UM ESTUDO DE FERRAMENTAS DE GERENCIAMENTO DE REQUISIÇÃO DE MUDANÇA

VAGNER CLEMENTINO

UM ESTUDO DE FERRAMENTAS DE GERENCIAMENTO DE REQUISIÇÃO DE MUDANÇA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR: RODOLFO F. RESENDE

Belo Horizonte

Dezembro de 2016

Clementino, Vagner

Um Estudo de Ferramentas de Gerenciamento de Requisição de Mudança / Vagner Clementino. — Belo Horizonte, 2016
xxv, 48 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientador: Rodolfo F. Resende

1. Computação — Teses. 2. Redes — Teses. I. Orientador. II. Título.

[Folha de Aprovação]

Quando a secretaria do Curso fornecer esta folha,
ela deve ser digitalizada e armazenada no disco em formato gráfico.

Se você estiver usando o `pdflatex`,
armazene o arquivo preferencialmente em formato PNG
(o formato JPEG é pior neste caso).

Se você estiver usando o `latex` (não o `pdflatex`),
terá que converter o arquivo gráfico para o formato EPS.

Em seguida, acrescente a opção `approval={`*nome do arquivo*`}`
ao comando `\ppgccufmg`.

Se a imagem da folha de aprovação precisar ser ajustada, use:
`approval=[`*ajuste*`][`*escala*`]{`*nome do arquivo*`}`
onde *ajuste* é uma distância para deslocar a imagem para baixo
e *escala* é um fator de escala para a imagem. Por exemplo:
`approval=[-2cm][0.9]{`*nome do arquivo*`}`
desloca a imagem 2cm para cima e a escala em 90%.

Dedicuum cest laborae a quelquis personatum que ajudorat a facirelo.

Agradecimentos

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias excepturi sint occaecati cupiditate non provident, similique sunt in culpa qui officia deserunt mollitia animi, id est laborum et dolorum fuga. Et harum quidem rerum facilis est et expedita distinctio. Nam libero tempore, cum soluta nobis est eligendi optio cumque nihil impedit quo minus id quod maxime placeat facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum hic tenetur a sapiente delectus, ut aut reiciendis voluptatibus maiores alias consequatur aut perferendis doloribus asperiores repellat.

"A verdade é o contrário da mentira,
e a mentira é o oposto da verdade."
(Autor desconhecido)

Resumo

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

Palavras-chave: Visão Computacional, Redes, Sabotagens.

Abstract

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

Keywords: Computer Vision, Networks, Sabotage.

Resumo Estendido

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Seção 1

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

Seção 2

At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias excepturi sint occaecati cupiditate non provident, similique sunt in culpa qui officia deserunt mollitia animi, id est laborum et dolorum fuga. Et harum quidem rerum facilis est et expedita distinctio. Nam libero tempore, cum soluta nobis est eligendi optio cumque nihil im-

pedit quo minus id quod maxime placeat facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et aut officiis debitis aut rerum necessitatibus saepe eveniet ut et voluptates repudiandae sint et molestiae non recusandae. Itaque earum rerum hic tenetur a sapiente delectus, ut aut reiciendis voluptatibus maiores alias consequatur aut perferendis doloribus asperiores repellat.

Lista de Figuras

1.1 Tipos de manutenção segundo a norma ISO/IEC 14764. Extraído de [ISO/IEC, 2006] 2
1.2 Evolução da manutenção de software como percentual do custo total. Extraído de [Engelbertink & Vogt, 2010] 4

3.1 Número de artigos incluídos durante o processo de seleção dos estudos. Baseado em [Petersen et al., 2015] 18
3.2 Esquema de classificação proposto 19
3.3 Taxonomia vertical para Modelos de Recuperação da Informação. Adaptado de [Cerulo & Canfora, 2004] 22
3.4 Total de artigos por categoria de problema 24

Lista de Tabelas

1.1 Exemplos de ferramentas e serviços da Internet. Adaptado de [Cavalcanti et al., 2014] 3

2.1 Categorias da Requisição de Mudanças. Adaptado de SWEBOK [?] 12

3.1 Número de Estudos Recuperados por Base de Dados 17
3.2 Taxonomia por Problemas na Manutenção de Software 23

4.1 Ferramentas e serviços da Internet selecionados. 27

A.1 Sentença de Busca por Base de Dados 48

Sumário

Agradecimentos **ix**

Resumo **xiii**

Abstract **xv**

Resumo Estendido **xvii**

Lista de Figuras **xix**

Lista de Tabelas **xxi**

1 Introdução **1**

1.1 Justificativa 3
1.2 Motivação 5
1.3 Problema 5
1.4 Visão Geral da Proposta 5
1.5 Metodologia de Pesquisa 5
1.6 Contribuições da Dissertação 6
1.7 Organização da Dissertação 6

2 Manutenção de Software: Uma Visão Geral **7**

2.1 Conceitos Fundamentais 11
2.2 Requisição de Mudança 12
2.3 O processo de Manutenção de Software 12
2.3.1 Manutenção de Software Tradicional 13
2.3.2 Manutenção de Software com Método dos Agilistas 13
2.4 Ferramentas de Gerenciamento de Requisições de Mudança (FGRM) . 13
2.4.1 Características Gerais 13
2.4.2 Extensões para FGRM 13

3 Mapeamento Sistemático da Literatura **15**
3.1 Introdução . . . 15
3.2 Metodologia de Pesquisa . . . 15
3.2.1 Questões de Pesquisa . . . 16
3.2.2 Pesquisa da Literatura . . . 16
3.2.3 Esquemas de Classificação . . . 17
3.3 Resultados . . . 23
3.3.1 Extensões para Problemas na Manutenção de Software . . . 23
3.3.2 Extensões com Suporte à Papeis . . . 24
3.3.3 Técnicas de IR Utilizadas pelas Extensões . . . 24
3.3.4 Ferramentais Estendidas . . . 24
3.4 Limitações e Ameaças à Validade . . . 24
3.5 Trabalhos Relacionados . . . 24
3.6 Resumo do Capítulo . . . 24

4 Caracterização das Ferramentas de Gerenciamento de Requisição de Mudança **25**
4.1 Introdução . . . 25
4.2 Objetivo do Capítulo . . . 25
4.3 Metodologia . . . 26
4.3.1 Seleção das Ferramentas . . . 26
4.3.2 Inspeção da Documentação . . . 27
4.3.3 Agrupamento das Funcionalidades . . . 27
4.4 Resultados . . . 28
4.5 Discussão . . . 28
4.6 Ameças à Validade . . . 28
4.7 Resumo do Capítulo . . . 28

5 Pesquisa com Profissionais **29**
5.1 Introdução . . . 29
5.2 Objetivo da Pesquisa com Profissionais . . . 29
5.3 Desenho da Pesquisa com Profissionais . . . 30
5.3.1 Questionário . . . 30
5.3.2 População, Amostra e Respostas . . . 31
5.4 Análise dos Dados . . . 31
5.5 Discussão . . . 32
5.6 Ameças à Validade . . . 32

5.7 Resumo do Capítulo 32

6 Extensões para Ferramentas de Gerenciamento de Requisição de Mudança 33

6.1 Introdução 33

6.2 Ameaças à Validade 33

6.3 Resumo do Capítulo 33

7 Avaliação das Extensões para Ferramentas de Gerenciamento de Requisição de Mudança 35

7.1 Introdução 35

7.2 Resumo do Capítulo 35

8 Conclusão 37

Referências Bibliográficas 39

Apêndice A Instrumentos do Mapeamento Sistemático 47

Capítulo 1

Introdução

Dentro do ciclo de vida de um produto de software o processo de manutenção tem papel fundamental. Devido ao seu alto custo, em alguns casos chegando a 60% do custo final [Kaur & Singh, 2015], as atividades relacionadas a manter e evoluir software tem sua importância considerada tanto pela comunidade científica quanto pela indústria. Neste sentido, este trabalho de dissertação se propõe a investigar e contribuir no entendimento de como as Ferramentas de Gerenciamento de Requisição de Mudança estão sendo melhoradas ou estendidas no contexto da transformação do processo de desenvolvimento e manutenção de software de um modelo tradicional para outro que incorpora cada vez mais as práticas propostas pelos agilistas. O intuito é analisar como as FGRM estão sendo modificadas com base na literatura da área em contraste com o ponto de vista dos profissionais envolvidos em manutenção de software.

A *Manutenção*, dentre outros aspectos, corresponde ao processo de modificar um componente ou sistema de software após a sua entrega com o objetivo de *corrigir falhas, melhorar o desempenho ou adaptá-lo devido à mudanças ambientais* [IEEE, 1990]. De maneira relacionada, *Manutenibilidade* é a propriedade de um sistema ou componente de software em relação ao grau de *facilidade* que ele pode ser corrigido, melhorado ou adaptado [IEEE, 1990].

As manutenções em software podem ser divididas em *Corretiva, Adaptativa, Perfectiva e Preventiva* [Lientz & Swanson, 1980, IEEE, 1990]. A Manutenção Corretiva lida com a reparação de falhas encontradas. A Adaptativa tem o seu foco na adequação do software devido à mudanças ocorridas no ambiente em que ele está inserido. A Perfectiva trabalha para detectar e corrigir falhas latentes antes que elas se manifestem como tal. A Perfectiva fornece melhorias na documentação, desempenho ou manutenibilidade do sistema. A Preventiva se preocupa com atividades que possibilitem aumento da manutenibilidade do sistema. A *ISO 14764* [ISO/IEC, 2006] propõe a divisão da

tarefa de manutenção nos quatro tipos descritos anteriormente e agrupa-os em um termo único denominado *Requisição de Mudança - Modification Request (RM)*, conforme pode ser visto pela Figura 1.1.

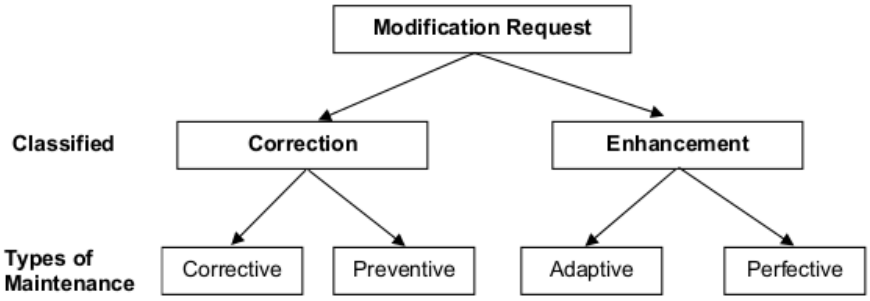

Figura 1.1: Tipos de manutenção segundo a norma ISO/IEC 14764. Extraído de [ISO/IEC, 2006]

Fazer uma discussão melhor sobre as diferenças entre manutenção e evolução de Software

Por conta do volume das Requisições de Mudança se faz necessária a utilização de ferramentas com o objetivo de gerenciá-las. Esse controle é geralmente realizado por Sistemas de Controle de Demandas (SCD)- Issue Tracking Systems, que auxiliam os desenvolvedores na correção de forma individual ou colaborativa de defeitos (bugs), no desenvolvimento de novas funcionalidades, dentre outras tarefas relativas à manutenção de software. Não existe na literatura uma nomenclatura comum para este tipo de ferramenta. Em alguns estudos é possível verificar nomes tais como Sistema de Controle de Defeito - Bug Tracking Systems, Sistema de Gerenciamento da Requisição - Request Management System, Sistemas de Controle de Demandas (SCD)- Issue Tracking Systems e diversos nomes afins. Todavia, de modo geral, o termo se refere as ferramentas utilizadas pelas organizações para *gerir as Requisições de Mudança*. Estas ferramentas podem ainda ser utilizadas por gestores, analistas de qualidade e usuários finais para atividades tais como gerenciamento de projetos, comunicação, discussão e revisões de código. Neste trabalho utilizaremos o termo `Ferramentas de Gerenciamento de Requisições de Mudança` (FGRM) ao referimos a este tipo de ferramenta. A Tabela 1.1 apresenta alguns exemplos de ferramentas que podem ser classificadas como FGRM. Também são listados serviços da Internet que oferecem funcionalidades presentes nas FGRM na forma de Software como Serviço.

