Automação de testes com aplicação inteligente

Pedro H. da S. Pereira[1]

[1] Departamento Acadêmico de Computação – Universidade Tecnológica Federal do Paraná (UTFPR)[)]

pedropereira.2000@alunos.utfpr.edu.br

Abstract. *This work will seek to raise evidence to validate the construction of a tool to support the construction of tests, with a set of samples previously selected in the test creation world, where we want to infer what are the most recurrent issues in the construction of tests for software, which will contribute to the analysis of data collected in a quantitative way, so that we will build a vision to enable the realization and implementation of a pilot case in one of the chosen samples, where the pilot case will offer a tool to assist professionals in the construction of tests for software, in order to validate the efficiency of the tool and its adherence to the developer process.*

Resumo. *Esse trabalho buscará levantar provas para validar a construção de uma ferramenta para apoio na construção de testes, com um conjunto de amostras previamente selecionadas no mundo de criação de testes, onde queremos inferir quais são as questões mais recorrentes na construção de testes para software, que contribuirá com a análise dos dados coletados de forma quantitativa, de forma que construiremos uma visão para viabilizar a realização e concretização de um caso piloto em uma das amostras escolhidas, onde o caso piloto oferecerá a uma ferramenta para auxiliar os profissionais na construção de testes para software, com o intuito de validar a eficiência da ferramenta e sua adesão ao processo do desenvolvedor.*

1. Introdução

Considerando que no planejamento da construção de um software é recorrente e indicado existir uma etapa ou fase focada somente em realizar testes com o software desenvolvido, podemos levantar a hipótese que a realização de testes bem construídos e bem aplicados pode aumentar a qualidade do produto e até diminuir gastos com manutenção. Pois se olharmos para um modelo de produção simples como o Cascada que foi desenvolvido pelo Herbert D. Benington, que depois de realizar análise, planejamento e desenvolvimento, temos uma etapa para testar o que foi desenvolvido e sabemos que parte do esforço da equipe tem que ser direcionado para tal.

Entretanto quando tratamos da construção de testes devemos salientar que dependerá de quem está desenvolvendo, pois nos dias atuais por mais que tenhamos modelos de testes que podem ser aplicados, ainda fica a cargo da análise do desenvolvedor escolher qual teste ele quer aplicar.

O que pode nos levar a pergunta, a construção de testes são sujeitos às escolhas do desenvolvedor, e a resposta é sim, por mais que exista testes focados em explorar a aplicação para encontrar brechas ou blocos onde um tipo específico de teste se aplica

melhor, não são todos os profissionais que querem investir seu tempo em testes exploratórios, e o mesmo vale para as empresas.

Com essas questões como tópicos, considerando que fosse possível criar uma ferramenta capaz de reconhecer padrões nos blocos de códigos, para assim indicar a um desenvolvedor responsável que pode haver um bug, ou até indicar para o mesmo que ele pode realizar um teste de caixa branca para cobrir aquele bug, seria um cenário muito mais favorável para as empresas e para os profissionais.

Entendendo sobre essa ferramenta proposta desejamos que ela possa atingir um assertividade para identificar bug, ou blocos de códigos que podem conter falhas e quais testes podem ser aplicados com 95% de certeza, desejamos também que a ferramenta possa se aprimorar a partir da ocorrência de novos bugs que a mesma não foi capaz de detectar, mas que o desenvolvedor poderá indicar para a ferramenta.

2. Problema

Dentro do aspecto da engenharia ou construção de um software há uma fase, ou período de tempo que é exclusivamente separada para que sejam construídos casos e planos de testes. Com essa fase ou período de tempo como foco, nos deparamos com um meio muito volátil podendo haver altas taxas de tempo para a construção de um caso de teste, baixa quantidade na detecção de falhas ou bugs, assim como empresas que conseguem resultados bons.

Mas se olharmos diretamente para o fato de que planejar e executar um determinado número de casos de teste ainda assim não é garantido com 100% de certeza que todas as falhas e bugs foram encontrados, pois a abrangência dos testes realizados depende do conhecimento de quem os produziu e atualmente que produz e executa os testes são desenvolvedores.

