

**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

© 2016, Vagner Clementino.
Todos os direitos reservados.

Clementino, Vagner

Um Estudo de Ferramentas de Gerenciamento de Requisição
de Mudança / Vagner Clementino. — Belo Horizonte, 2016
xxv, 102 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 adipiscing 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 adipiscing 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	Evolução da manutenção de software como percentual do custo total. Extraído de [Engelbertink & Vogt, 2010]	4
1.2	Dimensões de melhoria das FGRM's. Adaptado de [Zimmermann et al., 2005]	8
2.1	Tipos de manutenção segundo a norma ISO/IEC 14764. Extraído de [ISO/IEC, 2006]	15
2.2	IEEE 1219 - Processo de Manutenção de Software	17
2.3	ISO/IEC 14764 Processo de Manutenção de Software	17
3.1	Número de artigos incluídos durante o processo de seleção dos estudos. Baseado em [Petersen et al., 2015]	28
3.2	Esquema de classificação das melhorias propostas na literatura. Os retângulos representam as dimensões de melhorias e os polígonos de cantos arredondados representam as melhorias.	29
3.3	Taxonomia vertical para Modelos de Recuperação da Informação. Adaptado de [Cerulo & Canfora, 2004]	32
3.4	Número de estudos primários por ano de publicação.	34
3.5	Esquema de classificação das melhorias propostas na literatura. Os retângulos representam as dimensões de melhorias e os polígonos de cantos arredondados representam as melhorias.	35
3.6	Total de artigos por dimensão de melhoria	36
3.7	Total de artigos por tópico de melhoria	37
3.8	Total de artigos por papel na manutenção de software	44
3.9	Total de estudos que são extensões para FGRM	48
5.1	Os conceitos que compõem o Arcabouço Conceitual proposto por de Mello. Extraído de [de Mello et al., 2015]	67
5.2	Sentenças utilizadas para escolhas dos grupos do LinkedIn e de discussões no Stack Overflow	71

5.3	Ferramenta de coleta de dados da rede Stack Overflow	72
5.4	Histórico de relatos de uma RM do projeto Python	73

Lista de Tabelas

1.1	Exemplos de ferramentas e serviços da Internet. Adaptado de [Cavalcanti et al., 2014]	5
2.1	Categorias da Requisição de Mudanças. Adaptado de SWEBOK [442, 2004]	15
3.1	Número de Estudos Recuperados por Base de Dados	27
3.2	Lista de artigos de acordo com o esquema de classificação	35
4.1	Graus de Relevância	55
4.2	Ferramentas e sus respectivas documentação	56
4.3	Ferramentas utilizados no estudo	58
5.1	Fontes de Amostragem utilizadas no estudo.	69
A.1	Sentença de Busca por Base de Dados	98
B.1	Lista de Ferraentas	100
C.1	Lista de Ferraentas	102

Sumário

Agradecimentos	ix
Resumo	xiii
Abstract	xv
Resumo Estendido	xvii
Lista de Figuras	xix
Lista de Tabelas	xxi
1 Introdução	3
1.1 Motivação	5
1.2 Problema	6
1.3 Objetivos	9
1.4 Visão Geral do Estudo	10
1.5 Metodologia de Pesquisa	10
1.6 Contribuições do Estudo	11
1.7 Organização do Trabalho	11
2 Manutenção de Software: Uma Visão Geral	13
2.1 Conceitos Fundamentais	14
2.2 Requisição de Mudança	15
2.3 O processo de Manutenção de Software	16
2.3.1 Manutenção de Software Tradicional	16
2.3.2 Manutenção de Software com Método dos Agilistas	18
2.4 Ferramentas de Gerenciamento de Requisições de Mudança (FGRM) .	19
2.4.1 Extensões em FGRM	20

