XVI CONGRESSO DE INICIAÇÃO CIENTÍFICA DA UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

Usando Funções de Distância para Maximizar o Reaproveitamento de Testes no Contexto de MBT.

Thomaz Diniz Pinto de Morais[1], **Everton Leandro Galdino Alves** [2]

RESUMO

Model-Based Testing (MBT) é utilizado para gerar automaticamente testes a partir dos modelos de sistemas. Contudo, conforme um software evolui, seus modelos tendem a ser atualizados, o que normalmente acarreta em casos de testes obsoletos que são descartados. O descarte de casos de teste podem ser bastante custosos, uma vez que dados essenciais, como histórico de execução, são perdidos. Neste trabalho, investigamos o uso de funções de distância para tentar ajudar na redução do descarte de teste MBT. Executamos uma série de estudos empíricos usando artefatos de sistemas industriais e analisamos como dez funções de distância podem classificar o impacto de casos de uso no contexto de MBT. Nossos estudos demonstraram que funções de distância são efetivas para fazer uma classificação de edições de baixo impacto que resultam em casos de testes que podem ser reaproveitados com poucas alterações. Além disso, estabelecemos as configurações ótimas para cada função de distância. Por fim, fizemos um estudo de caso que mostrou que, usando funções de distância, conseguimos reduzir o descarte de casos de teste em 15%.

Palavras-chave: Engenharia de Software, Verificação e validação de software.

[1]Aluno de Ciência da Computação, Departamento de Sistemas e Computação, UFCG, Campina Grande, PB, e-mail: thomaz.morais@ccc.ufcg.edu.br
[2]Doutor, Professor do Magistério Superior, UASC, UFCG, Campina Grande, PB, e-mail: everton@computacao.ufcg.edu.br

Reducing the Discard of MBT Test Cases using Distance Functions

ABSTRACT

Model-Based Testing (MBT) is used for generating test suites from system models. However, as software evolves, its models tend to be updated, which often leads to obsolete test cases that are discarded.Test case discard can be very costly since essential data, such as execution history, are lost. In this paper, we investigate the use of distance functions to help to reduce the discard of MBT tests.For that, we ran a series of empirical studies using artifacts from industrial systems, and we analyzed how ten distance function scan classify the impact of MBT-centred use case edits. Our results showed that distance functions are effective for identifying low impact edits that lead to test cases that can be updated with little effort. Moreover, we found the optimal configuration for each distance function. Finally, we ran a case study that showed that, by using distance functions, we could reduce the discard of test cases by 15%.

Keywords: Software Engineering, Software verification and validation.

INTRODUÇÃO

Testes de software são de extrema importância no curso do desenvolvimento de um projeto. Uma vez que ajudam desenvolvedores a ganhar confiança que o software funciona como esperado. Além disso, testar é fundamental para reduzir riscos e garantir a qualidade do software (Kumar, 2016).

Por outro lado, testar pode ser uma atividade bastante custosa. Estudos descobriram que aproximadamente 50% dos custos de um projeto estão relacionados com testes (Kumar, 2016). Além disso suítes de testes podem combinar testes manuais e automáticos (Itkonen, 2009). Apesar de testes automáticos serem desejáveis, testes manuais ainda são bem importantes. Itkonen et al. afirma que testes manuais são de grande importância para a indústria de software e não podem ser completamente substituídos por testes automáticos. Um testador que executa testes manuais, por exemplo, tende a explorar melhor a interface de usuário e encontrar novas faltas.

Uma das estratégias adotadas para reduzir os custos e simplificar os testes manuais é o Model-Based Testing (MBT) (Mark, 2007). Com MBT, testes são gerados automaticamente a partir dos modelos que descrevem o comportamento do sistema. Estes modelos podem, por exemplo, especificar os possíveis estados do sistema, interações entre atores, e/ou comportamentos. No contexto desta pesquisa, focamos em artefatos modelados utilizando CLARET (Dalton, 2018). CLARET é uma linguagem de domínio específico (Van Deursen, 2000) e ferramenta, que promove a integração de um mecanismo de especificação simplificado para especificação de requisitos acompanhado de um processo de geração de casos de teste MBT, utilizando a ferramenta LTS-BT (Oliveira, 2008).

Apesar dos benefícios de MBT, seus uso pode trazer alguns problemas práticos. Silva et al. (2018) discutem que, em média, 86% dos casos de teste MBT dos projetos analisados em seus estudos tornaram-se obsoletos. Contudo, uma grande quantidade desses casos de teste não sofreram mudanças comportamentais e poderiam facilmente serem reaproveitados com pequenos reajustes. Nesse sentido, na prática, é comum que grandes quantidades de casos de teste sejam descartados a cada evolução do software/requisitos. Perder informações de testes

antigos pode atrapalhar no bom gerenciamento do projeto e más estratégias de testes (e.g., testes atrasados, testes que não trazem confiança, ou testes lentos) acabam aumentando o custo do processo de desenvolvimento do projeto em si, trazendo necessidade de refazer algo que já foi feito, o que desperdiça tempo e recursos (Kumar, 2016).

