UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
UNIDADE ACADÊMICA DE GRADUAÇÃO
CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ANDERSON SELESTINO CAMARGO

A MINERAÇÃO DE TEXTO APLICADA AO REGISTRO DE NÃO CONFORMIDADES: UMA PROPOSTA PARA A REDUÇÃO DE RETRABALHO NO DESENVOLVIMENTO DE SOFTWARE

São Leopoldo
2016

ANDERSON SELESTINO CAMARGO

A MINERAÇÃO DE TEXTO APLICADA AO REGISTRO DE NÃO CONFORMIDADES: UMA PROPOSTA PARA A REDUÇÃO DE RETRABALHO NO DESENVOLVIMENTO DE SOFTWARE

Artigo apresentado como requisito parcial para obtenção do título de Graduado em Análise e Desenvolvimento de Sistemas, pelo Curso de Análise e Desenvolvimento de Sistemas da Universidade do Vale do Rio dos Sinos - UNISINOS

Orientador (a): Ms. Rosemary Francisco

São Leopoldo

2016

A MINERAÇÃO DE TEXTO APLICADA AO REGISTRO DE NÃO CONFORMIDADES: UMA PROPOSTA PARA A REDUÇÃO DE RETRABALHO NO DESENVOLVIMENTO DE SOFTWARE

Anderson Selestino Camargo
Orientador (a): Ms. Rosemary Francisco

Resumo: No contexto de desenvolvimento de software erros podem ser frequentes. Desta forma, a cada erro de sistema encontrado, o *bug* (não conformidade) é classificado, reportado e repassado ao responsável. Contudo, muitas vezes não conformidades semelhantes ou iguais têm recorrência e geralmente são cadastradas novamente na ferramenta de *bug tracking*[1], isso acaba gerando cadastros duplicados e retrabalho (classificação do incidente já descrito, reanálise, suporte, e etc.). Neste contexto, este trabalho tem por objetivo, analisar como o *Text Mining* (mineração de texto) pode auxiliar na redução do retrabalho no registro de não conformidades em equipes de desenvolvimento e manutenção de software. Para a avaliação deste trabalho, aplicou-se questionário onde no resultado obtido, pôde-se perceber que a solução atingiu o objetivo: auxiliou na redução do retrabalho no registro de não conformidades em equipes de desenvolvimento e manutenção de software. Para a academia, o trabalho contribui com o estudo sobre o *Text Mining*, custo e retrabalho em desenvolvimento de software. Também contribui, abordando como que, a utilização de métodos que empregam *Text Mining*, pode colaborar para a redução de não conformidades em empresas que realizam o desenvolvimento e correções de software.

Palavras-chave: Retrabalho no Desenvolvimento de Software, *Text Mining*, Duplicidade de Não Conformidades, Garantia da Qualidade, Desenvolvimento de Software.

1. INTRODUÇÃO

Ao realizar o desenvolvimento de software, erros são encontrados, classificados e reportados. Contudo, nem sempre é tomado o devido cuidado ao reportar erros (não

[1] Sistema de *Bug Tracking* é um sistema que auxilia no processo de Garantia da Qualidade, gerenciando as não conformidades encontradas. Por ele, os envolvidos nos testes do sistema podem reportar falhas que encontraram no software.

conformidades[2]). (TIAN; SUN; LO, 2012). Em uma equipe de desenvolvimento de sistema, não conformidades são reportadas por todas as pessoas responsáveis pelo desenvolvimento (testadores, desenvolvedores, analistas de sistema e/ou clientes), o que pode dificultar a análise e/ ou correção de erros. (NETTO; BARROS; ALVIM, 2010).

Para melhorar a qualidade dos sistemas de software, os desenvolvedores muitas vezes permitem que os usuários finais forneçam *feedback* sobre não conformidades que encontrarem. (TIAN; SUN; LO, 2012). Os autores ainda citam que os usuários finais podem reportar erros encontrados cadastrando em um sistema de gerenciamento de não conformidades como o *Bugzilla*. Este processo de registro de não conformidades, no entanto, é descoordenado e distribuído, o que significa que muitos usuários poderão enviar relatórios de erros relatando o mesmo problema. De acordo com Tian, Sun e Lo (2012), não conformidades relatadas mais de uma vez para um mesmo problema são classificadas como duplicadas. Os autores também observam que a existência de muitas não conformidades duplicadas podem causar esforço manual desnecessário frequente, onde se devem classificar manualmente estas como duplicadas.

Outro desafio, é que muitas vezes há muitos relatórios de erros duplicados para o mesmo problema. Os desenvolvedores de software ou testadores normalmente tem que revisar esses relatórios de erros redundantes manualmente, o que é demorado e tem um custo ineficiente. (WU et al., 2011).

O processo de garantia da qualidade tem como objetivo “[...] assegurar que os produtos de trabalho e a execução dos processos estejam em conformidade com os planos, procedimentos e padrões estabelecidos. ” (MPS.BR, 2012, p.32). Empresas desenvolvedoras de software realizam verificações e validações nos softwares, gerando testes e com os testes executados, não conformidades são encontradas. Com isso, são alocados desenvolvedores para realizar correções dos erros reportados pela equipe. (ROUILLER et al., 2006; NETTO; BARROS; ALVIM, 2010).

De acordo com Boehm e Basili (2001), o custo destas correções pode consumir até 70% do orçamento previsto para o projeto de desenvolvimento do software. Tendo por exemplo a métrica de orçamento citada por Boehm e Basili (2001), simulando uma empresa desenvolvedora de software, onde cada não conformidade reportada tenha outra não

[2] No âmbito deste trabalho não conformidade é representada por qualquer anomalia no comportamento do software, sendo, portanto, erros de sistema.

conformidade similar ou igual, pode-se considerar que o orçamento a ser gasto pode ser superior aos 70% do orçamento previsto para projeto de desenvolvimento.

Além disso, segundo Glenford Myers (1979), o custo da correção de um erro em ambiente de codificação, equivale à dez vezes o valor da correção do mesmo erro na etapa de engenharia de requisitos. Tomando por base este valor, quando se tem o mesmo erro reportado duas vezes, este custo dobra, além de ser necessário, alocar o mesmo profissional ou outro, duas vezes para ver a mesma não conformidade.

Para este trabalho, o custo está baseado no esforço atrelado às não conformidades. Desta forma, diminuir a quantidade de retrabalho[3] no âmbito do desenvolvimento, torna-se fundamental, haja vista que isso auxilia no aumento da produtividade e uma redução no custo do desenvolvimento de software.

Com base nos motivos descritos acima, este trabalho se propôs a investigar a seguinte hipótese: "Como o *Text Mining* pode reduzir o retrabalho no registro de não conformidades no desenvolvimento de software em empresas que realizam melhorias e manutenções no software? "

Com o intuito de validar a hipótese, o objetivo geral do trabalho é analisar como o *Text Mining* pode reduzir o retrabalho no registro de não conformidades em equipes de desenvolvimento e manutenção de software. Para tanto, o trabalho foi desenvolvido de acordo com os seguintes objetivos específicos: a) mapear as possíveis causas de retrabalho no cadastro de não conformidades no desenvolvimento de software em empresas. b) identificar como o uso de *Text Mining* tem sido utilizado na literatura. c) elaborar uma abordagem utilizando *Text Mining* para reduzir o retrabalho no cadastro de não conformidades no desenvolvimento de software em empresas.

Tarefas duplicadas, ou similares, geram um esforço maior para serem corrigidas, e isto pode acarretar em atrasos na entrega do projeto. (CORDEIRO; FREITAS, 2011). Com a diminuição dos desperdícios, diminuem-se também os gastos decorrentes da má utilização dos recursos disponíveis, o que agrega valor à empresa e também à sua qualidade, gerando mais lucro e satisfação ao cliente. (CAMPÃO et al., 2012).

Já para Junior e Carvalho (2014), avaliar e classificar corretamente as não conformidades, contribui para a melhoria do controle, e também na redução de erros encontrados, culminando na otimização do processo de desenvolvimento de sistemas e para a

[3] Este trabalho acadêmico considera como retrabalho o registro de duas ou mais não conformidades iguais e/ou semelhantes que já tenham sido relatadas ou mesmo solucionadas.

qualidade do produto entregue ao cliente final. Além disso, toda a tentativa de reduzir a quantidade de retrabalho em empresas de desenvolvimento de software torna-se bem vista.

O presente trabalho está estruturado em 5 capítulos. O capítulo 2 apresenta a Revisão de Literatura. O capítulo 3 apresenta a Metodologia de Pesquisa utilizada. Já o capítulo 4, apresenta a Avaliação da Solução Proposta. E, o capítulo 5, apresenta a Conclusão deste trabalho.

2. REVISÃO BIBLIOGRÁFICA

O presente capítulo visa demonstrar o que a literatura apresenta sobre desenvolvimento de software, registro de não conformidades, retrabalho, custo de software, *Text Mining*, *Text Mining* no desenvolvimento de software e os trabalhos relacionados encontrados. O capítulo está dividido em 6 seções. O tópico 2.1 apresenta o Desenvolvimento de Software. O tópico 2.1.1 apresenta o Registro de Não Conformidades. O tópico 2.1.2 apresenta o Retrabalho e Custo de Software. O tópico 2.2 apresenta o *Text Mining.* Já o tópico 2.2.1, apresenta o *Text Mining* no Desenvolvimento de Software. E, o tópico 2.3, apresenta os Trabalhos Relacionados.

2.1. DESENVOLVIMENTO DE SOFTWARE

Na atualidade, o software destaca-se estrategicamente nas organizações, e vem sendo cada vez mais utilizado como instrumento de apoio às atividades e à tomada de decisão. (CORDEIRO, 2011). Com base na importância do software, Pressman (2006) destaca que a comunidade de desenvolvimento de software, busca desenvolver soluções que tornem o desenvolvimento de software mais fácil, mais rápido e menos dispendioso com qualidade, em um processo de melhoria contínua.

Desde a crise do software, surgiu a necessidade de tornar o desenvolvimento de Software um processo estruturado, planejado e orientado a padrões, para que as necessidades fossem atendidas e os gastos com a informatização de processos de informações se tornassem compensadores. (MAINART; SANTOS, 2010).

Segundo Carvalho e Mello (2012), o desenvolvimento de software tem as seguintes fases: requisitos, análise, projeto e entrega. Na fase de requisitos, por meio de reuniões com todos os envolvidos no projeto (clientes, investidores e parceiros), são levantadas pela equipe

de desenvolvimento, todas as necessidades do negócio e os recursos a serem desenvolvidos para o software. (CARVALHO; MELLO, 2012).

Pressman (2006) cita que a fase de análise tem por objetivo produzir a especificação de requisitos, que formalmente é um documento contendo a descrição funcional do sistema. O autor ainda cita que a fase de projeto de software compreende o desenvolvimento do sistema, onde é realizada a codificação a partir dos requisitos de sistema. E ele cita também que esta etapa é focada em 4 áreas: dados, arquitetura, interface e componentes de sistema. Carvalho e Mello (2012), lembram também que nesta etapa do processo de desenvolvimento os testes são iniciados.

Carvalho e Mello (2012) dizem que em metodologias ágeis a fase de entrega é realizada diversas vezes durante o projeto, culminando em um produto com melhorias incrementais em relação ao produto da entrega anterior. Os autores ainda citam que ao final da etapa de entrega, uma nova versão de sistema será entregue ao cliente final.

Com base nas observações vistas pelos autores sobre o desenvolvimento de sistemas, torna-se necessário aprofundar-se em como é realizado o registro de não conformidades encontradas durante o desenvolvimento de software. O tópico 2.1.1 abordará o que os autores da literatura observam sobre o registro de não conformidades.

2.1.1. REGISTRO DE NÃO CONFORMIDADES

Qualidade é definida pelo PMI como o grau até o qual um conjunto de características inerentes satisfaz as necessidades. (OLIMPIO, 2011). Segundo Mello (2002), o sistema de gestão da qualidade (SGQ) é referente a tudo o que uma empresa desempenha no gerenciamento de seus processos ou atividades.

Olimpio (2011) diz que não conformidades são entendidas como processos ou produtos que não estão em acordo com os requisitos determinados no Sistema de Gestão da Qualidade da organização. Olimpio (2011, p.17) também cita que “Quando um serviço ou produto realizado deixa de atender a uma ou mais necessidades configura-se uma não conformidade. ” Deming (1990) aborda que uma não conformidade é o não atendimento a um requisito especificado.

O MPS.BR (2012, p.32) descreve que o processo de garantia da qualidade, “[...] é assegurar que os produtos de trabalho e a execução dos processos estejam em conformidade com os planos, procedimentos e padrões estabelecidos. ”, ou seja o MPS.BR (2012) visa complementar a definição de não conformidade citada por Deming.

Para Rodrigues (2011, p. 478), “A verificação e validação (V&V) tem como objetivo mostrar que um software está em conformidade com sua especificação e atende às expectativas do cliente. ”, isto é, Rodrigues também denota a preocupação com a execução de testes, possibilitando assegurar a garantia da qualidade tal como o modelo MPS.BR (2012) descreve.

Segundo Sommerville (2007), a verificação envolve processos como, por exemplo, inspeções e revisões, a cada estágio do processo de software, desde a definição de requisitos do usuário até o desenvolvimento do programa. No processo de validação, os maiores esforços são centralizados nos testes, quando a versão operacional é testada. Os estágios do processo de teste são: teste de componente (ou unidade), teste de sistema e teste de aceitação. (SOMMERVILLE, 2007).

Portanto, equipes de testes, clientes, desenvolvedores e/ou analistas de sistemas, realizam testes durante todo o ciclo de desenvolvimento de sistemas, onde a cada teste realizado, não conformidades podem ser encontradas, registradas e reportadas aos responsáveis pela sua correção. (NETTO, 2010). Equipes de testes verificam a conformidade dos sistemas desenvolvidos, e têm por missão garantir que o produto entregue ao cliente esteja coeso com os requisitos do projeto e que atenda às expectativas dos usuários. (JUNIOR; CARVALHO, 2014).

