Pós-Graduação em Ciência da Computação

Cloves Alberto Chaves de Lima

SPt - Uma nova abordagem para revisão automática de artefatos de software e geração de planos de teste

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2019

Cloves Alberto Chaves de Lima

SPt - Uma nova abordagem para revisão automática de artefatos de software e geração de planos de teste

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Engenharia de software
Orientador: Alexandre Cabral Mota
Coorientadora: Flávia de Almeida Barros

Recife
2019

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

L732s Lima, Cloves Alberto Chaves de
SPt: uma nova abordagem para revisão automática de artefatos de software e geração de planos de teste / Cloves Alberto Chaves de Lima. – 2019.
78 f.: il., fig., tab.

Orientador: Alexandre Cabral Mota.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2019.
Inclui referências.

1. Engenharia de software. 2. Teste de software. 3. Mineração de texto. I. Mota, Alexandre Cabral (orientador). II. Título.

005.1 CDD (23. ed.) UFPE- MEI 2019-067

Tese de Mestrado apresentada por **Cloves Alberto Chaves de Lima** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**SPt - Uma nova abordagem para revisão automática de artefatos de software e geração de planos de teste**" **Orientador: Alexandre Cabral Mota** e aprovada pela Banca Examinadora formada pelos professores:

Profa. Flávia de Almeida Barros
Centro de Informática / UFPE

Prof. Lucas Albertins de Lima
Departamento de Computação / UFRPE

Prof. Juliano Manabu Iyoda
Centro de Informática / UFPE

Prof. Orientador: Alexandre Cabral Mota
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 08 de Fevereiro de 2019.

Prof. Ricardo B. C. Prudêncio
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

Dedico esta conquista a minha esposa Elizabeth pelo incentivo e apoio em todas as minhas escolhas e decisões.

AGRADECIMENTOS

Foi uma trajetória dura, mas em nenhum momento uma trajetória solitária. Por este motivo, tenho muito a agradecer a algumas pessoas.

A **Deus** por tudo que tem feito em minha vida, que sempre esteve comigo em toda essa jornada.

Sem nenhuma dúvida houve diversos professores que contribuíram para minha formação intelectual, dentre eles estão: **Prof. Dr. Ricardo Prudêncio** por ter dado inicio a toda essa jornada. Sou grato por tudo.

Agradeço ao meu orientador **Prof. Dr. Alexandre Mota** e em especial a minha co-orientadora **Prof. Dra. Flávia Barros** por ter participado dessa trajetória, me incentivando, apoiando e, principalmente orientando de maneira dura e terna, me corrigindo quando necessário (*sem nunca desmotivar*) para o enriquecimento deste trabalho. Uma grande profissional e admirável mulher, muito obrigado.

A minha **mãe Messelene** e **Tia Marilene**, pela minha formação moral e intelectual.

A minha querida **esposa Elizabeth** pelo companheirismo, incentivo e apoio incondicional, que apostou em mim mais do que ninguém, acreditando sempre no meu potencial. Nada disso teria sentido se você não estivesse em minha vida, te amo meu porto seguro.

Por fim, quero demonstrar o meu agradecimento a todos aqueles que, de um modo direto ou indireto tornaram possível a realização da presente dissertação.

RESUMO

Os produtos de software devem apresentar níveis de alta qualidade para serem bem-sucedidos em um mercado competitivo. Normalmente, a confiabilidade dos produtos é assegurada pelas atividades de teste. No entanto, o teste de software às vezes é negligenciado pelas empresas devido a seus altos custos, especialmente quando executados manualmente. Sob esta ótica, este trabalho investiga métodos inteligentes para automação de testes de software, com foco no processo de revisão de produtos/artefatos de software. Propomos um novo processo para criação de Planos de Teste com base na inspeção de documentos de software (em particular, Release Notes (RN)) usando técnicas de Mineração de Texto e Recuperação de Informação. O objetivo principal é identificar áreas relevantes da nova versão do software sob teste para serem examinadas pelas equipes de testes exploratórios. Essa informação é automaticamente extraída a partir das Change Request (CR) contidas no campo *bugs corrigidos* das RNs. Esse processo utiliza as áreas relevantes indicadas para sugerir *charters* (*i.e.*, casos de teste) para compor planos de teste mais eficientes e, desta forma, aumentar a produtividade das equipes de testes exploratório. O protótipo implementado, a ferramenta SWAT Plan tool (SPt), foi testado usando dados reais da *Motorola Mobility*, empresa parceira do CIn-UFPE. Os experimentos compararam o processo manual atual de revisão dos produtos/artefatos de software com o processo automatizado usando SPt, avaliando o tempo gasto e as áreas relevantes identificadas em ambos os métodos. Os resultados obtidos foram muito animadores.

Palavras-chaves: Revisão de Documentos. Teste de Software. Mineração de Texto.

ABSTRACT

Software products must show high-quality levels to succeed in a competitive market. Usually, products reliability is assured by testing activities. However, SW testing is sometimes neglected by companies due to its high costs - particularly when manually executed. In this light, this work investigates intelligent methods for SW testing automation, focusing on the software products/artifact review process. We propose a new process for test plan creation based on the inspection of SW documents (in particular, Release Notes (RN)) using Text Mining and Information Retrieve techniques. The implemented prototype, the SPt, automatically extracts from CR contained in the field *bugs fixed* of RN to be examined by exploratory tests teams. SPt also uses the relevant areas indicated to suggest *charters* (*i.e.*, test case for exploratory teams) to compose the most efficient test plan and thus increase the productivity of the exploratory test teams. SPt was tested using real-world data from *Motorola Mobility*, our partner Company. The experiments compared the current manual of SW products/artifact review process with the automated process using SPt, evaluating the time spent and the relevant areas identified in both methods. The obtained results were very encouraging.

Keywords: Document Review. Software Testing. Text Mining.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AM	Aprendizagem de Máquina
ARENA	Automatic RElease Notes generAtor
BIO	Beginning Inside Outside
CR	Change Request
CT	Caso de Teste
EI	Extração de Informação
IA	Inteligência Artificial
ICICLE	Intelligent Code Inspection in a C Language Environment
InspeQ	Inspecting software in phases to ensure Quality
MT	Mineração de Texto
NP	Nominal Phase
PLN	Processamento de Linguagem Natural
RI	Recuperação de Informação
RN	Release Notes
SMT	Sistema de Mineração de Texto
SPt	SWAT Plan tool
SRI	Sistema de Recuperação de Informação
SW	software
SWAT	Software Analyze Testing
VP	Verbal Phase

SUMÁRIO

1 INTRODUÇÃO

Garantir a qualidade dos produtos de software antes do seu lançamento é crucial para obter sucesso no atual mercado competitivo. Uma maneira de atingir os padrões desejados é através do teste de SW. Infelizmente, as atividades de teste são demoradas e muito caras. Várias pesquisas contribuíram para a melhoria das práticas de teste de SW implantadas pela indústria (MARTIN et al., 2007a), (ANDERSSON; RUNESON, 2002a). Ainda assim, esses trabalhos não abordaram adequadamente alguns processos importantes (GLASS et al., 2006), como a revisão de produtos/artefatos de SW.

De acordo com Katasonov e Sakkinen (2005) e Alberto (2014), uma das práticas centrais para garantia de qualidade é a revisão de produtos/artefatos de SW (documentos produzidos durante o processo de desenvolvimento do SW). A inspeção de documentos/artefatos de SW, uma das técnicas de revisão, é muito útil para reduzir o esforço na busca por defeitos (KOMSSI et al., 2010), uma vez que alguns desses documentos contêm informações detalhadas sobre as novas versões do SW a ser testado.

No entanto, na maioria das empresas, os analistas responsáveis pela revisão de documentos (na sua grande maioria, testadores ou desenvolvedores) não são bem treinados, o que contribui para o aumento de defeitos escapados (*i.e.*, defeitos que não foram encontrados durante a inspeção e fase de teste, e que ocorrem após o lançamento do produto) (MANTYLA; LASSENIUS, 2009), (CIOLKOWSKI; LAITENBERGER; BIFFL, 2003).

Além da falta de treinamento dos profissionais envolvidos na inspeção, outros fatores exercem uma influência negativa para a sua realização, como:

1. O tempo requerido - Em geral, essa tarefa é executada manualmente, tornando-se inviável em ambientes empresariais reais, devido a restrições de tempo.

2. A falta de incentivos - Os analistas (testadores ou desenvolvedores) não recebem incentivos financeiros ou de qualquer outro tipo que os motivem para a execução de uma inspeção (ALBERTO, 2014).

Assim sendo, a automação desse processo é uma solução para melhorar a produtividade das equipes e a qualidade final do SW sob teste.

1.1 OBJETIVOS

Este trabalho aborda a automação do processo de revisão de documentos SW usando técnicas de Mineração de Texto (MT) (IGNATOW; MIHALCEA, 2017), propondo um novo modelo de geração de planos de teste. Em particular, focamos na inspeção de documentos de RN, que contêm informações importantes sobre cada nova versão do SW (ABEBE; ALI; HASSAN, 2016), como alterações de código para corrigir bugs, funcionalidades adicionais,

etc. As RNs também trazem informações sobre CR, outro artefato central do SW que descreve os bugs encontrados por testadores.

O processo proposto inicialmente realiza uma análise dos comentários sobre as CRs contidas no RN. Sempre que um comentário possui informações relevantes, a CR associada é analisada e dela são extraídas áreas relevantes para análise por parte dos testadores. O objetivo principal é auxiliar as equipes de testes exploratórios: como não é viável testar totalmente cada nova versão do SW, os planos de teste exploratórios precisam se concentrar nas mudanças relatadas nas versões anteriores do SW.

1.2 CONTRIBUIÇÕES

O processo proposto foi implementado como um protótipo chamado de SWAT Plan tool (SPt), onde o termo SWAT se origina de *SW Analyze Testing*. O SPt foi construído com base em técnicas de Inteligência Artificial (IA) para classificação de texto (MCCALLUM; NIGAM, 1999), Processamento de Linguagem Natural (PLN) (JURAFSKY; MARTIN, 2009) e Métodos de Recuperação de Informação (RI) (BAEZA-YATES; RIBEIRO-NETO, 2011) para extração de informações, contando com cinco módulos de processamento detalhados no Capítulo 4.

Os experimentos foram realizados usando dados reais de uma empresa parceira do CIn-UFPE, a *Motorola Mobility*. Os experimentos compararam o processo manual usado antes da SPt, para analisar as RNs e criar planos de teste da Software Analyze Testing (SWAT) (*i.e.*, planos de teste da equipe de testes exploratórios) com a análise usando a SPt. Dois aspectos diferentes foram observados: o tempo gasto pelos testadores para inspecionar as RNs e gerar os planos de teste da SWAT com e sem o uso do SPt, e o número de áreas relevantes identificadas com e sem SPt. Como será visto no Capítulo 4, os resultados alcançados foram muito encorajadores.

Este trabalho gerou uma publicação em evento com classificação Qualis B2, ocorrido em outubro de 2018: Lima et al. (2018).

1.3 ORGANIZAÇÃO DO TRABALHO

Nesse capítulo introdutório buscou-se mostrar, em linhas gerais, o contexto ao qual o presente trabalho está inserido e um resumo das contribuições feitas com a sua elaboração.

O Capítulo 2 traz uma visão geral da revisão do produto/artefato de SW mostrando as principais abordagens para o modelo de inspeção de documento. Em seguida o mesmo capítulo aborda sobre o documento alvo a ser inspecionado, a RN e sobre o principal dado contido no campo de *bugs* corrigidos as CRs.

No Capítulo 3, discutimos sobre todas as técnicas utilizadas na construção do protótipo, dando enfase na MT e nos processos que a envolve. Além disso, uma breve descrição

sobre algumas técnicas de PLN e RI foram descritas para facilitar o entendimento dos processos que envolvem o trabalho proposto.

Por fim, o Capítulo 4 detalha o processo proposto para a criação de planos de teste da SWAT. Além do processo, detalhes da sua implementação para a criação da SPt foram abordados ao decorrer do capítulo. Ainda no Capítulo 4, detalhes do modelo de uso do protótipo com imagens são fornecidas para uma melhor compreensão de uso, e um experimento foi conduzido com clareza e sua metodologia e resultados foram disponibilizados.

A conclusão final é apresentada no Capítulo 5, juntamente com as limitações do trabalho e sugestões de continuidade a partir desta pesquisa.

2 REVISÃO DE PRODUTOS DE SOFTWARE, RELEASE NOTES E CHANGE REQUESTS

A revisão de produto de SW ou simplesmente revisão de documentos, consiste na análise de quaisquer artefatos resultantes das atividades de desenvolvimento de SW, como: código fonte, planos de teste, documentos de requisitos, CRs, RNs, etc (KOMSSI et al., 2010). Esta tarefa visa identificar padrões de código, novos recursos implementados ou áreas relevantes[1] mais suscetíveis a erros, ajudando não apenas os desenvolvedores, mas também as equipes de teste, tornando-as mais eficazes na detecção de defeitos, pois pode reduzir os custos de retrabalho em até 60% (GRADY; Van Slack, 1994).

Segundo Ciolkowski, Laitenberger e Biffl (2003), a revisão de um documento/produto de SW pode ser conduzida de duas formas pelo revisor. A primeira forma, conhecida como *walkthroughs*, consiste em o(s) revisor(es) discutem com os autores quais os possíveis dados relevantes contidos no documento. A segunda forma, conhecida como inspeção, é conduzida de forma mais sistemática, onde, em sua grande maioria, é definido um conjunto de regras a serem seguidas pelo revisor. Esta análise é realizada por apenas um revisor por documento, e todos os dados considerados como relevantes por ele devem ser devidamente armazenados. Se porventura for um defeito, o revisor deverá criar uma CR.

Ainda de acordo com Ciolkowski, Laitenberger e Biffl (2003), a inspeção é composta por modelos que são divididos em fases, que diferem de empresa para empresa. Os métodos mais bem sucedidos, que foram mais difundidos e documentados na literatura, são os processos originais de Fagan (FAGAN, 1976) e o processo ligeiramente melhorado de Gilb (GILB; GRAHAM; FINZI, 1993). Há também vários outros métodos como (WIEGERS, 2002) e (ANSI/IEEE, 1989), porém não são tão utilizados como os mencionados anteriormente. A inspeção de SW e outros métodos de revisão têm sido objeto de pesquisa nas últimas décadas (LAITENBERGER; DEBAUD, 2000). Várias modificações estruturais nos processos, bem como técnicas para apoiar a execução do processo, foram sugeridas em diferentes estudos. Embora o processo de inspeção tenha amadurecido, ainda há várias questões em aberto que visam a realização do processo de maneira mais eficaz (AURUM A., 2002).

Contudo, apesar de sua importância, essa tarefa muitas das vezes não é realizada em empresas de SW devido à sobrecarga de trabalho ou falta de incentivos financeiros àqueles que são alocados para essa tarefa (MÄNTYLÄ; ITKONEN; IIVONEN, 2012), que são os próprios desenvolvedores ou testadores. Por este motivo, conseguir conciliar a atividade de inspecionar documentos com as atividades diárias torna-se um tarefa árdua.

Com base em tais afirmativa, este estudo será conduzido sob a ótica da inspeção. O documento utilizado não será o código fonte contido em sistemas de controle de versão como em (Vaucher, Sahraoui e Vaucher (2008); Fischer, Pinzger e Gall (2003); Śliwerski,

[1] Neste caso, as áreas que são características de um sistema, por exemplo: o sistema de um smartphone tem muitos recursos como: câmera, rádio, wifi, etc.

Zimmermann e Zeller (2005), e sim no artefato RN, uma vez que este documento é repleto de informações relevantes sobre a nova versão do *software* como as CRs que foram corrigidas e suas correções devidamente integradas a essa nova versão do SW. Portanto, um analista pode identificar áreas do SW que possam ser exploradas na busca por novas CRs (VAUCHER; SAHRAOUI; VAUCHER, 2008). Outro fator é a falta de estudos na literatura que utilizem este documento como fonte principal de dados, utilizando-o apenas como um documento de validação, como em Vaucher, Sahraoui e Vaucher (2008).

2.1 INSPEÇÃO DE DOCUMENTOS

A inspeção de documentos de SW é comprovadamente conhecido como um dos meios para a garantia da qualidade, sendo considerado como um processo categórico que envolve técnicas de análise manual (CUNHA, 2009). A inspeção de documentos de SW envolve uma verificação cuidadosa do código, do design e de outros artefatos de SW, verificando-os com o intuito de encontrar características que possam ser consideradas como problemáticas, através das experiência passadas dos analistas (KOMSSI et al., 2010). Equipes de engenharia de SW definem a inspeção como um processo formal que deve ser conduzido metodicamente (KOMSSI et al., 2010). Segundo Martin et al. (2007b) e Andersson e Runeson (2002b), essa formalidade descrita na literatura é imposta apenas por algumas empresas, sendo um dos aspectos que diminuem a popularidade dessa prática na maioria das organizações.

Segundo Tervonen e Iisakka (1993), a inspeção deve ser considerada como uma parte inseparável do processo de desenvolvimento de SW, ao qual todos os artefatos devem ser submetidos. Dentro do ciclo de vida de produção de um SW, como mostra a Figura 1, apenas algumas etapas de inspeção são executadas. Essas etapas podem variar de acordo com o modelo[2] escolhido.

[2] Termo utilizado na literatura para as abordagens da inspeção, que serão discutidas nas seções posteriores.

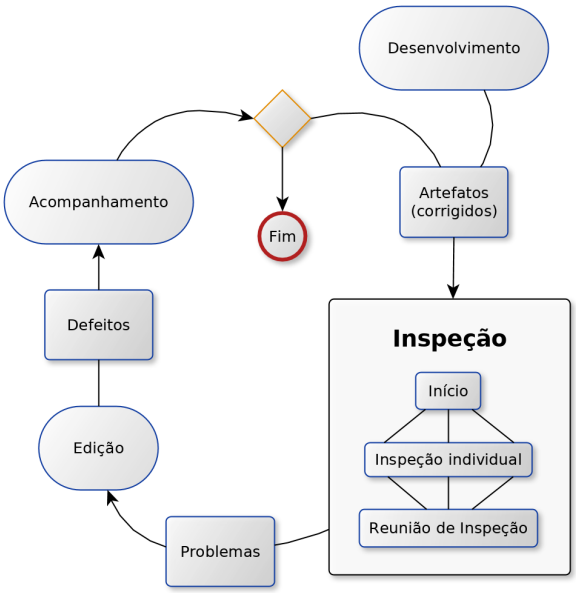

Figura 1 – O ciclo de vida central da produção de SW

Fonte: (LAITENBERGER; DEBAUD, 2000)

Em cada uma das etapas do processo de inspeção há um conjunto de metas a serem cumpridas pelo analista. Essas metas são definidas por meio de métricas prescritivas que descrevem quais ações os analistas devem tomar de acordo com o artefato analisado. Para isso, as inspeções devem ser adaptadas para se adequarem aos objetivos de cada equipe. Desta forma, é fundamental uso dos atuais modelos de inspeção e seus refinamentos para assim poder atender o processo, o produto à ser inspecionado, os papéis que os participantes têm em uma inspeção, assim como o tamanho de uma equipe, e as possíveis técnicas de leitura que podem ser utilizadas para a melhoria do desempenho de uma determinada equipe (LAITENBERGER; DEBAUD, 2000).

2.1.1 Modelo de inspeção manual abordagem de Fagan

Michael Fagan, há pouco mais de 40 anos, foi o responsável por introduzir na literatura o primeiro processo de inspeção de documentos de SW. O artigo original apresenta objetivos que vão além da detecção de erros nos documentos de software (FAGAN, 1976). Segundo Fagan (1976), as inspeções são “métodos formais, eficientes e de baixo custo que possuem o intuito de encontrar erros no projeto e no código”. A Figura 2, ilustra as cinco fases do processo de inspeção sugeridas por Fagan e seus respectivos objetivos.

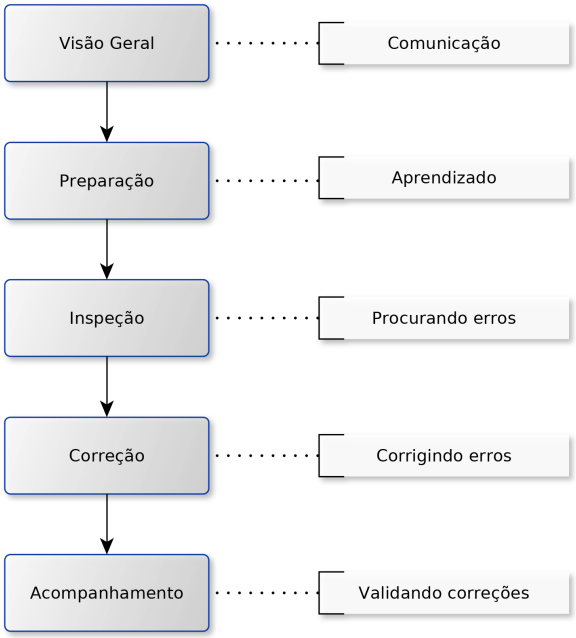

Figura 2 – Fases do processo de inspeção de Fagan

Fonte: (FAGAN, 1976)

De acordo com a Figura 2, é possível observar que o processo de inspeção segue uma estrutura metódica, como mencionado em Komssi et al. (2010), onde o revisor deve dar início ao processo respeitando a ordem das fases definidas. A primeira fase, chamada de visão geral, têm como principal objetivo tornar o produto inspecionado mais claro, ou seja, mais fácil de entender e de ser inspecionado pelos revisores. Esse primeiro contato com o artefato conta com a presença do criador do documento, que deverá explicar e tirar dúvidas sobre ele. No entanto, esta reunião inicial, conhecida como *Kickoff Meeting*, consome esforço e aumenta a duração de uma inspeção (FAGAN, 1976). Além disso, pode centrar a atenção dos inspetores em questões específicas, o que proíbe uma avaliação independente do artefato inspecionado (GILB; GRAHAM; FINZI, 1993). Já em contrapartida, Franz e Shih (1994) defende a relevância desta fase, sob argumento de que a visão geral faz com que todos os participantes da inspeção tenham a competência necessária para analisar facilmente o artefato inspecionado.