Grande parte da literatura de manutenção de software trata de técnicas e metodologias tradicionais da Engenharia de Software. Não obstante, é possível verificar um protagonismo das práticas propostas pelos agilistas em projetos de sucesso, mesmo

Ferramentas		Serviços da Internet	
Bugzilla	https://www.bugzilla.org/	SourceForge	https://sourceforge.net/
MantisBT	https://www.mantisbt.org/	Lauchpad	https://launchpad.net/
Trac	https://trac.edgewall.org/	Code Plex	https://www.codeplex.com/
Redmine	www.redmine.org/	Google Code	https://code.google.com/
Jira	https://www.atlassian.com/software/jira	GitHub	https://github.com/

Tabela 1.1: Exemplos de ferramentas e serviços da Internet. Adaptado de [Cavalcanti et al., 2014]

em áreas não relativas à Tecnologia da Informação [Serrador & Pinto, 2015]. Neste contexto, verifica-se uma tendência que os departamentos dedicados à manutenção de software se mostrem interessados nas metodologias dos agilistas e que tenham vontade de experimentá-las em suas atividades [Heeager & Rose, 2015]. Apesar da maioria dos textos em Engenharia de Software tratarem desenvolvimento e manutenção como atividades com natureza distintas, esta última pode adaptar características da primeira visando a melhoria do seu desempenho. Dentre as práticas propostas pelos agilistas passíveis de serem utilizadas em tarefas de manutenção é possível citar o desenvolvimento iterativo, maior envolvimento do cliente, comunicação face a face, testes frequentes, dentre outras.

Da mesma forma que ocorre no desenvolvimento de software, é possível verificar uma crescente adoção de técnicas da metodologia ágil na manutenção de software [Soltan & Mostafa, 2016, Devulapally, 2015, Heeager & Rose, 2015]. Neste contexto, é natural que ferramentas que dão suporte à manutenção, tal como as FGRM's, tenham que evoluir para se adaptar a esta nova forma de trabalhar. Mesmo em um ambiente tradicional de desenvolvimento e manutenção de software, verifica-se a necessidade de adequação das FGRM's. Uma das justificativas desta exigência se deve ao fato que a maioria desses sistemas são projetados em torno do termo "demanda"(bug, defeito, bilhete, recurso, etc.), contudo, cada vez mais este modelo parece estar distante das necessidades práticas dos projetos de software, resultando, por exemplo, que os desenvolvedores tenham um baixo entendimento da situação geral bem como das atividades das outras pessoas envolvidas no projeto [Baysal et al., 2013]

Apesar da inegável importância das FGRM's, percebe-se um aparente desacoplamento deste tipo de ferramenta com as necessidades das diversas partes interessadas (stakeholders) na manutenção e evolução de software. Um sinal deste distanciamento pode ser observado pelas diversas extensões (plugins) propostas na literatura [Rocha et al., 2015, Thung et al., 2014b, Kononenko et al., 2014].

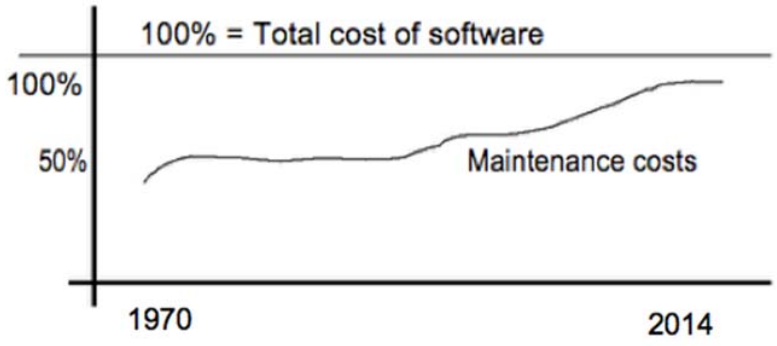

Figura 1.2: Evolução da manutenção de software como percentual do custo total. Extraído de [Engelbertink & Vogt, 2010]

1.1 Justificativa

Desde o final da década de 1970 [Zelkowitz et al., 1979] percebe-se o aumento do custo referente as atividades de manutenção de software. Nas décadas de 1980 e 1990 alguns trabalhos tiveram seu foco no desenvolvimento de modelos de mensuração do custo para manter o software [Herrin, 1985, Hirota et al., 1994]. Apesar da evolução das metologias de manutenção a estimativa é que nas últimas duas décadas o custo de manutenção tenha aumentado em 50% [Koskinen, 2010]. Esta tendência pode ser observada na Figura 1.2 no qual é possível verificar a evolução do custo da manutenção de software como fração do custo total do produto.

Diante da maior presença de software em todos os setores da sociedade existe um interesse por parte da academia e da industria no desenvolvimento de processos, técnicas e *ferramentas* que reduzam o esforço e o custo das tarefas de desenvolvimento e manutenção de software. Neste linha, o trabalho de Yong & Mookerjee [Tan & Mookerjee, 2005] propõe um modelo que reduz os custos de manutenção e reposição durante a vida útil de um sistema de software. O modelo demonstrou que em algumas situações é *melhor substituir um sistema do que mantê-lo*. Este problema é agravado tendo em vista que o custo de manutenção pode chegar a 60% do custo total do software [Kaur & Singh, 2015]. Este percentual reflete a fração de desenvolvedores dedicados à tarefas de manutenção de sistemas [Zhang, 2003].

A manutenção não necessariamente exige que o processo de software envolvido seja o tradicional. Percebe-se alguns exemplos de adoção das práticas ágeis para fins de manutenção e evolução do software [Kajko-Mattsson & Nyfjord, 2009a, Heeager & Rose, 2015, Devulapally, 2015, Naz et al., 2016]. Tal tendência não é surpreendente tendo em vista que os métodos "ágeis" enfatizam características úteis à

eficiência da implementação de software, tais como desenvolvimento incremental e teste contínuo que agregam valor para a evolução e manutenção eficaz de um sistema [Thomas, 2006]. Dentro desta tendência verifica-se a necessidade de que as ferramentas envolvidas no suporte à manutenção de software se adequem à este nova forma de manter software.

O desenvolvimento e a manutenção de software envolvem diversos tipos de métodos, técnicas e ferramentas. Em especial no processo de manutenção, um importante aspecto são as diversas Requisições de Mudanças que devem ser gerenciadas. Este controle é realizado pelas Ferramentas de Gerenciamento de Requisição de Mudanças (FGRM) cujo o uso vem crescendo em importância, sobretudo, por sua utilização por gestores, analistas da qualidade e usuários finais para atividades como tomada de decisão e comunicação.

A utilização de *"demanda"* como conceito central para Ferramentas de Gerenciamento de Requisição de Mudanças (FGRM) parece ser distante das necessidades práticas dos projetos de software, especialmente no ponto de vista dos desenvolvedores [Baysal et al., 2013]. Um exemplo deste desacoplamento das FGRM com a necessidade de seus usuários pode ser visto no trabalho proposto por Baysal & Holme [Baysal & Holmes, 2012] no qual desenvolvedores que utilizam o Bugzilla[1] relatam a dificuldade em manter uma compreensão global das RM's em que eles estão envolvidos. Segundo os desenvolvedores seria interessante que a ferramenta tivesse um suporte melhorado para a Consciência Situacional - Situational Awareness. Em síntese, eles gostariam de estar cientes da situação global do projeto bem como das atividades que outras pessoas estão realizando. Um outro sinal da necessidade de evolução deste tipo de ferramenta pode ser observado considerando as diversas extensões (plugins) propostas na literatura [Rocha et al., 2015, Thung et al., 2014b, Kononenko et al., 2014].

Neste contexto, é proposto neste projeto de dissertação a elaboração de um estudo das Ferramentas de Gerenciamento de Requisição de Mudança (FGRM) como o objetivo de *(i)* entender os requisitos comuns deste tipo de ferramenta; *(ii)* mapear as extensões para as FGRM que estão sendo propostas na literatura; *(iii)* avaliar sobre o ponto de vista dos profissionais a situação atual dos FGRM; *(iv)* propor novas extensões para as FGRM. Vamos discutir os aspectos que são considerados mais importantes a partir da literatura da área bem como do ponto de vista de profissionais envolvidos em manutenção de software. De forma particular, iremos estudar os mecanismos de personalização que algumas destas ferramentas permitem e tentaremos ainda criar exemplos de personalização para alguma possível extensão a ser identificada ao longo do

[1] `https://www.bugzilla.org`

trabalho.

1.2 Motivação

1.3 Problema

1.4 Visão Geral da Proposta

1.5 Metodologia de Pesquisa

O trabalho de dissertação pode ser dividido nas etapas listadas a seguir:

(i) Mapeamento Sistemático da Literatura [Keele, 2007]

(i) Caracterização das Ferramentas de Gerenciamento de Requisição de Mudança (FGRM)

(i) Pesquisa (Survey) com os desenvolvedores [Wohlin et al., 2012]

(i) Desenvolvimento de extensões para as FGRM's

1.6 Contribuições da Dissertação

1.7 Organização da Dissertação

Capítulo 2

Manutenção de Software: Uma Visão Geral

Uma tendência natural do software é evoluir a fim de atender aos novos requisitos e alterações no ambiente no qual ele está inserido. Em uma série de estudos Lehman propõe um conjunto de leis sobre a evolução do software. Dentre elas podemos destacar as leis da Mudança Contínua (Continuing Change) e da Complexidade Crescente (Increasing complexity). Segundo a lei da Mudança Contínua um programa que é utilizado em um ambiente real deve mudar ou se tornará progressivamente menos útil [Lehman, 1980]. A lei da Complexidade Crescente (Increasing complexity) afirma que quando um sistema em evolução muda, sua estrutura tende a se tornar mais complexa. Nesta situação, recursos extras devem ser disponibilizados a fim de preservar e simplificar a estrutura do software [Lehman, 1980]. As leis de Lehman tem sido validadas, especialmente aquelas relacionadas a tamanho e complexidade do software. Em um trabalho recente Yu & Mishra [Yu & Mishra, 2013] examinaram de forma empírica as Leis de Lehman em relação a evolução da qualidade do software. Os resultados dão suporte as Leis especialmente a que versa sobre a qualidade, na qual um produto de software decresce a sua aquele atributo ao longo do tempo, exceto que ele seja reestruturado.