Segundo Junior e Carvalho (2014, p. 163),

> “O registro das não conformidades encontradas em teste é realizado em dois níveis, primeiro o testador classifica a não conformidade de acordo com uma relação previamente elaborada, posteriormente o erro é descrito textualmente, bem como indicadas as ações tomadas. ”

Ao final, para cada não conformidade encontrada, ações de correção são definidas e acompanhadas até a sua efetiva solução ou correção. (MPS.BR, 2012).

Segundo o modelo MPS.BR (2012, p. 32), “Quando necessário, o escalamento das ações corretivas para níveis superiores é realizado, de forma a garantir sua solução. ” Ou seja, para cada não conformidade, deve-se existir uma resolução, e caso o responsável pela correção, por algum motivo não a fizer, a ação corretiva deve ser escalada a níveis superiores, de forma que a correção da não conformidade seja realizada.

Conforme abordado neste tópico, durante o desenvolvimento de sistemas não conformidades são encontradas e reportadas, o que resulta em um custo para a correção. Com isso, torna-se necessário buscar na literatura, a resposta para as seguintes perguntas: “Qual é o custo da correção de não conformidades? ” e “Existe retrabalho ao realizar a correção de não

conformidade? ". O tópico 2.1.2, visa buscar na literatura, sobre o que os autores abordam sobre Retrabalho e Custo de Software.

2.1.2. RETRABALHO E CUSTO DE SOFTWARE

De acordo com Boehm e Basili (2001), o custo das correções no processo de desenvolvimento de software pode consumir até 70% do orçamento previsto para o projeto. Pontes (2009) cita que as atividades de teste podem ser responsáveis por uma parcela considerável dos custos de um projeto.

Segundo Olimpio (2011), quando uma empresa desenvolvedora de software, segue alguma norma ou boa prática no desenvolvimento de software, como por exemplo a ISO 9000 isso garante que os processos são realizados da melhor maneira. O autor ainda cita que quando o registro de não conformidades é feito de forma manual, o processo de registrar não conformidades torna-se custoso e burocrático.

Utilizando a boa prática no processo de desenvolvimento de software, Lewis (2004, p.18), nos lembra que a definição para *Software Quality Assurance* (SQA) é "atividades sistemáticas fornecendo evidências para o uso pretendido para o produto total de software". Campos (2008) aborda que o SQA também é compreendido e composto por um grupo de pessoas relacionadas, estas sendo utilizadas através de todo o ciclo de vida de engenharia de software que positivamente influenciam e quantificam a qualidade do software que está sendo entregue. Já para Bartié (2002, p.29), "Um dos maiores desafios a ser considerados é estabelecer um modelo de custos relacionados à implantação de um processo de garantia da qualidade de software. "

No modelo apresentado por Bartié (2002), defende-se a criação de três categorias distintas para os custos relacionados à conformidade e não conformidade:

> 1. Custo da Detecção de Defeitos: Nesta categoria pode-se realizar referências para a expressão controle de qualidade, ou seja, o foco está no produto. Suas atividades executadas são orientadas ao produto concebido, e incluem: Revisões de requisitos, revisões de modelagem, revisões de planos de teste, inspeções de código e testes de software;
>
> 2. Custo da Prevenção de Defeitos: A prevenção de defeitos está associada à garantia da qualidade, ou seja, o foco está exatamente no processo. As atividades aqui realizadas são orientadas ao processo, e incluem: Definição de

> Metodologias, treinamentos, ferramentas de apoio ao processo de desenvolvimento, definição de políticas, procedimentos, padrões, especificações e convenções, planejamento do SQA, relatórios de qualidade para melhoria de processo.
>
> 3. Custo da Não conformidade: O custo da não conformidade está relacionado às perdas que o projeto terá, não optando pela detecção e prevenção de defeitos: Revisões, testes, correções de código-fonte e documentação muito constantes, reestruturação, redistribuição das versões do software, atrasos no cronograma, falhas na produção (ambiente do cliente).

Segundo Boehm (1983), quando os erros são descobertos em fases mais avançadas, a correção se torna mais difícil e várias etapas precisam ser refeitas. O retrabalho tem resultados negativos para o projeto, como o aumento dos custos e atrasos no cronograma.

Para Schiffauerova e Thomson (2003, p. 647), custo é definido por,

> "[...] não existe um acordo sobre a exata definição, mas usualmente é entendida como a soma dos custos da conformidade e dos da não conformidade. O custo da conformidade é o preço pago pela prevenção da má qualidade, e os custos da não conformidade são os custos da má qualidade causada pela falha do produto. "

O MPS.BR (2012, p.26) aborda que, "[...] O esforço e o custo para a execução das tarefas e dos produtos de trabalho são estimados com base em dados históricos ou referências técnicas". Ou seja, caso a equipe não possua dados históricos, experiência ou referências técnicas, o custo para a execução de tarefas, torna-se mais alto, haja vista que o desenvolvedor não conseguirá estimar assertivamente quanto tempo levará para corrigir a falha e/ou desenvolver uma nova funcionalidade.

Rouiller et al. (2006) abordam que retrabalho é alguma atividade que implique em alterar ou ajustar artefatos produzidos em etapas prévias do processo, por exemplo, durante a codificação precisa-se alterar um requisito mal definido. Por outro lado Silva, Souza e Camargo (2013), mencionam que no início da década de 1990, retrabalhos, acabavam por estourar os prazos dos projetos. Eles também abordam que os retrabalhos eram consequência de projetos que usavam muito tempo para documentar ao invés de codificar e testar.

Como visto na literatura, o desenvolvimento de software é uma atividade estruturada e desafiadora. Nas iterações do ciclo de desenvolvimento, não conformidades são encontradas e reportadas aos respectivos responsáveis por sua correção. Com isso, o custo de retrabalho com

relação ao desenvolvimento de software pode consumir até 70% do valor previsto para o projeto. Partindo desta afirmação, torna-se necessário estudar alguma solução para tentar diminuir este custo. No tópico 2.2, serão abordadas técnicas de *Text Mining*, a fim de fundamentar a utilização destes métodos na solução proposta por este trabalho como uma possível resposta ao problema do custo de retrabalho revisado na literatura.

2.2. *TEXT MINING*

Segundo Webber, Cristofoli e Lima (2013), a mineração de texto (*Text Mining*) é uma subárea da mineração de dados (*Data Mining*) que busca encontrar, de forma autônoma, informações (padrões e irregularidades) em dados não estruturados, sendo a maior parte em forma de texto. Rezende, Marcanini e Moura (2011) lembram que o *Text Mining* possibilita transformar uma grande quantia de dados textuais não estruturados em informação útil, muitas vezes inovadora para as empresas. E eles ainda abordam que até 2008, foi contabilizado que a humanidade produziu 487 hexabytes de informação digital.

Kuechler (2007) lembra que cerca de 80% da quantidade dos dados existentes atualmente, está salva em formato não estruturado, sendo uma grande quantidade em texto. E ele ainda aborda que as informações incluem: e-mails, arquivos eletrônicos gerados por softwares editores de texto, páginas web, etc.

Fonseca, Farias e Silva (2012) lembram que o formato de texto não estruturado foi criado para ser visualizado por humanos e são inadequados para a manipulação desta informação contida por meios computacionais. No geral estas informações são muito relevantes às organizações, haja vista que segundo Han e Kamber (2006), compõem um precioso repositório organizacional, pois compreendem o registro de histórico de atividades, memorandos, documentos internos, e-mails, projetos, estratégias e a própria compreensão adquirida.

Para Lopes (2004), Text *Mining* é o processo de retirar padrões relevantes e incomuns ou informação a partir de documentos em textos não estruturados. Segundo Fonseca, Farias e Silva (2012), o *Text Mining* permite converter parte desse conteúdo não estruturado em conhecimento útil para as empresas. Eles ainda definem o *Text Mining* como um conjunto de métodos e ações que visam encontrar conhecimento inovador em textos. Os autores ainda mencionam que esta técnica está sendo empregada em projetos de várias áreas.

Fonseca, Farias e Silva (2012) abordam que o *Text Mining* compreende uma série de técnicas que buscam extrair conteúdo de dados não estruturados com o objetivo de obter

informações, sendo que por diversas vezes, a informação contida nestes documentos não está contida de forma explícita. Os autores ainda citam que o *Text Mining* foi inspirado pelo Data Mining ou mineração de dados, onde o objetivo é descobrir padrões emergentes em banco de dados estruturados, e neste contexto o *Text Mining* tem por objetivo a extração de conhecimento útil em dados não estruturados ou semiestruturados.

Para Konchady (2006) tanto a mineração de dados quanto o *Text Mining* têm por objetivo buscar por informações escondidas, e ambos, empregam algoritmos semelhantes de Inteligência Artificial, aprendizagem de máquina, e estatística. Entretanto, o autor ainda salienta que a mineração de dados lida com dados estruturados, enquanto o *Text Mining* lida com dados não estruturados. Sendo, portanto, uma ampliação da área de *Data Mining* com o foco na análise de textos.

Wives (2002) descreve o processo de descoberta de conhecimento em textos não estruturados como uma evolução natural da recuperação de informações. O autor ainda cita que os sistemas de recuperação da informação passaram a seguir algumas técnicas de análise de informações e de aprendizado de máquina, muitas das quais vindas da área de descoberta de conhecimento em bases de dados. E ele ainda menciona que ao invés do usuário ter que examinar quais dos documentos devolutos são realmente importantes, o próprio software faria essa análise e retornaria as informações de forma sintética e concisa.

Conforme os autores abordam, o uso de técnicas de *Text Mining* torna-se atrativo quando é necessário obter o conhecimento através de texto não estruturados. Posto isso, é necessário entender como é aplicado o *Text* Mining no Desenvolvimento de Software. O tópico 2.2.1, visa mostrar o que existe na literatura sobre o uso de técnicas de *Text Mining* no desenvolvimento de software.

2.2.1. *TEXT MINING* NO DESENVOLVIMENTO DE SOFTWARE

Junior e Carvalho (2014) utilizaram uma ferramenta que foi criada utilizando o *Text Mining* como uma forma de analisar como estavam sendo classificadas não conformidades. Eles ainda reiteram que com a utilização da ferramenta foi possível melhorar o controle do processo de desenvolvimento de sistemas, reduzir a quantidade de não conformidades encontradas nos testes, reduzir o tempo de desenvolvimento e aumentar a qualidade do produto.

Por sua vez, Netto (2010) utilizou princípios do *Text Mining* para elaborar um algoritmo que sugere um cronograma autônomo considerando apenas tarefas de correção de

não conformidades. O algoritmo utiliza heurística e considera os tempos médios de tarefas que o desenvolvedor realizou para sugerir o cronograma. Eles também citam que pós-aplicação desta solução, o método proposto foi capaz de sugerir um cronograma menor que o de fato aplicado no projeto, contribuindo para que um maior número de não conformidades fosse solucionado, aprimorando a qualidade da nova versão do software, e antecipando correções de não conformidades mais críticas sob a perspectiva do cliente.

Wang et al (2008) apresentam uma técnica para verificar se novas não conformidades reportadas podem ser classificadas como registros duplicados de outras não conformidades anteriormente cadastradas na base de dados. Eles mencionam que a cada registro de não conformidade, as descrições textuais e os erros registrados são selecionados pelo sistema, do início até o evento da não conformidade em evidência. Os autores ainda descrevem que para cada atributo, a técnica os trata como um documento de texto e que a partir de técnicas de processamento de linguagem natural (NLP – *Natural Language Processing*), é gerado em um vetor de palavras chave.

Wang et al (2008) também falam que a cada novo registro de não conformidade, calcula-se índices de similaridade do novo registro com cada não conformidade já cadastrada. Eles citam que quando algum dos resultados calculados for igual ou maior a um limite pré-definido, então a nova não conformidade é considerada como registro duplicado. Os autores também mencionam que os testes realizados utilizando como indicador as não conformidades dos sistemas Eclipse e Firefox, exibiram níveis de assertividade de 67% e 93%, simultaneamente. E eles denotam que a utilização de técnicas de *Text Mining* para o ambiente de desenvolvimento de software, torna-se extremamente atrativa quando se busca reduzir o número de não conformidades que sejam duplicadas e/ou similares, o que se espera com isso, melhorar o processo de desenvolvimento de software.

A literatura mostra que o uso de *Text Mining* é atrativo para a melhoria no processo de desenvolvimento de sistemas, contudo, é necessário entender na literatura quais soluções foram elaboradas para a área de *Text Mining* e que podem ser comparadas a este trabalho científico. O tópico 2.3 almeja mostrar na literatura os trabalhos relacionados utilizados como forma de comparação a este trabalho científico.

2.3. TRABALHOS RELACIONADOS

A pesquisa bibliográfica realizada permitiu a localização de diversos trabalhos científicos abordando o *Text Mining* com o registro de não conformidades.

Junior e Carvalho (2014) utilizaram a ferramenta SOBEK que aplica conceitos de *Text Mining* para analisar como são reportadas as não conformidades pelos testadores. Os autores observaram que utilizando o *Text Mining* tem-se a oportunidade de avaliar e melhorar a classificação de não conformidades. Eles também citam que com uma melhor classificação de não conformidades, há uma melhora no controle, redução nos erros encontrados, o que contribui diretamente na melhoria do procedimento de desenvolvimento e da qualidade do produto a ser entregue ao cliente. E eles ainda lembram que após o uso da ferramenta, foi possível reduzir a quantidade de não conformidades encontradas em testes, pois segundo estes autores, a classificação correta de não conformidades pode culminar em uma melhora da qualidade do produto entregue ao cliente haja vista que dados mais assertivos contribui na tomada de decisão para a redução de erros encontrados no software.

No trabalho de Webber, Cristofoli e Lima (2013) foi proposta a criação de uma ferramenta que utiliza conceitos de *Text Mining* para a comparação de textos e classificação destes textos em áreas científicas com base em um prévio aprendizado de máquina. Estes autores citam que com base em testes realizados, o uso de Sistemas Imunológicos Artificiais (SIA), no processo de *Text Mining*, mostrou-se de forma satisfatória por produzir resultados concisos e robustos na classificação de textos em áreas científicas. E eles ainda abordam que o sistema desenvolvido é capaz de trabalhar com a análise de diferentes textos.