Na fase de preparação, acontece a leitura individual do documento sob inspeção. Segundo Fagan (1976), o principal objetivo desta fase é de apenas entender o artefato por completo, não tendo a detecção de defeitos como uma meta explícita. Tal definição casou opiniões divergentes, como em Gilb, Graham e Finzi (1993), que a considera como parte significativa, afirmando que os defeitos são encontrados já durante esta fase, colocação esta que também é apoiada mais tarde por Sauer C. e Yetton (1996). Outros autores tornam o entendimento dessa fase não tão claro, como em Doolan (1992), que descreve o objetivo da fase de preparação com um estudo individual de um dado artefato sem

nenhuma meta envolvida.

A fase de inspeção propriamente dita, tem como objetivo principal a busca por defeitos (LAITENBERGER; DEBAUD, 2000). Toda a equipe participa e o artefato sob inspeção é lido com o máximo de atenção. Soluções para problemas não são discutidas profundamente e são registradas apenas se forem óbvias (EBENAU; STRAUSS, 1994). Os defeitos encontrados são categorizados, e sua gravidade é determinada como maior ou menor (FAGAN, 1976). O grande problema dessa fase são as calorosas discussões criadas na literatura sobre, como validar a veracidade de um erro (LAITENBERGER; DEBAUD, 2000). O maior questionamento é se a inspeção com o objetivo de detecção de defeitos é mais uma atividade, ou seja, se deve ser executada de forma individual ou em grupo, havendo a necessidade de uma reunião de grupo (reunião de inspeção) (LAITENBERGER; DEBAUD, 2000). Em um estudo um pouco mais recente (KOMSSI et al., 2010), a alternativa dada para resolver esse impasse é apenas adaptar o uso da inspeção individual ou em grupo de acordo com tipo do artefato ou do objetivo da inspeção, fazendo reuniões de validação com um curto intervalo de tempo, para não comprometer o custo, ou simplesmente montar uma equipe que os participantes possuam a experiência necessária para a tomada de decisão (KOMSSI et al., 2010).

Ao término da fase de inspeção, um relatório escrito dos resultados deve ser concebido em um período de um dia (FAGAN, 1976). Este relatório é proveniente da tomada de decisão sobre se um defeito é realmente um defeito (LAITENBERGER; DEBAUD, 2000). O relatório criado servirá de artefato para as fases de correção e acompanhamento. O autor do documento inspecionado corrige os problemas detectados na fase de correção ou simplesmente justifica a sua existência (SHIREY, 1992). Como último passo, temos finalmente a fase de acompanhamento, onde é assegurado que todas as correções necessárias foram realizadas com maestria (DOOLAN, 1992). Também é possível que o documento precise de reinspeção parcial ou completa (CIOLKOWSKI; LAITENBERGER; BIFFL, 2003).

Um outro ponto que vai além das fases de inspeção do modelo de Fagan é o treinamento das pessoas envolvidas na inspeção. Fagan (1976) enfatiza que o treinamento adequado é necessário para que todos os processos sejam realizados corretamente e, consequentemente, melhorando o resultado da inspeção.

2.1.2 Modelo de inspeção manual abordagem de Gilb e Graham

O modelo de Gilb, Graham e Finzi (1993) é baseada no modelo principal de Fagan (1976). A principal característica do modelo diz respeito às fases de verificação e registro (HARJUMAA, 2005). No método de Gilb, Graham e Finzi (1993), a maior parte da detecção de defeitos é realizada durante a fase verificação, e a fase reunião é organizada para coletar e categorizar inicialmente as descobertas, fase esta defendida por Ebenau e Strauss (1994). Outras descobertas podem surgir durante a fase reunião, mas seu principal objetivo é validar as descobertas que foram realizadas individualmente (STåLHANE; TANVEER, 2005).

A Figura 3 ilustra as fases e os principais objetivos do processo de Gil e Graham.

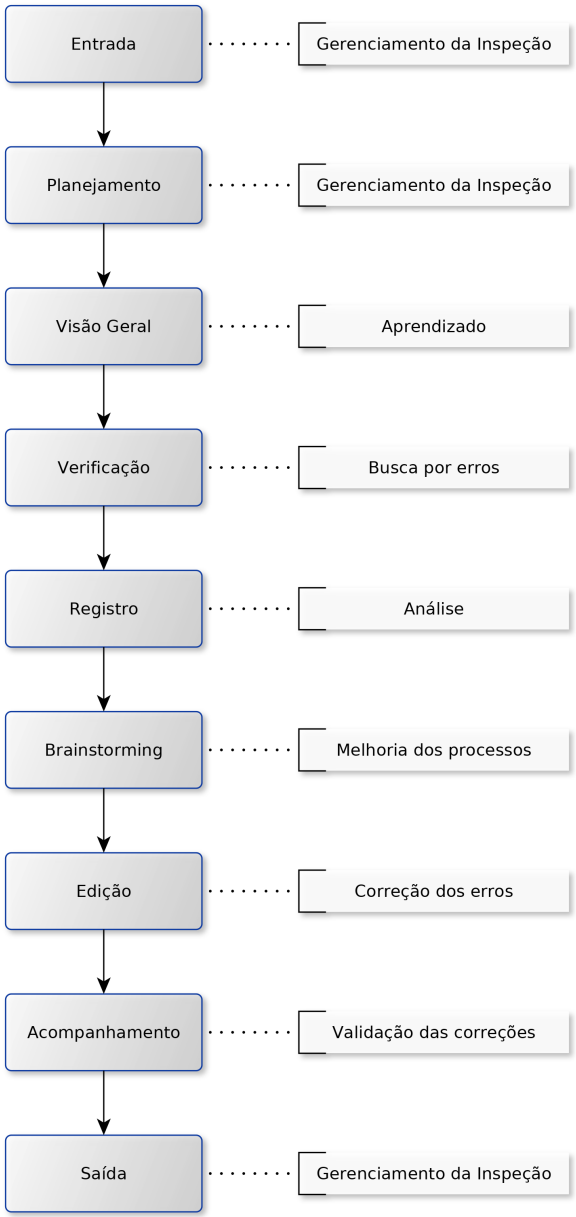

Figura 3 – Fases do modelo de Gil e Graham

Fonte: (GILB; GRAHAM; FINZI, 1993)

2.1.3 Modelo de inspeção manual - Processos Alternativos

Existem muitas variações do processo de inspeção, e a escolha do processo ideal torna-se difícil (WHEELER D. A.; MEESON, 1996). Desta forma, a heterogeneidade dos ambientes de trabalho e os diferentes tipos de artefatos fazem com que os processos alternativos se tornem a melhor escolha (HARJUMAA, 2005). A rigidez dos processos convencionais, motivou os pesquisadores a investigarem formas flexíveis de inspeção (LAITENBERGER;

DEBAUD, 2000). A flexibilidade do processo se dá através da adequação dos modelos existentes de acordo com a necessidade das empresas (CIOLKOWSKI; LAITENBERGER; BIFFL, 2003).

A fundamentação básica de um processo alternativo é a mudança aplicada a algumas fases como, as etapas do processo de inspeção, a quantidade de participantes em uma equipe, as ferramentas, os materiais de apoio aos inspetores e quais os reais papéis dos revisores (KOMSSI et al., 2010). Outro ponto importante é a relação entre as ações individuais aplicadas pelo inspetor e as reuniões conjuntas (HARJUMAA, 2005). O uso de diferentes documentos, também levantou discussões sobre as técnicas de leitura e o uso de ferramentas que pudessem automatizar o processo de inspeção (LAITENBERGER; DEBAUD, 2000).

2.1.4 Inspeção automatizada de documento

De acordo com as dificuldades encontradas no processo de inspeção e com a visão de torná-la mais eficiente, uma série de ferramentas e técnicas foram propostas segundo Harjumaa (2005). O mesmo autor ainda descreve em seu trabalho, que há dois aspectos em particular que foram ativamente estudados, que são:

- **Técnicas de leitura**: O uso de técnicas destinadas a inspetores individuais para ajudar a encontrar defeitos mais efetivamente;
- **Ferramentas de SW**: Ferramentas que ajudem a gerenciar o processo de inspeção e dados relacionados, permitindo que as inspeções possam ser realizadas de forma eficaz em relação às inspeções realizadas apenas manualmente.

Segundo Porter (1997), a variação de metodologia usada no processo de inspeção não afeta nos resultados relacionados à detecção de defeitos, e sim ajuda o inspetor individual a reduzir o seu esforço e melhorar sua eficiência. A escolha de uma abordagem de inspeção que reduza o esforço sem perder a sua qualidade é um desafio para os gerentes, dentro de um ambiente real de trabalho (BRIAND et al., 1998). Desta forma, para decidir as reais necessidades do uso da inspeção, isto é, para determinar se vale a pena gastar esforço para tal tarefa, deve-se considerar como a inspeção pode afetar a qualidade do produto de software, bem como o custo e a duração do projeto (LAITENBERGER; DEBAUD, 2000).

Uma ferramenta de inspeção é um sistema especialmente projetado para colaboração no processo de inspeção (HARJUMAA, 2005). Ainda segundo o autor, o uso de tais ferramentas é uma parte importante para ajudar na redução do esforço gerado no processo de inspeção manual. No âmbito geral, as ferramentas são desenvolvidas para fins de gerenciamento dos processos, detecção automática de defeitos e geração automática de relatórios como em Intelligent Code Inspection in a C Language Environment (ICICLE) (BROTHERS; SEMBUGAMOORTHY; MULLER, 1990) e Inspecting software in phases

to ensure Quality (InspeQ) (MACDONALD et al., 1995). Essa automação também pode contribuir para os registros de comentários incluídos diretamente no artefato inspecionado, classificar os defeitos de acordo com a sua severidade e categoria, gerenciamento de lista de verificação e relatório de eficiência de inspeção (LAITENBERGER; DEBAUD, 2000).

Além das funcionalidades mencionadas anteriormente, as ferramentas de inspeção devem ser capazes de integrar-se à infraestrutura existente e de lidar com todos as categorias de artefatos de SW (HEDBERG, 2004). Alguns produtos de SW proprietários também podem auxiliar na inspeção como, por exemplo, o Microsoft Word, que possui recursos que permitem a revisão distribuída de documentos (HARJUMAA, 2005) e o Adobe Acrobat, que possui recursos de compartilhamento de informação e anotação de documentos (HARJUMAA; TERVONEN; VUORIO, 2004).

Cada ferramenta de inspeção tem seus pontos fortes e cada implementação dá ênfase a algum aspecto específico dos modelos de inspeção de documentos. Tais ferramentas podem ser consideradas como um suporte computacional, podendo possuir um modelo próprio separado da inspeção tradicional, executada de maneira distribuída e inclusiva com fases assíncronas e síncronas (HARJUMAA; TERVONEN; VUORIO, 2004).

2.2 RN PARA PRODUTOS DE SW

A RN é uma importante fonte de informação sobre uma nova versão do SW, e tem o objetivo de resumir as principais alterações realizadas com relação à versão anterior, como: a implementação de novos recursos, quais *bugs* foram corrigidos, as mudanças de licença, dentre outros (LATONYA, 2013).

A disposição do conteúdo em uma RN é bem variada, uma vez que não existe um padrão de escrita para esse documento. Desta forma, cada comunidade de SW produz a sua RN de acordo com as suas diretrizes. Segundo Moreno et al. (2017), as RN, na maioria das vezes, são compostas por uma lista de informações que descrevem algum tipo de mudança.

As informações contidas na RN servem como fonte de informação para diferentes profissionais de uma organização (ABEBE; ALI; HASSAN, 2016), que vão desde o desenvolvedor até os engenheiros de teste. Essas informações ajudam tanto os profissionais quanto os usuários finais, a entenderem como a versão atual do SW está e quais meios podem ser observados para a uma tomada de decisão. Apesar da importância das RN para as empresas de SW, há poucos estudos na literatura que abordam suas reais contribuições dentro de um ambiente de trabalho com seu uso para prover garantia da qualidade do SW, que possibilita a identificação de *bugs* implícitos (MÄNTYLÄ; ITKONEN; IIVONEN, 2012).

2.2.1 RN caracterização

As RN são artefatos de SW que resumem as principais mudanças ocorridas em um ciclo de desenvolvimento, desde a sua versão anterior **(ver Seção 2.2)**. Elas proporcionam informações relevantes para os diferentes grupos envolvidos no desenvolvimento e na garantia da qualidade de um produto. Abaixo, temos a descrição de profissionais/usuários envolvidos e como eles utilizam as RN no seu ambiente de trabalho/uso:

- **Desenvolvedores**: extraem das RN quais alterações ocorreram no código, e os recursos que foram implementados e quais foram deixados para lançamentos futuros, para com isso aumentarem o seu poder de decisão e conhecimento sobre as próximas ações a serem tomadas (ABEBE; ALI; HASSAN, 2016).
- **Usuários finais**: lêe as RNs para decidirem a viabilidade de instalar uma nova versão do SW, levando em consideração os pré-requisitos de hardware e possíveis mudanças no comportamento final do sistema (GERMáN, 2004).
- **Equipes de teste**: extraem informações que contribuem para o aumento da produtividade na busca por novos comportamentos indesejados. Geralmente, os testadores olham para as novas funcionalidades que foram integradas, *bugs* que foram corrigidos nas versões anteriores e as soluções que foram adicionadas na nova versão (MÄNTYLÄ; ITKONEN; IIVONEN, 2012).

No geral, as RNs são produzidas de forma manual e no formato de um arquivo *HyperText* contendo um conjunto de linhas/campos que recebem informações diversas. A sua criação é considerada custosa, uma vez que diversos profissionais, incluindo desenvolvedores de software de código aberto, elucidaram que a criação manual de um documento como este leva em torno de oito horas para ficar pronto (MORENO et al., 2017). A abordagem denominada como Automatic RElease Notes generAtor (ARENA) (MORENO et al., 2017), precisou elencar quais os campos contidos em uma RN que são mais comuns de serem utilizados pelas intuições. Ainda no estudo, eles analisaram 990 documentos de RN com o intuito de classificar e ordenar os campos com maior ocorrência, como mostra a Tabela 1.

Tabela 1 – Campos contidos em documento de RN ordenados pelo percentual de ocorrência - Fonte: (MORENO et al., 2017)

Tipo de Conteúdo	RNs	%
Bug corrigido	888	90%
Novas Funcionalidades	455	46%
Novos componentes de código	424	43%
Componentes de código modificado	397	40%
Funcionalidades modificadas	262	26%
Operações de refatoração	206	21%
Mudanças na documentação	200	20%
Biblioteca atualizada	157	16%
Componentes de código reprovados	97	10%
Componentes de código excluídos	88	9%
Mudanças nos arquivos de conf.	84	8%
Mudanças de visibilidade nos componentes de código	72	7%
Mudanças nos conjuntos de testes	70	7%
Problemas conhecidos	64	6%
Componentes de código substituídos	47	5%
Mudanças de arquitetura	29	3%
Alterações em licenças	18	2%

Através desta análise, é possível observar o quanto de informações que devem ser coletadas para a criação de um documento de RN, reforçando a afirmativa dos profissionais envolvidos em sua criação que consideram como dispendioso o tempo para criá-la. É possível observar que o campo *bugs corrigidos* aparece na primeira posição com um percentual de ocorrência de 90%, sendo possível concluir que este campo é de suma importância para todas as instituições que utilizam o RN. Outra informação importante são as novas funcionalidades integradas na nova versão do SW, que aparecem em 46% dos documentos analisados. Esses dados podem proporcionar para as equipes de teste subsídios para explorarem novas áreas. Outra abordagem (BRANDON R., 2014) mostra a criação dos documentos de RN de forma customizada, dando ao criador do documento um certo poder para adicionar apenas os campos mais usados pela instituição. Brandon R. (2014), em sua proposta, conta com um assistente para construção da RN. Este assistente considera apenas os campos que os usuários indicam como relevante, possibilitando que os documentos de RN possuam apenas os campos mais usados pela instituição. As novas atualizações que houver nos próximos documentos também irão respeitar os mesmos critérios definidos na versão anterior da RN, proporcionando uma forma mais simples e rápida de analisar os campos mais utilizados.

Segundo Mäntylä, Itkonen e Iivonen (2012), as RNs também podem contribuir dire-

tamente para a descoberta de defeitos. O estudo realizado classifica quais documentos ajudam as equipes de teste na busca por falhas. Um questionário realizado com profissionais da área de teste, contendo perguntas sobre a quantidade de CRs que eles abriram em um ano e quais documentos mais influenciaram no resultado final, obteve as seguintes respostas:

1º) Casos de teste
2º) RNs
3º) Manual do Produto
4º) Apresentação do produto ou material de treinamento
5º) Requisito de produto ou especificação de recurso
6º) Especificação técnica do produto
7º) Mensagem ou relatório (*e.g.*, e-mail, bilhete de solicitação do cliente, etc.)
8º) Eu não estou usando nenhum material escrito
9º) Outros

De acordo com as respostas é possível observar que a RN aparece como um dos documentos utilizados na busca por defeitos (MÄNTYLÄ; ITKONEN; IIVONEN, 2012). Com isso, podemos observar que há uma contribuição relevante nas RNs e que existem vários esforços para melhorar, padronizar e automatizar a sua construção, respeitando os campos que mais sejam relevantes para observação. As análises feitas em seu conteúdo podem proporcionar a diversas equipes envolvidas no desenvolvimento de um SW um aumento na produtividade. Desta forma, é possível usar o documento na busca por defeitos implícitos ou ajudar a sugerir áreas para serem exploradas por casos de teste (MÄNTYLÄ; ITKONEN; IIVONEN, 2012).

2.2.2 RN para teste de SW

Atualmente, não há na literatura muitos trabalhos que citam a importância e o uso das RNs dentro de ambientes de teste de SW como um documento que pode ajudar na detecção de defeitos.

De acordo com Mäntylä, Itkonen e Iivonen (2012), não há como definir quais os documentos que podem fornecer um maior número de informações que conduzam um revisor a encontrar mais defeitos. Essa afirmativa se dá devido a alguns pontos que devem ser levados em consideração, como: o número de indivíduos que usam o documento, o número total de defeitos detectados usando o documento ou um número médio de defeitos detectados por indivíduos usando o documento. Apesar desta afirmativa, o mesmo autor, baseado em um conjunto de CRs que foram abertas no período de um ano por vários profissionais da área de teste (estudo realizado utilizando 4 empresas privadas), pôde elu-

cidar que documentos, incluindo a RN, desafiam o papel dominante dos casos de teste na detecção de defeitos, como apresentado na Tabela 2.

Tabela 2 – Quantidade de defeitos encontrados de acordo com o documento usado – Fonte (MÄNTYLÄ; ITKONEN; IIVONEN, 2012)

Documento	N	Total de defeitos	Parte total	Média de defeitos
Casos de teste	58	804	18.2%	13.9
RN	31	206	4.7%	6.7
Produto Manual	54	507	11.4%	9.4
Apresentação do produto ou material de treinamento	29	138	3.1%	4.7
Exigências do produto ou especificações de recursos	66	711	16.1%	10.8
Especificações técnicas do produto	42	352	7.9%	8.4
Mensagem ou relatórios, e.g., e-mail, defeitos reportados, ticket de clientes, solicitações	61	969	21.9%	15.9
Eu não estou usando nenhum material escrito	37	685	15.5%	18.5
Outros	2	56	1.3%	27.8
Todos os documentos	380	4427	100%	11.7

Podemos observar na Tabela 2 que a RN é mais uma vez citado como um documentos utilizados na busca por defeitos, onde 31 profissionais destacaram um total de 206 *bugs* encontrados utilizando este artefato. Esses profissionais responderam na entrevista que a RN influenciou positivamente na abertura da CR. Assim, mais uma vez, vemos a presença da RN influenciando na identificação de defeitos, mesmo sendo um documento com baixo percentual (4.7%) de eficacia. Além disso, o estudo também comprova de forma detalhada que o uso de outros documentos ajudam na produtividade de equipes de testes, não desmerecendo os casos de testes, que aparecem na primeira posição da Tabela 2.

Existem outras abordagens para o uso das RN. Por exemplo, Vaucher, Sahraoui e Vaucher (2008) examina técnicas que podem descobrir padrões de evolução em grandes sistemas orientados a objetos. Para localizar tais padrões, o estudo utilizou técnicas de agrupamento para identificar as classes que tiveram o mesmo comportamento de mudança. O estudo usa um conjunto de técnicas para descobrir se há semelhança entre grupos de classes que foram agrupadas. O mesmo estudo utiliza o documento de RN para validar os padrões de mudanças que foram selecionados. Os novos padrões encontrados são úteis na medida em que podem ser associados a atividades de manutenção de um SW, como correções de erros.

Já George (2016) utiliza as RN para compor um repositório de documentos que serve como base de treinamento para um algoritmo de classificação. O intuito do estudo é prover uma ferramenta capaz de predizer *bugs* utilizando técnicas de PLN, IA. Essa predição é gerada como resposta para uma entrada em formato textual para a ferramenta, onde a entrada pode ser um documento contendo *bugs* corrigidos, trechos de código, dentre outros.