Este projeto tem como objetivo analisar a utilidade prática do uso de funções de distância para aprimorar o uso de testes no contexto de MBT, mais especificamente, durante o processo de evolução de um software. Para tal, um estudo empírico foi realizado analisando edições de documentos de requisitos de dois projetos reais a fim de automaticamente distingui-las entre tipos distintos: edições de alto impacto e de baixo impacto. Enquanto edições de alto impacto são edições que envolvem alterações de comportamento (e.g., adição ou remoção de funcionalidade); edições de baixo impacto, são edições que descrevem alterações que não geram impacto no comportamento das funcionalidades especificadas (e.g., alterações de texto, aprimoramento de sentenças, etc). Desta forma, casos de teste impactados por edições puramente de baixo impacto tendem a ser reaproveitados com simples alterações, sendo seu descarte desnecessário.

Neste contexto, nosso estudo busca analisar a utilização de um conjunto de funções de distância para automaticamente realizar a classificação entre edições de documentos de requisitos de baixo impacto ou de alto impacto, bem como avaliar também seu impacto na suíte de testes. A ideia é que esta pesquisa nos permita determinar quais estratégias de comparação de edições de documento de requisitos melhor qualificarão as edições do sistema e assim reduzir o descarte desnecessário de casos de teste.

MATERIAIS E MÉTODOS

Para a condução deste estudo, um conjunto de ferramentas foram selecionadas, desenvolvidas, e/ou reutilizadas:

1. CLARET

Linguagem e ferramenta para modelagem de requisitos usando uma linguagem própria simplificada que combina elementos próximos a estruturas de linguagens de programação com sentenças em linguagem natural. A Figura 1 apresenta um exemplo de especificação usando a linguagem CLARET. As especificações CLARET são transformadas em modelos LTS que serão úteis para a geração dos testes.

Figura 1. Exemplo de especificação CLARET

```
1   systemName "Email"
2   usecase "Log in User" {
3       version "1.0" type "Creation" user "Dalton" date "01/01/2018"
4       actor emailUser "Email User"
5       preCondition "There is an active network connection."
6       basic {
7           step 1 emailUser "launches the login screen"
8           step 2 system "presents a form with username and password fields and a submit button"
9           step 3 emailUser "fills out the fields and click the submit button" af[1]
10          step 4 system "displays a successful message" ef[1,2]
11      }
12      alternative 1 "Username is predicted" {
13          step 1 emailUser "selects a suggested user name, types password
              and click on the submit button" bs 4
14      }
15      exception 1 "User does not exist in database" {
16          step 1 system "alerts that user does not exist" bs 3
17      }
18      exception 2 "Incorrect username/password combination" {
19          step 1 system "alerts that username or password are incorrect" bs 3
20      }
21      postCondition "User successfully logged"
22  }
```

Fonte: Dalton (2018)

2. LTS-BT

Ferramenta para geração, seleção, redução e priorização de casos de teste em nível de modelos, além de ser capaz de produzir um modelo de suíte de testes a partir de um modelo LTS através da especificação gerada em CLARET.

DESENVOLVIMENTO

Como o objetivo geral desta pesquisa é avaliar a utilidade de funções de distância para classificar edições de documentos de requisitos do sistema que serão usados para a geração de testes MBT. A fim de nortear nossa pesquisa, as seguintes research questions (RQ) foram definidas:

- RQ1: Funções de similaridade são capazes de classificar evoluções de documentos de requisitos de um sistema?
- RQ2: Qual função de distância apresenta os melhores resultados, e segundo qual configuração?

Para fazer nosso estudo empírico, selecionamos dois sistemas industriais (SAFF e BZC) que foram desenvolvidos no contexto de uma cooperação entre o nosso laboratório de pesquisa e duas empresas, Ingenico do Brasil Ltda. e Viceri Solution Ltda. O projeto SAFF é um sistema de informação que gerencia pagamentos em terminais, e BZC é um sistema para otimizar logísticas de atividades de e-commerce.

Ambos os projetos utilizaram desenvolvimento ágil (Beck, 2001) para guiar o desenvolvimento e atualizações de artefatos de requisitos. Além disso, ambos os times utilizaram CLARET, para especificação de casos de uso e LTS-BT para gerar suítes MBT. Ambos os projetos utilizam casos de teste manuais do tipo caixa preta. Neste contexto, manter o histórico de execução dos casos de teste é extremamente importante, já que pode ajudar a melhor gerir a evolução do sistema e impedir regressão de funcionalidades. Apesar disso, os times reportaram que muitas vezes descartam casos de testes quando atualizam qualquer passo do sistema.

Como nosso estudo foca na utilização de funções de distância para evitar este descarte, selecionamos dez das mais conhecidas funções de distância que são utilizadas nos mais variados contextos: Hamming (Hamming, 1950), LCS (Han, 2007), Cosine (Huang, 2008), Jaro, Jaro-Winkler (De coster, 2011), Jaccard (Niwattanakul, 2013), Ngram (Kondrak, 2005), Levenshtein (Levenshtein, 1965), OSA (Damerau, 1964), and Sorensen Dice (Sørensen, 1948). A fim de realizar análises sistemáticas, os valores de distância foram normalizados de tal forma que seus valores variam de zero a um. Valores próximos de zero referem-se a alta

similaridade enquanto valores próximos de um referem-se a baixa similaridade. Implementações de código aberto foram reutilizadas para as dez funções.