A pesquisa de Tian, Sun e Lo (2012) utiliza o *Text Mining* e o aprendizado de máquina para localizar não conformidades do sistema de *bug tracking Bugzilla* que estejam duplicadas, isto é, uma não conformidade que fora reportada mais de uma vez. Eles abordam ainda que após a implementação do algoritmo foi possível aumentar a detecção de não conformidades em torno de 200%.

Netto, Barros e Alvim (2010) abordaram em suas pesquisas, a dificuldade que um gerente de projeto tem em propor um cronograma para a correção de erros encontrados no sistema. Com base nesta afirmação, os autores propõem um algoritmo que utiliza o *Text Mining* para gerar um cronograma automático escolhendo sempre o “melhor” desenvolvedor para a correção do erro de acordo com o componente. Neste caso eles abordam como o melhor desenvolvedor, aquele que realizou mais correções de determinado tipo de não conformidade, no menor prazo e com menor quantidade de defeito pós-solução aplicada (erros causados pela solução). Eles ainda citam que com a aplicação do algoritmo foi capaz de prever cronogramas mais realistas que o de fato aplicado, onde este algoritmo sugeriu bons cronogramas em poucos minutos, o que pode culminar em uma redução dos custos em manutenções de sistema e um melhor alinhamento de entregas para com o cliente.

A diferenciação no trabalho de Webber, Cristofoli e Lima (2013) com base neste trabalho acadêmico é que embora ambos utilizem o *Text Mining* como uma forma de solução, o trabalho dos autores acima é voltado à classificação de textos em áreas, já o presente artigo tem como seu principal objetivo, propor uma solução para a diminuição de retrabalho no ambiente de desenvolvimento de software, ou seja, a diminuição de não conformidades duplicadas e/ou similares. Já com relação à pesquisa de Junior e Carvalho (2014), este trabalho científico, diferencia-se, pois é proposta uma solução a não conformidades duplicadas e/ou similares, diferentemente dos autores que analisaram a classificação destas não conformidades.

Com relação ao trabalho de Tian, Sun e Lo (2012) cabe ressaltar que a proposta de solução é similar à solução proposta neste artigo: ambos abordam o problema de duplicidade de não conformidades e formas para a redução destas duplicidades. Contudo, os autores acima focam somente em não conformidades duplicadas e não abordam as similares. Este trabalho acadêmico visou também reduzir o número de não conformidades similares.

Netto, Barros e Alvim (2010) aplicam um algoritmo para a geração de cronogramas de correção de não conformidade de forma autônoma, o que por sua vez difere do foco deste artigo, pois neste abordar-se-á uma solução para o registro de não conformidades duplicadas e/ou similares. O quadro 1, sintetiza os trabalhos relacionados utilizados para este artigo.

Quadro 1 – Trabalhos Relacionados

Trabalho de pesquisa	**Utiliza os princípios do *Text Mining*?**	**O trabalho analisa as não conformidades reportadas?**	**Realiza a identificação de Texto Duplicado?**	**Realiza a identificação de Texto Similar?**
Utilização da ferramenta SOBEK para analisar como são reportadas as não conformidades, elaborada por Junior e Carvalho (2014)	Sim	Sim, os autores revisaram após o uso da ferramenta como as não conformidades eram reportadas.	Sim	Não
Ferramenta que realiza a comparação de textos e classificação de textos em áreas científicas com base em um prévio aprendizado de máquina, elaborada por Webber,	Sim	Não, o trabalho é voltado à classificação de textos em áreas científicas com base no aprendizado de máquina.	Não	Não

Cristofoli e Lima (2013)				
Ferramenta para localizar não conformidades do sistema de *bug tracking Bugzilla* que estejam duplicadas, elaborada por Tian, Sun e Lo (2012),	Sim	Sim	Sim	Não
Aplicação de algoritmo para a geração de cronogramas de correção de não conformidade de forma autônoma, elaborado por Netto, Barros e Alvim (2010)	Sim	Não, pois o trabalho aborda que o algoritmo desenvolvido visa gerar um cronograma de forma autônoma para tarefas de correção.	Não	Não
Solução proposta por este trabalho	Sim	Sim, a solução ao cadastrar novas não conformidades, revisa se não existem não conformidades similares ou idênticas já reportadas.	Sim	Sim

Fonte: Elaborado pelo autor.

3. METODOLOGIA DE PESQUISA

Este trabalho baseou-se no tipo de pesquisa Quali, Exploratória e Descritiva. Como métodos de pesquisa, foi adotado o *mix* de pesquisa bibliográfica e pesquisa documental com estudo de caso com questionário.

Para a coleta de dados, foi escolhida a empresa desenvolvedora de software A Telecon Sistemas de Automação Comercial LTDA que adota a metodologia ágil *SCRUM*. A empresa atua há 26 anos no mercado de automação comercial e como utiliza a metodologia ágil, tem seu *sprint*[4] de quatro semanas. Para a pesquisa documental, com a ferramenta de *bug tracking* da empresa, foi realizado um levantamento dos dados, por meio de consulta ao banco de dados Microsoft SQL Server, utilizado na organização, buscando tarefas que tenham sido cadastradas com atividades (“Não Identificada”, “Suporte Avançado”, “Correções Internas”,

[4] Pacote de tarefas que compõem uma nova versão de sistema.

"Correções Externas"), no período de 1 de janeiro de 2012 à 04 de agosto de 2016. Isto foi importante para comprovar a existência de tarefas finalizadas ou em aberto que estavam com a mesma descrição de outra tarefa que tenha sido cadastrada.

Na amostragem as tarefas foram revisadas manualmente e comparadas com tarefas que estavam com *status* "Pendente" ou "Finalizado". A comparação destas tarefas teve por objetivo, identificar as tarefas duplicadas ou similares às outras tarefas que já tinham alguma ação prévia (encerradas). A coleta de dados foi obtida através de consultas ao banco de dados da ferramenta de *bug tracking* por meio de consultas SQL (*Structured Query Language*, ou Linguagem de Consulta Estruturada), agrupando a descrição da tarefa. E com base nesta revisão, foram encontradas tarefas com a descrição igual à outra já cadastrada e possuindo alguma das atividades monitoradas[5].

Com a observância do problema de pesquisa, a solução proposta no trabalho foi aplicada buscando avaliar se com isso, existiria uma redução de tarefas cadastradas com a mesma descrição ou mesmo com a descrição similar. Desta forma, a ferramenta foi elaborada para salvar todos os registros de similaridade ou duplicidade encontrados, o que possibilita analisar os dados obtidos. A ferramenta foi utilizada pelos usuários do sistema (2 testadores, 3 analistas de testes, 7 desenvolvedores, 2 analistas de sistemas, 1 analista de qualidade e 1 gerente de projeto) durante o período de duas semanas.

Após a utilização como forma de avaliação da solução proposta, foi elaborado um questionário contendo seis perguntas com o escopo fechado e três perguntas com o escopo aberto, no intuito de avaliar a opinião dos usuários após o uso da ferramenta. No Apêndice A é detalhada as alternativas e possíveis repostas do questionário. O formulário foi disponibilizado no Google Formulários a 16 usuários que utilizaram a solução. Dos 16 usuários, 9 responderam o questionário. A avaliação da solução proposta por questões de tempo foi realizada em duas semanas.

O presente capítulo mostrou como foi realizada a metodologia de pesquisa, a coleta e análise de dados. Posto isso, o próximo capítulo 4, tem por objetivo demonstrar como foi concebida a solução proposta para este trabalho científico.

[5] Tarefas que estejam cadastradas com alguma das atividades: "Não Identificada", "Suporte Avançado", "Correções Internas", "Correções Externas".

4. CONCEPÇÃO DA SOLUÇÃO PROPOSTA

Ao iniciar o desenvolvimento da solução proposta para este trabalho, realizou-se pesquisas de modo a obter qual seria o melhor *framework*/ferramenta para o desenvolvimento do software. Para o desenho da solução, foi esperado encontrar *frameworks* ou funções em bancos de dados que possibilitassem desenvolver um aplicativo que utilizasse o *Text Mining* para encontrar texto similar ou duplicado em descrições de tarefas na ferramenta de *bug tracking*, do *case* utilizado como exemplo para este artigo. Com isso em mente, e pretendendo utilizar o banco de dados relacional Microsoft SQL Server 2014 (MSSQL) foi utilizado os termos "*similarity between two text sql server*" para encontrar assuntos relacionados para a elaboração da solução proposta no artigo, onde segundo Winter (2016), existem as funções de *Full-Text Search (FTS)* do MSSQL que trabalham com *Text Mining* com suporte a *Stopwords*[6] e análise léxica.

Rocha (2008) salienta que o FTS possibilita que análises de palavras em campos do banco de dados do tipo texto, XML ou binário, sejam efetuadas com flexibilizações semânticas. E ele ainda reitera que ao realizar alguma consulta ao campo do tipo texto utilizando o recurso de FTS, além de obter os registros que contenham a palavra pesquisada, também é obtido registros que incluam diferentes termos semânticos da mesma palavra, ou até mesmo outras palavras que são relacionadas aos termos pesquisados, sendo, portanto esta, a funcionalidade que o FTS entrega.

Com isso, utilizou-se para a solução proposta neste artigo, recursos do FTS, como as funções *CONTAINS* e *FREETEXTTABLE*. Maier et al. (2001) demonstram que a função *CONTAINS* é um dos recursos que o FTS que pode ser utilizada para encontrar correlações concisas ou difusas (com grau de evidência inferior) para vocábulos e frases únicas, vocábulos em certa distância entre si ou equivalências ponderadas no MSSQL. Os autores ainda abordam que a função *FREETEXTABLE*, outro recurso do FTS, possibilita executar a busca por igualdade, ou seja, ela retornará registros caso o vocábulo procurado exista no texto buscado no SGBD[7]. Maier et al. (2001) lembram que a função *FREETEXTABLE* ainda consegue realizar a busca semântica do vocábulo, isto é, buscar termos semelhantes ao vocábulo pesquisado, como por exemplo, caso seja utilizada a função, e buscado o termo "Azul", o SGDB irá procurar também pelos vocábulos "Azuis" e "Azulado", por exemplo,

[6] Palavras com muita frequência e com pouco sentido (artigos, preposições, algumas conjunções) que são desconsideradas no *Text Mining*.
[7] Sistema Gerenciador de Banco de Dados.

buscando também sinônimos do vocábulo procurado e além disso, esta função realiza uma classificação com os termos encontrados na consulta. Microsoft TECHNET (2016), define que a função *FREETEXTABLE* utiliza como base para classificação a fórmula OKAPI BM25, onde mesmo palavras geradas por flexibilização (palavras geradas como sinônimos ou similares vide exemplo acima), tem o mesmo peso para a classificação. A fórmula de classificação detalhada está disposta no apêndice B.

Com base nas funções *CONTAINS* e *FREETEXTTABLE*, a solução foi desenvolvida. O algoritmo que realiza a mineração de texto em tarefas com a descrição duplicada, utiliza a função *CONTAINS* para retornar tarefas que contenham as palavras digitadas pelo usuário. Já no caso de tarefas similares, a consulta utiliza a função *FREETEXTTABLE* que possibilita a pesquisa além do termo pesquisado, o radical do vocábulo, os sinônimos do vocábulo e a flexibilização. Ou seja, com isso é permitido buscar texto similar ao texto pesquisado pelo usuário.

Além das funções mencionadas, para realizar *Text Mining* e encontrar tarefas duplicadas, a solução utiliza os princípios do *Text Mining*. Inicialmente é removido as *Stopwords*, em seguida os termos do texto são separados, colocados em maiúsculo, removidos os caracteres especiais e é realizada a remoção da acentuação dos vocábulos. Após este processo, o sistema, realiza a consulta utilizando a função *CONTAINS*. Caso sejam retornados registros de tarefas, o sistema calcula o percentual de similaridade das tarefas encontradas com o texto digitado pelo usuário, onde cada palavra é comparada com a palavra encontrada na tarefa retornada do SGBD. Com isso o sistema soma o valor total de palavras iguais e divide pelo total de palavras da descrição fornecida pelo usuário. O cálculo da fórmula é fornecido abaixo:

D = Σ [Pt] / Σ [Pu] * 100,

Onde:

Pt = Total de Palavras da Tarefa Iguais à tarefa original

Pu = Total de Palavras da Tarefa Original

Caso a tarefa tenha o percentual de duplicidade inferior ao percentual que fora configurado no SGBD, o sistema remove a tarefa da lista de tarefas que serão exibidas como duplicadas ao usuário. Ao final do processo, o sistema realiza a soma do percentual de duplicidade e divide pelo número de tarefas encontradas, obtendo com isso a média de duplicidade. Caso esta média seja superior ou igual ao percentual configurado no SGBD e

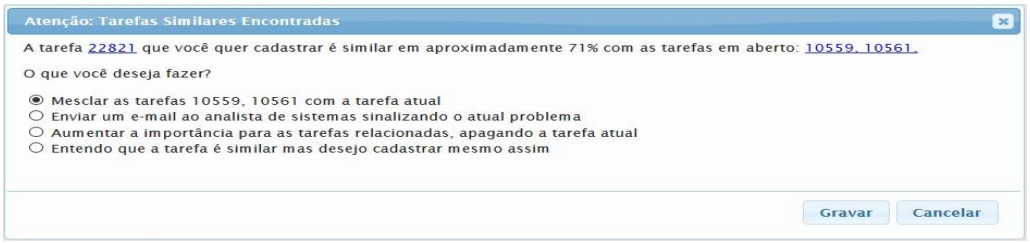
utilizado pela solução, então o sistema exibirá uma tela que solicitará ao usuário opções com relação às tarefas duplicadas. Estas opções serão detalhadas no tópico 4.1.