Percebida a importância do processo de manutenção de software, alguns trabalhos foram propostos visando mensurar o seu custo bem como propor processos com o objetivo de reduzir o esforço envolvido neste tipo de atividade.

No trabalho de Herrin [Herrin, 1985] foi proposto um modelo matemático com o objetivo de avaliar o impacto financeiro no orçamento de uma universidade devido às atividades de manutenção no sistema de processamento de dados da instituição. O modelo propõe que o valor disponível para desenvolvimento de um novo sistema é função inversa do custo de manutenção do software existente. Desta forma, o fato de se

manter um sistema durante muito tempo poderá impossibilitar a aquisição ou mesmo o desenvolvimento de um novo.

No estudo de Hirota et al. [Hirota et al., 1994] é proposta a utilização da técnica Análise de Ripple para estimar o custo da manutenção de software. O termo "efeito Ripple" foi utilizado pela primeira vez em um artigo publicado por Haney [Haney, 1972] para descrever a forma que a mudança em um módulo poderia causar alterações em outras partes do sistema [Bilal & Black, 2005]. A Análise Ripple é, portanto, uma técnica para analisar o fluxo de dados de variáveis dentro de um determinado programa. Os valores retornados pela aplicação do método são denominados Complexidade de Ripple. Os resultados demostraram que a Complexidade de Ripple está mais relacionada ao entendimento do software do que as métricas padrão, como linhas de código, complexidade ciclomática e pontos de função. Desta forma, a Complexidade de Ripple poderia ser utilizada, por exemplo, para predizer os custo de manutenção de um sistema, bem como a necessidade de substituição do mesmo.

Mediante o uso de Redes Neurais Shula & Misra [Shukla & Misra, 2008] propõe um estudo para medir o custo de manutenção de software. O trabalho discute a utilização de outras métricas além de linha de código e pontos de função para medir tamanho e custo do processo de manutenção. Os resultados demonstraram a possibilidade de construir um modelo para medir o custo utilizando Redes Neurais. Contudo, os resultados são sensíveis a escolha da arquitetura e parâmetros de treino, os quais idealmente deveriam ser preparados por um especialista no sistema (oráculo).

A dinamicidade do ambiente de negócios tem levado a diversas organizações a adotar as metodologias propostas pelos agilistas pelo fato delas auxiliarem no atendimento das exigências do cliente [Devulapally, 2015]. Esta tendência é mais forte no desenvolvimento de software e nos últimos anos vem ocorrendo de forma gradativa na manutenção.

No trabalho de Kajko-Mattsson & Nyfjord [Kajko-Mattsson & Nyfjord, 2009b] foi proposto um modelo ágil para manutenção que apropria diferentes práticas do Extreme Programming e do Scrum. Segundo os autores a junção desta duas metodologias possibilita a inclusão de práticas úteis tanto do ponto de vista do gerente do projeto bem como dos desenvolvedores. O modelo encoraja diversas práticas tais como *product backlog*, testes antes da codificação, planejamento iterativo, dentre outras.

A adoção na manutenção de software de algumas práticas propostas pelos agilistas foram analisadas durante 08 meses em estudo realizado por Svensson & Host [Svensson & Host, 2005]. Ao utilizar o Extreme Programming (XP) no processo de manutenção os autores concluíram que é muito difícil fazer uso do XP sem que sejam realizadas adequações no desenho de diversas práticas para desta forma adequar

às necessidades do time de desenvolvimento.

O estudo Heeager & Rose [Heeager & Rose, 2015] propõe um conjunto de nove heurísticas com o objetivo de ajudar aos profissionais da manutenção de software na adoção de práticas propostas pelos agilistas. O trabalho consistiu da inclusão do Scrum na rotina de trabalho do departamento de manutenção de software de uma organização de grande porte. Os autores argumentam que os métodos ágeis, quando aplicado ao trabalho de desenvolvimento, têm certas características relativamente bem compreendidas, no entanto o trabalho de manutenção difere do de desenvolvimento em certos aspectos e, portanto, é desafiador a implementação de métodos ágeis em um departamento de manutenção.

Diante da crescente importância das Ferramenta de Gerenciamento de Requisição de Mudanças (FGRM) no processo de manutenção de software, diversos trabalhos vêm sendo propostos com o objetivo de entender como elas estão sendo utilizadas bem como sugerir melhorias no desenho para desenvolver futuras FGRM's.

No trabalho de Junio et al. [Junio et al., 2011] é proposto um processo denominado PASM (Process for Arranging Software Maintenance Requests) que propõe lidar com tarefas de manutenção como projetos de software. Para tanto, utilizou-se técnicas de análise de agrupamento (clustering) a fim de melhor compreender e comparar as demandas de manutenção. Os resultados demostraram que depois de adotar o PASM os desenvolvedores tem dedicado um tempo maior para análise e validação. De outra forma, relacionada um menor tempo foi dedicado às tarefas de execução e codificação.

No estudo realizado por Bettenburg et al. [Bettenburg et al., 2008] foi desenvolvida uma pesquisa (*survey*) entre desenvolvedores e usuários dos projetos Apache[1], Eclipse[2] e Mozilla[3] a fim de verificar o que produziria uma boa FGRM. Os resultados demonstraram que do ponto de vista dos desenvolvedores eram consideradas úteis funcionalidades tais como reprodução do erro, rastros de pilhas (stack traces) e casos de testes. A partir deste resultado foi construído um protótipo capaz de conduzir os usuários na coleta e fornecimento de um maior número de informações úteis para a resolução do defeito reportado.

Avaliando o controle de demandas como um processo social, Bertram et al. [Bertram et al., 2010] realizaram um estudo qualitativo em FGRM's quando utilizados por pequenas equipes de desenvolvimento de software. Os resultados mostraram que este tipo ferramenta não é apenas um banco de dados de rastreamento de defeitos, recursos ou pedidos de informação, mas também atua como um ponto focal para a

[1] `http://www.apache.org/`
[2] `https://www.eclipse.org`
[3] `https://www.mozilla.org`

comunicação e coordenação para diversas partes interessadas (stakeholders) dentro e fora da equipe de software. Os clientes, gerentes de projeto, o pessoal envolvido com a garantia da qualidade e programadores, contribuem em conjunto para o conhecimento compartilhado dentro do contexto das FGRM's.

Em Zimmermann et al. [Zimmermann et al., 2009] é discutido a importância de que a informação descrita em uma Requisição de Mudança seja relevante e completa a fim de que o defeito reportado seja resolvido rapidamente. Contudo, na prática, a informação apenas chega ao desenvolvedor com a qualidade requerida após diversas interações com o usuário afetado. Com o objetivo de minimizar este problema os autores propõe um conjunto de diretrizes para a construção de um ferramenta capaz de reunir informações relevantes a partir do usuário e identificar arquivos que precisam ser corrigidos para resolver o defeito.

No trabalho de Breu et al.[Breu et al., 2010] o foco é analisar o papel dos FGRM's no suporte à colaboração entre desenvolvedores e usuários de um software. A partir da análise quantitativa e qualitativa de uma amostra de defeitos registrados em uma FGRM de dois projetos de software livre, foi possível verificar que os usuários desempenham um papel além de simplesmente reportar uma falha: a participação ativa e permanente dos usuários finais foi importante no progresso da resolução das falhas que eles descreveram.

O desenvolvimento de novas funcionalidades em FGRM's, mediante a capacidade de extensão propiciada por algumas delas vêm sendo explorada na literatura. *Buglocalizer* [Thung et al., 2014b] é uma extensão para o Bugzilla que possibilita a localização dos arquivos do código fonte que estão relacionados ao defeito relatado. A ferramenta extrai texto dos campos de sumário e descrição de um determinado erro reportado no Bugzilla. Este texto é comparado com o código fonte por meio de técnicas de Recuperação da Informação.

NextBug [Rocha et al., 2015] é uma extensão para o Bugzilla que recomenda novos bugs para um desenvolvedor baseado no defeito que ele esteja tratando atualmente. O objetivo da extensão é sugerir defeitos com base em técnicas de Recuperação de Informação.

No trabalho de Kononenko et al. [Kononenko et al., 2014] é apresentada uma ferramenta denominada *DASH* cujo objetivo é agrupar as demandas que são relevantes para as atividades de um desenvolvedor. Naturalmente todas as demandas ditas relevantes deveriam estar sob a responsabilidade de um mesmo programador. O principal objetivo desta ferramenta é aumentar a Consciência Situacional (Situational Awareness) dos desenvolvedores. Segundo os autores, o principal ganho do uso da ferramenta é que os programadores podem gerenciar melhor o excesso de informação e ficar mais

ciente da evolução das demais demandas do sistema.

Na ferramenta proposta por Thung et al. [Thung et al., 2014a] o foco é na determinação de defeitos duplicados. A contribuição deste trabalho é a integração do estado da arte de técnicas não supervisionadas para detecção de falhas duplicadas conforme proposto por Runeson et al. [Runeson et al., 2007]. A ferramenta utiliza o Modelo de Vetor Espacial (Vetor Space Model) como métrica de similaridade entre os defeitos e fornece aos desenvolvedores uma lista de possíveis duplicatas.

Once in operation, anomalies are uncovered, operating environments change, and new user requirements surface. The maintenance phase of the life cycle commences upon delivery, but maintenance activities occur much earlier.

2.1 Conceitos Fundamentais

Esta primeira seção introduz os conceitos e terminologias que ajudam no entendimento do papel e escopo da Manutenção de Software. De uma maneira geral, podemos definir atividade de manter software como a totalidade das ações necessárias para fornecer algum tipo de suporte ao produto de software. Não obstante, encontramos na literatura outras definições mais elaboradas sobre o conceito de Manutenção de Software.

Software maintenance is defined in the IEEE Standard for Software Maintenance, IEEE 1219, as the modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the to a modified environment. The standard also product addresses maintenance activities prior to delivery of the software product, but only in an information appendix of the standard.

The IEEE/EIA 12207 standard for software life cycle processes essentially depicts maintenance as one of the primary life cycle processes, and describes maintenance undergoing product software as the process of a “modification to code and associated documentation due to a problem or the need for improvement. The objective while is to modify the existing software product preserving its integrity.” ISO/IEC 14764, the international defines software standard for software maintenance, maintenance in the same terms as IEEE/EIA 12207 and emphasizes the pre-delivery aspects of maintenance, planning, for example.