Mineramos os repositórios dos projetos e coletamos todas as edições dos casos de teste. Cada edição impacta o projeto de alguma forma. Consideramos casos de teste como “impactado” qualquer caso de teste que inclua passos que foram atualizados durante a manutenção do modelo. Um total de 79 pares de versões de casos de uso foram analisadas em nosso estudo com um total de 518 edições. A Tabela 1 sumariza os dados coletados em nosso estudo, considerando o número de casos de uso, número de versões e quantidade de passos editados.

Tabela 1. Sumário dos artefatos utilizados em nosso estudo.

	Casos de Uso	Versões	Edições
SAFF	13	42	415
BZC	15	37	103
Total	28	79	518

Fonte: Dados coletados pelo primeiro autor

Classificamos manualmente cada passo entre edição de baixo impacto e edição de alto impacto. Uma edição de baixo impacto é uma atualização que não altera o comportamento do sistema (edição puramente sintática). Enquanto uma edição de alto impacto é uma edição que altera o comportamento do sistema (edição semântica). A Tabela 2 exemplifica essa classificação. Enquanto a edição da primeira linha muda semanticamente o requisito original, as outras duas demonstram melhoramentos semânticos (e.g., Melhoramento da forma como se descreve, correções gramaticais). Foram classificados 399 edições de baixo impacto e 27 edições de alto impacto para o sistema SAFF; 92 edições de baixo impacto e 11 de alto impacto para o BZC.

Tabela 2. Exemplo de mudanças identificadas e sua classificação.

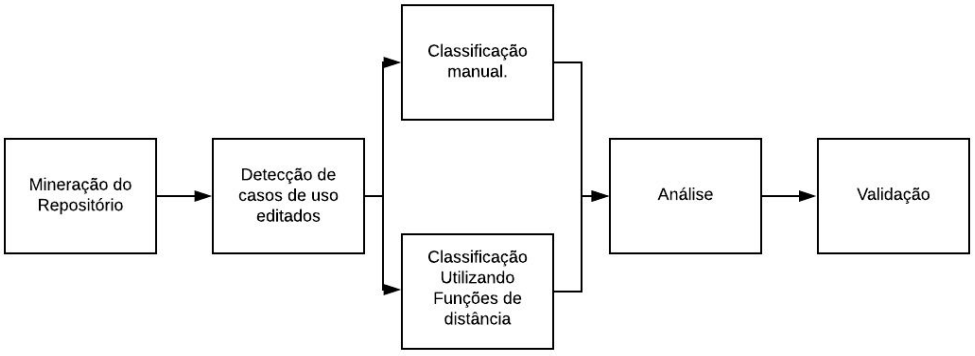
Original	Novo	Classificação
“Realiza extração em modo offline.”	“Exibe página que requisita dados do terminal.”	Alto Impacto
"Exibe página que requisita dados do terminal"	"Exibe pagina que requisita dados do terminal"	Baixo Impacto
"Clica no botão 'Edit'"	"Clica no botao 'Edit'"	Baixo Impacto

Fonte: Dados coletados pelo primeiro autor

Este resultado demonstra que muitas vezes os casos de uso evoluem de maneira a melhorar a estrutura. O que pode não ser uma boa justificativa para a grande quantidade de descarte de casos de teste em suítes MBT.

Depois disto, para cada versão das edições da especificação, executamos o código de funções de distância utilizando configurações variadas e observamos como estas classificam as edições em comparação com a nossa classificação manual. A Figura 2 sumariza as atividades executadas durante o nosso estudo experimental para cada projeto selecionado.