4.1. DESENHO DA SOLUÇÃO

A solução foi desenhada para que ao cadastrar uma tarefa com a atividade (“Não Identificada”, “Suporte Avançado”, “Correções Internas” ou “Correções Externas”), utilizando a mineração de texto e tendo por base na descrição fornecida, componente do sistema[8], alertar ao usuário quando a nova não conformidade for similar ou estar em duplicidade com outra não conformidade previamente cadastrada. Ao identificar uma não conformidade como duplicada e/ou similar à outra previamente cadastrada, a solução exibirá ao usuário algumas ações tais como: a) aumentar o esforço para a não conformidade cadastrada anteriormente; b) enviar um e-mail ao analista de sistemas avisando sobre a continuação do problema; c) mesclar ambas as tarefas (as identificadas pela ferramenta como similares) com a não conformidade a ser reportada; d) marcar a opção de cadastrar mesmo assim, neste caso a ferramenta buscará os códigos das tarefas similares ou duplicadas e, como uma forma de auxiliar no desenvolvimento da correção, colocará em passos para reproduzir da nova conformidade: “*Ver tarefa(s) x*”.

A figura 1 ilustra a tela da solução quando foram identificadas tarefas de não conformidades similares.

Figura 1 – Tarefas similares

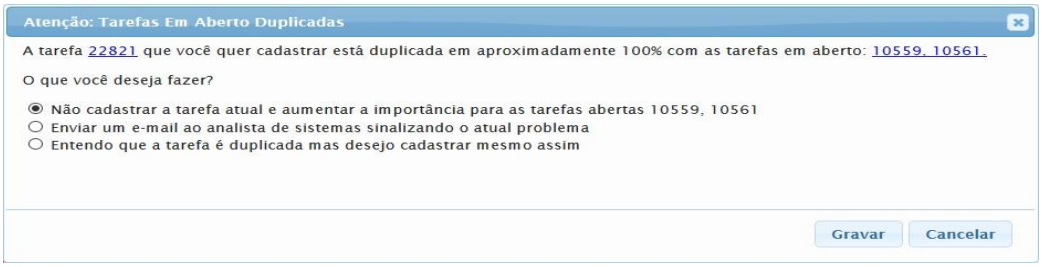


Fonte: Elaborado pelo autor.

Caso a solução identifique tarefas de não conformidades duplicadas em aberto, ao usuário será solicitado as opções conforme a figura 2.

Figura 2 – Tarefas em aberto duplicadas

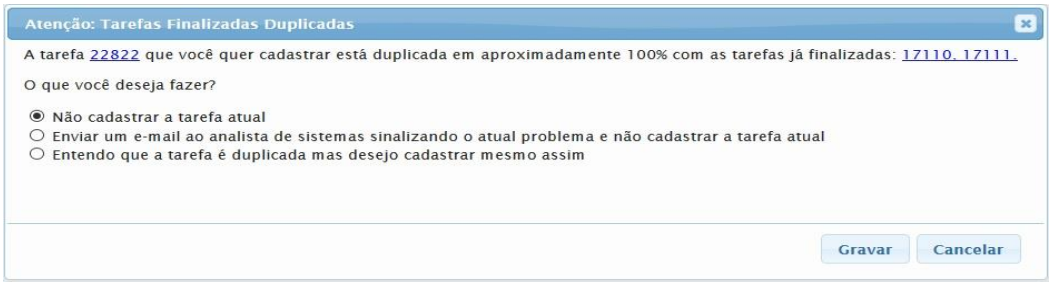
[8] Para o componente, a solução realiza a mineração de texto utilizando o módulo da nova tarefa, por exemplo, caso o componente cadastrado à nova tarefa seja Nota Fiscal de Saída, a solução buscará tarefas que possuam os componentes Notas Fiscais de Entrada, Notas Fiscais de Saída e Notas Fiscais.



Fonte: Elaborado pelo autor.

Já para as não conformidades cadastradas que possuam duplicidade com não conformidades já finalizadas, a solução proposta para este trabalho espera apresentar as opções conforme a figura 3.

Figura 3 – Tarefas finalizadas duplicadas



Fonte: Elaborado pelo autor.

Em todos os casos onde a ferramenta identificar a nova tarefa de não conformidade como similar ou duplicada à outra tarefa previamente cadastrada, o sistema exibe um *link* das tarefas encontradas, possibilitando ao usuário a comparação das tarefas encontradas.

O tópico 4.1 mostrou quais as ações que o sistema possibilita ao usuário caso a tarefa seja classificada como duplicada ou similar. Contudo, ainda não fora descrito como foi arquitetada a solução proposta neste artigo. O próximo tópico 4.2, objetiva demonstrar a arquitetura da solução proposta.

4.2. ARQUITETURA DA SOLUÇÃO

A solução foi desenvolvida utilizando as funções do banco de dados Microsoft SQL Server: *CONTAINS* E *FREETEXTTABLE*. O sistema realiza a mineração de texto para tarefas duplicadas ou similares, somente em tarefas que são cadastradas com as atividades monitoradas. Conforme dito, o sistema retorna apenas as tarefas onde a média de duplicidade for igual ou superior à média configurada para retornar as não conformidades (a média é configurada no banco de dados, na tabela “Configuracoes”, com o campo “PercentualDuplicidade”) e que possuam o mesmo módulo que a nova tarefa cadastrada. Com

relação à mineração de texto aplicada para encontrar as tarefas similares, o sistema considera também o módulo da nova tarefa e realiza o mesmo tratamento no aplicado para encontrar não conformidades duplicadas, contudo ao invés de utilizar a função *CONTAINS*, a solução utiliza a função *FREETEXTTABLE*, onde a consulta retorna apenas tarefas onde a classificação (fórmula fornecida no capítulo 4), dada pelo algoritmo *FREETEXTTABLE,* seja maior que 0. Após obter esta lista de tarefas, o algoritmo utiliza a biblioteca Lucene.Net (2016) com o método *GetDistance* da classe *JaroWinklerDistance*. Desta forma, o percentual de similaridade da palavra também é configurado via banco de dados (tabela "Configuracoes", campo "PercentualSimilaridadePalavra*"*). Então tomando por base o método *GetDistance*, é determinada a distância de similaridade entre duas palavras.

Este método utiliza o algoritmo *JaroWinklerDistance*, onde quanto maior a similaridade entre duas *strings*, a distância retornada será no máximo 1 (equivalência de 100%). O método utiliza a função Jaro-Winkler como base para o cálculo da similaridade entre *strings*.

Sendo ***c*** o número de correspondências entre os caracteres e ***t*** o número de transposições, a função Jaro, realiza o cálculo da similaridade entre duas *strings* por meio da função abaixo:

$$Jaro\ (a,b) = 1/3\ (c/a + c/b + ((c-t)\ /c)$$

Jaro-Winkler é uma variação da função Jaro que considera prefixos, de tamanho *p*, comuns em ambas *strings.* A função é definida pela equação abaixo:

$$JaroWinkler\ (a,b) = Jaro(a,b)\ + (p / 10)\ .\ (1 – Jaro\ (a, b))$$

Com isso, a solução separa as palavras das tarefas fornecidas pelo usuário, separa as palavras das tarefas encontradas e realiza a comparação de similaridade com o método RetornarDistanciaEntrePalavra da classe Texto. Ao final, caso o percentual de similaridade seja superior ou igual ao percentual configurado, é somada a variável de controle. A fórmula para retornar as tarefas é o somatório de palavras similares / número total de palavras * 100.

E para retornar as tarefas similares ao usuário, o sistema aplica a mesma fórmula já utilizada para retornar as tarefas em duplicidade, onde é somado o percentual de similaridade das tarefas / número total de tarefas encontradas. Caso esta média seja superior ou igual ao valor configurado no banco de dados, então o sistema retornará as tarefas encontradas.

Quem realiza o controle da lógica do sistema, é a biblioteca TextMining, com a classe controladora Controle pelo método AnalisarTarefa. Onde, inicialmente o sistema minera o

texto em busca de tarefas duplicadas e em aberto. Caso não encontre dados, o sistema realiza uma segunda busca, desta vez tentando encontrar tarefas duplicadas e com estado de finalizadas. Caso ainda não sejam retornadas tarefas, o sistema realiza a terceira mineração de texto, desta vez buscando tarefas similares. Após o processo, quando existem tarefas similares/duplicadas, o sistema por meio do método GravarTextMining, persiste os dados encontrados na tabela "TextMining*"* salvando o código da nova tarefa onde foi encontrado a duplicidade/similaridade, a descrição fornecida pelo usuário e os códigos das tarefas encontradas pelo algoritmo de mineração de texto. Ao final, o sistema exibe as opções de acordo com o tipo de tarefa, conforme o desenho da solução detalhado no tópico 4.1.

Com a exibição das opções, o usuário seleciona a opção que compreende como correta. Quando o usuário seleciona uma opção, o sistema então por meio do método AnalisarResposta, executa a ação solicitada pelo usuário, sendo que em tarefas que exijam alteração de dados, tais como "Mesclar Tarefas", "Aumentar a Importância" e "Excluir Tarefa", a biblioteca Agilis_web_biblioteca é utilizada para realizar a alteração e/ou exclusão da tarefa. Após, a resposta do usuário é persistida na tabela "Resposta*"*, salvando o código de usuário, o código do *Text Mining* encontrado e no campo "Acerto", se houve acerto ou erro.

Para as tarefas onde o usuário selecionar a opção "Entendo que a tarefa é duplicada/similar, mas desejo cadastrar mesmo assim", a resposta é persistida como erro, pois o usuário com a revisão manual, considerou errada a mineração de texto executada pela solução. Nas outras opções, será persistida como acerto. O Fluxo do sistema é detalhado no apêndice C. Já o diagrama de componentes, está anexo no apêndice D.

Conforme observado no tópico 4.2, a arquitetura da solução utilizou a biblioteca Lucene.NET para o cálculo da similaridade, recursos do banco de dados e a biblioteca Agilis_Web_Biblioteca para realizar a alteração de dados em tarefas. O próximo capítulo, visa abordar a avaliação da solução proposta, a análise e discussão dos resultados obtidos.

5. AVALIAÇÃO DA SOLUÇÃO PROPOSTA

5.1. CENÁRIO ATUAL DA ORGANIZAÇÃO

No atual cenário da organização as não conformidades podem ser reportadas em duas situações: durante o ambiente de testes ou durante o ambiente de produção. Quando a não conformidade é reportada durante o ambiente de testes, o testador cadastra uma nova tarefa com a descrição da não conformidade, estima a duração para esta tarefa, coloca importância

(unidade monetária utilizada no sistema de *bug tracking* para avaliar a urgência da correção, quanto maior, mais rápida será definida a correção) e atribui um desenvolvedor responsável pela correção da não conformidade encontrada. Em seguida, o desenvolvedor inicia a investigação da não conformidade, codificando se necessário a correção. Este processo é detalhado no apêndice E.

Portanto, não há uma revisão do testador com o intuito de verificar se a não conformidade reportada já não tenha sido reportada previamente. O mesmo ocorre com o desenvolvedor que ao solucionar a não conformidade acaba por dar baixa apenas na última tarefa de não conformidade, não revisando se já não existem tarefas de não conformidades similares ou iguais à última tarefa de não conformidade reportada. Este processo é realizado mais de uma vez durante o dia. Se, a cada nova não conformidade, existir outra não conformidade igual, o custo para a resolução destas não conformidades pode vir a duplicar. Com isso, aumenta-se o esforço de correção também.

Quando a não conformidade é encontrada no ambiente de produção, ela é reportada pelo cliente. O cliente entra em contado com o setor do suporte, afirmando ter descoberto uma não conformidade no sistema. Em seguida, o setor de suporte começa o atendimento, realizando uma bateria de testes para tentar descobrir a origem do problema. E, caso não consiga descobrir a origem da não conformidade ou a não conformidade é um erro de sistema, abre-se uma tarefa na ferramenta de *bug tracking* da empresa com a não conformidade descrita. Este processo é detalhado no apêndice F.

Para o processo de reportar de não conformidades durante os testes, não há uma revisão ao reportar uma nova não conformidade, verificando se já existe outra não conformidade com a mesma descrição ou similar já cadastrada. E neste caso, têm-se o mesmo problema do outro fluxo, onde poderão ser alocados mais de uma vez os recursos humanos para realizar a solução e testes da resolução das não conformidades reportadas. Nos dois fluxos mostrados, esta organização, tem como problema a não padronização de reportar não conformidades, seguindo os padrões da norma IEEE1044-2009 que descreve um padrão para reportar não conformidades. Além disso, caso a organização opte pela revisão de tarefas de não conformidade antes de abrir uma nova tarefa também de não conformidade, o processo de revisão é manual o que acaba por tornar o processo de revisar todas as tarefas de não conformidades custoso e pouco confiável.

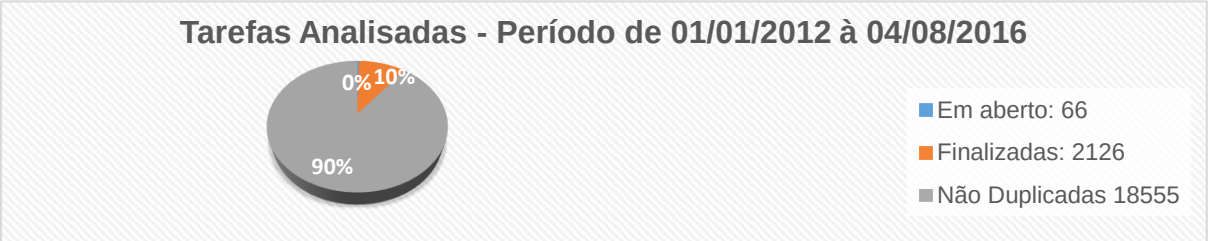
Conforme visto na seção 5.1 é demonstrado como são reportadas as não conformidades. Mas e quais dados serão analisados? Quais análises foram feitas em cima

destes dados? No próximo tópico 5.2 é objetivado demonstrar como fora realizada a análise dos dados utilizados para objetivar a importância da solução proposta.