Ambos trabalhos mencionados acima, tanto para a predição de bugs quanto para a identificação de padrões, deixaram a desejar no quesito de esclarecer quais os reais ganhos

obtidos através do uso da RN. A primeira proposta não descreve como a RN é utilizada para validar as mudanças de padrões, e nem quais campos foram utilizados para esse fim. Já a segunda proposta apresenta várias especulações de como usar técnicas de PLN, IA e Aprendizagem de Máquina (AM) para mitigar o problema proposto. Ainda neste estudo, George (2016) conduz um experimento em que ele armazena como dados de treinamento todo o conteúdo da RN. Desta forma, ele utiliza dados irrelevantes para o treinamento do classificador, uma vez que a sua proposta é predição de *bugs* e o campo mais indicado para treinamento seria o de *bugs* corrigidos apenas, como será descrito na proposta deste trabalho.

2.3 *CHANGE REQUEST*

Segundo Zhang et al. (2015), um relatório de *bug* é um documento/artefato de SW que descreve *bugs* de software, que pode ser submetido por um desenvolvedor, um testador ou um usuário final, podendo ser visto na literatura por várias nomenclaturas como *defect reports*, *fault reports*, *failure reports*, *change request*, dentre outras. A identificação e correção de *bugs* é considerada uma das atividades mais importantes para o desenvolvimento e manutenção do SW (XIA et al., 2014). Ainda segundo os autores, no geral, um relatório de *bug*, CR, contém muitos campos úteis, como status, resumo, descrição, repórter, produto, componente, resolução, gravidade, prioridade, sistema operacional (SO), versão, hora de criação e hora da modificação. Esses campos fornecem informações importantes para o processo de triagem e correção de erros (ZHANG et al., 2015), como mostra a Figura 4 e a Tabela 4 que possui uma breve descrição sobre os campos mencionados.

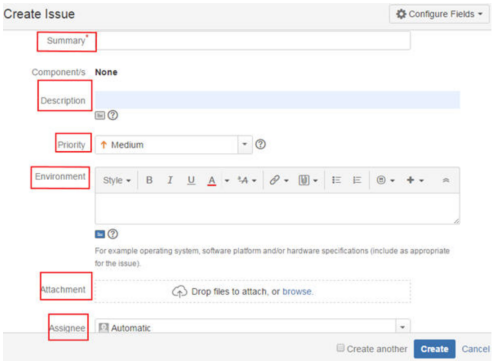

Figura 4 – Campos para abertura de CR no Jira

Fonte: Autor

Tabela 3 – Campos da CR – Fonte (XIA et al., 2014)

Campo	Detalhes	Exemplos
Status	status atual de uma CR que podem ser: “new”, “resolved”, “fixed”, “closed”, “invalid”, “duplicate”, etc.	“Resolved”,“Fixed”
Título	Texto em linguagem natural que é utilizada como uma breve descrição do bug.	.... falha no sistema
Descrição	Texto em linguagem natural que serve para descrever com mais detalhes todos os passos que conduziram o testador/desenvolvedor a encontrar o bug.	Teste falhou.... conjunto de mudanças: ...
Repórter	O desenvolvedor/testador que submeteu a CR	Alexandre Simmon
Produto	Produto afetado pelo Bug	Build versão 7
Componente	Componentes afetados pelo bug	Outros
Corrigido	Desenvolvedor que corrigiu o bug	Alexandre Simmon
Gravidade	Depois que o relatório de bug é enviado, o repórter consideraria sua gravidade para o sistema	Nulo
Prioridade	Depois que a CR é enviada e atribuída, os desenvolvedores devem considerar a prioridade para consertar esse bug	P2
Sistema operacional	O sistema operacional afetado pelo bug	Todos
Versão	Versão do código fonte onde o bug aparece	7.2
Hora da criação	A hora em que a CR é criado	06-03-3016 07:31
Hora de modificação	Hora em que o bug foi modificado	02-04-2016 9:31

É possível observar na Tabela 3 que os campos que possuem um quantidade significativa de texto em linguagem natural para a identificação do *bug* reportado são o titulo(*summary*) e a descrição (*description*). Desta forma, esses campos contribuem para que os desenvolvedores possam reproduzir, na maioria dos casos, os *bugs* descritos na CR.

De acordo com Thung et al. (2014) sempre que um usuário reporta uma nova CR, os desenvolvedores precisam ler o título e a descrição e, através desta informação, localizar manualmente os arquivos que possuem *bugs*. Ainda de acordo com os autores, eles descrevem que este processo, em sua grande maioria, são executados de forma manual e é frequentemente demorado e tedioso. Assim, vários estudos anteriores propuseram técnicas de localização de *bugs* para recuperar automaticamente arquivos potencialmente problemáticos reportados nas CRs.

Também é possível observar em Hooimeijer e Weimer (2007) a importância do título e da descrição da CR, um vez que a falta deste campos influenciam negativamente em análises onde se faz necessário o uso de linguagem natural para comparar informações de CRs atuais e dados históricos de uma determinada versão do SW.

2.4 CONSIDERAÇÕES FINAIS

Neste capítulo foi possível observar a necessidade da revisão de artefatos de SW para ajudar na garantia da qualidade. Também foi observado que a prática da inspeção de documentos pode ser aplicada de diversas formas e em diversos documentos. O documento apresentado como um dos artefatos mais importantes para um inspeção foi o RN, que

contém uma quantidade significativa de informação sobre as correções que foram implementadas em uma nova versão do SW e, com isso, fazer com que os testadores e todos que estão responsáveis na execução da inspeção possam gerar artifícios que os conduzam a encontrar novos *bugs* no SW em questão. No próximo capítulo serão apresentadas quais as principais técnicas na literatura que possam contribuir para que o processo da inspeção de documentos seja mais eficiente e automatizado. Uma vez que a maioria dos artefatos apresentados na **Seção 2.2.2** utilizam linguagem natural, há a necessidade da utilização de técnicas que estão inseridas no contexto da Mineração de Texto.

A inspeção de artefatos está tradicionalmente ligada à correção de defeitos no artefato inspecionado. Neste trabalho, utilizamos o termo inspeção de forma mais ampla, considerando-a uma atividade de análise de um artefato (RN e CRs) com o objetivo de produzir outro (Plano de Teste).

3 MINERAÇÃO DE TEXTO

Mineração de Texto é o processo de extração de conhecimento ou padrões significativos e não triviais de documentos de texto (TAN et al., 2000). Segundo Tan et al. (2000), dos 100% de informações referentes a ambientes empresariais, 80% estão armazenadas em formato textual, ao passo que apenas 20% dessa fatia são utilizadas para análises e nas tomadas de decisão organizacionais. Além disso, há diversas falhas nos resultados obtidos: documentos não relacionados, grande quantidade de informações irrelevantes, redundâncias, entre outros. A mineração de textos visa contribuir para mitigar essas falhas, oferecendo alguns meios que contribuem para a interpretação correta da linguagem natural dos textos, provendo formas de lidar com a sua imprecisão e incerteza (MACHADO et al., 2010).

A MT envolve vários segmentos da informática, como Estatística e PLN, sendo considerada uma área multidisciplinar (ARANHA C. E PASSOS, 2006 apud BARION E. E LAGO, 2008). Cada uma dessas áreas, ou a simples convergência entre elas, é capaz de transformar um simples texto em linguagem natural em uma estrutura que possa ser compreendida por uma máquina (HOTHO; NüRNBERGER; PAASS, 2005). As principais técnicas inseridas no contexto dessas áreas são a Extração de Informação (EI) (GAIZAUSKAS; WILKS, 1998 apud MACHADO et al., 2010), PLN (BERWICK; TENNY; ABNEY, 1991 apud MACHADO et al., 2010) e a RI (BAEZA-YATES; RIBEIRO-NETO, 2011). O presente trabalho irá discutir brevemente nas próxima seções as áreas de PLN e RI, uma vez que ambas fazem parte do escopo desta pesquisa.

De acordo com Martins et al. (2003), o processo de MT pode conter várias etapas, apontando apenas 4 como sendo as mais relevantes para o processo, que são: coleta de documentos (*i.e.*, formação da base de documentos, ou corpus), *pré-processamento* (*i.e.*, preparação ou transformação dos dados), *extração de conhecimento* (*i.e.*, busca de informações desconhecidas até o momento, mas que possam ser úteis para o domínio em questão) e *avaliação e interpretação dos resultados* (*i.e.*, análise humana). Em 2006 ARANHA C. E PASSOS (2006), em sua abordagem afirma que todo o processo de mineração de texto consiste em 5 etapas, que devem ser executadas de forma encadeada, seguindo a sequência ilustrada na Figura 5.

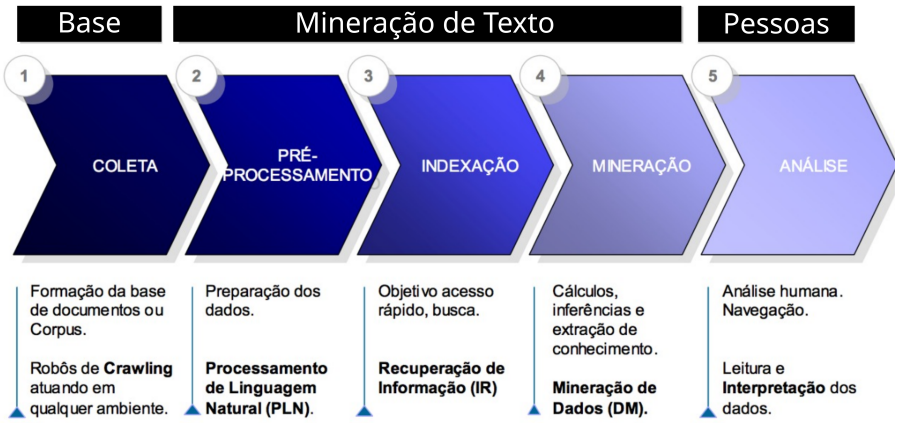

Figura 5 – Etapas do processo de MT

Fonte: (ARANHA C. E PASSOS, 2006)

Como é possível observar, há uma certa divergência entre as etapas citadas por Martins et al. (2003) e ARANHA C. E PASSOS (2006), onde a etapa de Extração de conhecimento foi substituída por outras duas etapas, a de Indexação (*i.e.*, um processo de RI) e de Mineração de Dados. O presente trabalho tem por base a abordagem definida por ARANHA C. E PASSOS (2006), ressaltando apenas uma reestruturação do processo que está representado na Figura 5 para melhores esclarecimentos das etapas nas seções posteriores.

3.1 COLETA

Definida como a primeira etapa da mineração de texto (ARANHA C., 2007), a coleta de dados tem como propósito formar o *corpus* (*i.e.*, coleção de documentos). É imprescindível que os documentos selecionados para compor a base estejam diretamente relacionados ao domínio do conhecimento a ser extraído, a fim de não dar origem a dados irrelevantes, nocivos à consistência do processo de Mineração (PEZZINI, 2017). Essa coleta pode ser feita de várias formas. Em Aranha C. (2007), a coleta dos documentos é feita de forma manual, analisando um conjunto de dados. Já em Junior (2003), a coleta de dados se dá de forma automática utilizando um *crawler* para extrair dados da Internet.

3.2 PROCESSAMENTO DE LINGUAGEM NATURAL

Segundo Aranha C. (2007), o PLN é uma das áreas de pesquisa mais importantes para a MT. Por definição, o PLN é a área de pesquisa que busca entender como os computadores são capazes de gerar e/ou analisar textos em linguagem natural, tendo como objetivo automatizar esses processos (PERNA C. L.; DELGADO, 2010). Portanto, o PLN pode ser entendido como um meio de comunicação entre um homem e uma máquina (TURBAN; MCLEAN; WETHERBE, 2004).

Com o objetivo de compreender o conteúdo e a estrutura gramatical dos texto, frequentemente ambígua, faz-se necessário criar representações mais simples que descrevem aspectos limitados da informação textual (CAMBRIA; WHITE, 2014). Segundo Collobert et al. (2011), na sua grande maioria, essas representações mais simples são motivadas por aplicações específicas que utilizam recursos/técnicas de PLN (*e.g.*, uso de variações da *bag-of-words*[1] para sistemas RI). Essas representações também podem descrever informações sintáticas obtidas através do uso de recursos/técnicas (*part-of-speech tagging*, *chunking*, e *parsing*) ou informações semânticas utilizando (desambiguação de sentido de palavra, rotulagem de papel semântico, extração de entidade nomeada, e resolução de anáfora) (CAMBRIA; WHITE, 2014)).

Todos os recursos/técnicas de PLN mencionados acima podem ser vistos como tarefas que atribuem rótulos às palavras (CAMBRIA; WHITE, 2014). Portanto, o PLN pode extrair da sentença textual um rico conjunto de recursos que podem ser utilizados por abordagens de MT e mineração de dados (JACKSON; MOULINIER, 2002).

As seções a seguir detalham as fases do PLN, com mais ênfase no pré-processamento do texto.

3.2.1 Pré-processamento

Segundo GOMES (2008), as coleções de textos devem ser submetidas a um Sistema de Mineração de Texto (SMT). Uma vez realizada a coleta dos dados, é necessário criar representações internas bem estruturadas, de modo a alimentar os algoritmos subsequentes. Nessa etapa os documentos passam por uma formatação para estruturá-los de maneira padronizada, mas sem perder suas características naturais. Desta forma, os algoritmos que serão utilizados nas próximas etapas serão capazes de manipular todos os documentos da mesma maneira (PEZZINI, 2017).

Tal processo é composto por alguns estágios. Segundo Runeson, Alexandersson e Nyholm (2007), os passos/processos utilizados na transformação de um documento são: tokenização, *stemming*, remoção de *stopwords* e criação da representação vetorial do documento. O presente trabalho irá elucidar com mais clareza alguns desses estágios nas seções seguintes.

3.2.1.1 Tokenização

A tokenização, também conhecida como atomização (FINATTO, 2005), examina um texto e particiona-o pelos termos (tokens) que o compõem em uma estrutura linear, preservando o conteúdo original do documento. Um token não necessariamente representa uma única palavra, podendo haver um agrupamento de vários tokens que representam termos compostos (*e.g.*, Rio de Janeiro)(CARRILHO, 2008).

[1] Técnicas de extração dos unigrams (um termo) do texto para criar uma lista desordenada de palavras (MANNING; RAGHAVAN; SCHUETZE, 2009).

Frase
Camera crashed while I was taking a picture
Representação pós-tokenização
|Camera| |crashed| |while| |I| |was| |taking| |a| |picture|

Figura 6 – representação pós tokenização

Fonte: autor

Na tokenização são utilizados delimitadores, como por exemplo, espaços em branco, quebras de linhas, tabulações e alguns caracteres especiais, sendo estes considerados tokens delimitadores (FELDMAN; SANGER, 2006). Sendo assim, pontuações e símbolos podem ser utilizados para dividir e/ou melhorar o entendimento dos tokens de acordo com o domínio escolhido. Por exemplo, o texto contido na Figura 6 "Camera crashed while I was taking a picture"é dividido em tokens da seguinte forma: |Camera| |crashed| |while| |I| |was| |taking| |a| |picture|.

3.2.1.2 Remoção de Stopwords

A remoção de stopwords envolve a eliminação de palavras irrelevantes para a recuperação dos documentos (BRITO, 2016). Tais palavras não são capazes de discriminar os documentos, tornando-as irrelevantes para alguns processos de RI e de AM, podendo ser chamadas de *palavras vazias* (WIVES LEANDRO E LOH, 1998). O mesmo autor ainda afirma que tais termos são úteis apenas para o entendimento e compreensão geral do texto. Classes de palavras como pronomes, artigos, conjunções e preposições são alguns dos exemplos que se encaixam na lista de stopwords (PEZZINI, 2017). Em alguns casos, há palavras cuja ocorrência é consideravelmente alta, aparecendo em quase todos os documentos de uma coleção. A identificação e a remoção dessa lista reduzem bastante o tamanho final do léxico, e consequentemente ampliam o desempenho do sistema em termos de consulta (CARRILHO, 2008). A Figura 7 mostra a evolução da Figura 6 com a utilização do procedimento de remoção de stopwords[2].

[2] Os exemplos apresentados no Capítulo 3 utilizam a língua inglesa, uma vez que o protótipo que será apresentado foi construído respeitando as normas gramaticais do idioma citado.

Frase
Camera crashed while I was taking a picture
Representação pós-tokenização
|Camera| |crashed| |while| |I| |was| |taking| |a| |picture|
Representação pós-Stopwords
|Camera| |crashed| |while| |I| |was| |taking| |a| |picture|
|Camera| |crashed| |taking| |picture|

Figura 7 – representação pós stopwords

Fonte: autor

De acordo com a Figura 7 é possível observar que o processo de remoção das stopwords possibilitou limpeza das *palavras vazias* que não iriam gerar valor para o pré-processamento do corpus.

3.2.1.3 Stemming

Segundo Jackson e Moulinier (2002), os *stemmers* são analisadores morfológicos que associam variantes do mesmo termo à sua forma raiz, utilizando na maioria das vezes a eliminação dos prefixos e sufixos, como plural e flexões verbais. Ainda de acordo com o autor, esta forma raiz pode ser considerada como o termo principal encontrado em um dicionário. Por exemplo, na língua inglesa os termos, 'go', 'goes', 'going', 'gone' e 'went' serão associado à sua forma raiz 'go'.

De acordo com Brito (2016), como se trata de um processo heurístico, que tem o intuito de remover as extremidades das palavras com a tentativa de alcançar um objetivo específico, os algoritmos utilizados nos *stemmers* não levam em consideração o contexto no qual as palavras se encontram. A Figura 8 a seguir utiliza a língua inglesa, uma vez que toda a pesquisa foi desenvolvida utilizando tal idioma.

Frase
Camera crashed while I was taking a picture
Representação pós-tokenização
|Camera| |crashed| |while| |I| |was| |taking| |a| |picture|
Representação pós-Stopwords
|Camera| |crashed| |while| |I| |was| |taking| |a| |picture|
|Camera| |crashed| |taking| |picture|
Representação pós-Stemming
|Camera| |crashed| |taking| |picture|
|Camer| |crash| |tak| |pictur|

Figura 8 – representação stemming

Fonte: autor

De acordo com a Figura 8 é possível observar que os tokens foram reduzidos para o seu radical.

3.2.2 Part-of-speech tagging

Segundo Jackson e Moulinier (2002), *Part-of-speech* (POS) *tagging* tem por objetivo rotular cada token de um corpus com o seu respectivo papel morfossintático, por exemplo, substantivo, verbos, advérbios, etc.

Frase
Camera crashed while I was taking a picture
Representação pós-POS tagging:
('Camera', 'NN'), ('crashed', 'VBN'), ('while', 'IN'), ('I', 'NN'),
('was', 'VBD'), ('taking', 'VBG'), ('a', 'DT'), ('picture', 'NN')

Figura 9 – representação pós-Pos tagging

Fonte: autor

Como mostrado na Figura 9, o uso da técnica liga o token ao seu respectivo papel morfossintático na frase acrescentando um rótulo, que nada mais é que a abreviatura de sua respectiva classe gramatical.

Embora o *POS tagging* seja muito utilizado, a ambiguidade nos idiomas atrapalha muitas das vezes esses algoritmos fazendo com que as rotulagens sejam equivocadas (JONES; SOMERS, 2013). Portanto, na literatura há uma quantidade significativa de estudos (RATNAPARKHI, 1996; OWOPUTI et al., 2013; GIMPEL et al., 2010) que possuem como objetivo principal o aprimoramento dessas técnicas, para assim realizar de forma mais precisa as rotulagens morfossintática em corpus de texto (JONES; SOMERS, 2013).

3.2.3 Chunking

Também chamado de rotulagem de papéis sintáticos, o *chunking* tem por objetivo rotular com papéis sintáticos os segmentos de uma sentença, como sintagmas nominais ou verbais (Nominal Phase (NP) ou Verbal Phase (VP)) (DIAB, 2009). Cada token recebe apenas uma tag única, representada por um identificador Beginning Inside Outside (BIO) da seguinte forma: tag de início (*e.g.*, B-NP), tag de parte interna (*e.g.*, I-NP) e tag de fora do argumento (*e.g.*, O) (HACIOGLU et al., 2004). Estas tags indicam quais sentenças fazem parte dos grupos/sintagmas indicados, como por exemplo, na criação de um sintagma nominal ou grupo nominal, os termos que dão inicio ao grupo recebem a tag "B"(*Beginning*), para os termos que fazem parte do grupo/sintagma e não encontram-se no inicio, são demarcados com a tag "I"(*Inside*) e todos os outros termos que não fazem parte do grupo/sintagma recebem a tag "O"(*Outside*).

Frase
Camera crashed while I was taking a picture
Representação pós-Chunking
('Camera','B-NP'), ('crashed','VP'), ('while', 'O'), ('I','B-NP'), ('was', 'VP'), ('taking', 'VP'), ('a', 'B-NP'), ('picture', 'I-NP')

Figura 10 – representação *chunking*

Fonte: autor

A Figura 10 mostra claramente como o *chunking* funciona, a técnica rotula cada termo da frase utilizando o marcador definido no algoritmo.

3.3 INDEXAÇÃO E BUSCA

A indexação é um processo em que o conteúdo dos documentos é convertido em um formato que permite a recuperação rápida de elementos neles contidos (BINDÁ; BRANDT; PIEDADE, 2013). Esse processo gera um índice como saída, que na sua grande maioria pode ser uma estrutura de dados que possibilita o acesso aos termos de cada documento, tendo a mesma funcionalidade de um índice de um livros (CORREIA, 2018). O desempenho de um sistema de RI está inteiramente ligado à qualidade do processo indexação (JONES; WILLETT, 1997).

Os índices, produtos do processo de indexação, são estruturas de dados antigas e bastante difundidas (BAEZA-YATES; RIBEIRO-NETO, 2011). Um índice equivale a uma coleção de palavras associada ao documento. A construção de forma eficiente de tais índices contribui para um alta performance nas consultas e, consequentemente, em uma maior qualidade no conjunto de respostas (GONZALEZ, 2002).