Figura 2. Fluxograma das atividades desenvolvidas

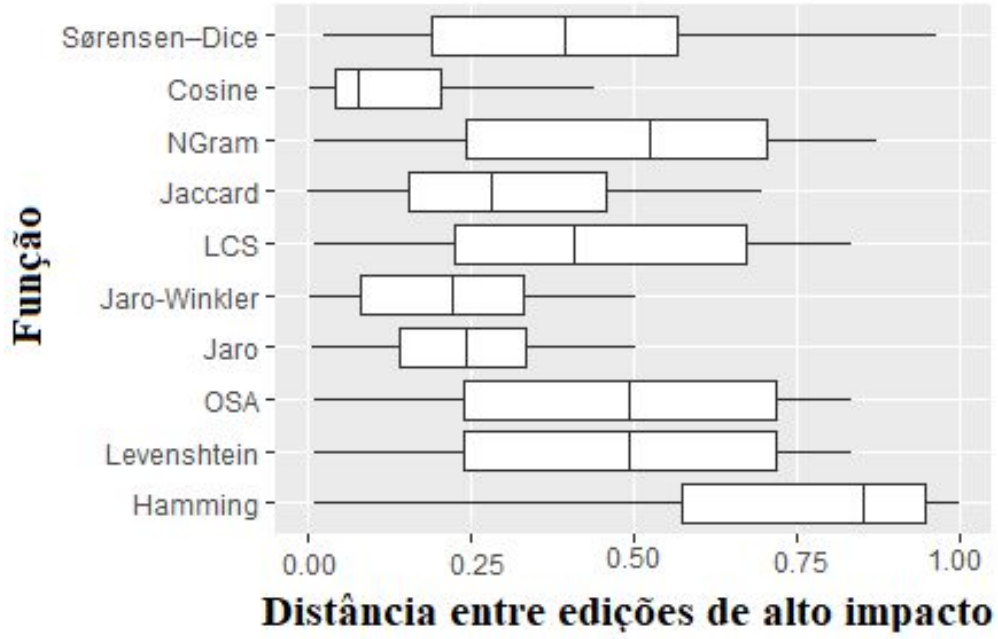


Fonte: Fluxograma desenvolvido pelo primeiro autor

RESULTADOS E DISCUSSÕES

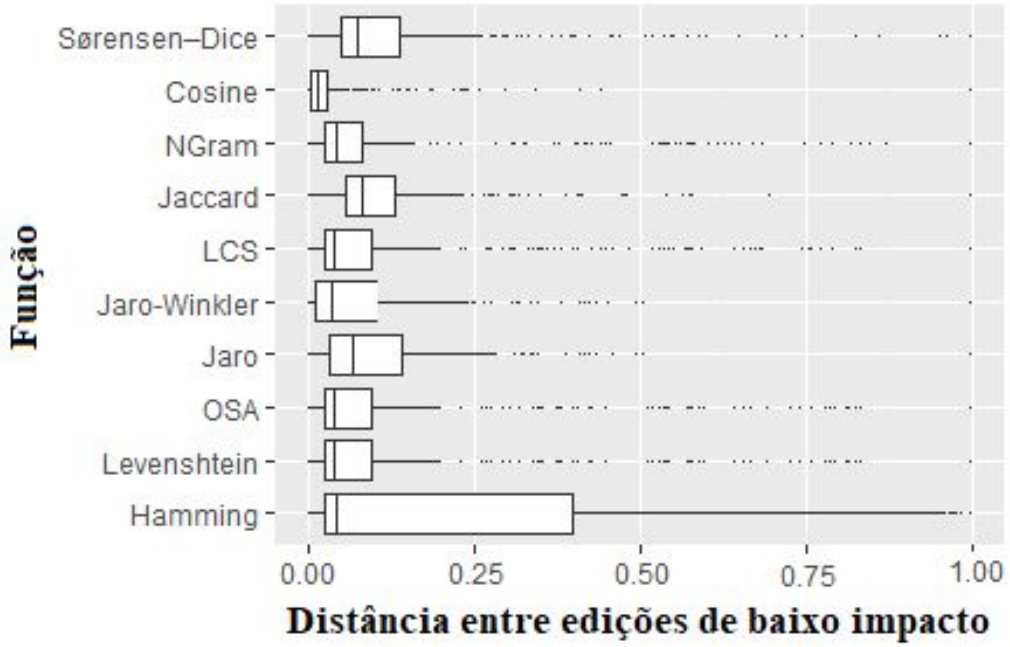
Para responder a RQ1, dividimos nossos dados em dois (edições de baixo e alto impacto) de acordo com nossa classificação manual. Então, executamos as funções de distância e plotamos os resultados. A Figura 3 mostra os boxplots de visualização desta análise para as edições de alto impacto, enquanto a Figura 4 mostra os boxplots para as edições de baixo impacto.

Figura 3. Distâncias entre edições de alto impacto



Fonte: Gráfico desenvolvidos pelo primeiro autor

Figura 4. Distâncias entre edições de baixo impacto



Fonte: Gráfico desenvolvidos pelo primeiro autor

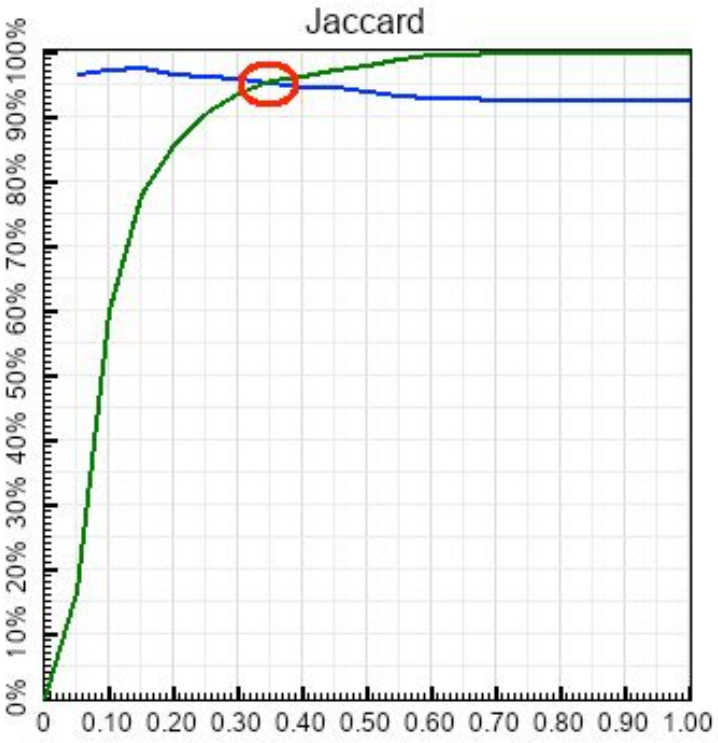
Como é possível observar, a maior parte das edições de baixo impacto possuem valores de distância pequenos (mediana menor que 0.1), para todas as