Por conseguinte, por mais que o desenvolvedor tenha conhecimento sobre a regra de negócio do produto, não conseguiria pensar em todas as possibilidades de escolha para um caso que pode ser construído no bloco de código e pode acabar se limitando a cenários que já lidou ou que consegue enxergar.

3. Justificativa

Considerando que os teste geralmente são referentes a um conhecimento prévio do desenvolvedor, podemos nos questionar se não existiria a possibilidade de automatizar este processo, ou no mínimo oferecer uma ferramenta para auxiliar o desenvolvedor a ter uma maior abrangência na construção de testes mais eficientes e na localização de bugs ou falhas no sistema.

Com essa ideia queremos, oferecer uma solução que se baseia na construção de uma máquina inteligente que se propõe a identificar padrões nos blocos de códigos recebidos para indicar falhas, bugs, falta de casos de teste para uma parte específica do código já propondo um caso de teste para cobrir a brecha, tudo a partir de um conhecimento prévio que a mesma adquiriu e tem como base.

Por conseguinte levando em consideração que o desenvolvedor não consegue visualizar todas as opções na construção dos testes, o que queremos mostrar é que seria diferente

com a utilização de uma máquina ou mais especificamente de uma inteligência programada para identificar padrões de bugs que já foram detectados e que podem ser assimilados.

4. Objetivos

Na estruturação do trabalho definiremos quais serão os objetivos os quais queremos compreender na conclusão e realização do trabalho, sendo eles um objetivo mais geral sobre o contexto de testes e objetivos específicos que vão compreender alguns itens que sabemos que poderiam ser benéficos para a aplicação.

4.1. Objetivos Gerais

O trabalho tem como objetivo levantar provas para a viabilidade de uma máquina capaz de identificar padrões na construção de um software a partir de outros bugs que já haviam sido encontrados ou a partir de bugs que já são tidos como comuns pela comunidade de testes.

4.2. Objetivos Específicos

Como objetivos mais específicos queremos atingir ao menos uma acurácia (assertividade da aplicação para indicar bug e falhas) de 95% com o funcionamento da nossa proposta, desejamos também que a aplicação possa se aprimorar com a ocorrência e detecção de novos bugs e falhas, possa gerar gráficos com informações que sejam benéficas para que a empresa possa aprimorar seus processos.

5. Procedimento Metodológico

Para a realização deste trabalho, fará-se necessário a concretização de uma entrevista por meio da técnica de coleta de dados intitulada padronizada ou estruturada, que diz respeito possuir um roteiro previamente estabelecido, assim colocando em enfoque questões que desejam ser respondidas no que diz respeito a automatização de testes (MORESI, 2003).

Esta entrevista será aplicada com amostras intencionais onde foram escolhidos casos para a amostra que representem o “bom julgamento” da população/universo(MORESI, 2003), os selecionados foram um grupo de empresas área de teste de software e profissionais dessas empresas, sendo 4 empresas e 20 profissionais do Estado de São Paulo.

O roteiro formulado fará uso dos recursos formulários e enquetes, onde os participantes responderam às questões propostas, para que possamos levantar dados referentes a questões que são mais recorrentes na implementação e execução de testes.

A tabulação dos dados coletados será de forma quantitativa, onde buscaremos elencar a partir de tópicos já mostrados em outros trabalhos como métricas recorrentes e ou as mais buscadas. Com a análise dos dados coletados queremos identificar se é possível afirmar que há uma necessidade de se aplicar a automação de testes dentro dos processos de produção aplicados pelas empresas, fazendo uso de um inteligência artificial, para auxiliar os profissionais no processo de construção e execução dos testes, o que também proporciona benefícios em questão de tempo e custos para a empresa.

Para complementar a metodologia pressuposta, será feita uma pesquisa bibliográfica em periódicos nacionais e internacionais sobre ferramentas, métodos e técnicas para a aplicação e utilização de inteligência artificial na construção ou execução de um teste bem como novas tecnologias ou propostas que forneçam um caminho confiável para alcançar a automação do processo.

Por fim, será escolhida uma empresa para receber um projeto piloto no qual a aplicação será desenvolvida, um padrão para a coleta e extração dos testes será definido, o principal objetivo deste piloto é para que possamos também identificar pontos de fragilidades e oportunidades.