Maintenance is needed to ensure that the software continues to satisfy user requirements. Maintenance. Lehman first addressed software maintenance and evolution of systems in 1969. Over a period of twenty years, his research led to the formulation of eight “Laws of Evolution”. [Leh97] Key findings include the fact that maintenance is evolutionary developments, and that maintenance decisions are aided by understand-

ing what happens to systems (and software) over time. Others state that maintenance is continued development, except that there is an extra input (or constraint)–existing large software is never complete and continues to evolve. As it evolves, it grows more complex unless some action is taken to reduce this complexity

2.2 Requisição de Mudança

Lientz & Swanson initially defined three categories of maintenance: corrective, This definition was later updated in the Standard for Software Engineering-Software Maintenance, ISO/IEC 14764 to include four categories, as follows: ? Corrective maintenance: Reactive modification of a software product performed after delivery to correct discovered problems ? Adaptive maintenance: Modification of a software product performed after delivery to keep a software product usable in a changed or changing environment ? Perfective maintenance: Modification of a software product after delivery to improve performance or maintainability ? Preventive maintenance: Modification of a software product after delivery to detect and correct latent faults in the software product before they become effective faults

ISO/IEC 14764 classifies adaptive and perfective maintenance as enhancements. It also groups together the corrective and preventive maintenance categories into a correction category, as shown in Table 1. Preventive maintenance, the newest category, is most often performed on software products where safety is critical.

Tabela 2.1: Categorias da Requisição de Mudanças. Adaptado de SWEBOK [**?**]

	Correção	**Melhoria**
Próativa	Preventiva	Perfectiva
Reativa	Corretiva	Adaptativa

2.3 O processo de Manutenção de Software

Modification requests are logged and tracked, the impact of proposed changes is determined, code and other software artifacts are modified, testing is conducted, and a new version of the software product is released

2.3.1 Manutenção de Software Tradicional

2.3.2 Manutenção de Software com Método dos Agilistas

2.4 Ferramentas de Gerenciamento de Requisições de Mudança (FGRM)

2.4.1 Características Gerais

2.4.2 Extensões para FGRM

Capítulo 3

Mapeamento Sistemático da Literatura

3.1 Introdução

OBJETIVO DA SEÇÃO: Esta seção visa apresenta uma visão geral do Mapeamento Sistemático realizado. Apresenta, de maneira sucinta, o contexto, o problema, a solução e os resultado deste estudo, de forma resumida, mas abrangente. Idealmente deverá ser a última seção a ser escrita neste Capítulo.

3.2 Metodologia de Pesquisa

Um *Mapeamento Sistemático da Literatura*, também conhecido como Estudo de Escopo (Scoping Studies), tem como objetivos fornecer uma visão geral de determinada área de pesquisa, estabelecer a existência de evidências de estudos sobre determinado tema e fornecer uma indicação da quantidade de trabalho da linha de pesquisa sob análise [Keele, 2007, Wohlin et al., 2012]. Nesta dissertação empregamos as diretrizes propostas por Petersen e outros [Petersen et al., 2008] de forma a produzirmos uma revisão de maneira sistemática afim de propiciar maior facilidade de replicação e extensão do mapeamento realizado. Em particular, definimos um conjunto de questões de pesquisa que foram utilizadas no processo de busca e seleção dos estudos primários. Em seguida, foram construídos esquemas de classificação com base nos dados extraídos dos artigos. Por fim, foi realizada a análise e sintetização dos dados com o objetivo de posicionar os estudos em suas respectivas classes na taxonomia. A estrutura desta seção está de acordo com o processo descrito por *Petersen e outros*, de modo que cada

subseção representa uma das etapas propostas pelos autores.

3.2.1 Questões de Pesquisa

O objetivo deste mapeamento sistemático é entender o estado da arte da pesquisa sobre FGRM. Em especial, o foco é identificar as extensões que estão sendo propostas para este tipo de ferramenta. No escopo deste trabalho, uma extensão é um componente de software que adiciona uma característica específica para um programa de computador[1]. Assim, a fim de alcançar e guiar os objetivos desta parte do trabalho, foram definidas as seguintes questões de pesquisa:

Incluir motivação das questões de pesquisa

- **Questão 01**: *Quais os problemas da atividade de manutenção de software as extensões das FGRM pretendem resolver?*
- **Questão 02**: *Quais papeis envolvidos no processo de manutenção de software as extensões visam dar suporte?*
- **Questão 03**: *Qual o modelo de Recuperação da Informação foi utilizada para desenvolver a extensão?*
- **Questão 04**: *Quais as FGRM disponíveis no mercados estão sendo estendidas?*

3.2.2 Pesquisa da Literatura

Com o objetivo de encontrar o conjunto de estudos mais relevantes, bem como eliminar aqueles que não são capazes de responder as questões de pesquisas propostas, adotamos os seguintes critérios para inclusão ou exclusão dos estudos:

- Critérios de Inclusão
  - Artigos publicados em conferências e periódicos (journals)
  - Estudos publicados a partir de 2010[2]
  - Artigos escritos em língua inglesa e portuguesa
  - Artigos disponíveis com texto completo
- Critérios de Exclusão

[1] `https://en.wikipedia.org/wiki/Plug-in_(computing)`

[2] Foram considerados neste estudo artigos publicados até maio/2016, data de realização da pesquisa nas base de dados.

- Livros e literatura cinza (gray literature)
- Artigos que não possuem relação com FGRM
- Estudos duplicados, neste caso apenas foi considerada a versão mais completa do trabalho

Os estudos primários foram coletados mediante a aplicação de sentenças de buscas nas seguintes bibliotecas digitais: *IEEE Explore, ACM Digital Library, Scopus, e Inspec/Compendex*. As bases de dados foram escolhidas com base na experiência reportada por Dyba e outros [Dybå et al., 2007] no qual verificou-se que o uso de apenas algumas bases era capaz de produzir um resultado similar a utilização de um conjunto maior de biblioteca digitais. As sentenças de buscas foram produzidas com base na metodologia PICO (Population, Intervention, Comparison and Outcomes) que é sugerida por Kitchenham e Charters [Keele, 2007] para ajudar pesquisadores na formulação de termos tomando como base as questões de pesquisa, que serão aplicados às bases de dados. As sentenças de busca aplicadas a cada base de dados são apresentadas na Tabela A do Apêndice A.

Após a condução da busca automatizada nas base de dados chegamos a um total de 286 artigos. A Tabela 3.1 exibe o conjunto inicial de estudos recuperados por base de dados. Os trabalhos coletados foram avaliados, através da ferramenta *JabRef*[3], em busca de possíveis duplicados tendo em vista a utilização de diferentes bases de dados. A busca por artigos duplicados resultou na exclusão de 81 documentos de modo que obtivemos um total de 205 estudos ao final do processo. Finalmente os artigos foram analisados com base na leitura do título e resumo. Nos casos em que o título e resumo não eram capazes de caracterizar o estudo uma leitura completa do texto foi realizada. O processo descrito resultou em 94 estudos incluídos neste trabalho. A Figura 3.1

Verificar o processo com objetivo de validar o total de artigo em cada etapa do processo.

Tabela 3.1: Número de Estudos Recuperados por Base de Dados

Base de Dados	**Total**
ACM Digital Library	109
IEEE Explore	100
Inspec/Compendex	22
Scopus	55

[3] `https://www.jabref.org/`

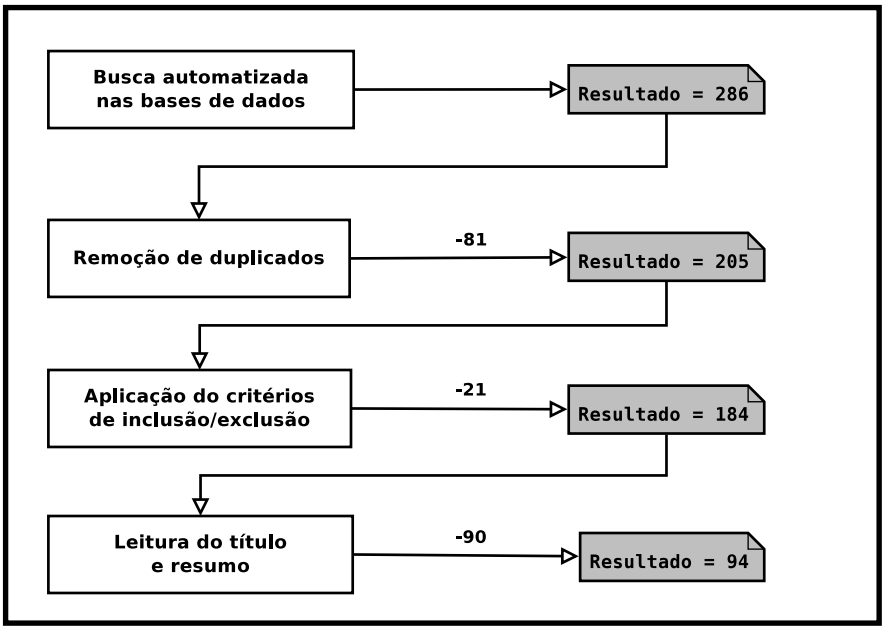

Figura 3.1: Número de artigos incluídos durante o processo de seleção dos estudos. Baseado em [Petersen et al., 2015]

3.2.3 Esquemas de Classificação

Com o objetivo de mapear os estudos sobre extensões das FGRM's foram propostos quatro esquemas de classificação. O primeiro esquema organiza os artigos pelo tipo de problema da atividade de manutenção de software a extensão se propõe solucionar. A segunda categorização distribuí os estudos pelo papel no processo de manutenção de software a extensão visa dar suporte. A terceira é uma classificação baseada na taxonomia para modelos de Recuperação da Informação (Information Retrieve - IR) proposta por Cerulo e Canfora [Cerulo & Canfora, 2004]. Em particular, utilizamos este esquema tendo em vista que grande parte das extensões propostas na literatura utilizam algum tipo de suporte de modelos de IR. O último esquema apresenta quais das ferramentas existentes na indústria estão sendo estendidas na literatura. Esta taxonomia nos fornece uma visão se as extensões propostas são com um foco propositivo, ou seja, sem que exista uma implementação concreta da extensão, ou se há algum tipo de ferramenta no qual existe um maior número de extensões propostas. Em seguida discutiremos cada esquema de classificação em detalhe.

3.2.3.1 Classificação por Tipo de Problema

Existem diversos problemas relacionados ao processo de Manutenção de Software. Já na década de 1980 pesquisas questionavam os profissionais envolvidos com manutenção de software quais os principais problemas da área[Lientz & Swanson, 1981]. Naturalmente

a percepção dos desafios envolvidos com a manutenção de software se altera com tempo, desta forma, é sempre válido revisar a literatura com o objetivo de entender quais os tipos de problemas estão sendo estudados.

Neste sentido foi proposto um esquema de classificação que relaciona os estudos pelo tipo de problema que a extensão pretende resolver. A construção da taxonomia se deu com base no processo definido por Petersen e outros [Petersen et al., 2008], o qual é composto de duas etapas:

I analisar as palavras-chaves e conceitos que identificam as contribuições do estudo por meio da analise do título e resumo

II após o término da etapa I, todas as palavras chaves são combinadas a fim de construir um conjunto de categorias para no qual os artigos devem ser classificados.

Os autores recomendam que nos casos em que o resumo e o título do estudo não sejam capazes de caracterizá-lo, as seções de introdução e conclusão também devem ser analisadas. Para as bases de dados onde era informado mais de um conjunto de palavras-chaves para um mesmo artigo, utilizamos aquelas que foram informadas pelos autores. Mediante a aplicação do processo foi construído o esquema de classificação apresentado na Figura **??**.