funções de distância. Este resultado nos dá evidências que baixos valores de distância podem ter relação com edições de baixo impacto. No caso de edições semânticas, não conseguimos encontrar relação entre as edições de alto impacto e os valores de distância das funções utilizadas. Desta forma, podemos responder a RQ1, dizendo que funções de distância podem nos dar certa segurança na hora de classificar as edições de baixo impacto.

RQ1: Funções de similaridade são capazes de classificar evoluções de documentos de requisitos de um sistema? Edições de baixo impacto estão normalmente relacionados com baixos valores de distância. Desta forma, funções de distância podem ser utilizadas para classificar edições de baixo impacto.

Para efetuar uma classificação automática, necessitamos definir um limiar de corte de classificação, sendo limiar de corte de classificação um valor limite (limiar) que define quando classificar uma edição sintática. Por exemplo, considere a função *f* utilizando o limiar de corte de *x* para analisar uma edição. Ao executar a função de distância nesta edição, se for obtido um valor maior que *x* de distância, esta edição será considerada de alto impacto, caso contrário, esta edição será considerada de baixo impacto. Realizamos um estudo empírico onde variamos ,para cada função, seus limiares entre 0 a 1 para encontrarmos a melhor configuração. Nesta análise utilizamos as métricas Precisão e Recall e consideramos o ponto de intercepção entre recall e precisão como um ponto de valor ótimo, uma vez que reflete um cenário com poucos erros de classificação.

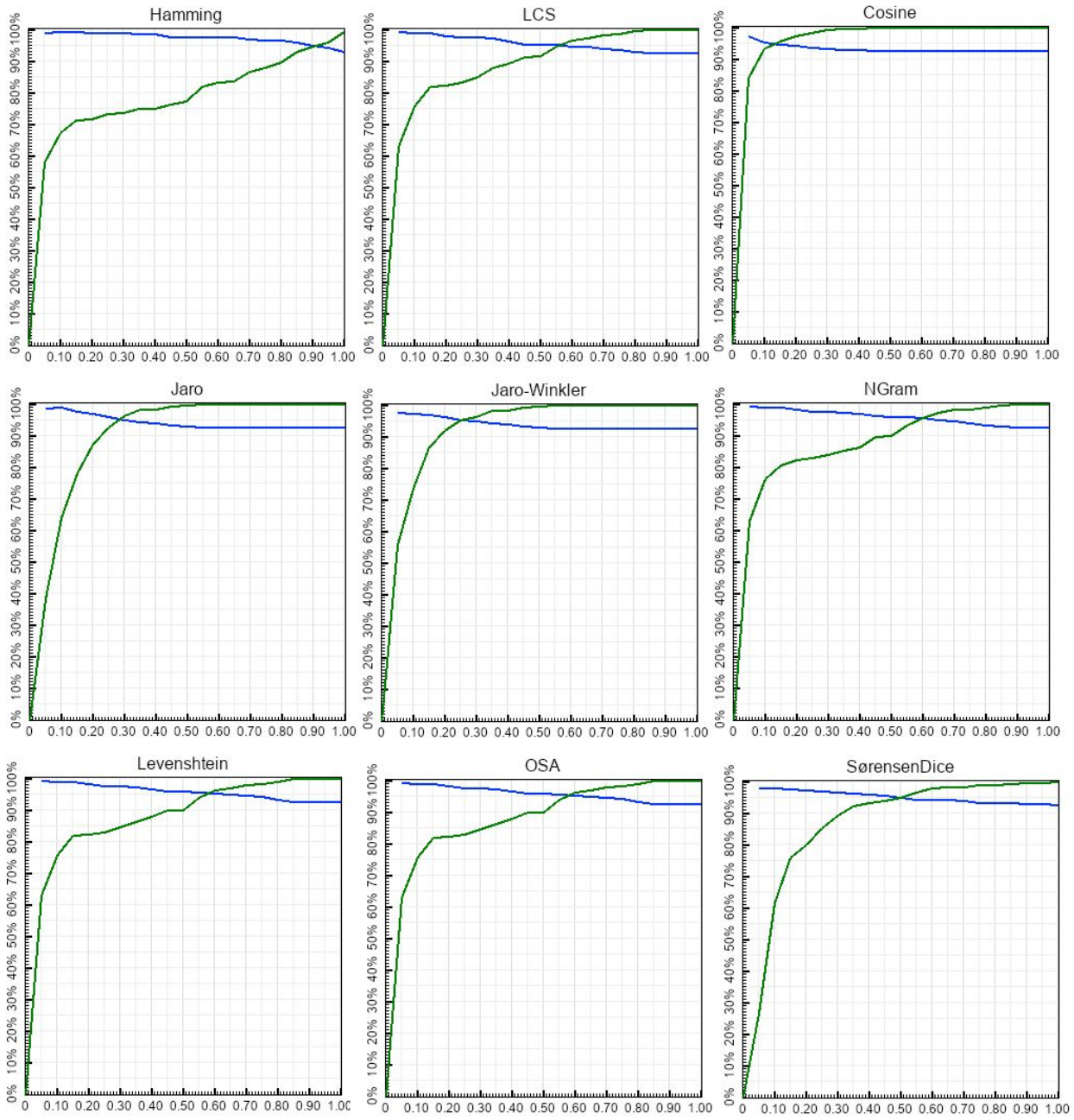
Figura 5 - Melhor limiar de corte para a função Jaccard