A construção de um arquivo de indexação se dá através de 3 técnicas, que são: Arquivos invertidos, árvore de sufixo e os arquivos de assinatura (BAEZA-YATES; RIBEIRO-NETO, 2011). Ainda segundo o autor, os arquivos de assinatura são construídos dividindo os documentos em blocos, onde cada bloco contém uma quantidade determinada de palavras. Ainda segundo o mesmo autor, cada palavra recebe uma assinatura, que é obtida através de uma função de hash. A concatenação dos blocos de assinatura geram o arquivo de assinatura, que é utilizado para a busca dos termos contidos no documento.

De acordo com Gonzalez (2002), um dos mecanismos que possui uma maior complexidade na sua construção e manutenção, embora a sua rapidez nas operações é considerada como muito boa é a árvore de sufixo. As árvores de sufixo são construídas a partir de todas as possíveis subsequências de caractere do texto (começando em um determinado ponto do texto) (BAEZA-YATES; RIBEIRO-NETO, 2011). Ainda de acordo com o autor, em um texto com n caracteres, a árvore irá conter n nós folha e n-1 nós internos, os arcos indicam a representação do caractere. Já os nós internos representam o número de posições entre

o caractere anterior e o posterior. Cada um dos nós folhas indicam a posição no texto da sequência que representa o caminho do nó-raiz até o nó-folha.

O índice invertido (ou arquivo de índice invertido) é, segundo Gonzalez (2002), a técnica mais utilizada atualmente, que consiste de um processo orientado à palavra para a indexação de um conjunto de documentos textuais, tendo como objetivo dar agilidade ao processo de busca. Segundo Jones e Willett (1997), os arquivos invertidos possuem dois componentes principais que são: *Dicionário de termos* (lista de todas as palavras-chaves, todos os termos classificados, em uma base que possa ser usada para recuperação) e os *Arquivos de endereçamento* (conjunto de listas que possuem identificadores de todos os documentos que contêm o termo corrente, onde cada lista está ligada a uma entrada do dicionário, também é corriqueiro que seja armazenado além da presença dos termos no documento, a sua localização no mesmo, desta forma, dando eficiência no uso da busca).

A Figura 11, descreve as etapas que levam a indexação e consequentemente a criação dos índices.

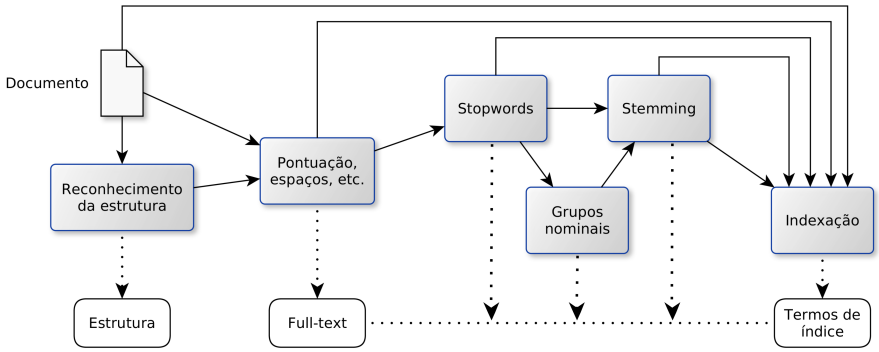

Figura 11 – Construção de termos de índice

Fonte: (GONZALEZ, 2002)

As etapas apresentadas na Figura 11 são responsáveis pelo pré-processamento da estrutura textual dos documentos que foram previamente selecionados, uma vez que, para a indexação, é necessário que todos os documentos estejam devidamente estruturados (BAEZA-YATES; RIBEIRO-NETO, 2011). A indexação irá filtrar o texto, através de um processo de identificação, as principais características do documento a ser indexado. Esta etapa pode ser conduzida por um Sistema de Recuperação de Informação (SRI) de três formas: a tradicional, quando os termos são selecionados de forma manual, o *full-text*, quando os termos que compõem o documento são utilizados como parte do índice e usando tags, quando os termos são selecionados de forma automática (BLUMETTI et al., 2007).

Como mencionado anteriormente a indexação está totalmente ligada à garantia da praticidade de um sistema de RI que nada mais é do que um ambiente mediador da comunicação entre fontes de informação e os seus solicitantes (GONZALEZ, 2002). Garantir eficiência em um SRI depende do controle apropriado da linguagem utilizada na repre-

sentação dos itens de informação, se os documentos foram devidamente indexados com seus índices criados corretamente e das pesquisas dos usuário (FERNEDA, 2003).

Segundo Cardoso (2004), a RI é uma área da Ciência da Computação orientada a estudar técnicas que tratam da representação, armazenamento organização e acesso a itens de informação, como documentos, catálogos online, registros semiestruturados e estruturados, etc, provendo aos usuários o acesso fácil às informações de seus interesses. Complementando a afirmativa acima, de acordo com Machado et al. (2010) a RI pode ser considerada como uma busca por documentos que utiliza um conjunto de medidas e métodos estatísticos para processar automaticamente os dados textuais. Ainda segundo Machado et al. (2010), a RI se baseia em perguntas e respostas, tendo como objetivo recuperar documentos como na maioria dos motores de busca, onde tal recuperação pode se dar através de sistemas baseados em palavras-chaves.

De acordo com Baeza-Yates e Ribeiro-Neto (2011), a RI vai além do que foi citado anteriormente, tendo atualmente pesquisas voltadas às áreas de *classificação de texto*, *arquitetura de sistemas*, *modelagens*, dentre outras. Ainda de acordo com os autores, a RI pode ser dividida em duas grandes áreas de pesquisa, que são: *centrada no computador*, que possui ênfase na construção de índices eficientes (já mencionado anteriormente no processo de indexação), no processamento das consultas e no desenvolvimento de algoritmos que sejam capazes de atuarem para o ranqueamento dos documentos; *centrado no usuário*, que consiste em analisar o comportamento do usuário no sistema de RI, para assim entender as suas principais necessidades, e determinar como esse comportamento pode afetar o sistema. Em ambas as áreas, a recuperação dos documentos de forma prática e eficaz é o ponto principal do uso da RI.

A Figura 12 mostra com detalhes um ciclo básico da tarefa de RI

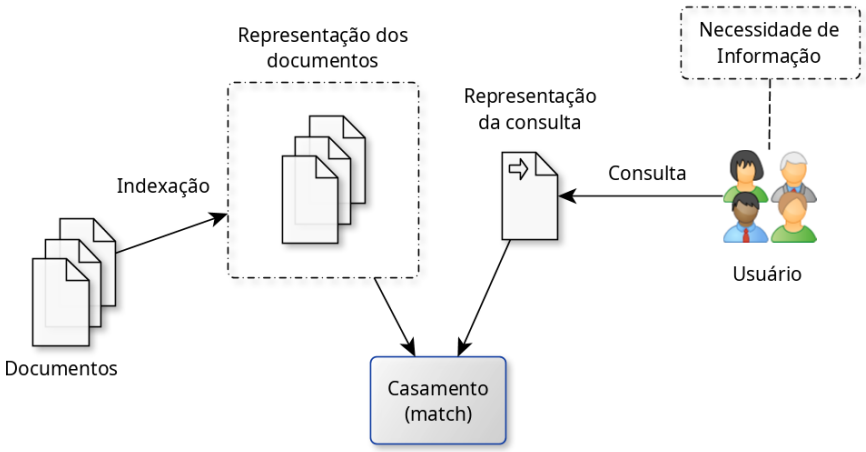

Figura 12 – Tarefa de RI

Fonte: autor

Na Figura 12 é possível observar duas fases na tarefa de RI que são, a representação

do documento (*i.e.*, índice invertido derivado do processo de indexação) e a representação da consulta (*i.e.*, na sua grande maioria é composta por um texto em linguagem natural definido pelo usuário para busca de documentos). Em um SRI como já mencionado anteriormente, se faz necessário a preparação para a criação da base de índices que foi mostrada na Figura 10. Com a base de índices criada, uma consulta pode ser feita a esta base. A Figura 13 mostra como um SRI se comporta na execução de consulta à base de índices.

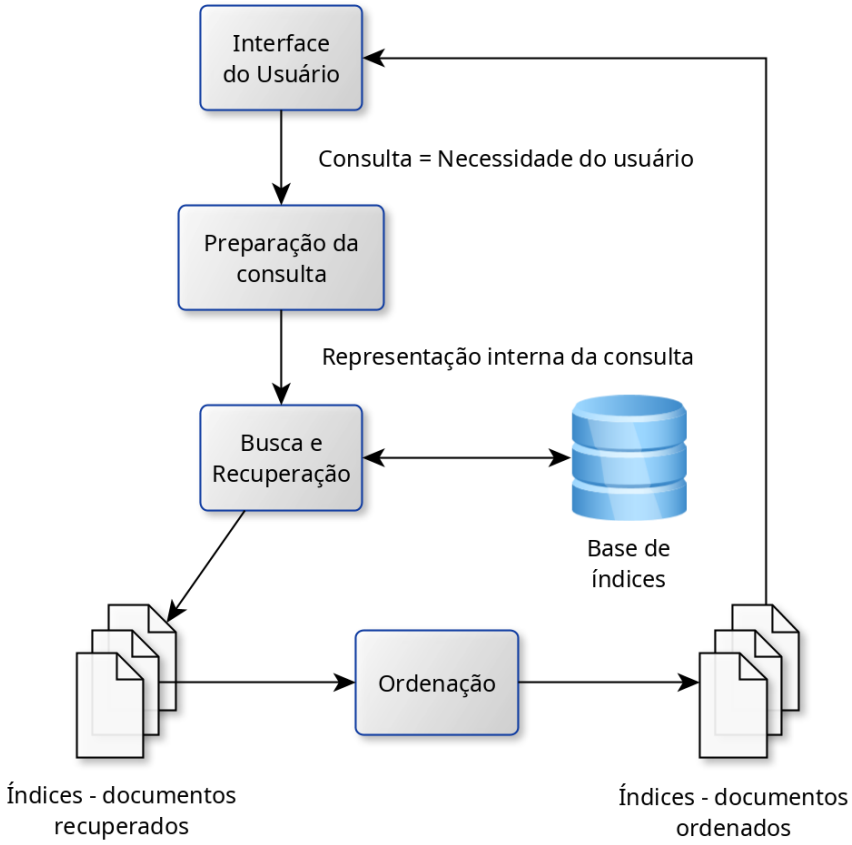

Figura 13 – Consulta à base de índices

Fonte: autor

De acordo com a Figura 13, o usuário constrói a consulta (query) para fornecer ao SRI como entrada de busca. Após a busca, os documentos que fizerem parte do casamento de padrão (match) com a entrada do usuário serão recuperados, ordenados e entregues ao usuário (FERNEDA, 2003).

Pelos motivos citados acima, vemos que os SRI com a MT se completam. Os SRI contribuem auxiliando a busca por informações úteis/válidas para um determinado usuário, em meio a uma grande quantidade de dados. Desta forma, há a necessidade da utilização da MT em conjunto aos SRI devido ao fato de haver atualmente uma grande quantidade de dados textuais, desestruturados ou parcialmente estruturados que precisam ser

recuperados de forma eficiente.

3.4 MINERAÇÃO

A Mineração consiste na aplicação de técnica voltadas para a AM com o intuito de obter novos conhecimentos (WITTEN; FRANK; HALL, 2011). De acordo com Feldman e Sanger (2006), a mineração surge da necessidade de organizar e padronizar automaticamente textos visando melhorar a análise dos mesmos. Nessa etapa na MT é possível escolher qual deverá ser a tarefa mediante a necessidade do usuário (Brito, 2016). Ainda segundo o autor, se a necessidade do usuário for analisar a similaridade e a construção de grupos naturais, a melhor tarefa a ser utilizada é o processo de clusterização, mas se por outro lado, estes grupos já forem conhecidos, seja através de uma análise prévia de um especialista ou pela execução de algoritmos de extração de conhecimento, a orientação de qual rótulo o documento deve receber para ser etiquetado de acordo com o seu conteúdo, é conseguida através do uso de algoritmos de classificação.

3.4.1 Classificação de texto

A classificação textual é uma das áreas de pesquisas de RI que ajudam a rotular esses documentos e assim, contribui de forma eficiente e inteligente para a sua recuperação, provendo um meio que permite organizar as informações, melhorando a compreensão e interpretação dos dados (MADUREIRA, 2009).

Segundo Baeza-Yates e Ribeiro-Neto (2011), a classificação textual consiste na identificação da classe de um documento a partir do seu conteúdo. Tal conteúdo estará relacionado a uma classe/categoria que servirá para rotular o documento e assim agrupar os documentos de uma base de forma coerente de acordo com o seu tópico. Desta forma, é correto afirmar que o processo de categorização dos documentos em classes, podendo associar o mesmo documento a um ou mais rótulos, é intitulado como classificação de texto (MADUREIRA, 2009).

Existem duas grandes abordagens para a construção de classificadores: manual, baseada em regras explícitas; baseada em aprendizagem de máquina. Os da abordagem de aprendizagem de máquina podem ser divididos em duas grandes áreas: supervisionadas (*i.e.,* a princípio, as classes dos documentos são conhecidas) e não supervisionada (*i.e.,* não existem dados rotulados para o treinamento) (MCCALLUM; NIGAM, 1999).

Quando há a necessidade de rotular manualmente grandes quantidades de dados de treinamento, ou simplesmente, quando esses dados não estão disponíveis, é utilizada a abordagem da classificação baseada em regras explícitas com um conjunto de palavras chaves (MCCALLUM; NIGAM, 1999). Ainda de acordo com esses autores é mais fácil e rápido criar um conjunto regras baseadas em palavras chaves do que rotular um grande conjunto de documentos manualmente.

Segundo Camilo (2009), a classificação baseada em regras é composta por uma estrutura condicional *SE* junto a uma estrutura conclusiva *ENTÃO*, assemelhando-se a uma regra de associação[3]. Ainda segundo esse autor, esse modelo de classificação, em sua grande maioria, é recuperado de uma árvore de decisão, para interpretar os seus resultados de forma menos complexa. De acordo com Paulo (2013), o modelo de classificação baseada em regra é concebido da seguinte forma:

SE **(determinada condição é verdade)** *ENTÃO* **(faça determinada ação)**

Onde o *antecessor* da regra é o lado que contém a condição e o *consequente* da regra é o lado que contém a ação. O mesmo autor ainda descreve que no lado do *antecessor* é comum encontrarmos mais de uma atributo, desta forma, uma dada regra r cobre um determinado registro x se os atributos de x satisfazerem o *antecessor* r.

De acordo com Cazes e Feitosa (2018), as regras para tal modelo de classificação devem ser definidas por um especialista de domínio, uma vez que o processo deve ser similar a uma classificação manual. O mesmo autor ainda descreve que o conjunto de regras possuem a finalidade de expressar de forma objetiva o raciocínio humano usado durante um procedimento manual.

3.5 ANÁLISE

A análise/interpretação dos dados resultantes dos processos de MT se dá através da validação da eficiência dos processos como um todo. Em outras palavras, a análise permite avaliar o objeto resultante dos processos para descobrir se um novo conhecimento foi adquirido, a partir dados analisados (BRITO, 2016). Desta forma, é correto afirmar que a análise precisa da intervenção de um especialista de domínio para descobrir se os resultados obtidos atendem às necessidades impostas e, caso negativo, quais etapas do processo de MT poderiam ser refeitas para melhorar os resultados (MARTINS et al., 2003).

3.6 CONSIDERAÇÕES FINAIS

Neste capítulo, foram abordados os conceitos elementares sobre Mineração de Texto, Recuperação de Informação e Processamento de Linguagem Natural. As ideias consideradas aqui foram estruturadas de modo a estabelecer diretrizes para o desenvolvimento de um protótipo que abarque os conhecimentos acima citados. Os conceitos de MT contribuíram de modo a dar a conhecer a necessidade do usuário frente a uma recuperação inteligente e eficiente de documentos textuais e os subsídios que são fornecidos para atendimento dessa necessidade, em termos computacionais. Os conceitos de MT ajudaram no tocante às in-

[3] Buscar por padrões de associações entre um determinado item e uma categoria, utilizando o conceito de frequência de temos (CAMILO, 2009).

ferências de conhecimento que se pode obter a partir da análise de certas características em dados textuais, com o intuito de aprimorar os procedimentos de busca em um SRI. Por último, vem a Aprendizagem de Máquina, a qual contribui fornecendo conhecimentos da aplicação de algoritmos inteligentes para detecção de padrões, cujas técnicas são utilizadas para automatizar os procedimentos de Mineração. A seguir, vem a Descrição e Desenvolvimento da Proposta, com o intuito de apresentar as motivações que culminaram na realização deste trabalho.

4 SPT: UM PROCESSO PARA CRIAÇÃO DE PLANOS DE TESTE EXPLORATÓRIO

O processo de verificação de uma nova versão de um SW contém várias fases. Essas fases, na maioria das vezes, são definidas de acordo com a técnica de teste escolhida pelo engenheiro de SW. Tais fases condizem com a área de atuação ao qual a aplicação está inserida, e com o nível de maturidade dela. Em geral, a manutenção de uma nova versão de SW é um processo que pode ser dividido em 3 fases:

- **Fase de criação do Plano de Teste** - Consiste em escolher quais técnicas e casos de teste serão usados para testar o SW. Normalmente, esta fase é executada manualmente, sendo assim muito demorada.
- **Fase de teste** - Os casos de teste no Plano de Teste são executados, a fim de identificar erros e defeitos no SW.
- **Fase de correção de *bugs*** - Os desenvolvedores corrigem o código de acordo com as CRs atuais. O novo código é então integrado à última versão do SW e as CRs corrigidas são mencionadas no documento de RNs (no campo *bugs* corrigidos).

Na primeira fase, dentro de um ambiente de produção, é possível identificar alguns problemas que interferem na qualidade do processo. Um desses problemas se refere à dificuldade em escolher os casos de teste que melhor atendem os erros reportados nas CRs atuais do SW sob teste, uma vez que testá-lo por completo não é uma boa prática, devido aos altos custos associados à tarefa de teste. O engenheiro de teste pode se deparar com milhares de testes disponíveis para diferentes linhas de produtos, tendo que escolher apenas aqueles que mais se adequam à nova versão do SW.

Outro fator que influencia no processo de criação do plano de teste é o tempo para criá-lo. Geralmente, a seleção dos casos de teste é feita de forma manual, resultando em um gasto de tempo diário por parte do engenheiro de teste ou do testador responsável para executar tal tarefa.

Uma boa prática indicada no auxílio à criação de planos de teste mais objetivos, que possam cobrir áreas mais propensas a defeitos de uma dada versão do SW, é a inspeção de documentos. Com essa inspeção, os engenheiros e testadores conseguem conhecer a nova versão do SW e consequentemente podem escolher casos de teste que possam ser mais eficazes. O grande problema que envolve a implantação do processo de revisão nas empresas se dá devido à maneira de execução da análise, que na sua grande maioria é feita de forma manual. Quando unimos o tempo necessário para realizar a revisão de documentos com o tempo gasto na escolha dos casos de teste, esse processo tende a tornar-se uma prática inviável para os ambientes de produção.

Com o intuito de contribuir para a melhoria do processo de criação de planos de teste, propomos um novo processo que irá auxiliar as equipes de testes exploratórios na criação de seus planos. Dentro da abordagem proposta, os planos de teste são construídos com base em palavras-chave relacionadas a áreas relevantes do SW que foram mencionadas em documentos de RN de uma nova versão de SW (e, portanto, devem ser exploradas por testadores). Em particular, focamos em áreas extraídas de comentários sobre CRs nas RNs.

O processo proposto consiste em 5 etapas (Seção 4.1): (1) Recuperação e Extração de dados a partir das RNs, (2) Classificação de CRs, (3) Seleção de áreas relevantes, (4) Seleção de casos de teste (*charters*), e (5) Criação de planos para SWAT (*SW analyze testing*).

Com base nesse processo, um protótipo foi criado para auxiliar as equipes de teste exploratório a criar planos de teste mais adequados, com foco em áreas consideradas mais suscetíveis a falhas. O SPt (SWAT Plan Tool) foi implementado usando a linguagem Python (LUTZ; ASCHER, 2007), e se baseia em métodos de Mineração de Texto e Recuperação de Informação, bem como de Processamento de Linguagem Natural e classificação de texto.

4.1 VISÃO GERAL

O processo proposto foi inspirado no trabalho de uma equipe de testes exploratórios, que inspeciona manualmente as RNs geradas diariamente para as novas versões de SW a serem testadas. Em seguida, eles criam planos de teste com base nessa inspeção.

A seguir, apresentamos o processo para criação de planos de teste, implementado pelo protótipo SPt. A Figura 14 mostra o diagrama de fluxo de dados do processo proposto.