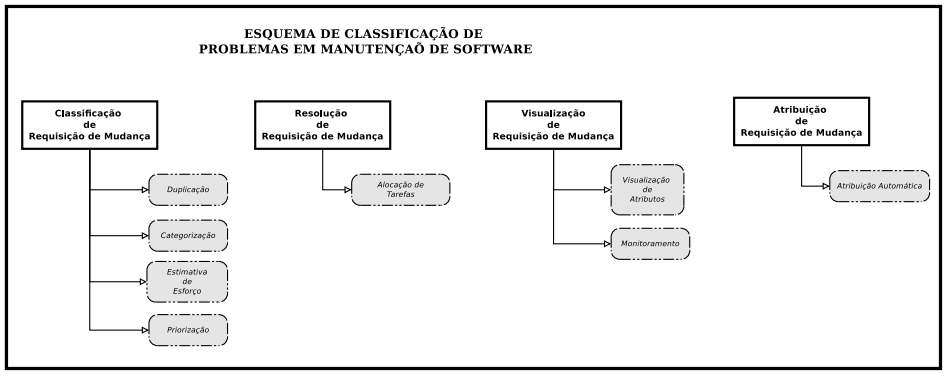

Figura 3.2: Esquema de classificação proposto

Incluir no anexo uma tabela demonstrando como as palavras-chaves foram combinadas

3.2.3.2 Classificação por Suporte ao Papel da Manutenção de Software

Da mesma forma que uma extensão proposta para as FGRM's visa resolver determinado problema, supomos ainda que a extensão pode dar suporte a determinado papel desempenhado no processo de Manutenção de Software. Para este fim utilizamos uma

classificação modificada da proposta por [Polo et al., 1999b]. No trabalho de Polo e outros o objetivo era definição de uma estrutura adequada da equipe de manutenção de software mediante a clara identificação das tarefas que cada membro deve executar. Os papéis propostos pelo no estudo é produto da aplicação da metodologia MANTEMA [Polo et al., 1999a] para manutenção em projetos de software bancários espanhóis, em especial aqueles em que a área de manutenção foi terceirizada (outsourcing). Os autores reforçam que apesar da taxonomia de papéis ter sido criada em um contexto específico, ela pode ser adequada para aplicação em outras situações.

No escopo deste trabalho a taxonomia utiliza a proposta de Polo e outros com algumas adequações. Em especial, foram removidos os papéis que segundo o nosso entendimento estão mais vinculados a um contexto de manutenção terceirizada (outsourcing). Além disso, dividimos o papel "time de manutenção' (maintenance team) em *Desenvolvedor e Analista de Qualidade* por entendemos que são papéis comuns a muito dos processos de manutenção existentes. Os papéis que compõe a taxonomia proposta estão descritos a seguir:

Usuário Afetado : Indivíduo que utiliza o sistema do qual será produzida uma Requisição de Mudança.

Reportador : Responsável por registrar a Requisição de Mudança na FGRM.

Gerente de Requisição de Mudança(Maintenance-request manager) : Responsável por decidir se uma Requisição de Mudança será aceita ou rejeitada e qual tipo de manutenção deverá ser aplicada. Posteriormente cabe a ele/ela encaminhar a RM para o Agendador.

Agendador(Scheduler) : Deve planejar a fila de Requisições de Mudança aceitas. Também estão no rol de responsabilidades deste papel a atribuição das RM's para o desenvolver mais apto.

Desenvolvedor : Responsável realizar as ações que irão solucionar a Requisição de Mudança.

Analista de Qualidade : Avaliam uma Requisição de Mudança que foi solucionada por um Desenvolvedor afim de verificar se a RM foi corretamente resolvida.

Líder da Manutenção (Head of Maintenance) : Tem por responsabilidade definir os padrões e procedimentos que compõe o processo de manutenção que será utilizado.

Apesar da taxonomia de papéis utilizada derivar de um contexto de manutenção de software específico (setor bancário e empresas com a área de manutenção terceirizada), ela é capaz de acoplar com outros tipos de processo de manutenção de software, como aquele proposto por Ihara e outros [Ihara et al., 2009]. Naquele estudo foi criada uma representação de um processo de modificação de bugs tomando como base as diversas situações que um bug possui em uma FGRM no contexto de projetos de código aberto. O processo resultante é facilmente acoplável com a taxonomia utilizada em nosso estudo.

Cabe ressaltar que está fora do escopo deste estudo elaborar uma taxonomia de papeis envolvidos na Manutenção de Software em função de supormos que isto corresponde a um esforço bem extenso. Nossa ação é identificar quais artigos trabalham com a noção de quais papeis a extensão pretender dar suporte, ou seja, relatar se existem papeis e quais são eles, sem com isso, envolver em uma consolidação definitiva.

3.2.3.3 Classificação por Técnicas de Recuperação da Informação

Um ponto em comum entre as diversas extensões para FGRM propostas na literatura é o fato delas utilizarem algum suporte de modelos de Recuperação de Informação (Information Retrieve - IR). Um modelo de IR visa solucionar um uma necessidade informacional de um usuário, representada como um conjunto de termos, no qual uma lista dos documentos mais relevantes devem ser recuperadas de uma coleção [Baeza-Yates et al., 1999].

Com o objetivo de obter uma visão geral das técnicas que estão sendo utilizadas para implementar as extensões para as FGRM, realizamos a classificação dos estudos através da taxonomia proposta por Canfora e Cerulo[Cerulo & Canfora, 2004]. O esquema de classificação consiste em duas visões sobrepostas: uma taxonomia vertical que classifica os modelos de IR com relação ao seu conjunto de características básicas; e uma taxonomia horizontal que classifica os objetos de IR com respeito as suas tarefas, forma e contexto[Cerulo & Canfora, 2004]. Nesta dissertação utilizamos a classificação vertical tendo em vista que estamos interessados na técnica utilizada na implementação da extensão.

Verificar a necessidade de definir melhor alguns termos de IR. Ex. documento, consulta do usuário, problema da representação de similaridade

Verificar o total de artigos que utilizam algum técnica de IR. Apesar de grande parte utilizar não serão todos os artigos

A taxonomia vertical é construída explorando duas características básicas de um Modelo de Recuperação da Informação: *representação (representation)*, que é a

forma adotada para representar ao mesmo tempo o documento e consulta do usuário; *raciocínio (reasoning)*, ao qual se refere ao framework adotado para resolver o problema da representação de similaridade, ou seja, trata-se do conjunto de métodos, modelos e tecnologias utilizadas para realizar o casamento entre um documento e a consulta do usuário. O esquema de classificação proposto pelos autores é apresentado na Figura 3.3.

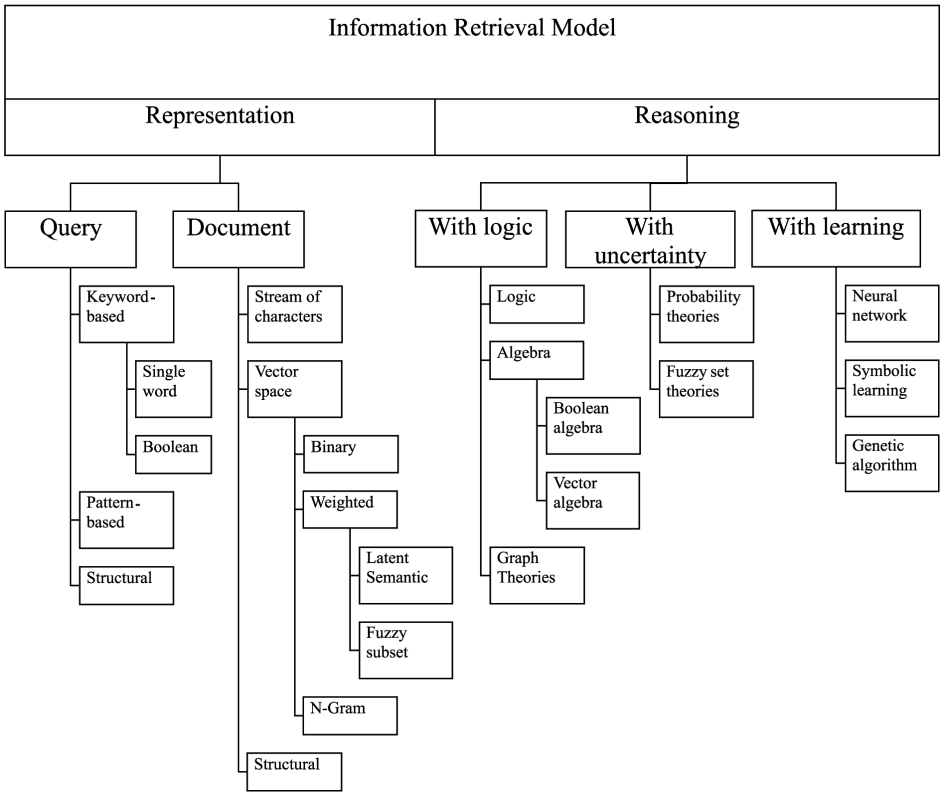

Figura 3.3: Taxonomia vertical para Modelos de Recuperação da Informação. Adaptado de [Cerulo & Canfora, 2004]

3.2.3.4 Classificação por Ferramenta Estendida

Do ponto de vista dos profissionais envolvidos em Manutenção de Software é importante entender quais as FGRM estão sendo estendidas. Por outro lado, para os pesquisadores é importante descobrir aquelas ferramentas mais "amigáveis para o desenvolvimento de novas extensões bem como a melhoria das existentes. Neste sentido, realizamos a classificação dos estudos pelas ferramentas que foram utilizados para o desenvolvimento das extensões. Uma ferramenta é entendida como utilizada por determinado estudo se ela foi utilizada no processo de validação ou se seus dados foram utilizados na avaliação da extensão.

Para produzirmos a taxonomia realizarmos a leitura do resumo e da introdução do artigo. Nos casos em que não foi possível determinar a ferramenta utilizada a partir da leitura das seções citadas, procedemos com a leitura com a parte avaliação do estudo. A Tabela exibe as ferramentas utilizadas para cada estudo analisado.

3.3 Resultados

Nesta seção apresentamos os resultados do Mapeamento Sistemático. Cada uma das taxonomias propostas é analisada mediante a apresentação dos estudos que possam exemplificar a classificação adotada. Iniciamos pela classificação por problemas na Manutenção de Software, o qual dividimos os estudos por área e tópico de pesquisa. Seguimos com a análise dos estudo pelo papel ao qual a extensão visa dar suporte. Posteriormente analisamos os modelos de IR utilizados para implementar algumas das extensões propostas na literatura. Finalizamos esta análise com as ferramentas existentes no mercado que estão efetivamente sendo entendidas.

3.3.1 Extensões para Problemas na Manutenção de Software

Nesta etapa do trabalho estamos interessados no estado da arte do estudo dos problemas encontrados na Manutenção de Software. Em especial, o nosso foco é entender que tipo de extensão para FGRM estão sendo proposta para solucionar este tipo de problema. Com base no mapeamento sistemático realizado a classificação do estudos é apresentada na Tabela 3.2.