Fonte: Gráfico desenvolvido pelo primeiro autor

A Figura 5 apresenta a análise feita para a função jaccard (limiar de corte ótimo em 0.33 de distância) – a linha verde apresenta os valores de precisão; enquanto a linha azul apresenta os valores de recall; e o círculo vermelho representa nosso ponto ótimo do limiar de classificação. A Figura 6 apresenta a mesma análise considerando as demais funções.

Figura 6 - Estudo exploratório com as funções de distância.



Fonte: Gráficos desenvolvidos pelo primeiro autor

A Tabela 3 apresenta a configuração ótima para cada função com seus respectivos valores de Precisão, Recall e Acurácia (métrica que combina Precisão e Recall). Estes resultados reforçam os conclusões de RQ1 já que todas funções apresentaram uma acurácia acima de 90%.

Tabela 3. Melhor configuração para cada função

Função	Limiar de Corte	Precisão	Recall	Acurácia
Hamming	0.91	94.59%	94.79%	90.14%
Levenshtein	0.59	95.22%	95.42%	91.31%
OSA	0.59	95.22%	95.42%	91.31%
Jaro	0.28	95.01%	95.21%	90.93%
Jaro-Winkler	0.25	95.21%	95.21%	91.12%
LCS	0.55	94.99%	94.79%	90.54%
Jaccard	0.33	95.22%	95.42%	91.31%
NGram	0.58	95.41%	95.21%	91.31%
Cosine	0.13	95%	95%	90.73%
Sorensen-Dice	0.47	94.99%	94.79%	90.54%

Fonte: Dados coletados pelo primeiro autor

Além disso, observando a Tabela 3 podemos parcialmente responder a RQ2. No contexto dos nossos dados, as dez funções testadas foram funções com bons resultados e, para completar nossa análise, investigamos qual função se comporta melhor em sua melhor configuração (limiar de corte). Para tal, executamos um teste de proporção comparando cada função, par-a-par, e, com 95% de confiança, não conseguimos encontrar nenhuma diferença estatística entre as funções. O que significa que a classificação deve ser efetiva, independente da função escolhida (RQ2). Na prática, o desenvolvedor pode selecionar qual função gostaria de utilizar baseado na conveniência (e.g, facilidade de implementação, mais rápida).

RQ2: Qual função de distância apresenta os melhores resultados, e segundo qual configuração? Na prática todas as dez funções testadas foram funções com bons resultados em sua melhor configuração e não existem diferenças estatísticas entre elas.

Para validar nossos resultados, fizemos um segundo estudo empírico considerando terceiro projeto: o TCOM que é um projeto também desenvolvido em cooperação entre os nossos laboratórios e a Ingenico do Brasil LTDA. para execução de testes em hardware. Mineramos o repositório do projeto e coletamos

todas as edições dos casos de teste e especificação CLARET, com o sumário dos dados obtidos demonstrado na Tabela 4.

Tabela 4. Sumário do projeto TCOM.

	Casos de Uso	Versões	Edições
TCOM	7	32	133

Fonte: Dados coletados pelo primeiro autor

Em seguida, manualmente, classificamos as edições em edições de baixo e alto impacto. Então executamos as classificações utilizando as configurações obtidas dentro dos nossos estudos iniciais (Tabela 3, segunda coluna). O resultado, juntamente com a precisão, recall e acurácia pode ser observado na Tabela 5.

Tabela 5. Validando os limiares encontrados com o projeto TCOM.

Função	Limiar de Corte	Precisão	Recall	Acurácia
Hamming	0.91	87.59%	94%	84.96%
Levenshtein	0.59	87.85%	94%	85.71%
OSA	0.59	87.85%	94%	85.71%
Jaro	0.28	89.52%	94%	87.22%
Jaro-Winkler	0.25	94%	89.52%	87.22%
LCS	0.55	89.62%	95%	87.97%
Jaccard	0.33	89.52%	94%	87.22%
NGram	0.58	87.85%	94%	85.21%
Cosine	0.13	88.68%	94%	86.47%
Sorensen-Dice	0.47	88.68%	94%	86.47%

Fonte: Dados coletados pelo primeiro autor

Como pode ser observado, os limiares de corte nos dão uma acurácia acima de 87%. Este resultado nos dão evidências de que funções de distância são efetivas para classificar edições (RQ1) e que os limiares de corte encontrados foram efetivos mesmo considerando um sistema totalmente diferente (RQ2).