5.2. ANÁLISE E DISCUSSÃO DOS RESULTADOS

Com base na revisão das tarefas, foram observados os seguintes dados: No período de 01 de janeiro de 2012 à 04 de agosto de 2016, foram cadastradas 20747 tarefas. Destas, 2202 estavam duplicadas, isto é a descrição da tarefa é exatamente igual à outra tarefa previamente cadastrada. 2126 tarefas foram finalizadas em duplicidade. E, com o *status* em aberto foram encontradas 66 tarefas. Os dados podem ser melhor observados no gráfico 1.

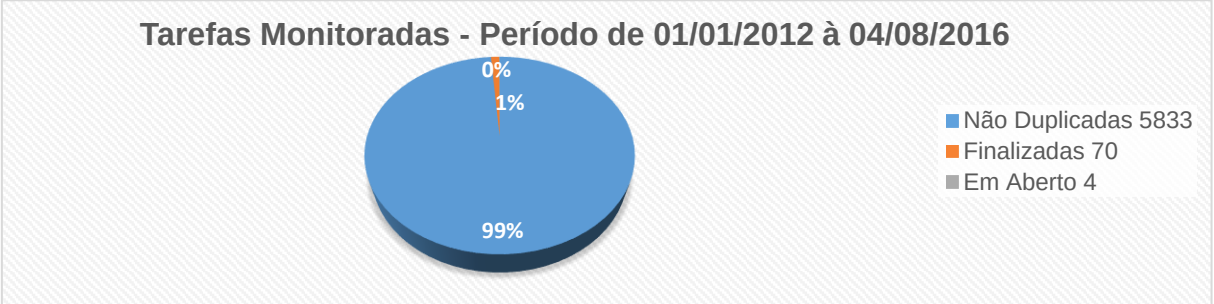
Gráfico 1 – Tarefas Analisadas no Período



Fonte: Elaborado pelo autor.

Das tarefas monitoradas por este trabalho, isto é, tarefas que tenham sido cadastradas com as atividades de "Não Identificada", "Suporte Avançado", "Correções Internas", ou "Correções Externas", no período analisado, foram cadastradas 5907. Destas tarefas com as atividades monitoradas mencionadas acima, 70 tarefas foram finalizadas em duplicidade. Além disso, havia 4 tarefas que estavam cadastradas com as atividades monitoradas, com *status "*Aberto*"*. O gráfico 2 demonstra a comparação em percentual dos resultados obtidos.

Gráfico 2 – Tarefas Monitoradas no Período

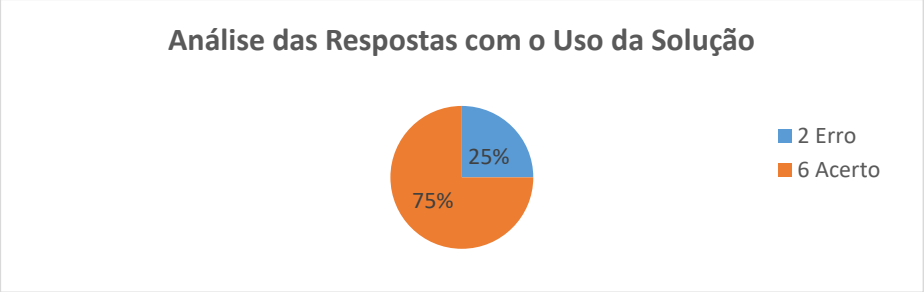


Fonte: Elaborado pelo autor.

Desta forma, buscando avaliar a solução proposta neste trabalho conforme abordado na metodologia, utilizou-se análise dos resultados persistidos pelas respostas dos usuários

com o uso da solução e questionário entre os usuários que utilizaram o sistema. A solução ao identificar uma tarefa como similar ou duplicada, persiste a resposta do usuário no banco de dados, se a mineração de texto fora satisfatória ou insatisfatória, conforme o que foi descrito no tópico 4.2. Assim, por meio de consulta SQL ao banco de dados foram analisados os registros existentes no período de 25/10/2016 à 09/11/2016. Nesta tabela foram obtidos 8 registros, dos quais 6 foram considerados acertos pelos usuários e 2 como erros. O gráfico 3 ilustra os resultados obtidos.

Gráfico 3 – Resultados Obtidos

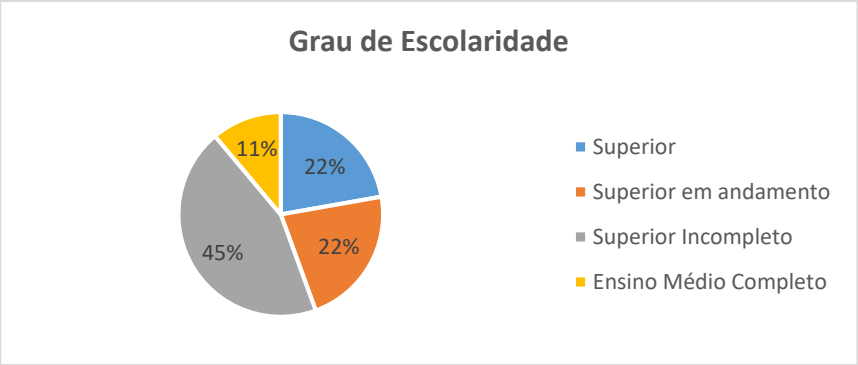


Fonte: Elaborado pelo autor.

No segundo caso foi elaborado um questionário com 9 alternativas. O objetivo deste questionário foi obter a percepção dos usuários pós aplicação da solução. Das 9 alternativas, 3 eram de escopo aberto para avaliar o perfil dos usuários e 6 de escopo fechado. O questionário foi disponibilizado de forma *online* no Google Formulários, a 16 usuários avançados, sendo 2 testadores, 3 analistas de testes, 7 desenvolvedores, 2 analistas de sistemas, 1 analista de qualidade e 1 gerente de projeto que utilizaram a ferramenta no período de 25/10/2016 a 09/11/2016. Destes, 9 usuários responderam ao questionário. Para avaliar o perfil destes usuários, os usuários responderam às afirmações de escopo aberto: “a) Qual é o seu grau de escolaridade? ”, b) “Qual é a sua idade? ” e c) “Qual é a sua profissão? ”.

Como respostas da alternativa a, observa-se que 4 (45%) dos respondentes possuem o nível Superior Incompleto, 2 (22%) dos respondentes possuem o nível Superior, 2 (22%) dos respondentes possuem o nível Superior em andamento e 1 (11%) respondente possui o Ensino Médio Completo. O gráfico 4 ilustra os resultados obtidos.

Gráfico 4 – Resultados da alternativa a



Fonte: Elaborado pelo autor.

Analisando a afirmação b, 4 (45%) dos respondentes têm a idade entre 22 e 27 anos, 4 (45%) dos respondentes tem a idade entre 28 e 37 anos e 1 (10%) tem entre 15 e 18 anos. Para a afirmação c, 4 (45%) dos respondentes são desenvolvedores, 2 (22%) dos respondentes são analistas de teste, 2 (22%) dos respondentes são testadores e 1 (11%) respondente é analista de sistemas.

Já para cada afirmação de escopo fechado, os respondentes escolheram uma das seguintes alternativas: i) "Discordo Plenamente"; ii) "Discordo"; iii) "Concordo"; e iv) "Concordo Plenamente". O quadro 2 lista as perguntas contidas no questionário e a resposta da maioria dos respondentes.

Quadro 2 – Análise dos Resultados do Questionário

Nº	**Afirmação**	**Alternativa Escolhida pela Maioria**	**Quantidade de Respondentes**
1	O sistema de *Text Mining* é de fácil compreensão.	Concordo	6 (66,7%)
2	O sistema é de fácil utilização.	Concordo	5 (55,6%)
3	O recurso de Identificar tarefas duplicadas é relevante para auxiliar na identificação de tarefas duplicadas.	Concordo	5 (55,6%)
4	O recurso de Identificar tarefas similares é relevante para auxiliar na identificação de tarefas similares.	Concordo Plenamente	5 (55,6%)
5	O sistema facilita o trabalho de comparação de tarefas similares ou duplicadas.	Concordo	4 (44,4%)
6	A solução desenvolvida auxilia ao tentar evitar o retrabalho no	Concordo Plenamente	5 (55,6%)

	cadastro de novas tarefas de não conformidades.		

Fonte: Elaborado pelo autor.

Analisando os dados obtidos pelas respostas, na afirmação 1, 8 (88,9%) dos respondentes concordam ou concordam plenamente e 1 (11,1%) respondente discorda ou discorda plenamente, da afirmação de que o sistema é de fácil compreensão. Para as afirmações 2, 3 e 4, houve uma igualdade de afirmações, onde 9 (100%) dos respondentes concordam ou concordam plenamente com as afirmações de que o sistema é de fácil uso, o recurso de identificar tarefas duplicadas é relevante ao auxiliar na identificação de tarefas duplicadas e o recurso de identificar tarefas semelhantes é relevante ao auxiliar tarefas a identificação de tarefas similares, respectivamente. Com relação à afirmação 5, 7 (77,7%) dos respondentes concordam ou concordam plenamente e 2 (23,3%) dos respondentes discordam ou discordam plenamente da afirmação de que o sistema facilita a comparação de tarefas similares ou duplicadas. E na alternativa 6, 8 (88,9%) dos respondentes concordam ou concordam plenamente e 1 (11,1%) respondente discorda ou discorda plenamente da afirmação de que o sistema auxilia ao tentar evitar o retrabalho no cadastro de novas tarefas de não conformidades.

Cabe salientar, com relação aos respondentes que discordaram da afirmação de que o sistema é de fácil compreensão pode ter ocorrido viés em suas respostas por terem confundido algum recurso da solução. Cabe salientar, os usuários que discordam da afirmação de que o sistema facilita a comparação de tarefas similares, podem ter creditado o ônus por não terem entendido o recurso. E por findar, cabe salientar, os usuários que discordam da afirmação de que o sistema auxilia ao tentar evitar o retrabalho no cadastro de novas tarefas de não conformidades, o viés da resposta pode ter ocorrido por não terem entendido o propósito da solução.

Com os resultados obtidos, é possível perceber que as respostas obtidas pela ferramenta e pelo questionário são condizentes, isto é, na visão dos usuários do sistema, a solução proposta atinge seu objetivo geral ao tentar reduzir o retrabalho, haja vista que por mais que a amostragem de dados tenha sido pequena, o número de acertos fora superior aos erros conforme a análise dos dados persistidos, e, comparando com o questionário respondido entre os usuários, a maioria dos usuários compreendeu o sistema, o considerou de fácil usabilidade, com os recursos de identificar tarefas similares e duplicadas relevantes, facilitando o trabalho de comparação em tarefas que possam estar duplicadas ou similares, e

o que é de grande valia para a análise da solução, considerou que o sistema auxilia ao tentar evitar o retrabalho no registro de novas não conformidades. Isso é importante pois com esta visão, os usuários percebem que o sistema é de grande valia como uma tentativa de diminuição de retrabalho.

5. CONCLUSÃO

Não conformidades são decorrentes do processo de desenvolvimento de software. Conforme visto na literatura, não conformidades duplicadas e/ou similares no desenvolvimento de software, ocasionam em problemas, onde é gerado um retrabalho na revisão de não conformidades já vistas, acabando por dobrar o custo de manutenção do projeto.

Como objetivo geral o trabalho analisou "como o *Text Mining* pode reduzir o retrabalho no registro de não conformidades em equipes de desenvolvimento e manutenção de software", como resposta à esta pergunta, foi buscado na literatura os conceitos de custo, retrabalho, desenvolvimento de software e *Text Mining*, onde a pesquisa verificou que, aplicando os conceitos de mineração de texto em um sistema, era possível prevenir o cadastro de tarefas com descrições idênticas ou similares às tarefas já cadastradas.

Para o objetivo específico de "mapear as possíveis causas de retrabalho no cadastro de não conformidades no desenvolvimento de software em empresas", obteve-se como resultado que ao cadastrar uma nova não conformidade, não há uma revisão com o intuito de verificar se a não conformidade reportada já não tenha sido reportada previamente e também para a organização utilizada como exemplo, a falta de um padrão adotado, para reportar não conformidades, facilitando com isso a revisão de não conformidades existentes.

Para o objetivo específico de "identificar como o uso de *Text Mining* tem sido utilizado na literatura", como resultado, revisou-se na literatura, trabalhos que abordassem o *Text Mining* e também quais trabalhos relacionados poderiam ser utilizados para a comparação por este artigo. Para tanto, também se elaborou um quadro comparativo com os trabalhos encontrados.

E por fim, para o objetivo específico de "elaborar uma abordagem utilizando *Text Mining* para reduzir o retrabalho no cadastro de não conformidades no desenvolvimento de software em empresas", foi desenvolvido um sistema que ao cadastrar uma nova tarefa de não conformidade, realiza a mineração do texto e verifica se existem tarefas com a descrição similar ou idêntica à nova tarefa cadastrada. Posto isto, é interessante que o sistema

permaneça sendo evoluído/aperfeiçoado após a conclusão deste trabalho por tratar-se de um sistema com o código livre. Assim, o código fonte do sistema foi disponibilizado em: https://github.com/andercamargo/TextMining.

Como métodos de pesquisa, utilizou-se a pesquisa bibliográfica, documental e questionário com 6 questões abertas e 3 questões fechadas, respondido por 9 usuários para avaliar a solução. A avaliação da solução foi decorrida no período de duas semanas. Além disso, a solução também salvou os dados de respostas dos usuários como erros ou acertos na mineração de texto realizada. Como resultado para o questionário, 8 (88,9%) dos usuários compreenderam o sistema, 9 (100%) dos usuários, o consideraram de fácil uso, e 8 (88,9%) dos usuários consideraram que o sistema auxilia ao tentar evitar o retrabalho no registro de novas não conformidades, o que denota que os usuários do sistema, viram utilidade para a solução proposta. Na análise dos dados salvos pela solução, foram identificadas 8 tarefas como duplicadas ou similares, destas 6 (75%) os usuários creditaram como acerto a identificação, e, portanto, a solução deste trabalho contribuiu para diminuição do retrabalho pois evitou-se a abertura de 6 novas tarefas duplicadas ou similares.