O total de artigos por área de pesquisa e seus respectivos tópicos pode ser observado na Figura **??**

Tabela 3.2: Taxonomia por Problemas na Manutenção de Software

Taxonomia		Artigos	Total
Classificação de Requisições de Mudança	*Duplicação*	[Alipour et al., 2013, Hindle et al., 2016, Sun et al., 2010]	3
	Categorização	[Behl et al., 2014, Zhang & Lee, 2011, Chawla & Singh, 2015]	3
	Estimativa de Esforço	[Hosseini et al., 2012]	1
Atribuição de Requisições de Mudança	*Alocação Automática*	[Hosseini et al., 2012]	1
Resolução de Requisições de Mudança	*Alocação de Tarefas*	[Netto et al., 2010]	1
Visualização de Requisições de Mudança	*Visualização de Atributos*	[Dal Sassc & Lanza, 2013]	1
	Monitoramento	[Takama & Kurosawa, 2013]	1

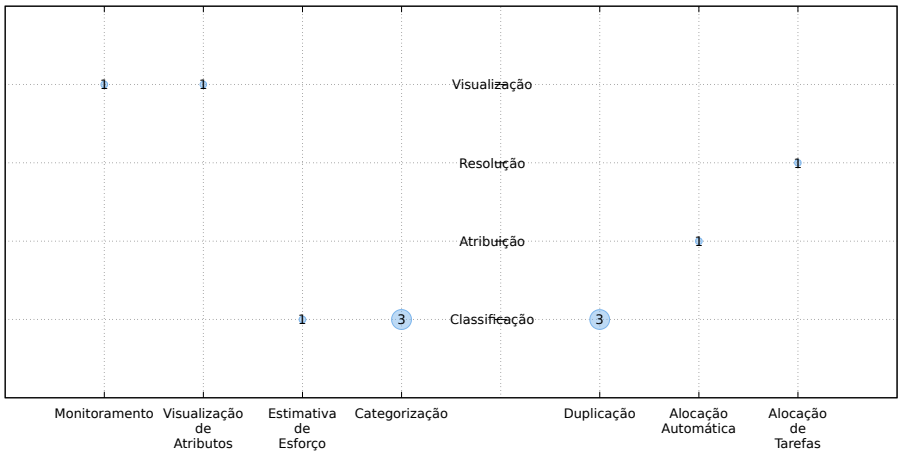

Figura 3.4: Total de artigos por categoria de problema

3.3.1.1 Classificação de Requisições de Mudança

3.3.1.2 Duplicação

3.3.1.3 Categorização

3.3.1.4 Estimativa de Esforço

3.3.2 Extensões com Suporte à Papeis

3.3.3 Técnicas de IR Utilizadas pelas Extensões

3.3.4 Ferramentais Estendidas

3.4 Limitações e Ameaças à Validade

3.5 Trabalhos Relacionados

3.6 Resumo do Capítulo

Capítulo 4

Caracterização das Ferramentas de Gerenciamento de Requisição de Mudança

4.1 Introdução

Definir conforme livro do wohlin(é assim que escreve??) o conceito de estudo exploratório

Esta etapa do trabalho consiste de um estudo exploratório com o objetivo de determinar quais são as funcionalidades comuns às Ferramentas de Gerenciamento de Requisição de Mudança (FGRM). O estudo consistirá na leitura da documentação de alguns FGRM para que de forma sistemática seja levantado quais são as funcionalidades oferecidas pela ferramenta em análise. O método de escolha das FGRM será avaliado posteriormente, todavia, um possível ponto de partida é a lista disponível na Wikipédia que compara diversas FGRM[1].

4.2 Objetivo do Capítulo

Neste capítulo realizamos um estudo exploratório com o objetivo de determinar as funcionalidades comuns as Ferramentas de Gerenciamento de Requisição de Mudanças existentes no mercado. A partir deste conjunto de compartilhado de funcionalidades é possível avaliar a contribuição das extensões que estão sendo propostas na literatura, conforme discutido no Capítulo3.

[1] `https://en.wikipedia.org/wiki/Comparison_of_issue-tracking_systems`

Acreditamos que o resultado deste estudo permitirá compreender melhor este tipo de ferramenta tomando como base as suas funcionalidades em comum. Também será possível propor extensões para as FGRM tendo em vista a possibilidade de determinar o conjunto mínimo de funções deste tipo de sistema. Uma outra possível contribuição é a possibilidade de criação de um esquema de caracterização com base nas funcionalidades oferecidas.

4.3 Metodologia

O processo de descoberta das funcionalidades comuns às FGRM é composto das etapas descritas a seguir:

- Seleção das Ferramentas
- Inspeção da Documentação
- Agrupamento das Funcionalidades

Cada uma etapas é explicada em detalhes nas próximas seções.

4.3.1 Seleção das Ferramentas

A primeira etapa consistiu na definição das ferramentas que seriam utilizadas neste estudo. Em uma inspeção inicial, verificamos a existência de mais de 50 FGRM disponíveis comercialmente ou de código aberto[2]. Contudo, devido a dificuldade de realizar a análise de cada uma daquelas ferramentas, optamos por escolher um subconjunto das ferramentas que fosse mais representativas, segundo o nosso conhecimento na área. A representatividade neste caso corresponde a notoriedade que a ferramenta possui dentro da literatura (Ex. Bugzilla) ou mesmo na industria (Ex. Github).

Conforme sugestão do professor Rodolfo seria interessate avaliar quais das ferramentas têm real notoriedade com base na opinião de profissionais envolvidos em manutenção de software

As FGRM selecionadas foram inicialmente classificadas como *"ferramentas"* e *"serviços da internet"*. O primeiro grupo representa os softwares capazes de serem implantados na infraestrutura do seu cliente e do qual permite algum grau de personalização de alguns componentes, como por exemplo o bando de dados utilizado. No segundo grupo estão os software que ofertam a gerência das RM's mediante uma

[2] `https://en.wikipedia.org/wiki/Comparison_of_issue-tracking_systems`

arquitetura do tipo Software como Serviço (Software as Service), onde certos tipos de alterações no comportamento do software por parte do usuário são mais restritas. As ferramentas selecionadas são apresentadas por tipo de categoria na Tabela 4.1

Incluir referência sobre Software como Serviço

Tabela 4.1: Ferramentas e serviços da Internet selecionados.

Ferramentas		**Serviços da Internet**	
Bugzilla	`https://www.bugzilla.org/`	SourceForge	`https://sourceforge.net/`
MantisBT	`https://www.mantisbt.org/`	Lauchpad	`https://launchpad.net/`
Trac	`https://trac.edgewall.org/`	Code Plex	`https://www.codeplex.com/`
Redmine	`www.redmine.org/`	GitLab	`https://gitlab.com`
Jira	`https://www.atlassian.com/software/jira`	GitHub	`https://github.com/`

4.3.2 Inspeção da Documentação

Neste etapa do trabalho realizamos a leitura do material disponível na internet para cada uma das ferramentas. Entre estes materiais inclui manual do usuário, manual do desenvolvedor, notas de lançamento e etc. Para cada uma das FGRM optamos por analisar a última versão estável do software a fim de analisarmos o que há de mais novo disponível aos usuários. A Tabela apresenta a versão analisada e o elo de ligação para cada uma da documentação utilizada neste estudo. Para aquelas ferramentas que apresentam documentação em mais de um idioma optamos sempre por utilizar aquela escrita em inglês por entendermos ser a que possivelmente mais atualizada.

4.3.3 Agrupamento das Funcionalidades

Esta etapa tem por objetivo agrupar as funcionalidades que aparecem com nomenclatura distintas em diferentes ferramentas, mas que apresentam o mesmo significado. Cabe ressaltar que o agrupamento de algumas funcionalidades pode depender de uma análise subjetiva de quem está realizando a atividade. Neste sentido, a fim de evitar algum tipo de viés o agrupamento foi realizado em duas etapas:

Análise Individual Neste etapa o autor e um outro especialista realizam de forma separada os agrupamentos que acharem necessários

Anaĺise Compartilhada Em um segundo momento tanto o auto quanto o especialista discutem as possíveis divergências até que um consenso seja obtido.

Após as duas etapas descritas anteriormente chegamos ao conjunto de funcionalidades descritos na Tabela. A coluna agrupamento exibe o nome comum para representas as funções descritas na coluna "nomenclatura original"

4.4 Resultados

4.5 Discussão

4.6 Ameças à Validade

4.7 Resumo do Capítulo

Capítulo 5

Pesquisa com Profissionais

5.1 Introdução

Com o objetivo de coletar os aspectos mais importantes das FGRM's do ponto de vista dos profissionais ligados à manutenção de software será realizada uma pesquisa (survey). O planejamento e o desenho da pesquisa seguirá as diretrizes propostas em[Wohlin et al., 2012].

A população da pesquisa proposta é a comunidade envolvida com o processo de manutenção de software e que faça uso de FGRM's. Neste contexto, seriam possíveis amostras, os desenvolvedores envolvidos com tarefas de manutenção nos projetos da Mozilla[1] ou da Eclipse Foundation[2].

A importância deste tipo de trabalho está na possibilidade de avaliar se as pesquisas relativas a evolução das FGRM estão em consonância com as necessidades dos profissionais envolvidos em manutenção de software, reduzindo, desta forma, a distância entre o estado da arte e o estado da prática.

usar a expectativa de sermos aderentes às demandas do praticante

5.2 Objetivo da Pesquisa com Profissionais

Em linhas gerais, o objetivo desta etapa do estudo é analisar, através da percepção e opinião dos profissionais envolvidos em manutenção de software, a situação das funcionalidades atualmente oferecidas pelas FGRM, bem como a relevância das extensões propostas na literatura. Estruturando melhor o objetivo, conforme propõe a metodolo-

[1] `https://bugzilla.mozilla.org/`
[2] `https://bugs.eclipse.org/bugs/`

gia GQM (Goal, Question e Metric)[Basili et al., 1994], *o propósito deste estudo avaliar as funcionalidade oferecidas e as extensões propostas nas literatura para as FGRM do ponto de vista dos profissionais envolvidos em manutenção de software no contexto de projetos de software de código aberto e uma empresa pública informática de médio porte.*

Com intuito de atingir os objetivos propostos fora definidas as seguintes questões de pesquisa:

Incluir uma descrição para cada questão de pesquisas

Questão 01 Qual o perfil dos profissionais envolvidos em Manutenção de Software?

Questão 02 Qual a opinião dos profissionais envolvidos em Manutenção de Software com relação as funcionalidades oferecidas pelas FGRM?

Questão 03 Na visão dos profissionais envolvidos em Manutenção de Software quais das extensões propostas na literatura teriam maior relevância em suas atividades atuais?

As questões de pesquisas foram respondidas mediante a realização de uma pesquisa baseada em questionário (survey). O desenho da pesquisa é detalhada na próxima seção onde discutimos a estrutura do questionário bem como a amostra a população e a sua respectiva amostra utilizada no estudo.