Em seguida, rodamos um estudo de caso para avaliar como nosso método de utilizar funções de distância podem reduzir o descarte de testes. Para isso

mantivemos utilizando a especificação CLARET dos artefatos do TCOM e definimos a seguinte questão de pesquisa:

- RQ3: Funções de distância podem ser utilizadas para reduzir descarte em testes MBT?

Para responder a RQ3, utilizamos os testes MBT gerados por LTS-BT. Como todas as funções se mostraram estatisticamente equivalentes, selecionamos *Levenshtein* para classificar automaticamente as edições. Em um cenário comum, qualquer caso de teste que inclui um passo que corresponde a um trecho modificado do documento de use case seria descartado. Então, no contexto deste estudo, utilizamos a seguinte estratégia: “Apenas casos de testes que contém edições de alto impacto deveriam ser descartados, enquanto casos de teste com edições de baixo impacto podem ser reutilizadas com pouca ou nenhuma edição”. A razão por trás disso é que edições de baixo impacto trazem pouca ou nenhuma mudança ao comportamento do sistema. Utilizamos esta estratégia e a fizemos a classificação utilizada por Oliveira et al (2008) que divide os testes entre três tipos: Obsoletos - Casos de testes impactados, Reusáveis - casos de testes que não foram impactados e Novos - Casos de testes que previamente não existiam, mas com a mudança no sistema agora existem.

Um total de 1477 casos de testes MBT foram coletados do projeto TCOM. onde 333 foram classificados como novos (23%), 724 foram classificados como obsoletos (49%) e 420 foram classificados como reusáveis (28%). Em um cenário comum todos os casos de teste obsoleto seriam descartados.

Reclassificamos os testes obsoletos entre: Obsoletos de baixo impacto - casos de teste que incluem somente passos editados que foram classificados como de baixo impacto; obsoletos de alto impacto - aqueles que possuem somente passos editados que foram classificados de alto impacto; e obsoletos de impacto misturado - que possuem tanto casos editados classificados como de alto quanto de baixo. Desta análise, 109 casos de teste foram de baixo impacto e tendem a ser reaproveitados. Este número pode parecer baixo (15%), porém é válido destacar que estes testes seriam descartados; 196 foram classificados como de alto impacto (27%) e 419 como misturado (58%).

Tabela 6. Matriz de confusão da classificação dos casos de teste do TCOM

	Baixo impacto previsto	Alto Impacto Previsto	Impacto Misturado Previsto	Total
Baixo Impacto Real	69	4	21	94
Alto Impacto Real	3	37	27	67
Impacto Misturado Real	37	155	371	563
Total	109	196	419	724

Fonte: Dados coletados pelo primeiro autor

Para verificar se nossa classificação foi de fato efetiva, apresentamos a matriz de confusão na Tabela 6. Em geral, nossa classificação foi 66% efetiva (precisão). Uma precisão menor já era esperada, quando comparada com as precisões anteriores, já que consideramos que todas as edições podem afetar um caso de uso como um todo, enquanto anteriormente estávamos analisando a classificação de edições passos individualmente. Apesar disso, nossa classificação foi efetiva para os casos de alto impacto e de baixo impacto, e a maior parte dos erros foi para os casos de impactos misturados (que possuem ambos).

Então os 15% dos obsoletos (retirados da classificação de baixo impacto) poderiam ser reclassificados como reutilizáveis. Além disso, acreditamos que este valor pode subir ainda mais se considerarmos a possibilidade de uma melhor classificação dos obsoletos com edições de impacto misturados. Encontramos casos onde, apesar de termos passos misturados, havia indícios de ser na verdade um teste que poderia ser reutilizado. Por exemplo, um teste com 104 passos de execução onde apenas um passo necessitou de revisão por conta de edições de alto impacto.

Finalmente, podemos responder a RQ3 e dizer que a classificação automática pode, de fato, reduzir o número de testes descartados em, no mínimo 15%. Este número pode incrementar ainda mais com estudos considerando os casos de teste de alto e baixo impacto misturados.

RQ3: Funções de distância podem ser utilizadas para reduzir descarte em testes MBT? Utilizar funções de distância no contexto MBT pode reduzir o descarte de testes em, no mínimo, 15%.

A fim de tornar os resultados da nossa pesquisa acessíveis a comunidade, desenvolvemos uma ferramenta[3]. Nossa ferramenta tem por objetivo ajudar testadores a reduzir o descarte de testes utilizando as funções de distância discutidas neste trabalho. A ferramenta recebe como entrada diferentes versões de especificação CLARET e uma suite de teste. A partir desses dados, conseguimos automaticamente identificar quais casos de teste são reutilizáveis. Em um futuro próxima essa ferramenta estará inserida no ambiente de CLARET.

CONCLUSÃO

Neste estudo, descrevemos uma série de estudos empíricos que executamos em sistemas industriais com o intuito de avaliar a utilização de funções de distância para classificar automaticamente o impacto de edições nos modelos de especificação e testes. Nossos resultados mostraram que funções de distância são efetivas para classificar edições de baixo impacto.