6. Aporte Teórico

Tendo como objeto a construção de um software ou a engenharia por trás do mesmo é levado em conta uma etapa que se prova a eficiência e funcionalidade do mesmo, sendo esta o teste, que para ser eficiente Bardin (KOSMATOV; MARCOZZI; DELAHAYE, 2021) consideram que "utilizar-se da geração de testes automatizados que mostrou-se ser um direcionamento preciso e específico com objetivos para completar".

Entretanto pode-se valer de técnicas que preferem invalidar o objeto testado assim como Bardin (KOSMATOV; MARCOZZI; DELAHAYE, 2021) alegou que, "quando uma opção é construída, pode ser resolvida usando uma alternativa que não é usada para resolver o problema, diminuindo a expectativa para o valor de entrada ou provendo um caminho que é inviável se a resposta for insatisfatória".

Por sua vez, segundo Ravelo (ESCOBAR; LINARES, 2021) "mesmo sendo um cenário muito simples, pode ser necessária a comunicação de duas aplicações diferentes (como usuário e o motorista) executando em dispositivos diferentes na ordem para completar o cenário estipulado".

É de suma ressaltar que segundo Belli (BUDNIK; HOLLMANN; TUGLULAR; ERIC, 2016) no âmbito de testes "há um esforço para geração de testes e geração de casos de teste, que tem sido complicados pelas ferramentas, representando somente a menor parte no custo de tempo comparado com o atual tempo de execução dos testes e este resultado tem sido negligenciado".

Por conseguinte técnicas de teste que lidam com objetivos e caminhos muitos específicos assim como:

> Técnicas caixa-branca ATG comuns poderiam enfrentar questões como lidar eficientemente com a diversidade de testes objetivos que são confrontados na prática. Por exemplo, DSE na maioria das vezes seguindo a exploração por exaustão do programa com base nos testes, visando tipicamente uma maior cobertura na execução de caminhos para ceder ou falhar. Enquanto muitos caminhos orientados a provas por exploração concretizadas em alguns contextos, é de conhecimento que o resultado da suíte de testes pode ter um comportamento desinteressante relacionado com os dados em vez do que com o controle dos testes. (BARDIN; KOSMATOV; MARCOZZI; DELAHAYE, 2021).

Tendo em isto em vista é de interessante elencarmos que segundo Oliveira (ZEYDA; CAVALCANTI, 2010) um sistema de controle condiz com o cenário de

“serem frequentemente usados em aplicações seguras ou críticas onde é verificado o estado de um funcionamento muito importante ou interessante constantemente”.

Vamos neste trabalho considerar então que testes e casos de testes em alguns cenários devem ser muito bem definidos e estruturados pois esta necessidade já ocorreu na construção de outros trabalhos como exemplo para um trabalho teve-se como base três estudos de caso com aplicações industriais e comerciais, convertendo sistemas interativos, reativos e proativos (BELLI; BUDNIK; HOLLMANN; TUGLULAR; ERIC, 2016).

Com a necessidade de se ter alguma definição na construção dos casos de teste quando comentamos sobre testes caixa branca:

> É importante que muitos objetivos dos testes caixa branca sejam definidos na escolha estrutural, i.e. expressado em termos de artefatos de código genérico (e.g. cobrir todas instruções, todas decisões, todas condições, entre outras), sem falar levar em conta a semântica dos programas. (BARDIN; KOSMATOV; MARCOZZI; DELAHAYE, 2021).

Pois criar este tipo de teste para scripts é algo tipicamente linear, no sentido que eles não incorporam caminhos alternativos servindo somente para um único propósito de rebobinar (ARRUDA; BARROS; SAMPAIO, 2019).

