quadro 1:

Quadro 1 - Caso de Teste Criação de Requisição de Acesso Gerado Após Ação da Inteligência Artificial

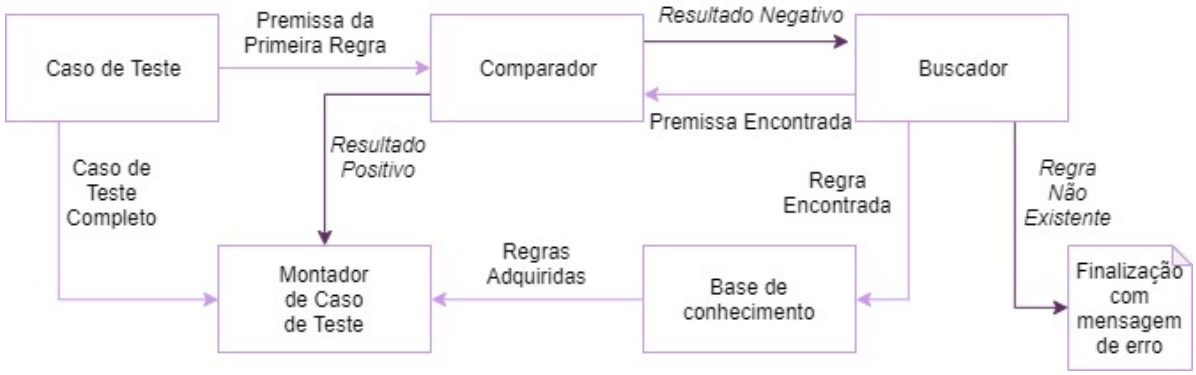
Condição Inicial	Ação	Resultado Esperado
Página [Home] visível	Clicar em aba [Administração]	Página [Administração] visível
Página [Administração] visível	Clicar em [Requisição de acesso]	Página [Requisição de acesso] visível
Página [Requisição de acesso] visível	Clicar em [Nova requisição de acesso]	Página [Criar requisição de acesso] visível
Página [Criar requisição de acesso] visível	Preencha o campo [País] com [Brasil]	Campo [País] preenchido com [Brasil]
...	...	...

Fonte: A autora

Esse fluxo de funcionamento pode ser verificado na Figura 2 , na qual a premissa da Primeira Regra do CT é mandada para um Comparador que verifica se a premissa recebida é igual à página Inicial configurada, em caso negativo essa premissa é enviada para o Buscador que procurará na Base de Conhecimento uma regra que possua como resultado esperado a mesma premissa recebida, caso o Buscador encontre uma regra que satisfaça a busca, esse a salva na memória de trabalho e devolve a premissa encontrada para o

Comparador, sendo essa premissa equivalente ao Home, a busca é finalizada, as regras encontradas pelo Buscador são enviadas para o Montador de Caso de Teste e as regras do CT inicial são anexadas a elas gerando o CT final. Caso o Buscador não seja capaz de encontrar uma regra que satisfaça a busca então a operação é finalizada apresentando um erro.

Figura 2 - Funcionamento da Inteligência Artificial na Metodologia Proposta



Fonte: A autora

Para possibilitar esse funcionamento torna-se necessário o uso de uma árvore de busca utilizando o encadeamento regressivo, ou *backward*, na busca de regras, dado que a regra analisada terá como foco sua conclusão e esta será procurada entre as premissas das demais regras.

5 CONSIDERAÇÕES FINAIS

A metodologia de teste proposta neste trabalho se apresenta por meio de uma interface com o usuário, a aplicação exemplo HappyPath, que pode ser utilizada em testes longos e repetitivos, deixando o processo mais ágil e intuitivo. Sua relevância se dá pelo uso de uma interface amigável que permite armazenar funções comumente usadas na construção de automações de teste dando a liberdade ao testador de usar a linguagem de programação que melhor lhe convier, dado que não há limitações na metodologia quanto a esse aspecto.

É esperado que essa metodologia possa ser amplamente usada em projetos de desenvolvimento de software que exijam testes longos e/ou repetitivos, potencializando assim a capacidade e efetividade do testador ou da equipe de testes.

REFERÊNCIAS

GIL, Antonio Carlos. **Métodos e Técnicas de Pesquisa Social**. 6. ed. São Paulo: Editora Atlas S.a., 2008. 220 p. Disponível em: <http://197.249.65.74:8080/biblioteca/bitstream/123456789/707/1/M%C3%A9todos%20de%20Pesquisa%20Social.pdf>. Acesso em: 19 nov. 2019.

HEINZLE, Roberto. **PROTÓTIPO DE UMA FERRAMENTA PARA CRIAÇÃO DE SISTEMAS ESPECIALISTAS BASEADOS EM REGRAS DE PRODUÇÃO.** 1995. 161 f. Dissertação (Mestrado) - Curso de Engenharia, Departamento de Engenharia de Produção e Sistemas, Universidade Federal de Santa Catarina, Florianópolis, 1995. Cap. 2. Disponível em: <https://repositorio.ufsc.br/bitstream/handle/123456789/157957/100877.pdf?sequence=1&isAllowed=y>. Acesso em: 16 mar. 2019.

LUO, Lu. **Software Testing Techniques:** Technology Maturation and Research Strategy. Pittsburgh: Institute for Software Research International Carnegie Mellon University, 2001. 19 p. Disponível em: <http://www.cs.cmu.edu/~luluo/Courses/17939Report.pdf>. Acesso em: 27 out. 2019.

MILLER, Edward F. **Introduction to Software Testing Technology**, Tutorial: Software Testing & Validation Techniques, Second Edition, IEEE Catalog No. EHO 180-0, pp. 4-16.

MONARD, Maria Carolina; BARANAUKAS, José Augusto. “Indução de regras e árvores de decisão”. **Sistemas Inteligentes Para Engenharia**. Rezende: So Editora Manole Ltda., 2003. Disponível em: <http://dcm.ffclrp.usp.br/~augusto/publications/2003-sistemas-inteligentes-cap5.pdf>. Acesso em: 16 mar. 2019.

MYERS, Glenford J.; BADGETT, Tom; SANDLER, Corey. **The Art Software Testing.** 3. ed. New Jersey: John Wiley & Sons, 2012. 256 p.

RUSSELL, Stuart J.; NORVIG, Peter. **Artificial Intelligence:** A Modern Approach. 2. ed. New Jersey: Pearson Education, Inc., Upper Saddle River, 2003. 1112 p.

SOMMERVILLE, Ian. Agile software development. In: SOMMERVILLE, Ian. **SOFTWARE ENGINEERING.** 9. ed. Boston: Addison-wesley, 2011. Cap. 3. p. 56-62.