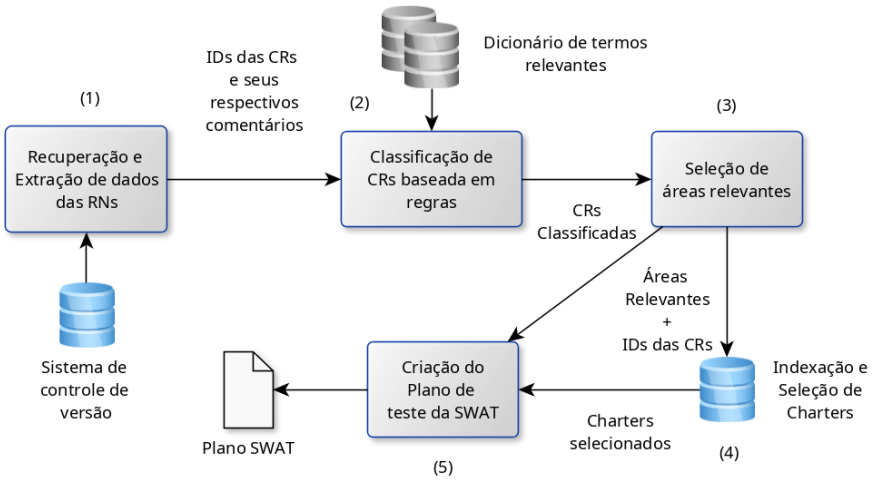

Figura 14 – Processo de criação de planos de teste

Fonte: autor

Abaixo, temos uma breve descrição das 5 etapas do processo apresentado na Figura 14, que serão detalhadas nas seções a seguir.

1. **Recuperação e extração de dados** - coleta os documentos de RN relacionados ao SW sob teste a partir de um repositório, e extrai as informações textuais (*e.g.*, ID da CR corrigida e os seus respectivos comentários) contidas na áreas de *bugs* corrigidos. Essas informações serão passadas para a etapa de classificação de CRs;

2. **Classificação de CRs baseada em regras** - nesta etapa, as informações textuais extraídas pela etapa (1) são usadas para classificar as CRs entre irrelevantes ou relevantes (quando se referem a *bugs* relevantes). As CRs relevantes são passadas para a etapa (3);

3. **Seleção de áreas relevantes** - Esta etapa acessa o conteúdo das CRs classificadas como relevantes, para verificar a ocorrência de termos positivos (*i.e.*, termos que se referem a áreas do SW que devem ser testadas). Caso algum termo positivo seja identificado, ele e o ID da respectiva CR são passados como entrada para a próxima etapa;

4. **Indexação e Seleção de *charters*** - os termos relevantes extraídos passam por uma validação do usuário, e a seguir são usados para criar consultas(*queries*) enviadas a uma base indexada de casos de teste(*charters*). Essas consultas recuperam os *charters* que cobrem as áreas relevantes indicadas pela etapa anterior;

5. **Criação do Plano de teste da SWAT** - na última etapa, os dados gerados nas etapas anteriores são agrupados, e é criado um Plano de Teste da SWAT. Com a última etapa concluída, os dados ficam disponíveis para os testadores utilizarem.

4.2 RECUPERAÇÃO E EXTRAÇÃO DE DADOS

Uma parte considerável das empresas de desenvolvimento de SW armazena as versões de software em repositórios de controle de versão. Tais repositórios também armazenam RNs juntamente com a respectiva versão do software à qual ela está relacionada. Assim, a etapa inicial do processo consiste em coletar as RNs relacionados às versões do SW a serem testadas, contidas no repositório de controle de versão. Com base na RN recuperada, a etapa de extração de informação preenche um template com informações textuais relevantes para o processo geral. As informações relevantes foram identificadas com base nas análises manuais feitas por uma equipe de teste exploratório. A partir dessas análises, foi possível observar que o melhor campo da RN a ser inspecionado é a área de *bugs* corrigidos, uma vez que ela contém comentários textuais sobre as CRs. Esses comentários podem ser o título da CR ou simplesmente um comentário feito pelo desenvolvedor responsável pela correção da CR. Contudo, pode haver inconsistências entre as informações textuais

contidas nos comentários das CRs corrigidas (os comentários disponíveis na área de bug corridos da RN) e o seu real conteúdo. Cada RN dá origem a um template diferente de forma automatizada. Esses dados servem como entrada para a etapa de classificação das CRs. A *Motorola* também possui um SGBD relacional para armazenar o conteúdo das CRs que estão nos seu *bugtracker*. Esta estratégia facilita o acesso de APIs ao conteúdo dessas CRs de forma estrutural, com base em *queries SQL*. O SGBD utilizado é o *Bigquery*[1], que pode ser acessado facilmente com as devidas autenticações. Esse ambiente é utilizado na fase 3 do processo proposto (ver Seção 4.4).

4.2.1 Implementação

Esta etapa inicial consiste em coletar as RNs relacionadas às versões do SW a serem testadas. Cada versão de SW (também denominada como *build*) é identificada exclusivamente por uma chave *id_build* (também conhecida como *job*). O *job* é um *alias* (*i.e.*, apelido utilizado para facilitar a identificação de *devices* ou projetos) para o nome do *device* (aparelho) que está sob teste. Através dessa identificação, é possível coletar do repositório de *builds* a versão mais recente do SW para o *device* sob teste. A nossa empresa parceira disponibiliza as novas versões do SW para download através do *Jenkins*[2].

Para a recuperação da RN, foi necessário criar um *script* na linguagem *Python* utilizando a biblioteca *jenkinsapi*[3]. O *script* é capaz de acessar o repositório e coletar a versão mais recente da *build* usando o *job* disponibilizado pelo testador. Este *script* sempre irá retornar a RN da *build* mais recente, já que o processo de revisão de documento é feito utilizando a *daily build* (*build* lançada no dia corrente).

Com base na RN recuperada, o processo de extração de informação preenche um template que possui os seguintes campos: *build_version*, *ID_CRs* e comentários. As RNs são disponibilizadas como arquivos de hipertexto. Assim, para o preenchimento automático desse template, foi criado um *parser HTML* em *Python* usando a biblioteca *Beautifulsoup*[4]. O parser vasculha o conteúdo da RN até achar o campo de *bugs* corrigidos. Ao encontrá-lo, o algoritmo identifica em qual *tag HTML* as informações do modelo estão contidas, e então as extrai.

Cada RN dá origem a um template correspondente, que será utilizado pela etapa de classificação das CRs entre relevantes ou não.

4.3 CLASSIFICAÇÃO DAS CRS

As CRs são de importância central para a criação de planos SWAT, uma vez que elas transmitem solicitações para melhorias do SW ou correção de erros. No entanto, nem

[1] https://cloud.google.com/bigquery/
[2] https://jenkins.io/
[3] https://jenkinsapi.readthedocs.io/en/latest/
[4] https://www.crummy.com/software/BeautifulSoup/bs4/doc/

todas as CRs em uma RN são relevantes para o nosso processo. Estamos interessados em CRs que se referem a problemas encontrados no SW (já que elas contêm palavras-chave usadas para identificar áreas relevantes a serem testadas). As CRs referentes a requisitos ou recursos descontinuados, versões de SW que não estão mais na fase de teste, por exemplo, não são relevantes.

Esta seção detalha o processo de classificação das CRs relevantes. Inicialmente, construímos classificadores usando técnicas de Aprendizado de Máquina supervisionado (BAEZA-YATES; RIBEIRO-NETO, 2011). No entanto, os resultados obtidos com os classificadores usados não foram satisfatórios, devido à falta de um corpus suficientemente grande de CRs etiquetadas.

Diante disso, optamos por construir um classificador baseado em regras explícitas e dicionários de termos de domínio (abordagem baseada em conhecimento para classificação de texto). Essa solução obteve maior precisão na tarefa de classificação, também demonstrando ser extensível e fácil de atualizar.

Realizamos uma classificação binária de CRs (relevante / não relevante) com base nos templates preenchidos na etapa (1)**(ver Seção 4.2)** utilizando dois dicionários de termos do domínio (um com palavras positivas – relevantes, e outro com palavras negativas –não relevantes). Como esses dicionários não estavam disponíveis anteriormente, eles precisaram ser construídos por meio de um processo de aquisição de conhecimento. A criação e a atualização dos dicionários não estão representadas na Figura 14, pois não participam do processo de criação de planos de teste para SWAT. Trata-se de um processo separado, que é guiado pelo funcionário responsável pela manutenção da ferramenta (veja detalhes abaixo).

4.3.1 Criação e atualização dos dicionários

Como já mencionado, o processo de classificação está baseado em dois dicionários: um positivo e um negativo. O dicionário positivo contém termos que ocorrem nas CRs consideradas relevantes para serem examinadas. Esses termos podem ser substantivos, adjetivos ou verbos que apontam um problema na versão do software sob teste (*e.g.*, *falha*, *não funciona*). Por outro lado, o dicionário negativo contém termos que ocorrem nas CRs não relevantes (como nomes de requisitos ou recursos descontinuados, versões de SW não mais na fase de teste, etc.).

Os dicionários foram construídos por meio de um processo semiautomático de aquisição de conhecimento. Inicialmente, equipes de diferentes áreas forneceram palavras-sementes / termos considerados essenciais para classificar uma CR como relevante ou não relevante. No entanto, análises iniciais revelaram a necessidade de atualizar os dicionários, a fim de levar em conta as mudanças frequentes nos requisitos e recursos do software. Assim, um procedimento semiautomático foi implementado para prover o crescimento dos dicionários

desse processo visando melhorar a precisão da classificação. Observe que este procedimento de atualização é executado sob demanda.

O procedimento de atualização do dicionário negativo extrai termos relevantes de CRs e RNs antigas, procurando por requisitos descontinuados ou recursos para atualizar o dicionário negativo. Para melhorar o dicionário positivo, o procedimento recupera palavras relacionadas a erros e defeitos, montando e identificando siglas, e verificando as possíveis variações de escrita para o mesmo termo na versão em teste. Esses termos são extraídos do campo de resumo e descrição das CRs e do campo de comentários das RNs, que são produzidos manualmente por desenvolvedores e testadores (e, portanto, transmitem informações muito relevantes). O procedimento de extração de novos termos para atualização dos dicionários é baseado em técnicas de Processamento de Linguagem Natural, tais como POS-Tagging, RegEx, Chunk e N-grams (MANNING et al., 2014).

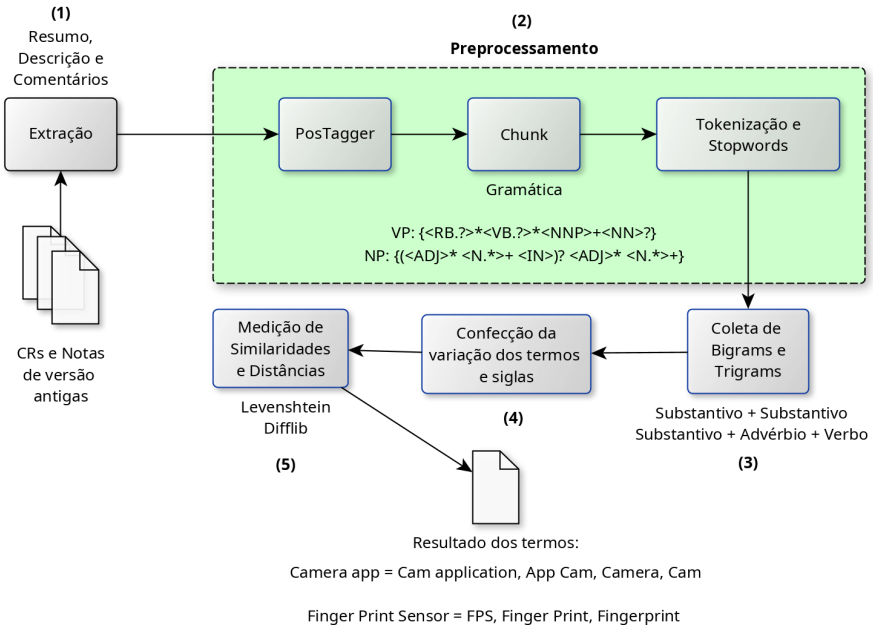

Figura 15 – Extração de palavras relevantes - Diagrama de fluxo

Fonte: autor

A Figura 15 mostra o diagrama de fluxo do algoritmo para extração de novos termos positivos e negativos, com o intuito de atualizar o dicionário de termos. O processo conta com 5 fases, detalhadas a seguir.

Extração - Nesta fase, o algoritmo extrai das CRs o título e a descrição do *bug*, e extrai das RNs os comentários sobre as CRs corrigidas. Esses textos são concatenados e transformados em um só *string*. Para a etapa extração foi desenvolvido um *script* na linguagem *Python* capaz de acessar o conteúdo de uma CR e extrair apenas o seu título e a descrição, concatenando-os em um único texto. Para as RNs antigas, foi necessário que os arquivos

HTML das RNs fossem passados como entrada para o algoritmo, em vez de apenas um job, para que o *parser HTML* (i.e., mesmo *script* utilizado na primeira etapa do processo de criação do plano de teste SWAT **(ver Seção 4.2.1)** pudesse extrair os dados referentes aos comentários das CRs corrigidas.

Pré-processamento - Nesta etapa, são usados alguns scripts criados utilizando a linguagem Python juntamente com a biblioteca *NLTK*[5], que implementa técnicas de PLN necessárias para esta etapa. A princípio, é utilizada a técnica *POS-tagging*, que realiza a etiquetagem morfossintática dos termos. As marcações foram realizadas utilizando a estrutura gramatical da língua inglesa, devido ao domínio do projeto ser voltado para textos em inglês. A seguir, utiliza-se a técnica de *chunk* para extrair do texto grupos nominais e verbais respeitando a estrutura morfossintática existente (Figura 15). Após a extração dos grupos nominais, é necessário realizar uma normalização das estruturas, usando a tokenização e a remoção de *stopwords*. A saída desta etapa é uma lista de tokens com suas estruturas morfossintáticas associadas.

Coleta de bigrams e trigrams – Esta etapa recebe a lista de tokens produzida na etapa anterior, e cria uma representação mais sofisticada em forma de bigrams e trigrams (respectivamente, com 2 e 3 tokens agrupados). Apenas os tokens que obedecem aos seguintes padrões morfossintáticos são considerados: (substantivo + substantivo) ou (substantivo + advérbio + verbo). Os grupos de tokens formados são enviados para a próxima etapa, onde são usados na criação de variações e abreviaturas que podem ser encontradas nos documentos de CRs e RNs. Para este fim, foi desenvolvido um *script Python* capaz de identificar a estruturas morfossintáticas obtidas anteriormente, da seguinte forma:

Lista de tokens fornecida como entrada para o *script*:
[('Cam2', 'NNP'), ('App', 'NNP'), ('dual', 'JJ'), ('camera', 'NN'), ('feature', 'NN')]

Saída do *script*:
[Cam2 App, camera feature, Cam2 camera, Cam2 feature, camera App, ...]

Geração de variações dos termos e de siglas – Os documentos de CRs e RNs podem conter palavras e siglas escritas de diferentes formas (inclusive neologismos). Como exemplo, citamos o termo *FingerPrint* Sensor, que pode aparecer como a sigla *FPS*, com as duas primeiras palavras separadas (*Finger print* Sensor) ou simplesmente omitindo a última palavra (*Fingerprint*). Assim, a fim de aumentar a precisão do processo de clas-

[5] https://www.nltk.org/

sificação, esta etapa cria siglas e variações para os termos coletados na etapa anterior. Esses novos tokens serão validados e, se forem adequados, serão inseridos nos dicionários pertinentes. Os exemplos abaixo ilustram essas duas fases.

Um bigrama, como por exemplo, *Advance Calling* sofre as seguintes transformações:

1. Soma dos termos:
    a) *Advance* + *Calling* = **AdvanceCalling**
2. Subtração de um dos termos:
    a) *Advance* - *Calling* = **Advance**
    b) *Advance* - *Calling* = **Calling**
3. Uso do *RegEx* para gerar abreviações:
    a) *AdvanceCalling* = **AdvCalling**, **AdvanceCall**
4. Uso de *RegEx* para extrair siglas:
    a) *AdvanceCalling* = **ADC**, **ACA**

As mesmas fases são executadas com os trigrams, com uso de procedimentos relativos a uma análise combinatória, para assim gerarem variações que contenham todos os termos. Após as variações e as siglas serem construídas, uma lista com todos os termos candidatos é passada para a etapa (5). É importante esclarecer que, inicialmente, os termos originais dos dicionários manualmente criados também foram submetidos a essa etapa de criação de variações e siglas. O objetivo foi de ampliar esses dicionários originais antes de inserir novos termos extraídos de novos documentos de CRs e RNs. A partir daí, os dicionários entraram na rotina padrão de manutenção descrita aqui nesta seção.

Medição de Similaridade e Distância de edição – Essa etapa realiza uma validação automática inicial dos termos coletados até o momento (palavras isoladas, termos compostos, bigrams, trigrams, variações e siglas). Essa validação se dá a partir da medição da similaridade baseada em cosseno (que irá transforma os tokens em vetores, e utilizar tais valores para calcular o cosseno da similaridade entre os tokens (LOPES, 2011)) e da distância de edição (que irá utilizar um número de operações necessárias para transformar um token em outro e, e desta forma calcular a distância de similaridade entres eles (ESTELA, 2007)) entre os tokens coletados/criados e os termos já existentes nos dicionários, buscando identificar as variações válidas para um determinado termo. Um exemplo de termos encontrados por esta etapa foi:

1. O dicionário continha o termo *Camera app*;

2. Criando variações com os bigramas e trigramas coletados, foi possível identificar e montar os seguintes tokens: *Cam application*, *camera*, *appcam*, *cameraapp*, *cam*, *camera radio*, *rc*, *cameraradio*, *radiocamera*, dentre outros;

3. A seguir, esta etapa identificou os termos mais similares a *Camera app*, que foram: *Cam application*, *camera*, *appcam*, *cameraapp* e *cam*. Tais variações tornaram-se candidatas a serem adicionadas aos dicionários.

Os termos que foram aprovados pela validação automática descrita acima foram avaliados por um analista de teste antes de serem incluídos nos respectivos dicionários, para garantir a precisão. Os usuários também têm permissão para incluir novos termos nos dicionários de forma manual. Este procedimento só é executado quando o analista de teste identifica um termo positivo (*e.g.*, o nome de um novo recurso) ou um termo negativo muito relevante que ainda não se encontra no dicionário correspondente.

Foi desenvolvido um *script* em *Python* para medir a similaridade entre os tokens (usando a função cosseno) e as distâncias de edição entre esses tokens (utilizando a biblioteca *difflib*[6]). Esse *script* disponibiliza para o especialista responsável pela a validação final os tokens candidatos e o valor da medição de similaridade. Vejamos o exemplo a seguir.

Entrada:
[Cam application, camera, appcam, cameraapp, cam, camera radio, rc, cameraradio, radiocamer]

Saída (medição feita com os termos contidos no dicionário positivo):
[Cam application - 90%, 93% - (Camera app) , camera - 72%, 58% - (Camera app), appcam - 53%, 90% - (Camera app), ...]

O resultado da execução desse *script* apresenta os resultados da medição de similaridade entre os tokens de entrada e o termo do dicionário correspondente. O primeiro percentual indica a similaridade baseada em cosseno, e o segundo valor indica o resultado para a distância entre os termos utilizando o *difflib*. O uso das duas medições foi necessário para que os neologismos pudessem ser identificados com maior facilidade. Um exemplo é o resultado para o termo *appcam*, que obteve uma similaridade de cosseno de 53%, porém obteve 90% de similaridade com o uso da distância entre os termos. Essa medição permitiu a indicação do termo (que tivesse um valor de similaridade maior ou igual a 75%) como um possível token candidato. O valor indicado anteriormente para considerar um token como válido, foi definido junto ao especialista de domínio, que analisou termo a termo todas os resultados gerados pelo processo. Com os termos devidamente avaliados, uma

[6] https://docs.Python.org/2/library/difflib.html

lista foi gerada com todos os termos candidatos. Este processo não é executado no ato das inspeções, e sim esporadicamente, devido a uma análise realizada pelo especialista de domínio que observou a necessidade da execução do processo a cada 3 meses.

4.3.2 Processo de Classificação das CRs

A classificação parte do conteúdo dos comentários relacionados às CRs nas RNs. O conteúdo coletado é submetido a um processo de intersecção com os termos existentes nos dicionários, como será explicado nesta seção. Os dados do template que representa a CR são pré-processados e normalizados, a fim de permitir uma correspondência correta com os termos nos dicionários. Os tokens resultantes são comparados com os termos nos dicionários. Duas regras de classificação foram definidas com o parecer dos analistas seniores, como mostra a Figura 15.

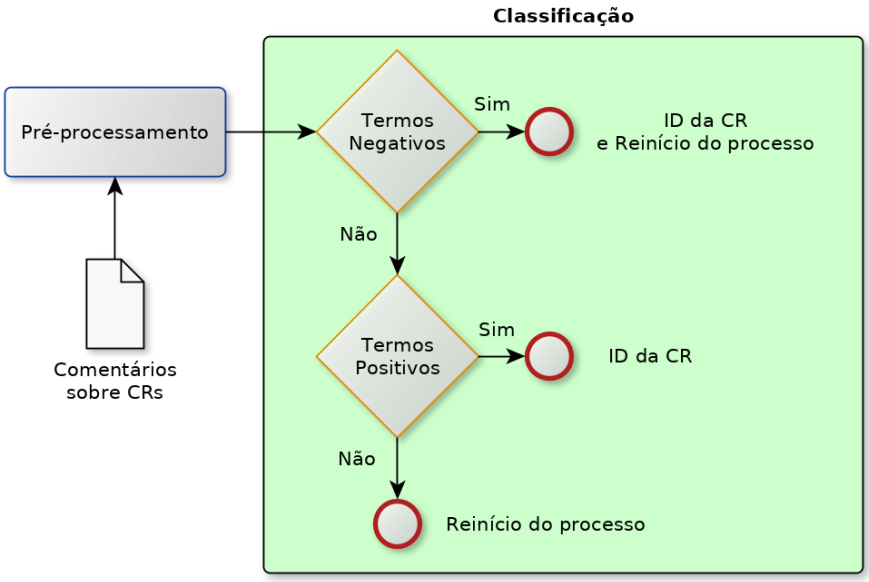

Figura 16 – Processo de classificação das CRs

Fonte: autor

De acordo com a Figura 16, o processo de classificação se dá da seguinte maneira:

1. Se os dados do template contiverem apenas termos no dicionário positivo, a CR associada será classificada como relevantes;

2. Se o template contiver pelo menos um termo negativo, a CR associada será considerada como não relevante.

O processo de classificação cria uma lista com os identificadores das CRs classificadas como positivas e outra lista com identificadores das CRs negativas. A lista positiva é enca-

minhada para a etapa de Seleção de Áreas Relevantes, e a lista negativa ficará disponível para vistoria do usuário testador, se desejado.