Entretanto há algumas técnicas recentes que mostram sua valia nos testes realizados em trabalhos acadêmicos onde o principal objetivo do estudo seria propor um modelo baseado em testes que tentam provocar falhas no script testado, com uma visualização de caixa preta presumindo que não se tem acesso ao código fonte (BELLI; BUDNIK; HOLLMANN; TUGLULAR; ERIC, 2016), ou então:

> Testes de mutação que estavam originalmente propostos para se basearem na estratégia caixa branca, e com uma técnica baseada em injeção de falhas para implementação baseada em teste de software, especialmente em nível de unidade. Trabalhos recentes têm também aplicado isso para outros níveis de testes, tal como um nível de integração, e outros artefatos como uma especificação de programa. (BELLI; BUDNIK; HOLLMANN; TUGLULAR; ERIC, 2016).

Com uma abordagem proposta pela MBMT o uso de gráficos baseados em modelo é exemplificado para uso e explora mais detalhes para dois operadores elementares de mutantes que foram propostos (BELLI; BUDNIK; HOLLMANN; TUGLULAR; ERIC, 2016).

Já com a proposta da ferramenta Kraken no trabalho do Ravelo que foca no âmbito da geração de dados falsos fazendo uso dos parâmetros Gherkin e transformadores o Kraken na ordem de uso ou reuso dos cenários de testes pode gerar randomicamente nomes, números, e-mails e campos de dados e para a manter e utilizar os relatórios na execução dos arquivos de teste foi gerado um diretório que contém alguns relatórios como detalhes de execução por aparelho (RAVELO; ESCOBAR; LINARES, 2021).

Mas a respeito da validação externa desde sua evolução da abordagem proposta pelo MBMT baseado-se em dados empíricos, tornando os discutíveis onde muitos não são generalizados para todo sistema (BELLI; BUDNIK; HOLLMANN; TUGLULAR; ERIC, 2016).

Quando voltamos nossos olhares para o âmbito acadêmico, surgem algumas técnica muito valiosas que se baseiam na gramática para gerar uma certa automação na construção e execução nos casos de testes onde:

> Para executar ação de sinalização, quando o GAF lê uma dessas instruções nos arquivos de teste, a execução é pausada e notifica o motor de sinal (SE). Esse motor coordena cada dispositivo e habilita suas comunicações para usar um arquivo baseado em protocolos de sinal; cada dispositivo tem um arquivo de texto em um arquivo de sistema interno chamado caixa de entrada onde é possível receber sinais de outros dispositivos. Esses sinais são palavras escritas para os arquivos na caixa de entrada que indicam uma ação completa em outro dispositivo. (RAVELO; ESCOBAR; LINARES, 2021).

Considerando as atividades propostas por Ravelo e considerando o envolvimento da gramática em meio a execução do teste deve se salientar que:

> Desde testes de atividade a códigos, são frequentemente interpretados como comandos para serem executados em um SUT, e o que devemos refletir é sobre a gramática de sentenças imperativas. Além disso, devemos também mapear a saída para as sentenças de entrada dentro da estrutura dos quadros definidas para nossos mecanismos de raciocínio. (ARRUDA; BARROS; SAMPAIO, 2019).

Foi demonstrado também que cada verbo define uma frase, desde que o verbo complementar possa variar (ARRUDA; BARROS; SAMPAIO, 2019), levando em consideração aplicação dessas atividades de teste a consistência do passo a passo pode ser checada justamente analisando se está coerente com o proposto.

Portanto o conceito principal que resguarda a performance é responsivo com o tempo da aplicação dos mecanismos de semântica (ARRUDA; BARROS; SAMPAIO, 2019), posto isto devemos considerar que a entrada de um semântica nova poderá ser nociva para a performance desta automação proposta.

Pensando nisso é então mencionado que é válido uma união dinâmica executada automaticamente trocando quando são novas entradas gramaticais, desenvolvedores evitando intervir ou devido à inviabilidade para geração de análises (ARRUDA; BARROS; SAMPAIO, 2019).

Observando um trabalho como o de Bardin que utiliza de mais testes que em muitos estudos onde tornou-se um padrão realizar uma confirmação dos dados empíricos, já no estudo em questão usou-se uma generalização onde foram utilizados 12

programas de benchmark padrões (BARDIN; KOSMATOV; MARCOZZI; DELAHAYE, 2021).