Como limitações, destaca-se a falta de dados e o pouco tempo para avaliar a solução proposta neste artigo, haja vista que a avaliação da solução fora decorrida em apenas duas semanas. Todavia, se a solução pudesse ter sido avaliada anteriormente, o percentual de assertividade das respostas pelos usuários seria mais preciso. Para a academia, o trabalho contribui com o estudo sobre o *Text Mining*, custo e retrabalho em desenvolvimento de software, sendo que o *Text Mining* é um assunto importante podendo ser explorado por áreas com propostas diferentes a este artigo. Também contribui, abordando como que, a utilização de métodos que empregam *Text Mining,* na área de desenvolvimento de software, pode colaborar para a redução de não conformidades em empresas que realizam o desenvolvimento e correções de software, onde se espera como resultado, a diminuição do retrabalho ao realizar a manutenção do software.

Como trabalhos futuros, a correção de palavras incorretas digitadas pelo usuário pode ser aplicada. Esta pesquisa fora iniciada com o intuito de propor correções de palavras que o usuário tenha escrito incorretamente, contudo pela falta de tempo, este recurso não fora desenvolvido na solução testada pelo usuário final. Além disso, a solução poderia gerar um relatório com todas as tarefas que foram evitadas de serem abertas com o status duplicado ou similar. Isto é importante para criar-se um indicador possibilitando realizar a comparação de ganho de tempo ao não ter de realizar a correção mais de uma vez, em uma mesma não conformidade que já tenha sido cadastrada. Também poderia sugerir, utilizando os conceitos

da mineração de texto, o melhor desenvolvedor para realizar a correção de uma não conformidade. E para findar, a solução poderia informar ao usuário, utilizando a mineração de texto, qual o componente de sistema onde são cadastradas mais não conformidades e quais não conformidades são mais comuns, facilitando com isso a tomada de decisão.

THE TEXT MINING APPLIED TO THE REGISTRATION OF NON-CONFORMITIES: A PROPOSAL FOR THE REDUCTION OF RETRIEVAL IN SOFTWARE DEVELOPMENT

Abstract: *In the context of software development errors can be frequent. With each system error found, the bug (non-compliance) is classified, reported and passed on to the responsible party. However, often non-conformities of the same or similar are recurring and are usually re-registered in the bug tracking tool, which results in duplicate registrations and rework (incident classification, reanalysis, support, etc.). In this context, this paper aims to analyze how Text Mining (text mining) can help reduce rework in the register of nonconformities in software development and maintenance teams. For the academy, it contributes a study on Text Mining, cost and rework in software development. It also contributes, such as the use of methods that employ Text Mining, can collaborate to reduce non-conformities in companies that perform development and software patches.*
For the evaluation of this work, a questionnaire was applied and data were analyzed in the database with the results of the users' responses. The questionnaire was answered by professionals who work in the software development area of a systems development company. In the obtained result, it was possible to perceive that the solution reached the objective: it helped in the reduction of the rework in the registry of nonconformities in software development and maintenance teams.

Keywords: Rework in Software Development, Text Mining, Duplicity of Nonconformities, Quality Assurance, Software Development.

REFERÊNCIAS

BARTIÉ, Alexandre. **Garantia da qualidade de software**. 1 ed. São Paulo: ELSEVIER EDITORA, 2002.

BOEHM, B.; BASILI, V. R. Top 10 list [software development]. **Computer**, v. 34, n. 1, p. 135–137, 2001.

CAMPÃO, C. A., et al. **Análise dos custos da qualidade: um estudo de caso em uma empresa alimentícia**. Fonte: **Revista Espacios**. Disponível em: <http://www.revistaespacios.com/a12v33n03/123303261.html>. Acesso em: 26 jun. 2016.

CAMPOS, Fábio. **Qualidade, Qualidade de Software e Garantia da Qualidade de Software são as mesmas coisas?** Disponível em: <http://www.linhadecodigo.com.br/artigo/1712/qualidade-qualidade-de-software-e-garantia-da-qualidade-de-software-sao-as-mesmas-coisas.aspx>. Acesso em: 08 jun. 2016.

CARVALHO, B. V. DE; MELLO, C. H. P. Aplicação do método ágil scrum no desenvolvimento de produtos de software em uma pequena empresa de base tecnológica. **Gestão & Produção**, v. 19, n. 3, p. 557–573, 2012.

CORDEIRO, A. G.; FREITAS, A. L. P. Priorização de requisitos e avaliação da qualidade de software segundo a percepção dos usuários. **Ciencia da Informacao**, v. 40, p. 160–179, 2011.

DEMING, Edwards W. **Qualidade**: a revolução na produtividade. Rio de Janeiro: Marques Saraiva, 1990.

FONSECA, L. C. C.; FARIAS, M. P.; SILVA, R. DE J. DA. Sistema de Recomendação de Links para o fomento de discussões em fóruns de um Ambiente Virtual de Aprendizagem. n. 2012, p. 1–10, 2012.

HAN, J. KAMBER, M. Data Mining: Concepts and Techniques, 2nd ed. Morgan Kaufmann, 2006.

JUNIOR, E. D.; CARVALHO, D. R. EXTRAINDO CONHECIMENTO A PARTIR DE BASE DE GERENCIAMENTO DE FÁBRICA DE SOFTWARE. p. 162–171, 2014.
KUECHLER, W. L. Business applications of unstructured text. **Communications of the ACM**, v. 50, n. 10, p. 86–93, 2007.

KONCHADY, Manu. Text Mining Application Programming. Thomson. 2006.

KUECHLER, W. L. Business applications of unstructured text. Communications of ACM, v.50, n.10, p. 86–93, 2007.

IEEE. **1044-2009 - IEEE Standard Classification for Software Anomalies**. Disponível em: <http://ieeexplore.ieee.org/document/5399061/>. Acesso em: 14 nov. 2016.

LEWIS, William E. "**Software testing and continuous quality improvement**". Florida: Auerbach, 2004.

LOPES, M. C. S. Mineração de Dados Textuais Utilizando Técnicas de Clustering para o Idioma Português. PhD thesis, Universidade Federal do Rio de Janeiro, 2004.

Lucene.NET. **Lucene.Net 3.0.3.** Disponível em:

<http://lucenenet.apache.org/docs/3.0.3/db/d12/_jaro_winkler_distance_8cs_source.html>. Acesso em: 08 nov. 2016.

MAIER, A. et al. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering. **IEEE Data Eng. Bull.**, v. 24, n. 4, p. 44, 2001.

MAINART, D. DE A.; SANTOS, C. M. Desenvolvimento de Software : Processos Ágeis ou Tradicionais ? Uma visão crítica. **Enacomp2010**, 2010.

Microsoft TechNet. **Como os resultados de consultas de pesquisa são classificados (Pesquisa de Texto Completo).** Disponível em: <https://technet.microsoft.com/pt-br/library/ms142524(v=sql.105).aspx>. Acesso em: 07 nov. 2016.

MELLO, C. H. P.; SILVA, C. E. S.; TURRIONI, J. B. & SOUZA, L. G. M. ISO 9001:2000 Sistema de Gestão da Qualidade para Operações de Produção e Serviços. São Paulo: Editora Atlas S. A., 2002.
MYERS GJ. The Art of Software Testing. Wiley Interscience, 1979.

NETTO, F.; BARROS, M. O. M.; ALVIM, A. C. F. F. **An Automated Approach for Scheduling Bug Fix Tasks**. 2010 Brazilian Symposium on Software Engineering. **Anais**...IEEE, set. 2010Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5631516>

NETTO, F. DE C. Um Método Automático para Geração de Cronogramas de Tarefas de Correção de Bugs. p. 90, 2010.

OLIMPIO, J. Módulo de Registro e Controle de Não Conformidades Segundo a Norma Iso 9001 : 2008, 2011.

PRESSMAN, R. S. Software Engineering: A Pracctiitioner's Approach. In: PRESSMAN R. S. Engenharia de Sofware (R. D. Penteado, Trad.) 6 ed, São Paulo: Mcgraw-Hill, 2006.

REZENDE, S. O.; MARCACINI, R. M.; MOURA, M. F. O uso da Mineração de Textos para Extração e Organização Não Supervisionada de Conhecimento. **Revista de Sistemas de Informação da FSMA**, v. 7, p. 7–21, 2011.

ROCHA, Pericles, **Consultas Full-Text em Português no SQL Server 2005.** Fonte: **SQL Server: um Endpoint Brasileiro.** Disponível em: <https://blogs.msdn.microsoft.com/procha/2008/05/15/consultas-full-text-em-portugus-no-sql-server-2005/>. Acesso em: 06 nov. 2016.

RODRIGUES, N. N. Praticando Qualidade de Software: Ensinando e Aprendendo seus Valores através de Ambiente Real. n. 2011, p. 475–486, 2011.

ROUILLER, A. C. et al. ProQualiti: Núcleo de Estudos em Engenharia e Qualidade de Software. v. 2, 2006.

SANTOS, R. E. S. et al. Técnicas de processamento de linguagem natural aplicadas ao processo de mineração de textos: resultados preliminares de um mapeamento sistemático. p. 116–125, 2014.

SCHIFFAUEROVA, A. THOMSON, V. A review of research on cost of quality models and best pratices. Department of Mechanical Engineering, McGill University, Montreal, Canada. International Journal of Quality & Reliability Management, Vol. 23 Iss 6 pp. 647 – 669, 2003.

SILVA, D. E. DOS S.; SOUZA, I. T. DE; CAMARGO, T. Metodologias Ágeis Para O Desenvolvimento De Software: Aplicação Eo Uso Da Metodologia Scrum Em Contraste Ao Modelo. **Revista Computação Aplicada**, 2013.

SOFTEX. MPS.BR: Guia Geral MPS de Software. Associação para promoção da excelência do software brasileiro – SOFTEX. [S.l.], 2012.

SOMMERVILLE, Ian. (2007). **Engenharia de Software**. São Paulo: Pearson Addison-Wesley.

TIAN, Y.; SUN, C.; LO, D. Improved duplicate bug report identification. **Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR**, p. 385–390, 2012.

WEBBER, C. G.; CRISTOFOLI, L. G.; LIMA, M. DE F. W. DO P. UnderMine Text Miner – Uma Ferramenta de Mineração de Texto para Área Educacional. **CINTED-UFRGS**, p. 10, 2013.

WINTER, R. **Practical Text Mining with SQL using Relational Databases**. Fonte: **SlideShare**. Disponível em: <http://pt.slideshare.net/RalphWinters/ralph-winters-text-analytics-sql-relational-database>. Acesso em: 06 nov. 2016.

WIVES, L. K. Tecnologias de descoberta de conhecimento em textos aplicadas à inteligência competitiva. **Universidade Federal do Rio Grande do Sul**, 2002.

WU, L. et al. BUGMINER : Software Reliability Analysis Via Data Mining of Bug Reports. **Seke**, p. 95–100, 2011.

APÊNDICE A – Quadro de afirmações de escopo fechado utilizadas para a Avaliação da Solução Proposta

Nº	**Afirmação**	**Alternativas**
1	O sistema de *Text Mining* é de fácil compreensão.	a. Concordo Plenamente b. Concordo c. Discordo d. Discordo Plenamente
2	O sistema é de fácil utilização.	a. Concordo Plenamente b. Concordo c. Discordo d. Discordo Plenamente
3	O recurso de Identificar tarefas duplicadas é relevante para auxiliar na identificação de tarefas duplicadas.	a. Concordo Plenamente b. Concordo c. Discordo d. Discordo Plenamente
4	O recurso de Identificar tarefas similares é relevante para auxiliar na identificação de tarefas similares.	a. Concordo Plenamente b. Concordo c. Discordo d. Discordo Plenamente
5	O sistema facilita o trabalho de comparação de tarefas similares ou duplicadas.	a. Concordo Plenamente b. Concordo c. Discordo d. Discordo Plenamente
6	A solução desenvolvida auxilia ao tentar evitar o retrabalho no cadastro de novas tarefas de não conformidades.	a. Concordo Plenamente b. Concordo c. Discordo d. Discordo Plenamente

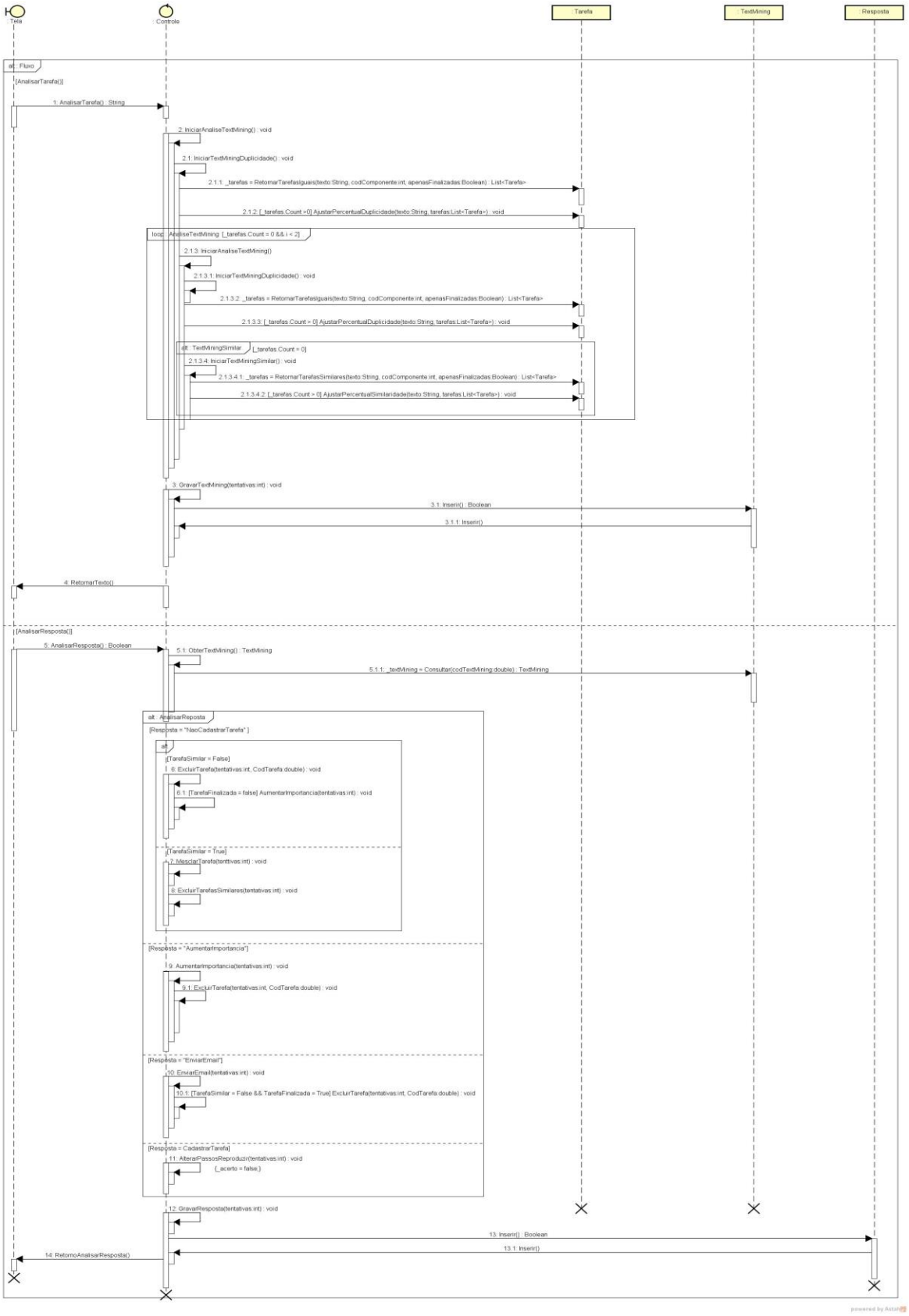
Fonte: Elaborado pelo autor.