5.3 Desenho da Pesquisa com Profissionais

Esta Pesquisa com Profissionais (survey) consistiu de um estudo exploratório sem uma hipótese prévia a ser avaliada. Realizamos um survey com um desenho transseccional[Kitchenham, 2002] onde a nossa população de interesse são os profissionais envolvidos em manutenção de software e estamos especialmente interessado na experiência deles no uso das FGRM.

A pesquisa foi realizada através um questionário auto-administrável disponível pela Internet[3]. O formulário foi enviado aos participantes mediante o e-mail previamente coletado.

5.3.1 Questionário

O formulário enviado aos participantes foi estruturado em três parte, cada uma com o objetivo de coletar um conjunto de informações. Na primeira parte estamos inter-

[3] `https://www.google.com/forms/about/`

essados na formação de base (background) dos profissionais. O segundo conjunto de perguntas visa obter a percepção dos participantes sobre as funcionalidades atualmente oferecidas pelas FGRM. A terceira parte é do formulário contêm as perguntas sobre a percepção dos participantes sobre as extensões propostas na literatura.

A fim de obtermos um formulário que conseguisse atingir os objetivos deste estudo, realizamos um processo de avaliação em quatro etapas. O formulário resultante de uma etapa anterior foi utilizado como entrada de uma etapa posterior. Desta forma, utilizamos um processo iterativo para produzirmos o formulário.

(i) Avaliação por Pesquisadores: Nesta etapa o formulário inicialmente proposto foi enviado para dois pesquisadores experientes na área de manutenção de software.

(ii) Avaliação por Profissionais O formulário resultante da análise anterior foi encaminhado a dois profissionais experientes envolvidos com manutenção de software.

(iii) Piloto da Pesquisa O formulário obtido após a fase anterior foi utilizado em um piloto com dez profissionais envolvidos da manutenção de software de uma empresa pública de informática - PRODABEL[4]

(iv) Tradução do Formulário Em cada uma das etapas de anteriores o formulário foi aplicado em português, tendo em vista que alguns profissionais envolvidos no processo de avaliação não ter fluência em língua inglesa, em especial na fase "Piloto da Pesquisa'. Neste sentido, a última etapa consistiu na tradução do formulário para a língua inglesa. Esta etapa foi conduzida com o suporte de um pesquisador experiente na área de Engenharia de Software.

Acho que devemos tentar um grupo não PUBLICO prodabel e também um grupo PRODABEL (onde seja possível comentar sobre possíveis especificidades!!)

Avaliar o impacto de ter um questionário em inglês e outro em português

5.3.2 População, Amostra e Respostas

Avaliar a utilização de um ranqueamento para aplicar o questionário em desenvolvedores de projetos de código aberto

[4] `http://www.prodabel.pbh.gov.br`

5.4 Análise dos Dados

Neste seção apresentamos os resultado obtidos da aplicação do questionário. Os foram divididos pela questão de pesquisa ao qual visa responder. Por se tratar de um estudo exploratório, no qual não foi proposta determinada tese a ser provada, a análise dos resultados é feita mediante o uso de gráficos representando a escala de Likert. Este tipo de grafo é recomendado para visualizar dados na escala de Likert tendo em vista que possibilita o entendimento da divergência entre as respostas dos participantes [Robbins & Heiberger, 2011]

5.5 Discussão

5.6 Ameças à Validade

5.6.0.1 Ameaças Internas

5.6.0.2 Ameaças Externas

5.7 Resumo do Capítulo

Capítulo 6

Extensões para Ferramentas de Gerenciamento de Requisição de Mudança

6.1 Introdução

A partir dos resultados do Mapeamento Sistemática, do Estudo de Caracterização das ferramentas e da Pesquisa com o profissionais pretende-se desenvolver uma ou mais extensão (plugins) para determinada FGRM. Cabe ressaltar que esta parte do trabalho será realizada caso o esforço seja compatível com os prazos e recursos disponíveis. No caso da implementação de um plugin este será apresentado e avaliado mediante a realização de um *Experimento Controlado* [Wohlin et al., 2012] utilizando a base de dados de demandas de manutenção de uma empresa de software real. Este experimento será conduzido com o objetivo de avaliar a utilização de uma extensão em um ambiente de desenvolvimento e manutenção de software real.

6.2 Ameaças à Validade

6.3 Resumo do Capítulo

Capítulo 7

Avaliação das Extensões para Ferramentas de Gerenciamento de Requisição de Mudança

7.1 Introdução

7.2 Resumo do Capítulo

Capítulo 8

Conclusão

A Manutenção de Software é um processo complexo e caro e, portanto, merece atenção da comunidade acadêmica e da indústria. Desta forma, emerge a necessidade do desenvolvimento de técnicas, processo e ferramentas que reduzam o custo e o esforço envolvidos nas atividades de manutenção e evolução de software. Neste contexto, as Ferramentas de Gerenciamento de Requisição de Mudança desempenham um papel fundamental que ultrapassa a simples função de registrar falhas em software. Neste sentido é proposto o estudo para entender o papel desta ferramenta, analisar a literatura sobre o assunto e discutir os aspectos que são considerados mais importantes do ponto de vista dos profissionais. Para alcançarmos este objetivo é proposto um Cronograma de Atividade conforme exibido no Anexo **??**. As atividades que compõe cada etapa do trabalho estão descritas no Anexo **??**.

Referências Bibliográficas

[Alipour et al., 2013] Alipour, A.; Hindle, A. & Stroulia, E. (2013). A contextual approach towards more accurate duplicate bug report detection. Em *Proceedings of the 10th Working Conference on Mining Software Repositories*, pp. 183--192. IEEE Press.

[Baeza-Yates et al., 1999] Baeza-Yates, R.; Ribeiro-Neto, B. et al. (1999). *Modern Information Retrieval*, volume 463. ACM press New York.

[Basili et al., 1994] Basili, V. R.; Caldiera, G. & Rombach, H. D. (1994). The goal question metric approach. Em *Encyclopedia of Software Engineering*. Wiley.

[Baysal & Holmes, 2012] Baysal, O. & Holmes, R. (2012). A qualitative study of mozillas process management practices. *David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada, Tech. Rep. CS-2012-10*.

[Baysal et al., 2013] Baysal, O.; Holmes, R. & Godfrey, M. W. (2013). Situational awareness: Personalizing issue tracking systems. Em *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pp. 1185--1188, Piscataway, NJ, USA. IEEE Press.

[Behl et al., 2014] Behl, D.; Handa, S. & Arora, A. (2014). A bug mining tool to identify and analyze security bugs using naive bayes and tf-idf. Em *Optimization, Reliabilty, and Information Technology (ICROIT), 2014 International Conference on*, pp. 294--299. IEEE.

[Bertram et al., 2010] Bertram, D.; Voida, A.; Greenberg, S. & Walker, R. (2010). Communication, collaboration, and bugs: The social nature of issue tracking in small, collocated teams. Em *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work*, CSCW '10, pp. 291--300, New York, NY, USA. ACM.

[Bettenburg et al., 2008] Bettenburg, N.; Just, S.; Schröter, A.; Weiss, C.; Premraj, R. & Zimmermann, T. (2008). What makes a good bug report? Em *Proceedings*

of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering, pp. 308--318. ACM.

[Bilal & Black, 2005] Bilal, H. & Black, S. (2005). Using the ripple effect to measure software quality. Em *SOFTWARE QUALITY MANAGEMENT-INTERNATIONAL CONFERENCE-*, volume 13, p. 183.

[Breu et al., 2010] Breu, S.; Premraj, R.; Sillito, J. & Zimmermann, T. (2010). Information needs in bug reports: Improving cooperation between developers and users. Em *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work*, CSCW '10, pp. 301--310, New York, NY, USA. ACM.

[Cavalcanti et al., 2014] Cavalcanti, Y. C.; Mota Silveira Neto, P. A.; Machado, I. d. C.; Vale, T. F.; Almeida, E. S. & Meira, S. R. d. L. (2014). Challenges and opportunities for software change request repositories: a systematic mapping study. *Journal of Software: Evolution and Process*, 26(7):620--653.

[Cerulo & Canfora, 2004] Cerulo, L. & Canfora, G. (2004). A taxonomy of information retrieval models and tools. *CIT. Journal of computing and information technology*, 12(3):175--194.

[Chawla & Singh, 2015] Chawla, I. & Singh, S. K. (2015). An automated approach for bug categorization using fuzzy logic. Em *Proceedings of the 8th India Software Engineering Conference*, pp. 90--99. ACM.

[Dal Sassc & Lanza, 2013] Dal Sassc, T. & Lanza, M. (2013). A closer look at bugs. Em *Software Visualization (VISSOFT), 2013 First IEEE Working Conference on*, pp. 1--4. IEEE.

[Devulapally, 2015] Devulapally, G. K. (2015). Agile in the context of Software Maintainability.

[Dybå et al., 2007] Dybå, T.; Dingsøyr, T. & Hanssen, G. K. (2007). Applying systematic reviews to diverse study types: An experience report. Em *ESEM*, volume 7, pp. 225--234.

[Engelbertink & Vogt, 2010] Engelbertink, F. P. & Vogt, H. H. (2010). How to save on software maintenance costs. *Omnext White Paper. Accessed.*

[Haney, 1972] Haney, F. M. (1972). Module connection analysis: a tool for scheduling software debugging activities. Em *Proceedings of the December 5-7, 1972, fall joint computer conference, part I*, pp. 173--179. ACM.

[Heeager & Rose, 2015] Heeager, L. T. & Rose, J. (2015). Optimising agile development practices for the maintenance operation: nine heuristics. *Empirical Software Engineering*, 20(6):1762--1784. ISSN 15737616.

[Herrin, 1985] Herrin, W. R. (1985). Software maintenance costs: A quantitative evaluation. Em *Proceedings of the Sixteenth SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '85, pp. 233--237, New York, NY, USA. ACM.

[Hindle et al., 2016] Hindle, A.; Alipour, A. & Stroulia, E. (2016). A contextual approach towards more accurate duplicate bug report detection and ranking. *Empirical Software Engineering*, 21(2):368--410.

[Hirota et al., 1994] Hirota, T.; Tohki, M.; Overstreet, C. M.; Hashimoto, M. & Cherinka, R. (1994). An approach to predict software maintenance cost based on ripple complexity. Em *Software Engineering Conference, 1994. Proceedings., 1994 First Asia-Pacific*, pp. 439--444. IEEE.

[Hosseini et al., 2012] Hosseini, H.; Nguyen, R. & Godfrey, M. W. (2012). A market-based bug allocation mechanism using predictive bug lifetimes. Em *Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on*, pp. 149--158. IEEE.

[IEEE, 1990] IEEE (1990). IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*, pp. 1–84.

[Ihara et al., 2009] Ihara, A.; Ohira, M. & Matsumoto, K.-i. (2009). An Analysis Method for Improving a Bug Modification Process in Open Source Software Development. Em *Proceedings of the Joint International and Annual ERCIM Workshops on Principles of Software Evolution (IWPSE) and Software Evolution (Evol) Workshops*, IWPSE-Evol '09, pp. 135--144, New York, NY, USA. ACM.

[ISO/IEC, 2006] ISO/IEC (2006). International Standard - ISO/IEC 14764 IEEE Std 14764-2006 Software Engineering 2013; Software Life Cycle Processes 2013; Maintenance. *ISO/IEC 14764:2006 (E) IEEE Std 14764-2006 Revision of IEEE Std 1219-1998)*, pp. 01–46.