Também descobrimos que edições de baixo impacto podem ser facilmente reutilizadas com pouco esforço de edição. Acreditamos que estes resultados podem ajudar os testadores a reduzir o desperdício de testes e a melhor trabalharem com artefatos MBT.

Além disso, os trabalhos desenvolvidos neste projeto renderam um artigo científico a ser publicado no XXXIII Brazilian Symposium on Software Engineering.

AGRADECIMENTOS

O presente trabalho foi realizado com apoio do CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico – Brasil, no contexto do programa PIBIC/CNPq-UFCG.

REFERÊNCIAS

[3] https://github.com/ThomazDiniz/Tools-to-compare

Anderson G.F. Silva, Wilkerson L. Andrade, and Everton L.G. Alves. **A study on the impact of model evolution in mbt suites**. In Proceedings of the III Brazilian Symposium on Systematic and Automated Software Testing, SAST '18, pages 49–56, New York, NY, USA, 2018. ACM.

Beck K., M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham,M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, Jon Kern,Brian Marick, Robert C. Martin, Steve Mallor, Ken Shwaber, and Jeff Sutherland. **The Agile Manifesto**. Technical report, The Agile Alliance,2001

Booch, Grady and Rumbaugh, James and Jacobson, Ivar, **Unified Modeling Language User Guide, The (2Nd Edition)** (Addison-Wesley Object Technology Series), 2005 Addison-Wesley Professional (0321267974)

Dalton N. Jorge, Patrícia Machado, Everton L. G. Alves, and Wilkerson Andrade. Claret - **Central artifact for requirements engineering and model-based testing**. / in proceedings of the 24th tools session / 8th brazilian conference on software: Theory and practice. fortaleza, ce, br,41–48. In-, 2017.

Damerau, Fred J. **A Technique for Computer Detection and Correction of Spelling Errors** (1964) - Commun. ACM 171 - 176

De Coster, Xavier and De Groote, Charles and Destine, Arnaud and Deville, Pierre and Lamouline, Laurent and Leruitte, Thibault and Nuttin, Vincent, **Mahalanobis distance, Jaro-Winkler distance and Dollar in Usi Gesture**, 2011

Divya Kumar and KK Mishra. 2016. **The Impacts of Test Automation on Software's Cost, Quality and Time to Market**. Procedia Computer Science 79 (2016), 8–15.

Emanuela Cartaxo, Wilkerson Andrade, Francisco de Oliveira Neto, and Patrícia Machado. **Lts-bt: A tool to generate and select functional test cases for embedded systems.** In-, pages 1540–1544, 01 2008.

Hamming, Richard W. "**Error detecting and error correcting codes**" (1950) - The Bell system technical journal volume 29 147-160

Han T.S., Ko SK., Kang J. (2007) **Efficient Subsequence Matching Using the Longest Common Subsequence with a Dual Match Index**. In: Perner P. (eds) Machine Learning and Data Mining in Pattern Recognition. MLDM 2007. Lecture Notes in Computer Science, vol 4571. Springer, Berlin, Heidelberg

Huang, Anna. "**Similarity measures for text document clustering**." Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand. Vol. 4. 2008.

Itkonen J., M. V. Mantyla, and C. Lassenius. 2009. **How do testers do it? An exploratory study on manual testing practices.** In 2009 3rd International Symposium on Empirical Software Engineering and Measurement. 494–497. https: //doi.org/10.1109/ESEM.2009.5314240.

Karim O Elish and Mahmoud O Elish. 2008. **Predicting defect-prone software modules using support vector machines**. Journal of Systems and Software 81, 5 (2008), 649–660

Kondrak, Grzegorz, "**N-gram similarity and distance**" - International symposium on string processing and information retrieval (115-126) 2005.

Levenshtein, Vladimir Iosifovich, **Binary codes capable of correcting deletions, insertions, and reversals**. Soviet Physics Doklady, Doklady Akademii Nauk SSSR, V163 No4 845-848 1965

Mark Utting and Bruno Legeard. **Practical Model-Based Testing: A Tools Approach**. Morgan Kaufmann Publishers Inc., San Francisco,CA, USA, 2007

Niwattanakul, Suphakit and Singthongchai, Jatsada and Naenudorn, Ekkachai and Wanapu, Supachanun, **Using of Jaccard coefficient for keywords similarity** - Proceedings of the international multiconference of engineers and computer scientists (2013) 380-384

Roger Pressman. 2005. **Software Engineering: A Practitioner's Approach (6 ed.)**. McGraw-Hill, Inc., New York, NY, USA

Sørensen, Thorvald. "**A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons.**" *Biol. Skr.* 5 (1948): 1-34.

Todd Sedano, Paul Ralph, and Cécile Péraire. **Software development waste**. In-, 05 2017.

van Deursen, Arie and Klint, Paul and Visser, Joost, 2000, pages 26-36, "**Domain-Specific Languages: An Annotated Bibliography**" SIGPLAN Notices

Wiegers, K.E. and Beatty, J. "**Software Requirements**" 2013 Microsoft Press (9780735679665).