APÊNDICE B – Fórmula de classificação utilizada pela função *FREETEXTTABLE*

Rank = **Σ [Termos na Consulta] w (((k1 + 1) tf) / (K + tf)) * ((k3 + 1) qtf / (k3 + qtf)))**
Onde:
w é o peso Robertson-Sparck Jones.
Na forma simplificada, w é definida como:
w = log10 (((r + 0.5) * (N – R + r + 0.5)) / ((R – r + 0.5) * (n – r + 0.5))
N é o número de linhas indexadas pela propriedade que está sendo consultada.
n é o número de linhas contendo a palavra.
K is (k1 * ((1 – b) + (b * dl / avdl))).
dl é o comprimento da propriedade, na ocorrência de palavras.
avdl é a média do comprimento da propriedade que está sendo consultada, na ocorrência de palavras.
k1, b, e k3 são constantes 1.2, 0.75, e 8.0, respectivamente.
tf é a frequência da palavra na propriedade consultada em uma linha específica.
qtf é a frequência do termo na consulta.

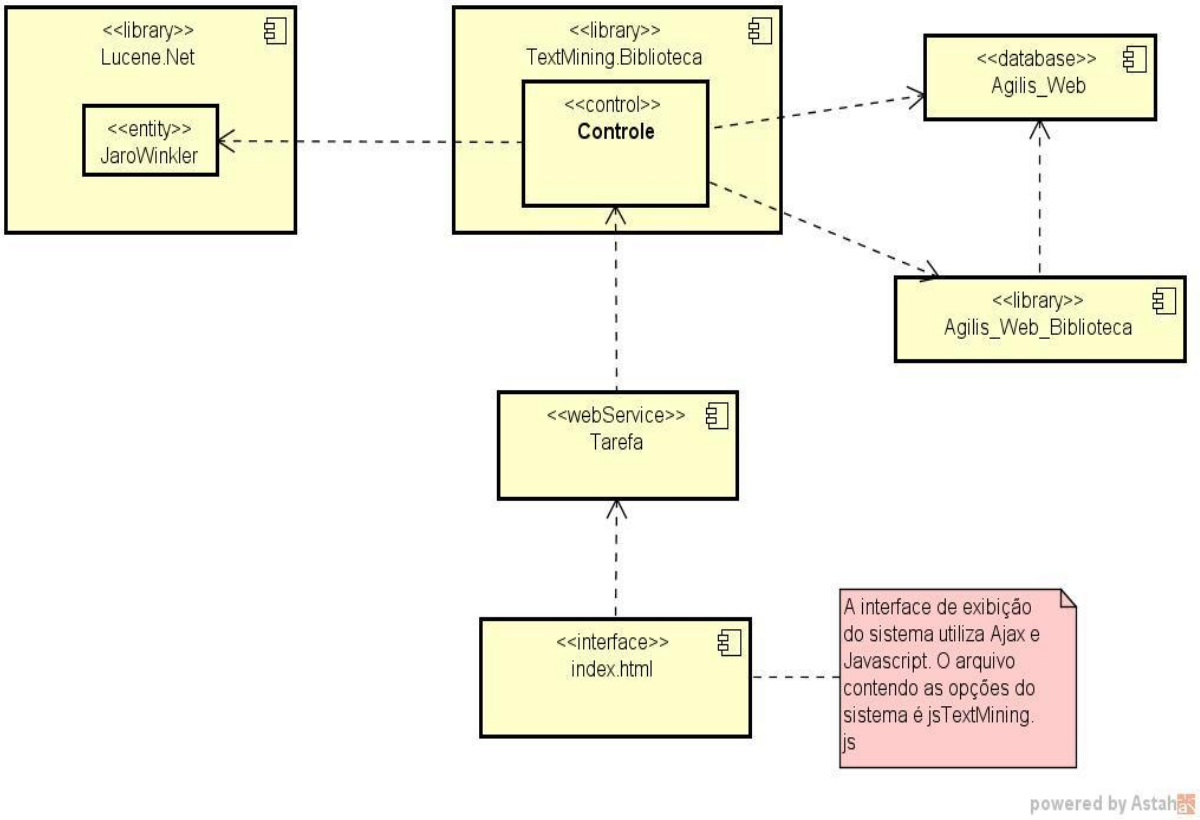
Fonte: Microsoft TECHNET (2016).

APÊNDICE C – Diagrama de sequência da Solução



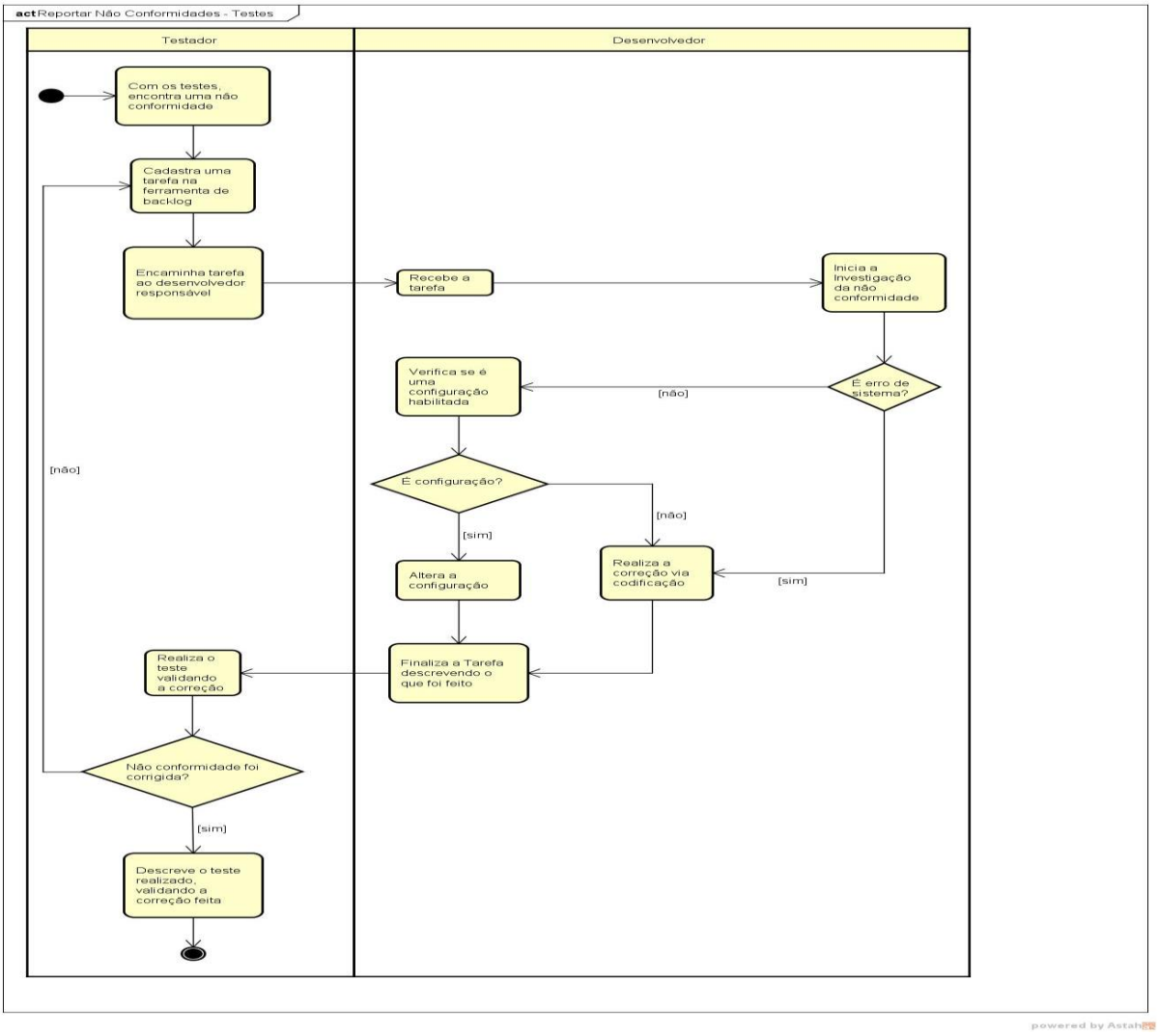
Fonte: Elaborado pelo autor.

APÊNDICE D – Diagrama de Componentes da Solução



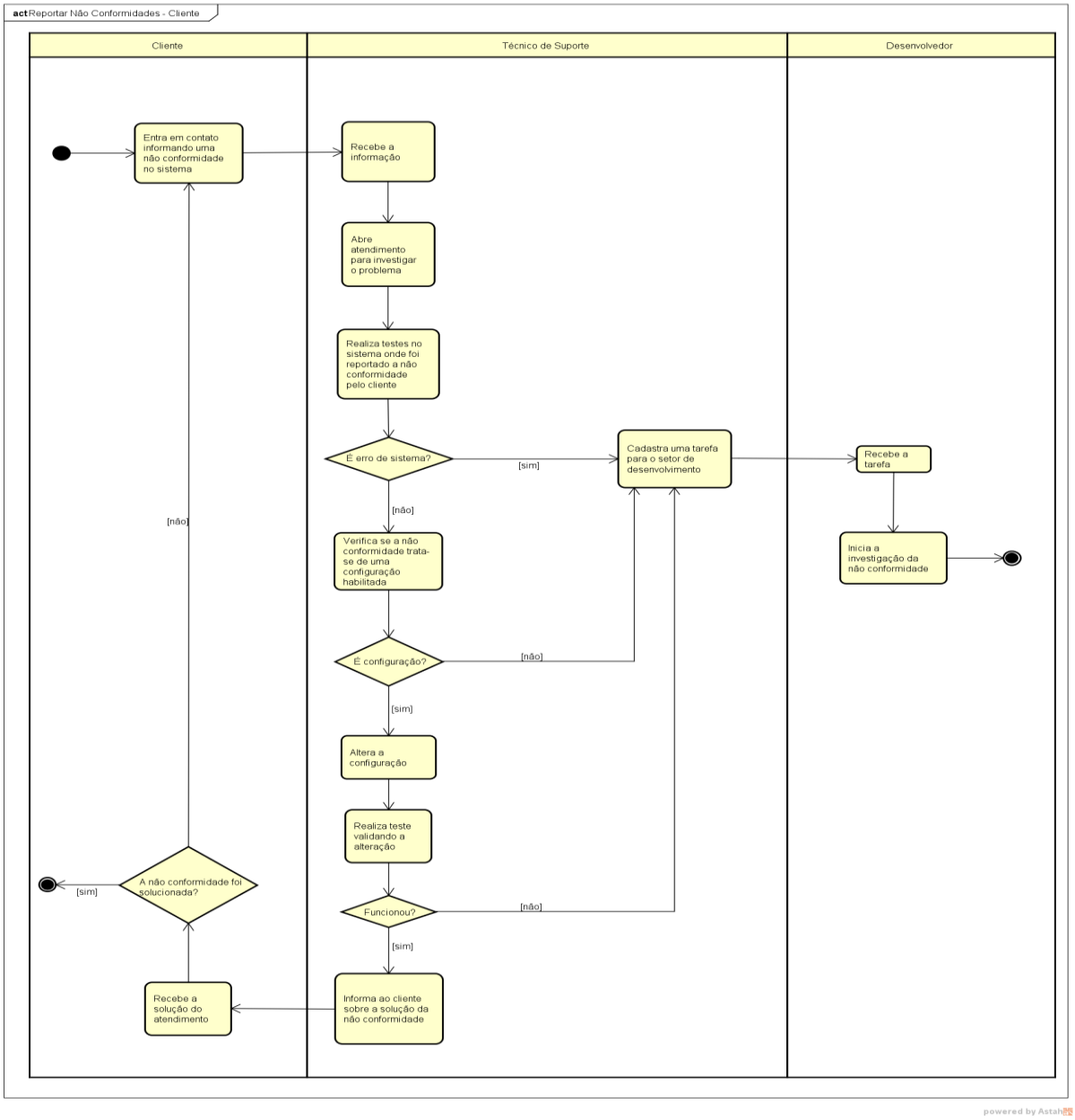
Fonte: Elaborado pelo autor.