Para esta etapa do processo, foram desenvolvidas rotinas utilizando a linguagem *Python*. Abaixo descrevemos as fases desta etapa.

1. O *script* de pré-processamento recebe o comentário contido no template. Esta fase conta apenas com a tokenização e remoção de stopwords, e também foi desenvolvido utilizando a linguagem *Python* com o uso da biblioteca *NLTK*;

2. Com o comentário do template devidamente pré-processado, o *script* acessa o conteúdo dos dicionários positivo e negativo e os armazena em duas listas distintas;

3. As regras comparam os tokens gerados pelo pré-processamento dos comentários com dos dicionários, obedecendo o fluxo apresentado na Figura 16;

4. Em seguida, todos os tokens relevantes nos comentários são guardados em uma lista junto com o ID da sua respectiva CR;

5. O *Script* cria um arquivo *log* que armazena as listas com os termos e CRs positivas e com os termos e CRs negativas. Esse arquivo tem por objetivo guardar dados etiquetados para uma possível adaptação da etapa de classificação, que poderia fazer uso de alguma abordagem de Aprendizagem de máquina supervisionada.

Ao final do processo, o *script* fornece para a próxima etapa uma lista de IDs de CRs que possuem comentários com termos positivos, desta forma classificando-as como CRs positivas.

4.4 SELEÇÃO DE ÁREAS RELEVANTES

Para a seleção de áreas relevantes foi necessário a estruturação do processo em algumas etapas. A figura abaixo ilustra as etapas que o constituem.

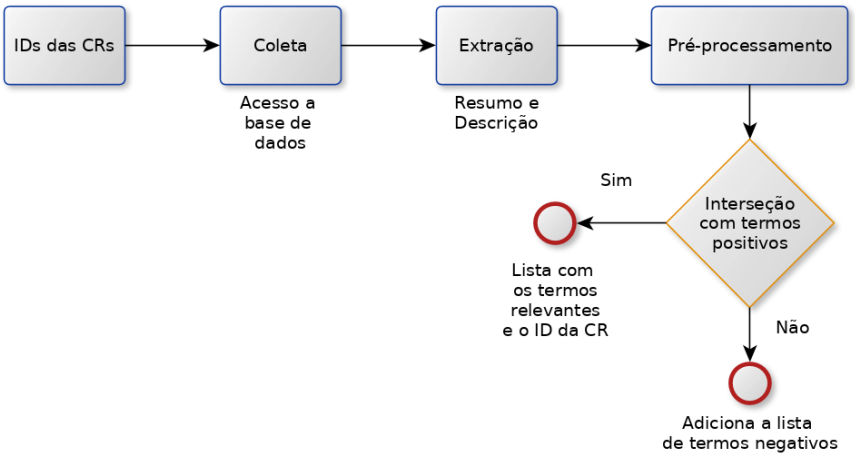

Figura 17 – Processo de seleção de áreas relevantes

Fonte: autor

Como mostra a Figura 17, esta etapa recebe como entrada os IDs das CRs classificadas como relevantes e retorna as áreas de SW que devem ser examinadas pela equipe de teste. Inicialmente, através de uma *query SQL*, o conteúdo das CRs que estão contidas no SBGD *Bigquery* é recuperado **(ver Seção 4.1)**. A *query* foi criada com o intuito de recuperar apenas os campos da CR que são usados no processo: resumo e descrição. Esses dois campos contêm informações válidas em linguagem natural sobre os defeitos identificados no SW, e as etapas de teste que levam à sua reprodução.

Os textos extraídos de todas as CRs de entrada são concatenados e submetidos a uma fase de pré-processamento (*RegEx*, tokenização, remoção de *stopwords*, *stemming*). As áreas relevantes a serem exploradas são obtidas através da intersecção entre a lista resultante de termos e o dicionário de termos positivos. Desta forma, todos os termos que representam áreas de SW e foram mencionados em qualquer CR relevante serão selecionados.

Esta etapa tem como saída uma lista de IDs das CRs positivas e as áreas relevantes identificadas.

4.4.1 Implementação

Para o processo de seleção de áreas relevantes, alguns *scripts* foram criados utilizando a linguagem *Python*. O pré-processamento, além de utilizar as técnicas anteriormente mencionadas **(ver Seção 3.1)**, necessitou realizar a remoção de algumas informações mais específicas, como algarismos, datas, nomes próprios, localidades, algumas palavras reservadas (*i.e.*, palavras que poderiam surgir como títulos de algumas fases na descrição da CR, podendo ser confundidas com um termo relevante) e links. Foi necessária a construção de um conjunto de expressões regulares em *Python* para remoção de algarismos,

datas e palavras reservadas. Para a remoção das localidades e nomes próprios foi construído um *script* em *Python* que utiliza a biblioteca *NLTK*, especificamente para realizar o reconhecimento de entidades nomeadas (que é todo e qualquer objeto do mundo real como, pessoas, localidade, endereços, etc.) As etapas de execução dos *scripts* seguem a seguinte ordem:

1. As autenticações utilizando a biblioteca *Python google.cloud*[7] são realizadas para que o *script* possa acessar a *Bigquery* através de uma *query SQL*;

2. A *query* é composta por um *select* que utiliza como parâmetro o ID da CR e algumas condições que foram observadas junto aos testadores experientes da *Motorola*, para que desta forma fossem recuperadas apenas as CRs coerentes com objetivo da análise. A *query* retorna o título da CR e a sua descrição;

3. Para a etapa anterior, é passada uma lista de IDs para ser utilizada na *query*, e como resposta da *query* é recebida uma lista de títulos e descrições dos IDs que estão de acordo com as especificações impostas na *query*;

4. Para cada CR, o seu respectivo título e descrição são concatenados, transformando-os em um único texto;

5. Os textos concatenados são submetidos ao primeiro filtro de pré-processamento, o *RegEx* para a remoção dos números, links e palavras reservadas;

6. Após o texto ter passado pelo *RegEx*, ele é submetido à fase de identificação e remoção das entidades nomeadas, removendo os nomes próprios e localidades;

7. Com o texto devidamente pré-processado, é realizada a interseção entre o texto e os termos do dicionário positivo;

Todos os termos resultantes desta interseção mais os IDs das CRs associadas ficam à disposição do usuário testador, para que ele possa analisar a relevância destes termos e utilizá-los na próxima etapa.

4.5 INDEXAÇÃO E SELEÇÃO DE *CHARTERS*

A partir do resultado da etapa anterior, os usuários (testadores) podem selecionar as áreas ou CRs que condizem com a necessidade do SW sob teste. Após a seleção das áreas de interesse, o processo geral sugere uma lista de *charters* que cobrem as áreas selecionadas. Os charters sugeridos são obtidos a partir de um repositório local da *Motorola*.

A empresa parceira mantém um extenso banco de dados de Casos de Teste, incluindo *charters*. Os *charters* são mais genéricos do que os CTs normalmente usados por outras

[7] https://cloud.google.com/Python /

equipes de testes, como mostra a Figura 18, e servem para cobrir os vários recursos diferentes (às vezes novos) de uma versão de SW em teste. Devido à grande quantidade de *charters* armazenados, a sua recuperação requer um resultado ordenado por relevância em relação à consulta. Porém, o SGBD da empresa não realiza essa ordenação.

Teste Charter 1
Charter - Análise de Login funcionalidade do website;
Áreas - Login com um usuário existente com **Nome de usuário** e **Senha**;
Login com um usuário existente com conta **Google**;
Login com um usuário existente com conta **Facebook**;
Entrar com **Nome de usuário** e **Senha** incorretos para verificar a mensagem de validação;
Bloquear o seu **Nome de usuário** e verificar a mensagem de validação;
Usar o **Esqueci a Senha** para resetar a **Senha**;
Duração - Normal (60 min)

Figura 18 – Exemplo de um *Charter*

Fonte: autor

Assim, foi necessário usar uma ferramenta de indexação e recuperação de informação (o Solr da Apache) para indexar todos os *charters*, e desta forma prover uma busca e recuperação mais eficientes. Como visto no **Capítulo 3**, para a indexação com o uso de técnicas de arquivos invertidos, temos etapas como: identificação do documento (coleta), reconhecimento da estrutura, pré-processamento e indexação (criação dos termos e índice) **(ver Figura 10)**. A recuperação dos documentos se dá a partir de consultas (em linguagem natural, na sua grande maioria), retornando documentos ordenados por relevância.

De acordo com os processos mencionados anteriormente, a Figura 19 retrata todas as etapas necessárias para a criação de uma base indexada de *charters* no Solr.

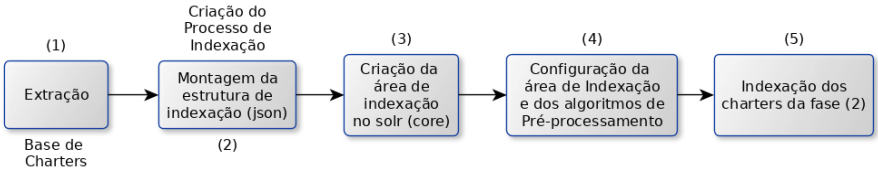

Figura 19 – Etapas da indexação dos *charters* no Solr

Fonte: autor

Como é possível observar na Figura 19, o processo que permite a indexação dos *charters* conta com 5 etapas, descritas a seguir:

Coleta/extração de dados — Esta fase consiste na coleta dos *charters* da base local da *Motorola*. Para este fim, foi criado um *script* em *Python* que pudesse acessar o repositório

e fazer a extração dos campos relevantes para nosso processo: ID, título e descrição do charter. O *script* conta com a biblioteca *Python JIRA*[8], uma vez que uma parte dos dados se encontra no *Jira framework*. O *script* conta com duas etapas:

1. Com o uso da biblioteca *Python JIRA*, foi possível abrir um conexão, utilizando as devidas credenciais, para coletar os IDs e títulos dos casos de testes específicos para a equipe de exploratório, que são os *charters* contidos no *Jira framework*. Para este fim, outras especificações internas da *Motorola* também foram necessárias para facilitar a identificação dos *charters*, uma vez que dentro do repositório há outros casos de testes que são aplicados a técnicas de regressão, *smoking*, dentre outros.

2. Com os IDs e títulos devidamente coletados, havia a necessidade de coletar as descrições dos *charters* que ficam armazenados em uma API proprietária à parte. Para tal, foi necessária uma requisição utilizando o protocolo HTTP POST para recuperar as descrições utilizando os IDs anteriormente coletados. A descrição dos *charters* contém vários campos e, por este motivo, o processo de extração coletou apenas as condições iniciais, passos, e resultados esperados, uma vez que são os mais relevantes para a nossa proposta.

A seguir, uma lista com as informações coletadas foi repassada para a próxima etapa.

Preparação dos dados — Os dados coletados foram organizados em uma estrutura JSON, para facilitar a utilização deles pela ferramenta *Solr*. O *Solr* conta com uma biblioteca para *Python* de nome *pysolr*[9], que permite a indexação dos dados a partir da estrutura JSON. Desta forma, os dados contidos na lista passaram por um processo de reestruturação para se adequarem à estrutura requerida, como mostra o exemplo a seguir:

```
{
        "chave": ID,
        "titulo": titulo,
        "descricao": passos,
        "condicaoinicial": condicao inicial,
        "resultadoesperado": resultado esperado
}
```

Com a estrutura devidamente finalizada, os dados foram enviados para a indexação no Solr.

Criação da área de indexação no Solr — O *Solr* conta com uma estrutura de indexação chamada de *core*. Um *core* pode ser criado tanto usando linhas de comandos diretamente em um computador, ou simplesmente usando a interface web do próprio *Solr*.

[8] https://jira.readthedocs.io/en/master/
[9] https://github.com/django-haystack/pysolr

Após a criação do *core*, um ambiente pré-configurado é definido dentro da estrutura do *Solr*. Este ambiente é formado por uma hierarquia de diretórios e arquivos de configuração.

Configuração da área de indexação — O *Solr* conta com vários recursos que contribuem para a criação de um ambiente de busca robusto. Essas configurações precisam ser definidas antes da indexação dos dados, e tais configurações devem respeitar uma estrutura que possa ser compartilhada para a busca, uma vez que será realizado o casamento de padrão entre os dados indexados e o conteúdo da *query* de busca. Toda a configuração necessária para a indexação e busca é feita dentro de um arquivo *.xml* que está dentro da pasta *conf* no diretório principal do *core*.

Nessa etapa, foi necessário definir quais algoritmos deveriam ser utilizados para o pré-processamento dos documentos a serem indexados, bem como das *queries* submetidas ao sistema. Para isso, as *tags* necessárias para tokenização, remoção de *stopwords* e *stemming* devem ser acrescentadas no arquivo responsável pela configuração do core de nome *managedschema.xml*.

Para que a remoção de *stopwords* possa ser executada pelo *Solr*, além da *tag*, também há a necessidade do fornecimento de uma lista de *stopwords* para arquivo *.txt* de mesmo nome, que está dentro da pasta *conf*.

Após a configuração, o core fica disponível para receber os dados que para a indexação.

Indexação dos dados — Por último, após o ambiente estar devidamente configurado, os dados obtidos na fase (2) são enviados para a indexação no *Solr*. O *script* responsável pelas fases (1) e (2) foi configurado para executar automaticamente uma vez por semana (para este fim, foi necessário uma análise do tempo médio de atualização de um charter no repositório da empresa parceira, e através deste estudo, foi observado que o tempo médio seria de uma semana para a execução automática das etapas (1) e (2)), para assim garantir que há integridade nos dados contidos no *Solr* de acordo com o repositório de *charters* da *Motorola*.

Com o *core* construído e configurado com os dados indexados, a ferramenta SPt pode usar as estruturas do *Solr* para recuperar os *charters* mais relevantes com base nas áreas indicadas. A consulta usa como dados de entrada as palavras-chave que representam as áreas relevantes extraídas e sugeridas pelo processo. Com base nas consultas, a ferramenta *Solr* retorna uma lista de *charters* classificados por similaridade para cada área de entrada.

A função de similaridade usada é a função nativa do *Solr*, de nome *BM25*(BM significa Melhor Correspondência)[10]. Os algoritmos utilizados pelo *BM25* implementam funções de ordenação baseadas em métodos de recuperação de informação probabilística, que podem obter melhores resultados de ordenação para bases indexadas de pequenos documentos quando comparadas a funções baseadas em *TF-IDF* (Frequência de Termo - Frequência

[10] https://en.wikipedia.org/wiki/Okapi_BM25

de Documento Inversa) (BAEZA-YATES; RIBEIRO-NETO, 2011).

A lista de charters ordenados por relevância é então disponibilizada para o testador, para que ele possa escolher livremente.

4.6 CRIAÇÃO DO PLANO DE TESTE DA SWAT – EXEMPLO DE USO DO PROTÓTIPO IMPLEMENTADO

No contexto deste trabalho, um Plano de teste da SWAT é um documento contendo palavras-chave que representam áreas de SW a serem exploradas, IDs de CRs e *charters* relacionados a cada área relevante no plano. Os planos de teste SWAT gerados pela SPt (SWAT Plan Tool) são usados pelos testadores para orientar o processo de teste manual. Cada nova RN originará um plano diferente.

As áreas relevantes e os IDs das CRs são obtidos a partir da RN atual que está sendo processada. Os *charters*, por sua vez, são recuperados da base indexada, criada via Solr, usando como consulta as áreas relevantes indicadas pela análise. Com o Plano de teste da SWAT devidamente construído, os dados ficam disponíveis para os usuários testadores conhecerem com mais detalhes quais são as áreas mais propensas a um defeito no *device* sob teste, e qual os *charters* mais indicados para a cobertura dessas áreas. As figuras a seguir apresentam a concepção do plano de teste da SWAT utilizando o protótipo.

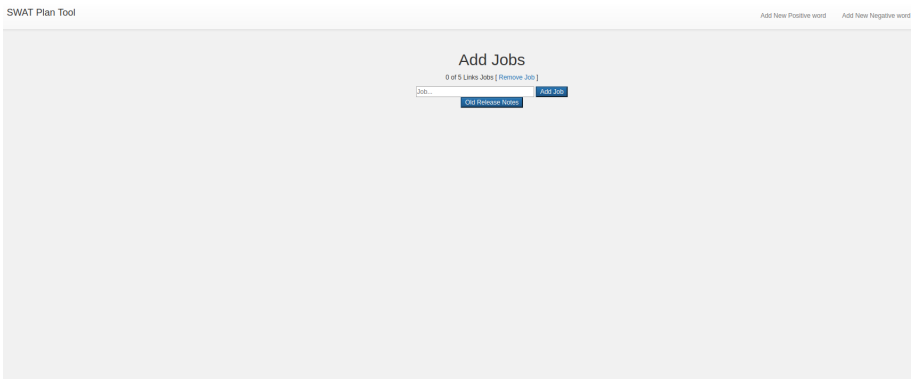

Figura 20 – Ambiente Inicial da SPt

Fonte: autor

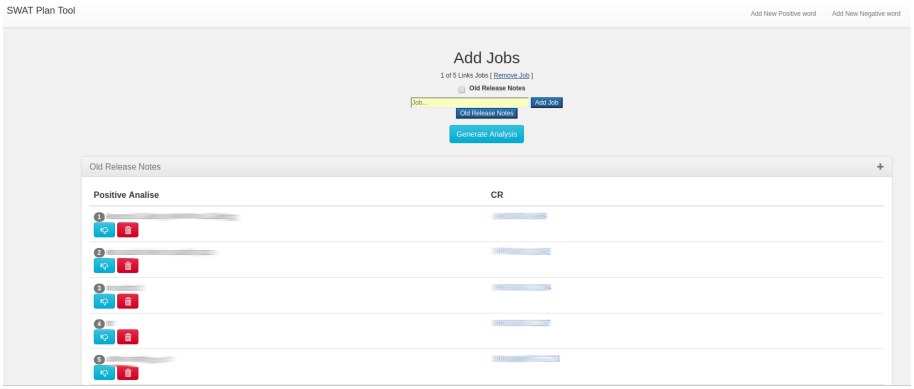

Figura 21 – Ambiente de resposta da análise produzida pela SPt

Fonte: autor

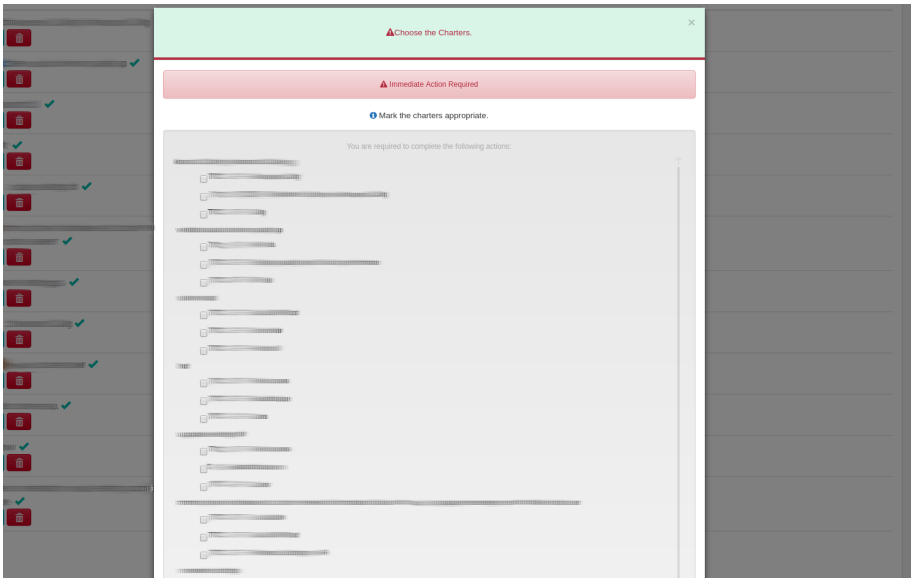

Figura 22 – Seleção dos *charters* indicados pela SPt

Fonte: autor

A Figura 20 apresenta a interface principal da SPt, onde o testador deverá fornecer um *job* referente à versão da *build* que ele deseja analisar **(ver Seção 4.2.1)**. Como adicional, a SPt também oferece a opção do uso do documento HTML da RN, desta forma possibilitando ao testador fazer análise em RNs mais antigas.

Na Figura 21, é possível observar o modelo de apresentação dos resultados após a análise. A SPt fornece, no lado esquerdo, as áreas relevantes (e disponibiliza o recurso de edição para os termos que foram indicados pela análise como relevantes). A Spt também conta com o recurso *deslike*, que permite o usuário deixar um comentário sobre as áreas fornecidas como relevantes. O intuito de tal funcionalidade, se dá devido a necessidade de armazenar *logs* de uso para criar dados históricos para possíveis implementações futuras.

Do lado direito da tela, são apresentados os IDs das CRs onde os termos foram identificados (os IDs estão em formato de links, que podem ser utilizados para acessar o seu conteúdo). Este ambiente também conta como a possibilidade de o testador remover e de dar feedbacks sobre as áreas indicadas pela ferramenta.

A Figura 22 apresenta a escolha dos *charters* de acordo com as áreas relevantes indicadas pela SPt **(ver Seção 4.5)**.

Após os *charters* terem sido selecionados, o testador pode gerar o plano de teste da SWAT apenas com um clique, como mostra a Figura 23. A SPt também disponibiliza ao testador a opção de copiar o resultado para a área de transferência, caso ele deseje armazenar o conteúdo da análise para consultas posteriores.