Colocando em ênfase o âmbito da detecção de falhas ou partes que são menos acessíveis faz-se o uso de análise estática que se mostrou muito mais eficiente entregando e classificando 8,3% dos objetivos inviáveis (BARDIN; KOSMATOV; MARCOZZI; DELAHAYE, 2021)

Contudo outro ponto relevante é que campos descritivos em testes objetivos acabam sendo um pouco menos convenientes, mesmo com eles se propondo a utilizar mais poder em teste de cobertura com critérios aprofundando mais em análises (BARDIN; KOSMATOV; MARCOZZI; DELAHAYE, 2021), acarretando na apresentação de campos de cobertura mais aparentes para tornar eficiente a automação que é expressiva e influenciável (BARDIN; KOSMATOV; MARCOZZI; DELAHAYE, 2021).

Com as indagações apresentadas pelo Bardin consideramos também uma proposta apresentada por Ravelo que mostrou uma ferramenta chamada Kraken oferecendo um caminho para manusear e orquestrar diferentes dispositivos executando aplicações ao mesmo tempo (RAVELO; ESCOBAR; LINARES, 2021), Ravelo ainda explica que é possível fazer a execução de diferentes GUI simultâneas, de eventos com rolagens batendo ou inserindo diversas entradas durante um tempo aleatório (RAVELO; ESCOBAR; LINARES, 2021).

Estudando o trabalho apresentado por Oliveira podemos dizer que é necessário um refinamento para a utilização da linguagem como uma tática onde o próprio Oliveira ainda diz que para o processo de Registrar que começa com o um valor zero, o que proporciona um comportamento de recursividade onde a ação de RegCycle com uma escolha externa: podendo armazenar um novo valor (OLIVEIRA; ZEYDA; CAVALCANTI, 2010).

Por conseguinte Registrar e RegCycle são linguagens utilizadas dentro da execução dos testes e Oliveira diz que com a inclusão de cláusulas opcionais, que simplificam os resultados da documentação e pela aplicação de táticas subjacentes auxiliam no processo mas no geral, para blocos de entrada que precisam de mais fluxo de execução são compartilhados linguagem e todos inputs compartilhados são considerados agora como atualizados a partir da linguagem definida (OLIVEIRA; ZEYDA; CAVALCANTI, 2010).

Ademais Oliveira acrescenta que se considerarmos que cada processo que é implementado está em um bloco, para cada um desse que é removido, o processo que define o diagrama, no paralelismo entre o processo que molda os blocos que eles implementam, resulta na criação de um único processo para cada processo (OLIVEIRA; ZEYDA; CAVALCANTI, 2010).

7. Considerações Finais

Considerando o andamento atual do trabalho, ainda precisamos estipular quais empresas farão parte da amostra e realizar a entrevista com a mesma para que possamos apontar considerações mais assertivas acerca da aplicabilidade da proposta.

Mais a frente no trabalho queremos construir uma aplicação piloto em conjunto com um dos elementos da amostra selecionada, onde iremos considerar dentre os elementos da amostra selecionada o que estiver mais disposto a participar do trabalho e receptivo a compartilhar seus dados do processo de construção de testes para que possamos estruturar nossa aplicação.

Consideramos também angariar a participação das amostras selecionadas despertando o interesse das mesmas em nossa proposta, para que futuramente possamos ampliar o piloto para outros elementos da amostra.

References

ARRUDA, F.; BARROS, F.; SAMPAIO, A. Automation and consistency analysis of test cases written in natural language: An industrial context, 2019.

BARDIN, S.; KOSMATOV, N.; MARCOZZI, M.; DELAHAYE, M. Specify and measure, cover and reveal: A unified framework for automated test generation, 2021.

BELLI, F.; BUDNIK, J. C.; HOLLMANN, A.; TUGLULAR, T.; ERIC W. W. Model-based mutation testing—Approach and case studies, 2016.

MORESI, E., Metodologia da Pesquisa, Brasília, Março de 2003.

OLIVEIRA, M.; ZEYDA, F.; CAVALCANTI, A. A tactic language for refinement of state-rich concurrent specifications, 2010.

RAVELO, M. W.; ESCOBAR, V. C.; LINARES, V. M. Kraken: A framework for enabling multi-device interaction-based testing of Android apps, 2021.