[Junio et al., 2011] Junio, G.; Malta, M.; de Almeida Mossri, H.; Marques-Neto, H. & Valente, M. (2011). On the benefits of planning and grouping software maintenance requests. Em *Software Maintenance and Reengineering (CSMR), 2011 15th European Conference on*, pp. 55–64. ISSN 1534-5351.

[Kajko-Mattsson & Nyfjord, 2009a] Kajko-Mattsson, M. & Nyfjord, J. (2009a). A model of agile evolution and maintenance process. Em *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*, pp. 1--10. IEEE.

[Kajko-Mattsson & Nyfjord, 2009b] Kajko-Mattsson, M. & Nyfjord, J. (2009b). A model of agile evolution and maintenance process. Em *System Sciences, 2009. HICSS '09. 42nd Hawaii International Conference on*, pp. 1–10. ISSN 1530-1605.

[Kaur & Singh, 2015] Kaur, U. & Singh, G. (2015). A review on software maintenance issues and how to reduce maintenance efforts. *International Journal of Computer Applications*, 118(1).

[Keele, 2007] Keele, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Em *Technical report, Ver. 2.3 EBSE Technical Report. EBSE.*

[Kitchenham, 2002] Kitchenham, B. A. (2002). Principles of survey research part 2: Designing a survey barbara a. kitchenham department of computer science keele university, staffordshire, uk. *ACM SIGSOFT Software Engineering Notes*, 27(1):18.

[Kononenko et al., 2014] Kononenko, O.; Baysal, O.; Holmes, R. & Godfrey, M. W. (2014). Dashboards: Enhancing developer situational awareness. Em *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, pp. 552--555, New York, NY, USA. ACM.

[Koskinen, 2010] Koskinen, J. (2010). Software maintenance costs. *Jyväskylä: University of Jyväskylä.*

[Lehman, 1980] Lehman, M. M. (1980). On understanding laws, evolution, and conservation in the large-program life cycle. *Journal of Systems and Software*, 1:213--221.

[Lientz & Swanson, 1980] Lientz, B. P. & Swanson, E. B. (1980). *Software Maintenance Management.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. ISBN 0201042053.

[Lientz & Swanson, 1981] Lientz, B. P. & Swanson, E. B. (1981). Problems in application software maintenance. *Commun. ACM*, 24(11):763--769. ISSN 0001-0782.

[Naz et al., 2016] Naz, R.; Khan, M. N. A. & Aamir, M. (2016). Scrum-Based Methodology for Product Maintenance and Support. (January):10--27.

[Netto et al., 2010] Netto, F.; Barros, M. O. & Alvim, A. C. (2010). An automated approach for scheduling bug fix tasks. Em *Software Engineering (SBES), 2010 Brazilian Symposium on*, pp. 80--89. IEEE.

[Petersen et al., 2008] Petersen, K.; Feldt, R.; Mujtaba, S. & Mattsson, M. (2008). Systematic mapping studies in software engineering. *EASE'08 Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering*, pp. 68--77. ISSN 02181940.

[Petersen et al., 2015] Petersen, K.; Vakkalanka, S. & Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1--18. ISSN 09505849.

[Polo et al., 1999a] Polo, M.; Piattini, M.; Ruiz, F. & Calero, C. (1999a). Mantema: a complete rigorous methodology for supporting maintenance based on the iso/iec 12207 standard. Em *Software Maintenance and Reengineering, 1999. Proceedings of the Third European Conference on*, pp. 178–181.

[Polo et al., 1999b] Polo, M.; Piattini, M.; Ruiz, F. & Calero, C. (1999b). Roles in the maintenance process. *ACM SIGSOFT Software Engineering Notes*, 24(4):84--86. ISSN 01635948.

[Robbins & Heiberger, 2011] Robbins, N. B. & Heiberger, R. M. (2011). Plotting likert and other rating scales. Em *Proceedings of the 2011 Joint Statistical Meeting*, pp. 1058--1066.

[Rocha et al., 2015] Rocha, H.; Oliveira, G.; Marques-Neto, H. & Valente, M. (2015). Nextbug: a bugzilla extension for recommending similar bugs. *Journal of Software Engineering Research and Development*, 3(1).

[Runeson et al., 2007] Runeson, P.; Alexandersson, M. & Nyholm, O. (2007). Detection of duplicate defect reports using natural language processing. Em *Proceedings of the 29th International Conference on Software Engineering*, ICSE '07, pp. 499--510, Washington, DC, USA. IEEE Computer Society.

[Serrador & Pinto, 2015] Serrador, P. & Pinto, J. K. (2015). Does Agile work? - A quantitative analysis of agile project success. *International Journal of Project Management*, 33(5):1040--1051. ISSN 02637863.

[Shukla & Misra, 2008] Shukla, R. & Misra, A. K. (2008). Estimating software maintenance effort: A neural network approach. Em *Proceedings of the 1st India Software Engineering Conference*, ISEC '08, pp. 107--112, New York, NY, USA. ACM.

[Soltan & Mostafa, 2016] Soltan, H. & Mostafa, S. (2016). Leanness and Agility within Maintenance Process. (January 2014).

[Sun et al., 2010] Sun, C.; Lo, D.; Wang, X.; Jiang, J. & Khoo, S.-C. (2010). A discriminative model approach for accurate duplicate bug report retrieval. Em *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, pp. 45--54. ACM.

[Svensson & Host, 2005] Svensson, H. & Host, M. (2005). Introducing an agile process in a software maintenance and evolution organization. Em *Ninth European Conference on Software Maintenance and Reengineering*, pp. 256–264. ISSN 1534-5351.

[Takama & Kurosawa, 2013] Takama, Y. & Kurosawa, T. (2013). Application of monitoring support visualization to bug tracking systems. Em *Industrial Electronics (ISIE), 2013 IEEE International Symposium on*, pp. 1--5. IEEE.

[Tan & Mookerjee, 2005] Tan, Y. & Mookerjee, V. (2005). Comparing uniform and flexible policies for software maintenance and replacement. *Software Engineering, IEEE Transactions on*, 31(3):238–255. ISSN 0098-5589.

[Thomas, 2006] Thomas, D. (2006). Agile evolution: Towards the continuous improvement of legacy software. *Journal of Object Technology*, 5(7):19--26.

[Thung et al., 2014a] Thung, F.; Kochhar, P. S. & Lo, D. (2014a). Dupfinder: Integrated tool support for duplicate bug report detection. Em *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, ASE '14, pp. 871--874, New York, NY, USA. ACM.

[Thung et al., 2014b] Thung, F.; Le, T.-D. B.; Kochhar, P. S. & Lo, D. (2014b). Buglocalizer: Integrated tool support for bug localization. Em *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2014, pp. 767--770, New York, NY, USA. ACM.

[Wohlin et al., 2012] Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M. C.; Regnell, B. & Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.

[Yu & Mishra, 2013] Yu, L. & Mishra, A. (2013). An empirical study of lehman's law on software quality evolution. *Int J Software Informatics*, 7(3):469--481.

[Zelkowitz et al., 1979] Zelkowitz, M. V.; Shaw, A. C. & Gannon, J. D. (1979). *Principles of Software Engineering and Design.* Prentice Hall Professional Technical Reference. ISBN 013710202X.

[Zhang, 2003] Zhang, H. (2003). *Introduction to Software Engineering,*. Tsinghua University Press.

[Zhang & Lee, 2011] Zhang, T. & Lee, B. (2011). A bug rule based technique with feedback for classifying bug reports. Em *Computer and Information Technology (CIT), 2011 IEEE 11th International Conference on*, pp. 336--343. IEEE.

[Zimmermann et al., 2009] Zimmermann, T.; Premraj, R.; Sillito, J. & Breu, S. (2009). Improving bug tracking systems. Em *Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on*, pp. 247–250.

Apêndice A

Instrumentos do Mapeamento Sistemático

Tabela A.1: Sentença de Busca por Base de Dados

Base de Dados	Setença de Busca
ACM Digital Library	("issue tracking" OR "bug tracking" OR "issue-tracking" OR "bug-tracking" OR "bug repository" OR "issue repository") AND ("issue report" OR "bug report" OR "bug priorization" OR "bug fix" OR "bug assigment" OR "bug reassigment" OR "bug triage" OR "duplicate bug" OR "reopened bug" OR "bug impact" OR "bug localization" OR "bug prediction" OR "bug risk" OR "bug severity" OR "bug classification") AND ("extension" OR "plugin" OR "add-on" OR "tool" OR "improving" OR "personalization")
IEEE Explore	("Document Title":"issue tracking") OR ("Document Title":"bug tracking") OR ("Document Title":"issue-tracking") OR ("Document Title":"bug-tracking") OR ("Document Title":"bug repository") OR ("Document Title":"issue repository") AND ("Document Title":"issue report" OR "Document Title":"bug report" OR "Document Title":"bug priorization" OR "Document Title":"bug fix" OR "Document Title":"bug assigment" OR "Document Title":"bug reassigment" OR "Document Title":"bug triage" OR "Document Title":"duplicate bug" OR "Document Title":"reopened bug" OR "Document Title":"bug impact" OR "Document Title":"bug localization" OR "Document Title":"bug prediction" OR "Document Title":"bug risk" OR "Document Title":"bug severity" OR "Document Title":"bug classification") AND ("Document Title":"extension" OR "Document Title":"plugin" OR "Document Title":"add-on" OR "Document Title":"tool" OR "Document Title":"improving" OR "Document Title":"personalization")
Inspec/Compendex	((("issue tracking" OR "bug tracking" OR "issue-tracking" OR "bug-tracking" OR "bug repository" OR "issue repository") WN KY) AND (("issue report" OR "bug report" OR "bug priorization" OR "bug fix" OR "bug assigment" OR "bug reassigment" OR "bug triage" OR "duplicate bug" OR "reopened bug" OR "bug impact" OR "bug localization") WN KY)) AND (("extension" OR "plugin" OR "add-on" OR "tool" OR "improving" OR "personalization") WN KY)) OR ((("issue tracking" OR "bug tracking" OR "issue-tracking" OR "bug-tracking" OR "bug repository" OR "issue repository") WN KY) AND (("bug prediction" OR "bug risk" OR "bug severity" OR "bug classification") WN KY)) AND (("extension" OR "plugin" OR "add-on" OR "tool" OR "improving" OR "personalization") WN KY))
Scopus	(TITLE-ABS-KEY (("issue tracking" OR "bug tracking" OR "issue-tracking" OR "bug-tracking" OR "bug repository" OR "issue repository"))) AND (TITLE-ABS-KEY ("issue report" OR "bug report" OR "bug priorization" OR "bug fix" OR "bug assigment" OR "bug reassigment" OR "bug triage" OR "duplicate bug" OR "reopened bug" OR "bug impact" OR "bug localization" OR "bug prediction" OR "bug risk" OR "bug severity")) AND (TITLE-ABS-KEY ("extension" OR "plugin" OR "add-on" OR "tool" OR "improving" OR "personalization"))