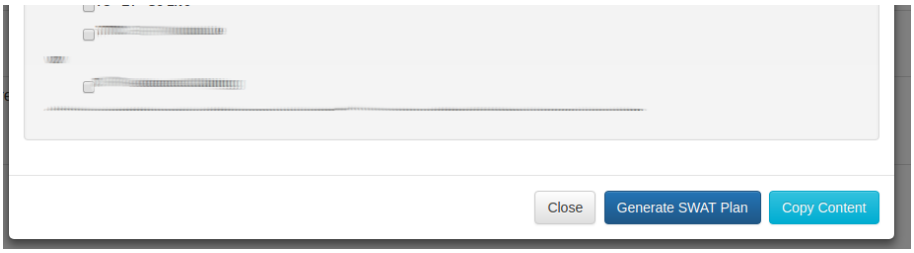

Figura 23 – Geração do plano de teste da SWAT

Fonte: autor

Como apresentado acima, o uso da SPt permite que, com poucos passos, os testadores consigam gerar um plano de teste conciso e objetivo a partir da inspeção do documento de RN.

4.7 ESTUDO EMPÍRICO E RESULTADOS

Esta seção descreve os estudos realizados para validar o protótipo desenvolvido. Como mencionado anteriormente, a SPt visa ajudar os testadores na tarefa de criar planos SWAT. Como tal, os estudos compararam o atual processo manual realizado para analisar as RNs e gerar planos de teste com o correspondente processo semiautomatizado utilizando a SPt. Dois aspectos diferentes foram observados:

1. O tempo gasto pelos testadores para processar as RNs e criar os planos SWAT com e sem o uso do protótipo SPt;

2. A quantidade de áreas relevantes identificadas com e sem o uso do SPt (cobertura).

Tal estudo também visa observar como a inserção do processo de revisão de documento pode contribuir para a criação de um plano de teste mais eficiente para equipes de teste exploratório.

4.7.1 Corpora utilizados no Estudo

Para comparar de forma justa a análise manual com os processos de análise automática, foi necessário construir dois corpora diferentes de RNs - caso contrário, o testador executaria mais rapidamente a segunda análise, já que ele já teria adquirido conhecimento sobre as áreas e *charters* relevantes a selecionar a partir da primeira análise. No entanto, para fornecer o mesmo nível de dificuldade em ambas as análises, os dois corpora precisavam ser muito semelhantes em relação às características essenciais das RNs selecionadas - por exemplo, elas deveriam se referir à mesma versão de SW e ter o mesmo número de CRs.

Cada corpus utilizado no estudo consiste em quatro RNs, todas relacionadas ao mesmo SW sob teste. Cada RN corresponde a uma única versão do SW (mesma compilação), e elas foram disponibilizadas em dias consecutivos. Essas RNs possuem todos os campos padrão de qualquer RN **(ver Seção 2)**.

4.7.2 Metodologia do Estudo

Os estudos se concentraram em dois aspectos:

1. Quantidade de áreas relevantes selecionadas manualmente *versus* quantidade de áreas relevantes automaticamente identificadas pela SPt (cobertura);

2. Tempo total gasto pelo testador para criar planos de teste da SWAT com e sem o uso do SPt.

O estudo foi replicado por três testadores diferentes que trabalham na empresa parceira (*Motorola*), cada um com um nível diferente de experiência: iniciante, intermediário e experiente. Esses três participantes são responsáveis por analisar manualmente as RNs diariamente, tendo o conhecimento para identificar as áreas a serem exploradas em testes de um determinado SW. Esses participantes foram indicados pelo gerente de testes do time de exploratório. Dessa forma, assumimos que eles eram adequados para participar desse estudo.

Um treinamento foi fornecido aos participantes para a utilização da SPt. Este treinamento teve duração de 30 minutos e foi apresentado os requisitos básicos para a execução do estudo empírico.

A ordem de execução também foi definida, onde o testador experiente primeiramente usou a Spt para o estudo e depois o processo manual, já o testador intermediário fez o processo inverso e o testador novato executou o estudo na mesma ordem que o testador experiente.

Todos os valores gerados pelo testador experiente foram utilizados como valores de referencia para os estudos realizados pelos demais.

Inicialmente, o ambiente foi configurado adequadamente e, para tornar o estudo o mais próximo possível do dia-a-dia dos testadores, foi pedido a cada um deles que buscassem as

RNs dentro do repositório de *builds*, como eles costumam fazer diariamente. Em seguida, cada testador realizou uma análise inteiramente manual do corpus 1 (4 RNs e as CRs relacionadas), para selecionar as áreas relevantes a serem exploradas. Cada área encontrada pelo testador era registrada em um arquivo *.txt*, que servia como documento de apoio para a análise manual. Essas áreas foram usadas para procurar *charters* no repositório da empresa, de modo a recuperar os *charters* mais relevantes a serem incluídos nos planos de teste da SWAT. Os testadores precisavam procurar área por área (utilizando os termos contidos no documento *.txt*) dentro do repositório *charters* de forma visual, selecionando um a um.

Em seguida, os testadores criaram os planos de teste da SWAT com base nos charters selecionados, e criaram manualmente no *JIRA framework* um relatório (transcrito em linguagem natural), para que todos os testadores do time de exploratório pudessem usar como referência os testes e as áreas que deveriam ser exploradas. É importante lembrar que cada RN origina um plano de teste da SWAT diferente. Portanto, foi necessário que todo o processo fosse repetido 4 vezes por cada testador alocado para o estudo.

A seguir, os mesmos testadores usaram o protótipo SPt para realizar uma análise semiautomatizada do corpus 2. Através da interface da SPt, os testadores passaram como entrada os *jobs* das RNs, e após a etapa de recuperação e extração de dados, os testadores receberam como saída as áreas relevantes automaticamente extraídas desses documentos. A seguir, através da interface da SPt, os testadores escolheram as áreas que julgaram mais relevantes para a versão do SW sob teste. As áreas escolhidas foram usadas para consultar, por meio da interface SPt, o repositório de *charters* indexados pelo Solr. A ferramenta então apresentou uma lista de 3 *charters* para cada área presente na consulta, sendo a lista ordenada pela similaridade entre o conteúdo do charter e as áreas usadas na consulta à base.

Por fim, o testador precisou apenas selecionar os *charters* de sua escolha com um click, para serem incluídos nos planos de teste da SWAT correspondentes. O plano de teste da SWAT foi criado contendo os IDs de CRs, áreas relevantes e seus respectivos *charters*, e um relatório foi construído no *JIRA framework* de forma automática.

4.7.3 Resultados Obtidos

As tabelas 4, 5 e 6, respectivamente, mostram os resultados dos estudos conduzidos pelos testadores selecionados (iniciante, intermediário e experiente). Cada tabela possui 7 linhas, com informações preenchidas pelos testadores.

1. ID da RN utilizada;
2. Número de áreas relevantes identificadas pela análise manual (M);
3. Número de áreas relevantes identificadas através do uso do protótipo (SPt);

4. Cobertura da seleção via protótipo em relação aos valores de referência (para este estudo, são considerados *valores de referência* (*i.e.*, valores corretos) a quantidade de áreas selecionadas manualmente pelo testador experiente a partir de cada RN analisada);

5. Total de tempo gasto pelo testador para concluir todo o processo de geração do plano de teste de forma manual (M);

6. Total de tempo gasto pelo testador para concluir todo o processo de geração do plano de teste usando o protótipo (SPt);

7. Percentual de economia de tempo com o uso do protótipo (SPt).

A medida de Cobertura é tradicionalmente obtida dividindo-se o número de itens relevantes retornados por um procedimento, pelo número total de itens relevantes na coleção (BAEZA-YATES; RIBEIRO-NETO, 2011). Para cada RN, a cobertura foi obtida dividindo-se o número de áreas relevantes identificadas em cada execução pelo valor de referência para a RN sob análise. Os valores de referência usados aqui foram determinados pela análise realizada pelo testador experiente (Tabela 4).

Tabela 4 – Estudo conduzido por um testador experiente

RN#	1	2	3	4	**Total**
(M) Número de áreas relevantes	15	8	16	11	**50**
(SPt) Número de áreas relevantes	15	6	14	11	**46**
(SPt) Cobertura em relação aos valores de referência	100%	75%	87.5%	100%	**90.6%**
(M) Total de tempo gasto	70min	35min	55min	45min	**205min**
(SPt) Total de tempo gasto	40min	20min	25min	25min	**110min**
% da economia de tempo com o uso da (SPt)					**46.3%**

Tabela 5 – Estudos conduzido pelo testador iniciante

RN#	1	2	3	4	**Total**
(M) Número de áreas relevantes	15	7	11	8	**41**
(SPt) Número de áreas relevantes	13	6	10	8	**37**
(SPt) Número de áreas relevantes (valores de referência)	15	6	14	11	**46**
(SPt) Cobertura em relação aos valores de referência	86.6%	100%	71.4%	72.7%	**82.6%**
(M) Total de tempo gasto	120min	55min	50min	55min	**280min**
(SPt) Total de tempo gasto	33min	20min	35min	30min	**118min**
% da economia de tempo com o uso da (SPt)					**57.8%**

Tabela 6 – Estudo conduzido por testador de intermediário

RN#	1	2	3	4	**Total**
(M) Número de áreas relevantes	11	5	16	11	**43**
(SPt) Número de áreas relevantes	11	6	13	11	**41**
(SPt) Número de áreas relevantes (valores de referência)	15	6	14	11	**46**
(SPt) Cobertura em relação aos valores de referência	73.4%	100%	92.8%	100%	**91.5%**
(M) Total de tempo gasto	102min	80min	45min	45min	**227min**
(SPt) Total de tempo gasto	80min	37min	25min	25min	**167min**
% da economia de tempo com o uso da (SPt)					**38.6%**

De acordo com os dados obtidos, foi possível observar que o número de áreas relevantes identificadas por cada testador mostrou algumas variações. Como esperado, o testador experiente identificou um número maior de regiões relevantes no total, com e sem o uso do SPt; e o testador iniciante identificou a menor quantidade de áreas, devido à falta de experiência. Como consequência, os planos de teste da SWAT criados pelo testador iniciante não cobriam todas as áreas críticas do SW a ser testado. Também foi possível observar que os valores médios de cobertura do testador iniciante com o uso da SPt ficaram acima de 74%, e o testador experiente alcançou 90,6% de cobertura com a SPt, o que é altamente satisfatório (particularmente quando observamos que a execução manual levou quase o dobro do tempo).

Surpreendentemente, o testador intermediário alcançou 38,6% de tempo economizado (menos do que o especialista). Esse baixo percentual foi devido à dificuldade desse testador em lidar com a ferramenta na primeira execução, mesmo tendo recebido o treinamento necessário (Tabela 6, 5ª linha, 1ª coluna). Note que as outras execuções foram mais rápidas (colunas 2, 3 e 4).

Finalmente, o testador experiente obteve os melhores resultados globais usando a SPt, economizando 46,3% do tempo e ainda alcançando 90,6% de cobertura. Note que, devido ao seu nível de especialização, esse testador realizou o Estudo manual em menos tempo que os outros. Assim, não seria viável obter uma taxa maior de economia de tempo e ainda manter uma cobertura alta.

Diante dos resultados dos estudos, podemos afirmar que os ganhos obtidos com o SPt foram muito encorajadores, demonstrando indicações que a SPt pode trazer ganhos de produtividade.

4.8 TRABALHOS RELACIONADOS

A seleção de casos de teste é uma áreas de pesquisa que, na sua grande maioria, está ligado aos Testes de Regressão (Rothermel; Harrold, 1996). Desta forma, podemos identificar na literatura relacionada vários trabalhos com foco nessa área. No entanto, este trabalho visa os Testes exploratórios propondo uma solução voltada na automação das técnicas de

inspeção de documentos para ajudar na seleção de *charters* usando MT e RI. Pesquisas associadas ao uso RI para a seleção de Caso de Teste (CT) podem ser encontradas nas obras Magalhães et al. (2016); Araújo et al. (2017), Magalhães et al. (2017); Magalhães, Mota e Maia (2016).

O trabalho de Magalhães et al. (2016) propôs um processo de seleção de CT baseado em RI, especificamente usando um sistema de RI, o Magalhães et al. (2017) e Magalhães, Mota e Maia (2016). Todo processo foi construído sobre a ótica de duas grandes fases: (i) automatização de várias atividades manuais; (ii) implementação do método de seleção de CT. O processo automatizado recebe como entrada um conjunto de CRs selecionados manualmente pela equipe de teste e retorna como saída uma lista ordenada de CTs que serão usados para realizar os testes de regressão. O nosso trabalho também utiliza as CRs, mas as coletamos de forma automatizada através do processo de extração e coleta de dados (**ver Seção 4.2**) da RN. Com isso, utilizamos apenas termos chaves contidas nas CRs para obtermos uma lista ordenada de CTs para cada CR. Desta forma, possibilitando que o plano teste de exploratório possua apenas testes que cubram as áreas relevantes. No trabalho de Magalhães et al. (2016) as palavras chaves utilizadas para a seleção dos CTs, são extraídas dos campos das CRs que contém informações significativas para a tarefa (por exemplo, título, componente do produto, descrição do problema). Ao contrário do nosso processo, todas as palavras contidas no texto são utilizadas para a seleção dos CTs. No nosso processo, apenas termos contidos no dicionário positivo (**ver Seção 4.3.1**) são utilizados para essa tarefa.

Em Araújo et al. (2017), a solução proposta para a seleção automática de CTs baseia-se no código-fonte que foi alterado na atual campanha de teste de regressão. O processo automatizado recebe como entrada um arquivo que indica onde ocorreram as alterações (como a linha completa do código-fonte ou o caminho da mudança) e retorna como saída uma lista de palavras-chave que serão usadas para executar uma seleção de CTs. A abordagem utilizada em (ARAúJO et al., 2017) para a extração de palavras chaves, se dá através da utilização de expressão regular baseada na regra *CamelCase* (CamelCase, 2019). Desta forma, as palavras chaves são extraídas do código fonte separando os termos, que formam o nome de um método, após ter identificar as letras em caixa alta. Ainda em (ARAúJO et al., 2017), o autor transcreve que, como algumas das palavras geradas pelo processo não fazem parte do domínio, foi necessário criar um conjunto de stopwords que pudesse validar os termos extraídos do código. Diferentemente da nossa abordagem, tais termos não foram construídos a partir de um princípio gramatical, ou seja, não foi priorizado nenhuma estrutura sintática para definir quais os melhores termos para o dicionário de *stopwords*. Ainda no trabalho de (ARAúJO et al., 2017), não houve nenhuma menção sobre como esse primeiro conjunto de *stopwords* foi concebido, ou se houve alguma atualização para a sua melhoria. A nossa abordagem parte de outro princípio. Os dicionários não servem para excluir termos que previamente já foram selecionado, como na abordagem de

(ARAúJO et al., 2017), e sim para identificar quais CRs são relevantes ou não no processo de classificação.

Os outros trabalhos mencionados acima, Magalhães et al. (2017) e Magalhães, Mota e Maia (2016), partem do mesmo princípio das abordagens de que utilizam cobertura de código para identificação de palavras chaves usadas como entrada para um sistema de RI. O sistema de RI utilizado nesses trabalhos foi o Apache Lucene, que utiliza o TF-IDF para o ranqueamento dos CTs. Já na nossa abordagem utilizamos o Apache Solr que, mesmo originado do Apache Lucene, utiliza outro algoritmo de ranqueamento, o BM25, que melhor se adapta a realidade de nossa pesquisa.

4.9 CONSIDERAÇÕES FINAIS

Neste capítulo, foi descrito o processo semiautomático de criação de planos de teste exploratório, bem como o protótipo desenvolvido (SPt). Destacamos a motivação para desenvolvimento deste trabalho, quais as técnicas utilizadas na implementação e a medição quantitativa e qualitativa do desempenho do protótipo em um ambiente real. Com isso, foi possível ressaltar a necessidade de implementação de um modelo automatizado para revisão de documentos dentro da fase de criação do plano de teste, uma vez que tal abordagem ajuda o testador a criar os seus planos de teste de forma eficiente. Neste Capítulo também descrevemos sobre os trabalhos relacionados, onde foi possível observar abordagens que utilizam sistemas de RI para seleção de CT.

A revisão de documento não é uma prática utilizada no processo de teste. Tal processo, na sua grande maioria, é realizado antes da etapa de teste, e por este motivo, foi observada a necessidade de unir as abordagens e gerar um processo mais eficiente. A SPt conseguiu não apenas criar um plano de teste da SWAT mais eficaz, mas também criar um novo ciclo de vida para a concepção de tais planos, atingindo resultados bastante animadores.

5 CONCLUSÃO

Esse capítulo trata do encerramento deste trabalho. Nele, os objetivos serão revistos para discutir se eles foram alcançados e as considerações finais com conclusões tanto do ponto de vista prático quanto o metodológico serão abordadas. Além disso, as oportunidades identificadas para embasar novas pesquisas e trabalhos serão apresentadas.

5.1 CONSIDERAÇÕES FINAIS

Conforme mencionado na introdução, o objetivo desse trabalho era construir um novo processo para compor planos de teste para equipes que executam testes exploratórios. Este processo baseia-se principalmente na automação da inspeção das RN, com foco na extração de áreas de SW relevantes a serem exploradas pelos testadores.

Do ponto de vista teórico, vale destacar que durante o levantamento bibliográfico, não foi identificado na literatura nenhum outro trabalho que se dispusesse a integrar técnicas de revisão de documentos/artefatos de SW para a geração automática de planos de teste, com base em áreas mais suscetíveis a defeitos. Assim, esse trabalho traz uma contribuição para seu campo de pesquisa acadêmica ao mostrar, de forma bem sucedida, que a arquitetura proposta pode ser utilizada em problemas de geração de planos de teste mais eficientes, mais especificamente, na detecção e extração de áreas relevantes que possam contribuir para a escolha dos melhores casos de teste (*i.e., charters*) para a nova versão do SW sob teste.

Para viabilizar o novo processo, mostrou-se necessário a construção de um protótipo de nome SPt, que foi implementado utilizando técnicas de MT e RI para classificar CRs contidas nos documentos de RN e, a partir das informações coletadas, extrair as áreas relevantes a serem exploradas. Após a concepção da SPt houve a necessidade de comprovarmos a sua eficácia através de um experimento que revelou um ganho significativo de produtividade pelos testadores, superando o processo manual adotado atualmente pela empresa parceira em relação ao esforço associado ao tempo de sua execução. Desta forma, a SPt pôde contribuir para a construção de um método mais eficiente de geração dos planos de teste para uma equipe de testes exploratório.

5.2 SUGESTÃO PARA PESQUISAS FUTURAS

Ao longo desta pesquisa, foi possível observar algumas frentes de trabalho que podem ser abordadas, como:

Utilização dos dados Históricos - A SPt gera um aglomerado de dados históricos das utilizações feitas pelos usuários testadores na concepção dos seus planos de teste da

SWAT. A metodologia de classificação atual pode ser revista, utilizando tais dados para um abordagem que utilize AM.

Geração automática de tesauros - Com base em uma das etapas da SPt um processo de geração automática de termos para os dicionários positivo e negativo foi criado. Para este, algumas melhorias ainda precisam ser feita, para que o uso do especialista de domínio seja menos requerido. Desta forma, a implementação de outras abordagens além da PLN apresentada poderia contribuir para o aumento da sua precisão.

Implementação do protótipo - Atualmente o protótipo foi implementada para equipes de teste exploratório, podendo ser expandido para outras equipes de teste como: regressão, *smoke*, dentre outros. Desta forma, isso comprovará que tal metodologia pode melhorar a produtividade de diferentes equipes de teste.

REFERÊNCIAS

ABEBE, S. L.; ALI, N.; HASSAN, A. E. An empirical study of software release notes. *Empirical Software Engineering*, v. 21, n. 3, p. 1107–1142, jun. 2016. ISSN 1382-3256, 1573-7616. Disponível em: <http://link.springer.com/10.1007/s10664-015-9377-5>.

ALBERTO, S. Quality of requirements specifications: A framework for automatic validation of requirements. In: *Proceedings of ICEIS'2014 Conference*. [S.l.]: SCITEPRESS, 2014.

ANDERSSON, C.; RUNESON, P. Verification and validation in industry - a qualitative survey on the state of practice. In: *Proceedings International Symposium on Empirical Software Engineering*. [S.l.: s.n.], 2002. p. 37–47.

ANDERSSON, C.; RUNESON, P. Verification and validation in industry—a qualitative survey on the state of practice. In: . [S.l.: s.n.], 2002. p. 37– 47. ISBN 0-7695-1796-X.

ANSI/IEEE. An american national standard, ieee standards for software reviews and audits. *ANSI/IEEE Std 1028-1988*, 1989.

ARANHA C. E PASSOS, E. A tecnologia de mineração de textos. *Revista Eletrônica de Sistemas de Informação*, v. 5, n. 2, 2006. ISSN 1677-3071. Disponível em: <http://www.periodicosibepes.org.br/index.php/reinfo/article/view/171>.

ARANHA C., V. M. Uma abordagem de pré-processamento automático para mineração de texto em português: sob o enfoque da inteligência computacional. 2007.

ARAúJO, J.; ARAúJO, J.; aES, C. M.; ANDRADE, J. a.; MOTA, A. Feasibility of using source code changes on the selection of text-based regression test cases. In: *Proceedings of the 2Nd Brazilian Symposium on Systematic and Automated Software Testing*. New York, NY, USA: ACM, 2017. (SAST), p. 8:1–8:6. ISBN 978-1-4503-5302-1. Disponível em: <http://doi.acm.org/10.1145/3128473.3128481>.