APÊNDICE E – Diagrama de atividades do fluxo de reportar não conformidade – Testes



Fonte: Elaborado pelo autor.

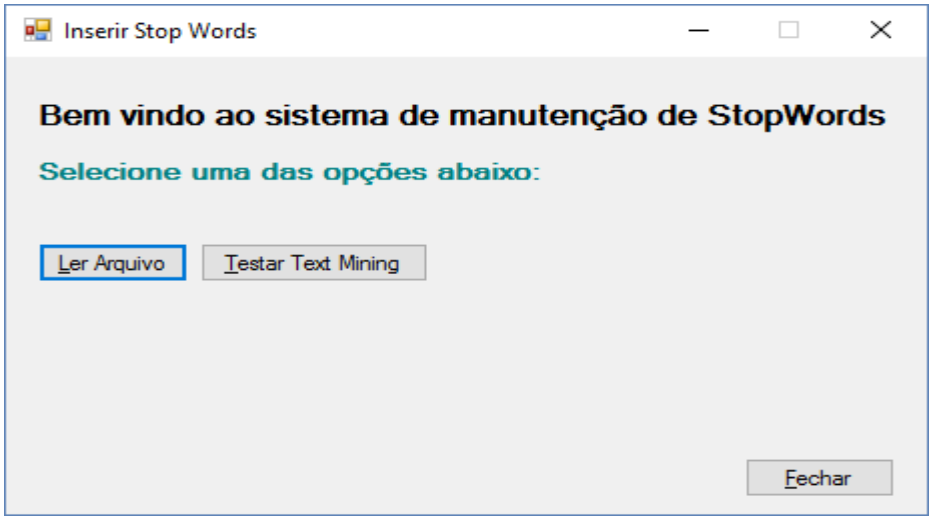
APÊNDICE F – Diagrama de atividades do fluxo de reportar não conformidade reportada – Cliente



Fonte: Elaborado pelo autor.

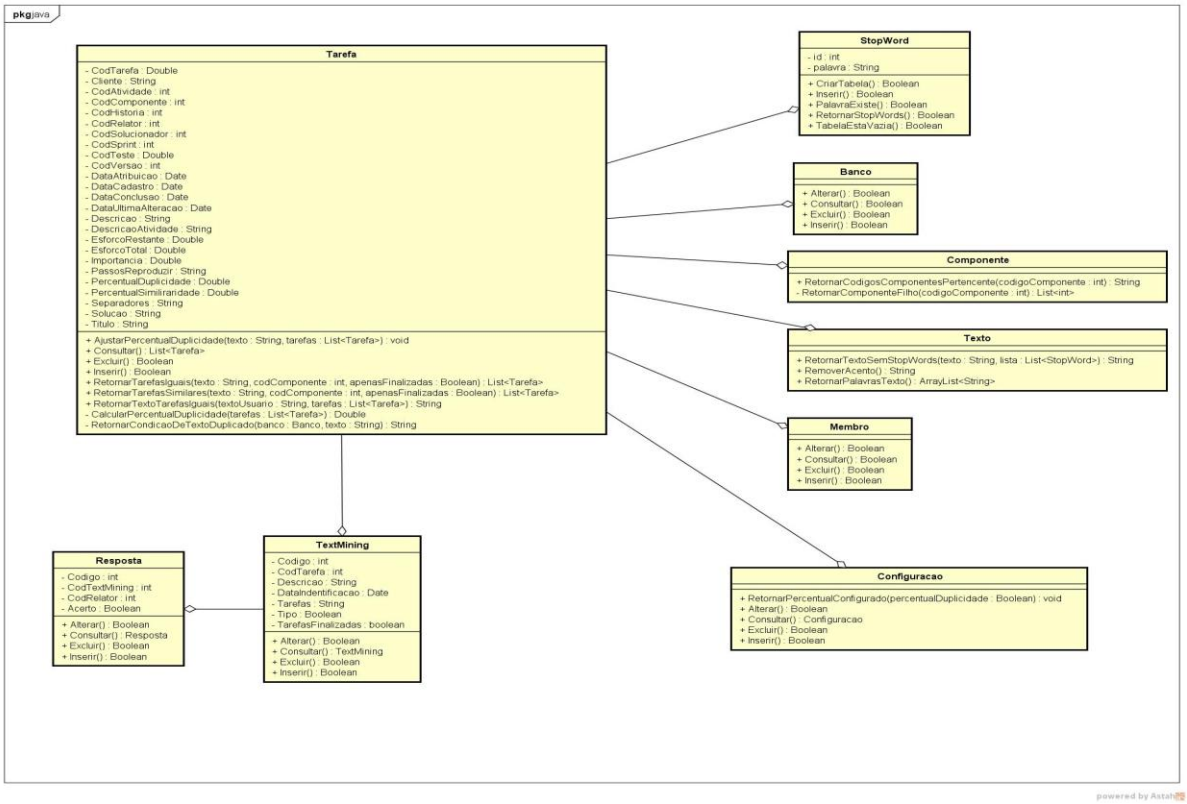
APÊNDICE G – Ferramenta para Manutenção de *Stopwords*, desenvolvida.

Para a Manutenção de *Stopwords*, fora desenvolvido também uma ferramenta que realiza a inserção de *Stopwords* com base na leitura de um arquivo texto, onde caso a palavra ainda não exista na tabela *Stopword* ela é inserida pela solução.



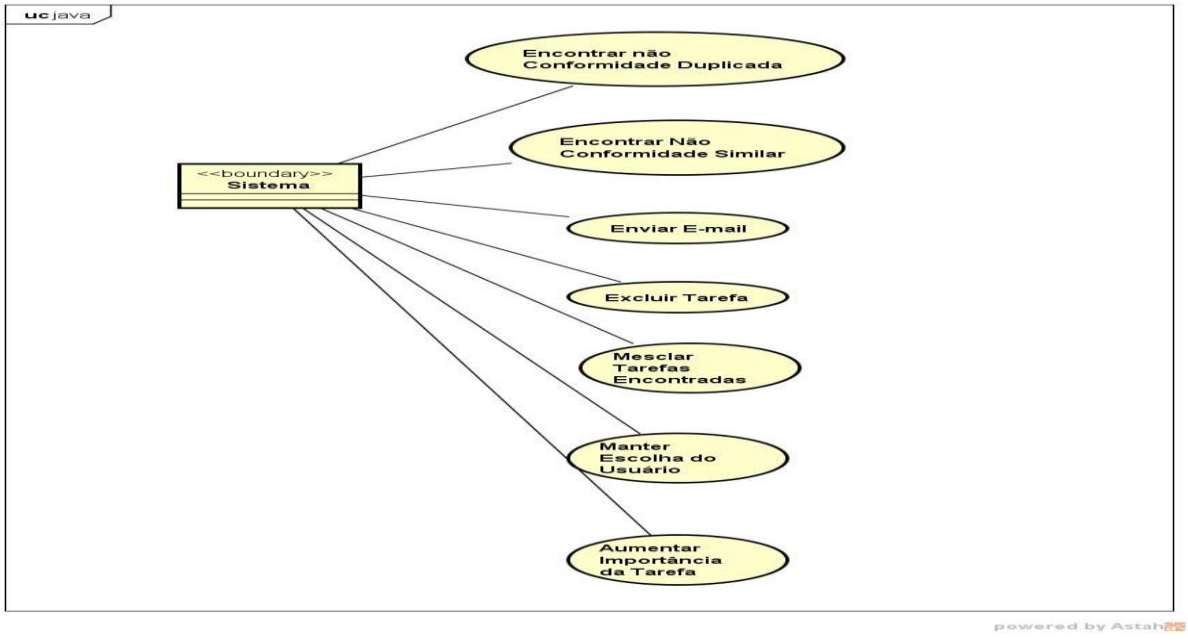
Fonte: Elaborado pelo autor.

APÊNDICE H – Diagrama de Classes da Solução de *Text Mining*



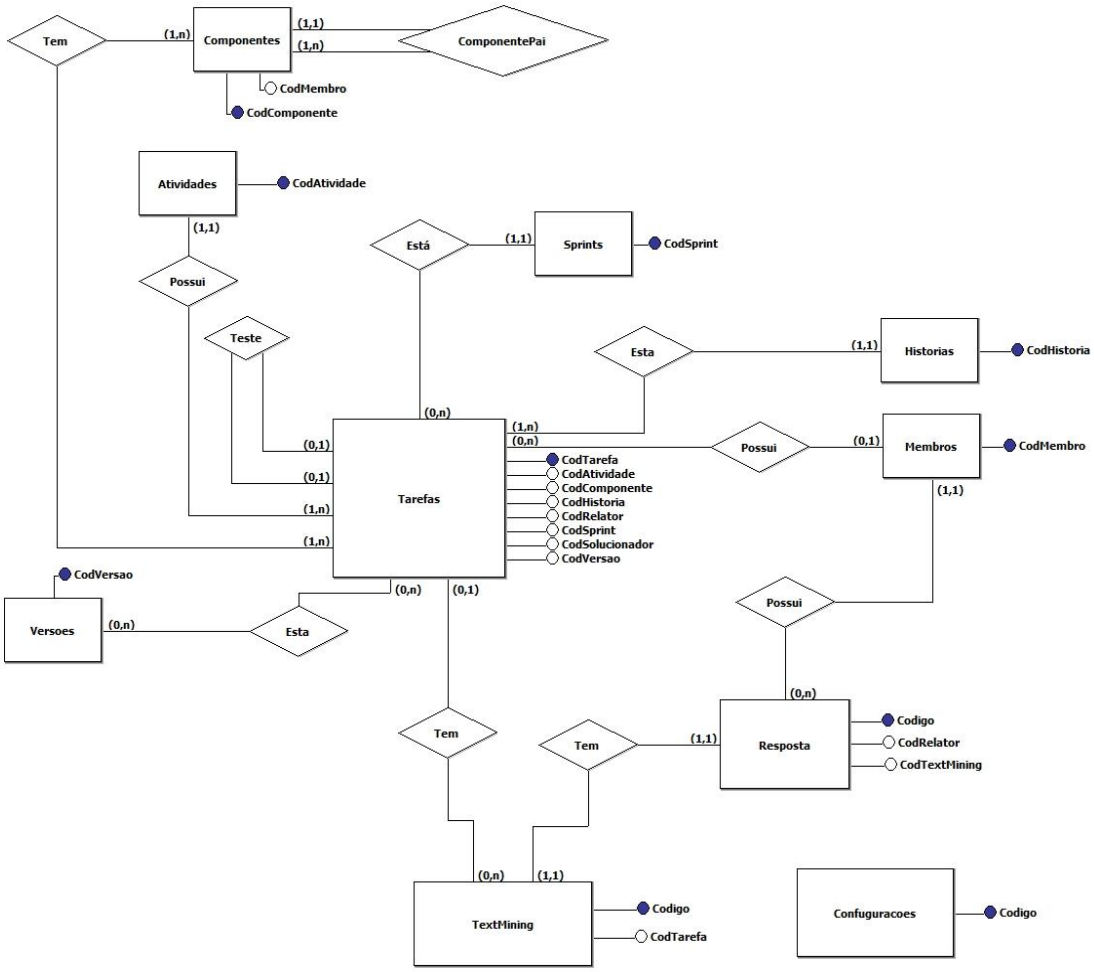
Fonte: Elaborado pelo autor.

APÊNDICE I – Diagrama de Casos de Uso da Solução de *Text Mining*



Fonte: Elaborado pelo autor.

APÊNDICE J – Diagrama E.R. da Solução de *Text Mining*



Fonte: Elaborado pelo autor.

ANEXO A – Termo de Confidencialidade para a Coleta de Informações de Empresa/Instituição

UNISINOS

UNIVERSIDADE DO VALE DO RIO DOS SINOS
Unidade Acadêmica de Graduação

TERMO DE CONFIDENCIALIDADE PARA COLETA DE INFORMAÇÕES DE EMPRESA/INSTITUIÇÃO.

Eu, Anderson Selestino Camargo, aluno(a) do **Curso de** Análise e Desenvolvimento de Sistemas da Universidade do Vale do Rio dos Sinos - Unisinos, matriculado(a) sob o número 1703931, **declaro que a Empresa/Instituição** A TELECON SISTEMAS DE AUTOMAÇÃO LTDA **objeto de estudo do Trabalho de Conclusão de Curso intitulado** A MINERAÇÃO DE TEXTO APLICADA AO REGISTRO DE NÃO CONFORMIDADES: UMA PROPOSTA PARA A REDUÇÃO DE RETRABALHO NO DESENVOLVIMENTO DE SOFTWARE entregue no semestre 2016/2, **permitiu a pesquisa e o uso de todos os dados que nele constam.**

Declaro, ainda, que as informações apresentadas são verdadeiras e correspondem à realidade da Empresa/Instituição estudada.

(X) A Empresa/Instituição autorizou a divulgação do seu nome fantasia/razão social.

( ) **A Empresa/Instituição não autorizou a divulgação do seu nome fantasia/razão social. Nesse caso, responsabilizo-me em preservar o nome da Empresa/Instituição de forma a que ela não seja passível de identificação no meu Trabalho.**

São Leopoldo, 21 de setembro de 2016.

Anderson S. Camargo
Assinatura do aluno

93.702.538/0001-29
A TELECON SISTEMAS DE AUTOMAÇÃO COMERCIAL LTDA
Rua Cristóvão Colombo, 121
B. Vila Fernandes - Cep 92110-450
CANOAS - RS

Ciência da empresa

CÁSSIO PENES DA ROCHA
Nome do responsável da Empresa/Instituição

Assinatura do Responsável da Empresa/Instituição
Carimbo ou CNPJ

Av. Unisinos, 950 Caixa Postal 275 CEP 93022-000 São Leopoldo Rio Grande do Sul Brasil
Fone: (51) 3591-1122 http://www.unisinos.br

Fonte: Unisinos (2016).