AURUM A., W. C. *Proceedings of Requirements Engineering for Software Quality*, Essen Germany, p. 9–10, dez. 2002.

BAEZA-YATES, R.; RIBEIRO-NETO, B. *Modern Information Retrieval: The Concepts and Technology Behind Search*. [S.l.]: Addison-Wesley Professional, 2011.

BARION E. E LAGO, D. Mineração de texto. *Revista Eletrônica de Sistema de Informação*, v. 3, 2008.

BERWICK, R. C.; TENNY, C.; ABNEY, S. P. Book. *Principle-based parsing : computation and psycholinguistics / edited by Robert C. Berwick, Steven P. Abney, Carol Tenny*. [S.l.]: Kluwer Academic Publishers Dordrecht ; Boston, 1991. vii, 408 p. : p. ISBN 0792316371 0792311736.

BINDÁ, J. M.; BRANDT, M. A. G.; PIEDADE, M. P. Análise da Aplicação de Sistemas de Recuperação de Informação Usando Android numa Base Bíblica. p. 10, 2013.

BLUMETTI, B.; LEMOS, C.; CARLOS, L.; ARAúJO, R. Seleção de informações usando text mining com ri. p. 13, 2007.

Denison F. Brandon R. *Systems and methods to provide customized release notes during a software system upgrade of a process control system*. 2014. USPTO 8,898,660. Disponível em: <https://patents.google.com/patent/US20050132349>.

BRIAND, L.; EMAM, K. E.; LAITENBERGER, O.; FUSSBROICH, T. Using simulation to build inspection efficiency benchmarks for development projects. In: *Proceedings of the 20th International Conference on Software Engineering*. Kyoto, Japan: IEEE Comput. Soc, 1998. p. 340–349. ISBN 978-0-8186-8368-8. Disponível em: <http://ieeexplore.ieee.org/document/671387/>.

BRITO, E. M. N. Mineração de Textos: Detecção automática de sentimentos em comentários nas mídias sociais. p. 54, 2016.

BROTHERS, L.; SEMBUGAMOORTHY, V.; MULLER, M. ICICLE: groupware for code inspection. In: *Proceedings of the 1990 ACM conference on Computer-supported cooperative work - CSCW '90*. Los Angeles, California, United States: ACM Press, 1990. p. 169–181. ISBN 978-0-89791-402-4. Disponível em: <http://portal.acm.org/citation.cfm?doid=99332.99353>.

CAMBRIA, E.; WHITE, B. Jumping NLP Curves: A Review of Natural Language Processing Research [Review Article]. *IEEE Computational Intelligence Magazine*, v. 9, n. 2, p. 48–57, maio 2014. ISSN 1556-603X. Disponível em: <http://ieeexplore.ieee.org/document/6786458/>.

CamelCase. 2019. Page Version ID: 54365217. Disponível em: <https://pt.wikipedia.org/w/index.php?title=CamelCase&oldid=54365217>.

CAMILO, C. O. Mineração de Dados: Conceitos, Tarefas, Métodos e Ferramentas. p. 29, 2009.

CARDOSO, O. Recuperação de informação. *INFOCOMP*, v. 2, n. 1, p. 33–38, 2004. ISSN 1982-3363. Disponível em: <http://www.dcc.ufla.br/infocomp/index.php/INFOCOMP/article/view/46>.

CARRILHO, J. Desenvolvimento de uma metodologia para mineraÇÃo de textos. 2008. Disponível em: <https://www.maxwell.vrac.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=11675@1>.

CAZES, T. B.; FEITOSA, R. Incorporação de conhecimento do especialista através de regras para a classificação de imagens de sensores remotos de alta resolução. 12 2018.

CIOLKOWSKI, M.; LAITENBERGER, O.; BIFFL, S. Software reviews: The state of the practice. *IEEE Software*, v. 20, n. 6, p. 46–51, nov. 2003. ISSN 0740-7459. Disponível em: <http://ieeexplore.ieee.org/document/1241366/>.

COLLOBERT, R.; WESTON, J.; BOTTOU, L.; KARLEN, M.; KAVUKCUOGLU, K.; KUKSA, P. Natural Language Processing (Almost) from Scratch. *NATURAL LANGUAGE PROCESSING*, p. 45, ago. 2011.

CORREIA, M. F. B. (Ed.). *Recuperação de informação*. [S.l.]: Informática aplicada: ITI 4301, 2018.

CUNHA, S. *Revisão de Software*. 2009. Disponível em: <https://smcufmg.wordpress.com/2009/09/17/revisao-de-software/>.

DIAB, M. T. Second Generation AMIRA Tools for Arabic Processing: Fast and Robust Tokenization, POS tagging, and Base Phrase Chunking. p. 4, jan. 2009.

DOOLAN, E. P. Experience with Fagan's inspection method. *Software: Practice and Experience*, v. 22, n. 2, p. 173–182, fev. 1992. ISSN 00380644, 1097024X. Disponível em: <http://doi.wiley.com/10.1002/spe.4380220205>.

EBENAU, R. G.; STRAUSS, S. H. *Software Inspection Process*. New York, NY, USA: McGraw-Hill, Inc., 1994. ISBN 0-07-062166-7.

ESTELA, M. XSimilarity : Uma ferramenta para consultas por similaridade embutidas na linguagem XQuery. p. 47, 2007.

FAGAN, M. E. Design and code inspections to reduce errors in program development. *IBM Syst. J.*, IBM Corp., Riverton, NJ, USA, v. 15, n. 3, p. 182–211, set. 1976. ISSN 0018-8670. Disponível em: <http://dx.doi.org/10.1147/sj.153.0182>.

FELDMAN, R.; SANGER, J. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. [S.l.: s.n.], 2006. v. 34. ISBN 9780521836579.

FERNEDA, E. Recuperação de Informação: estudo sobre a contribuição da Ciência da Computação para a Ciência da Informação. p. 147, 2003.

FINATTO, M. Analise textual assistida por computador: Reconhecimento linguistico - terminologico do texto tecnico-cientifico de quimica em portugues da coesão a enunciação (textquim). *Relatório Final de Atividades, CNPQ*, Nature Publishing Group, 2005.

FISCHER, M.; PINZGER, M.; GALL, H. Populating a release history database from version control and bug tracking systems. In: *Proceedings of the International Conference on Software Maintenance*. Washington, DC, USA: IEEE Computer Society, 2003. (ICSM '03), p. 23–. ISBN 0-7695-1905-9. Disponível em: <http://dl.acm.org/citation.cfm?id=942800.943568>.

FRANZ, L.; SHIH, J. Estimating the value of inspections for early testing for software projects. v. 45, 1994.

GAIZAUSKAS, R. J.; WILKS, Y. Information extraction: Beyond document retrieval. *Journal of Documentation*, v. 54, p. 70–105, 1998.

B. George. *Automatic pre-detection of potential coding issues and recommendation for resolution actions*. 2016. USPTO 9,519,477. Disponível em: <https://patentimages.storage.googleapis.com/53/97/01/36ad9be8c95c81/US9519477.pdf>.

GERMáN, D. M. Using software trails to reconstruct the evolution of software. *Journal of Software Maintenance*, v. 16, p. 367–384, 11 2004.

GILB, T.; GRAHAM, D.; FINZI, S. *Software inspection*. Wokingham, England ; Reading, Mass: Addison-Wesley, 1993. ISBN 978-0-201-63181-4.

GIMPEL, K.; SCHNEIDER, N.; O'CONNOR, B.; DAS, D.; MILLS, D.; EISENSTEIN, J.; HEILMAN, M.; YOGATAMA, D.; FLANIGAN, J.; SMITH, N. A. *Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments:*. Fort Belvoir, VA, 2010. Disponível em: <http://www.dtic.mil/docs/citations/ADA547371>.

GLASS, R. L.; COLLARD, R.; BERTOLINO, A.; BACH, J.; KANER, C. Software testing and industry needs. *IEEE Softw.*, v. 23, n. 4, p. 55–57, 2006.

GOMES, R. M. Mineração de texto na desambiguação. 2008.

GONZALEZ, M. A. I. Recuperação de Informação e Processamento da Linguagem Natural. p. 83, 2002.

GRADY, R. B.; Van Slack, T. Key lessons in achieving widespread inspection use. *j-IEEE-SOFTWARE*, v. 11, n. 4, p. 46–57, jul. 1994. ISSN 0740-7459 (print), 0740-7459 (electronic).

HACIOGLU, K.; PRADHAN, S.; WARD, W.; MARTIN, J. H.; JURAFSKY, D. Semantic Role Labeling by Tagging Syntactic Chunks. p. 4, 2004.

HARJUMAA, L. *Improving the software inspection process with patterns*. Tese (Doutorado) — Oulun yliopisto, Oulu, 2005. OCLC: 936699425.

HARJUMAA, L.; TERVONEN, I.; VUORIO, P. Improving software inspection process with patterns. In: *Fourth International Conference onQuality Software, 2004. QSIC 2004. Proceedings.* [S.l.: s.n.], 2004. p. 118–125.

HEDBERG, H. Introducing the Next Generation of Software Inspection Tools. Springer Berlin Heidelberg, Berlin, Heidelberg, v. 3009, p. 234–247, 2004. Disponível em: <http://link.springer.com/10.1007/978-3-540-24659-6_17>.

HOOIMEIJER, P.; WEIMER, W. Modeling bug report quality. In: *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering - ASE '07*. Atlanta, Georgia, USA: ACM Press, 2007. p. 34. ISBN 978-1-59593-882-4. Disponível em: <http://portal.acm.org/citation.cfm?doid=1321631.1321639>.

HOTHO, A.; NüRNBERGER, A.; PAASS, G. A brief survey of text mining. *LDV Forum - GLDV Journal for Computational Linguistics and Language Technology*, v. 20, p. 19–62, 01 2005.

IGNATOW, G.; MIHALCEA, R. *An Introduction to Text Mining: Research Design, Data Collection, and Analysis*. [S.l.]: SAGE Publications, 2017.

JACKSON, P.; MOULINIER, I. *Natural language processing for online applications: text retrieval, extraction and categorization*. Amsterdam: Benjamins, 2002. (Natural language processing, 5). OCLC: 845507522. ISBN 978-90-272-4988-3 978-1-58811-249-1 978-1-58811-250-7 978-90-272-4989-0.

JONES, D. B.; SOMERS, H. *New Methods In Language Processing*. [S.l.]: Routledge, 2013. ISBN 978-1-134-22738-9.

JONES, K. S.; WILLETT, P. (Ed.). *Readings in Information Retrieval*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997. ISBN 1-55860-454-5.

JUNIOR, J. M. Classificação de páginas na internet. p. 89, 2003.

JURAFSKY, D.; MARTIN, J. H. *Speech and Language Processing (2Nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2009. ISBN 0131873210.

KATASONOV, A.; SAKKINEN, M. Requirements quality control: a unifying framework. *Requirements Engineering*, v. 11, n. 1, p. 42–57, 2005.

KOMSSI, M.; KAUPPINEN, M.; PYHAJARVI, M.; TALVIO, J.; MANNISTO, T. Persuading Software Development Teams to Document Inspections: Success Factors and Challenges in Practice. In: . IEEE, 2010. p. 283–288. ISBN 978-1-4244-8022-7. Disponível em: <http://ieeexplore.ieee.org/document/5636543/>.

LAITENBERGER, O.; DEBAUD, J.-M. An encompassing life cycle centric survey of software inspection. *Journal of Systems and Software*, v. 50, n. 1, p. 5–31, jan. 2000. ISSN 01641212. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0164121299000734>.

LATONYA, P. *The Benefit of Software Release Notes and Why Your Company Should Use Them*. 2013. Disponível em: <https://www.seguetech.com/benefit-software-release-notes/>.

LIMA, C.; SANTOS, I.; BARROS, F.; MOTA, A. Spt: A text mining process to extract relevant areas from sw documents to exploratory tests. In: . [S.l.]: In 2018 7th Brazilian Conference on Intelligent Systems (BRACIS) IEEE., 2018. p. 254–259.

LOPES, G. Um framework para a construÇÃo de mediadores oferecendo eliminaÇÃo de duplicatas. 2011. Disponível em: <https://www.maxwell.vrac.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=16775@1>.

LUTZ, M.; ASCHER, D. *Aprendendo Python, 2.ed.* Bookman, 2007. ISBN 9788577800131. Disponível em: <https://books.google.com.br/books?id=FloUD_kfc1cC>.

MACDONALD, F.; MILLER, J.; BROOKS, A.; ROPER, M.; WOOD, M. Automating the Software Inspection Process. p. 39, 1995.

MACHADO, A. P.; FERREIRA, R.; BITTENCOURT, I. I.; ELIAS, E.; BRITO, P.; COSTA, E. Muneração de texto em redes sociais aplicada à educação a distância. v. 6, p. 21, 2010.

MADUREIRA, F. A. P. Classificação de documentos. *Dissertação submetida para obtenção do grau de Mestre em Engenharia de Informática. Universidade de Nova Lisboa, Portugal*, 2009.

MAGALHãES, C.; ANDRADE, J.; PERRUSI, L.; MOTA, A. Evaluating an automatic text-based test case selection using a non-instrumented code coverage analysis. In: *Proceedings of the 2Nd Brazilian Symposium on Systematic and Automated Software Testing*. New York, NY, USA: ACM, 2017. (SAST), p. 5:1–5:9. ISBN 978-1-4503-5302-1. Disponível em: <http://doi.acm.org/10.1145/3128473.3128478>.

MAGALHãES, C.; BARROS, F.; MOTA, A.; MAIA, E. Automatic selection of test cases for regression testing. In: *Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing*. New York, NY, USA: ACM, 2016. (SAST), p. 8:1–8:8. ISBN 978-1-4503-4766-2. Disponível em: <http://doi.acm.org/10.1145/2993288.2993299>.

MAGALHãES, C.; MOTA, A.; MAIA, E. Automatically finding hidden industrial criteria used in test selection. In: *SEKE*. [S.l.]: KSI Research Inc. and Knowledge Systems Institute Graduate School, 2016. p. 470–473.

MANNING, C.; RAGHAVAN, P.; SCHUETZE, H. Introduction to Information Retrieval. p. 581, 2009.

MANNING, C.; SURDEANU, M.; BAUER, J.; FINKEL, J.; BETHARD, S.; MCCLOSKY, D. The Stanford CoreNLP Natural Language Processing Toolkit. In: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Baltimore, Maryland: Association for Computational Linguistics, 2014. p. 55–60. Disponível em: <http://aclweb.org/anthology/P14-5010>.

MÄNTYLÄ, M.; ITKONEN, J.; IIVONEN, J. Who tested my software? Testing as an organizationally cross-cutting activity. *Software Quality Journal*, v. 20, p. 145–172, mar. 2012.

MANTYLA, M. V.; LASSENIUS, C. What types of defects are really discovered in code reviews? *IEEE Trans. Software Eng.*, v. 35, n. 3, p. 430–448, 2009.

MARTIN, D.; ROOKSBY, J.; ROUNCEFIELD, M.; SOMMERVILLE, I. 'good' organisational reasons for 'bad' software testing: An ethnographic study of testing in a small software company. In: *29th International Conference on Software Engineering (ICSE'07)*. [S.l.: s.n.], 2007. p. 602–611. ISSN 0270-5257.

MARTIN, D.; ROOKSBY, J.; ROUNCEFIELD, M.; SOMMERVILLE, I. 'good' organisational reasons for 'bad' software testing: An ethnographic study of testing in a small software company. In: *Proceedings of the 29th International Conference on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2007. (ICSE '07), p. 602–611. ISBN 0-7695-2828-7. Disponível em: <https://doi.org/10.1109/ICSE.2007.1>.

MARTINS, C. A.; GODOY, D.; MONARD, M. C.; MATSUBARA, E. T.; AMANDI, A. Uma experiência em minerção de textos utilizando clustering probalistico clustering hierárquico. p. 22, 2003.

MCCALLUM, A.; NIGAM, K. Text classification by bootstrapping with keywords, em and shrinkage. In: *Workshop On Unsupervised Learning In Natural Language Processing*. [S.l.: s.n.], 1999. p. 52–58.

MCCALLUM, A.; NIGAM, K. Text Classification by Bootstrapping with Keywords, EM and Shrinkage. p. 7, 1999.

MORENO, L.; BAVOTA, G.; PENTA, M. D.; OLIVETO, R.; MARCUS, A.; CANFORA, G. Arena: An approach for the automated generation of release notes. *IEEE Transactions on Software Engineering*, v. 43, n. 2, p. 106–127, Feb 2017. ISSN 0098-5589.

OWOPUTI, O.; O'CONNOR, B.; DYER, C.; GIMPEL, K.; SCHNEIDER, N.; SMITH, N. A. Improved Part-of-Speech Tagging for Online Conversational Text with Word Clusters. In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, 2013. p. 380–390. Disponível em: <http://www.aclweb.org/anthology/N13-1039>.

PAULO, S. Dissertação apresentada ao Instituto de Matemática e Estatística da Universidade de São Paulo para obtenção do título de Mestre em Ciências. p. 117, 2013.

PERNA C. L.; DELGADO, H. K. F. M. J. *Linguagens Especializadas em Corpora: modos de dizer e interfaces de pesquisa*. [S.l.]: EDIPUCRS, 2010.

PEZZINI, A. MINERAÇÃO DE TEXTOS: CONCEITO, PROCESSO E APLICAÇÕES. *Revista Eletrônica do Alto Vale do itajaí*, v. 5, n. 8, p. 058–061, fev. 2017. ISSN 23164190. Disponível em: <http://www.revistas.udesc.br/index.php/reavi/article/view/6750/6415>.

PORTER, A. A. Assessing Software Review Meetings: Results of a Comparative Analysis of Two Experimental Studies. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, v. 23, n. 3, p. 17, 1997.

RATNAPARKHI, A. A Maximum Entropy Model for Part-Of-Speech Tagging. In: *Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 1996. Disponível em: <http://aclweb.org/anthology/W/W96/W96-0213>.

Rothermel, G.; Harrold, M. J. Analyzing regression test selection techniques. *IEEE Transactions on Software Engineering*, v. 22, n. 8, p. 529–551, Aug 1996. ISSN 0098-5589.

RUNESON, P.; ALEXANDERSSON, M.; NYHOLM, O. Detection of Duplicate Defect Reports Using Natural Language Processing. In: *29th International Conference on Software Engineering (ICSE'07)*. Minneapolis, MN, USA: IEEE, 2007. p. 499–510. ISBN 978-0-7695-2828-1. Disponível em: <http://ieeexplore.ieee.org/document/4222611/>.

SAUER C., J. R. L. L.; YETTON, P. A behaviourally motivated programme for empirical research into software development technical review. 1996.

SHIREY, G. C. *In Proceedings of the Ninth International Conference on Testing Computer Software*, p. 151–159, 1992.

ŚLIWERSKI, J.; ZIMMERMANN, T.; ZELLER, A. When do changes induce fixes? In: *Proceedings of the 2005 International Workshop on Mining Software Repositories*. New York, NY, USA: ACM, 2005. (MSR '05), p. 1–5. ISBN 1-59593-123-6. Disponível em: <http://doi.acm.org/10.1145/1082983.1083147>.

STåLHANE, T.; TANVEER, H. Improving the software inspection process. p. 163–74, 2005.

TAN, A.-H.; RIDGE, K.; LABS, D.; TERRACE, H. M. K. Text mining: The state of the art and the challenges. 11 2000.

TERVONEN, I.; IISAKKA, J. Monitoring software inspections with prescriptive metrics. p. 11, 1993.

THUNG, F.; LE, T.-D. B.; KOCHHAR, P. S.; LO, D. BugLocalizer: integrated tool support for bug localization. In: *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2014*. Hong Kong, China: ACM Press, 2014. p. 767–770. ISBN 978-1-4503-3056-5. Disponível em: <http://dl.acm.org/citation.cfm?doid=2635868.2661678>.

TURBAN, E.; MCLEAN, E.; WETHERBE, J. *Tecnologia da informação para gestão: transformando os negócios na economia digital*. Bookman, 2004. ISBN 9788536303918. Disponível em: <https://books.google.com.br/books?id=Nr5EuQAACAAJ>.

VAUCHER, S.; SAHRAOUI, H.; VAUCHER, J. Discovering new change patterns in object-oriented systems. In: *Proceedings of the 2008 15th Working Conference on Reverse Engineering*. Washington, DC, USA: IEEE Computer Society, 2008. (WCRE '08), p. 37–41. ISBN 978-0-7695-3429-9. Disponível em: <https://doi.org/10.1109/WCRE.2008.32>.

WHEELER D. A., B. B.; MEESON, R. N. Software inspection - an industrial best practice. 1996.

WIEGERS, K. E. *Peer reviews in software: a practical guide*. Boston, MA: Addison-Wesley, 2002. (The Addison-Wesley information technology series). ISBN 978-0-201-73485-0.

WITTEN, I.; FRANK, E.; HALL, M. *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier Science, 2011. (The Morgan Kaufmann Series in Data Management Systems). ISBN 9780080890364. Disponível em: <https://books.google.com.br/books?id=bDtLM8CODsQC>.

WIVES LEANDRO E LOH, S. Recuperação de informações usando a expansão semântica e a lógica difusa. 1998.

XIA, X.; LO, D.; WEN, M.; SHIHAB, E.; ZHOU, B. An Empirical Study of Bug Report Field Reassignment. p. 10, 2014.

ZHANG, J.; WANG, X.; HAO, D.; XIE, B.; ZHANG, L.; MEI, H. A survey on bug-report analysis. *Science China Information Sciences*, v. 58, n. 2, p. 1–24, fev. 2015. ISSN 1674-733X, 1869-1919. Disponível em: <http://link.springer.com/10.1007/s11432-014-5241-2>.