3	Mapeamento Sistemático da Literatura	23
3.1	Introdução	23
3.2	Metodologia de Pesquisa	25
3.2.1	Questões de Pesquisa	25
3.2.2	Pesquisa da Literatura	26
3.2.3	Esquemas de Classificação	27
3.3	Resultados	33
3.3.1	Frequência das Publicações	33
3.3.2	Extensões para Problemas na Manutenção de Software	33
3.3.3	Classificação por Melhorias com Suporte à Papéis da Manutenção de Software	43
3.3.4	Ferramentas Estendidas	47
3.4	Limitações e Ameaças à Validade	48
3.5	Trabalhos Relacionados	50
4	Caracterização das Ferramentas de Gerenciamento de Requisição de Mudança	51
4.1	Introdução	51
4.2	Objetivo do Capítulo	52
4.3	Metodologia	53
4.3.1	Seleção das Ferramentas	53
4.3.2	Inspeção da Documentação	56
4.3.3	Agrupamento das Funcionalidades	57
4.4	Resultados	57
4.4.1	Ferramentas Escolhidas	57
4.4.2	Categorização das Ferramentas	57
4.5	Discussão	60
4.6	Ameaças à Validade	60
4.7	Resumo do Capítulo	61
5	Pesquisa com Profissionais	63
5.1	Introdução	63
5.2	Objetivo da Pesquisa com Profissionais	65
5.3	Desenho e Metodologia da Pesquisa com Profissionais	66
5.3.1	Conceitos Básicos	66
5.3.2	Metodologia	68
5.3.3	População, Amostra e Respostas	75

5.4	Resultados	75
5.4.1	Perfil dos Participantes	76
5.5	Discussão	76
5.6	Ameças à Validade	76
5.7	Resumo do Capítulo	76
6	Extensões para Ferramentas de Gerenciamento de Requisição de Mudança	77
6.1	Introdução	77
6.2	Ameças à Validade	77
6.3	Resumo do Capítulo	77
7	Conclusão	79
	Referências Bibliográficas	81
	Apêndice A Instrumentos do Mapeamento Sistemático	97
	Apêndice B Lista de Ferramenta de Gerenciamento de Requisição de mudanças	99
	Apêndice C Formulário Aplicado para Seleção de Ferramentas	101

Notes

■ O objetivo de seção é introduzir ao leitor na disciplina de Manutenção de Software, em especial quanto aos tipos de manutenção descritos na literatura e a utilização de uma ferramenta para o seu gerenciamento	3
■ O objetivo desta seção é apresentar a motivação do estudo. Em síntese, tenta responder a seguinte pergunta: Por que dentro do contexto da manutenção de software estudar as Ferramentas de Gerenciamento de Requisição de Mudança é IMPORTANTE?	5
■ OBJETIVO: Apresentar o problema que esta dissertação pretende resolver. O problema deverá ser definido claramente ou deverão ser apresentadas provas da importância do mesmo dentro do escopo da Engenharia de Software . .	6
■ OBJETIVO: Apresentar de forma sucinta o trabalho realizado nesta dissertação com o objetivo de apresentar uma solução para o problema declarado na seção anterior.	10
■ A medida que o mapeamento for concluído irei adicionar os resultados obtidos.	10
■ Incluir os resultados da caracterização das ferramentas de forma sucinta. . .	10
■ Incluir os resultado de forma sucinta da pesquisa com profissionais	10
■ OBJETIVO: Em linhas gerais apresenta como o problema descrito na seção anterior foi resolvido mediante a apresentação da metodologia científica utilizada	10
■ Aguardar o andamento do estudo para melhor discutir as contribuições efetivas do mesmo	11
■ Vamos aguardar o desenvolvimento deste trabalho para definirmos a estrutura do texto.	11
■ Descreve melhor análise de impacto	18
■ Incluir alguma referência de trabalho que propõe atribuição automática de RM	21
■ Resultado parcial, devemos posteriormente reavaliar esta frase	21
■ Incluir motivação das questões de pesquisa	26

■ Verificar o processo com objetivo de validar o total de artigo em cada etapa do processo.	27
■ Incluir no anexo uma tabela demonstrando como as palavras-chaves foram combinadas	29
■ Verificar o total de artigos que utilizam algum técnica de IR. Apesar de grande parte utilizar não serão todos os artigos	31
■ Incluir citação para o artigo Impact analysis by mining e Identifying the start impact set of a	42
■ Inserir citação de Software defect association mining and defects e Minig software code repositories and bug database	42
■ Verificar os artigos com N/A e não especificado	43
■ reavaliar a quantidade de estudos	47
■ Inserir anexo com o formulário que representa os cartões	57
■ Incluir uma descrição para cada questão de pesquisas	65
■ Incluir definição de Unidade de Observação	66

Capítulo 1

Introdução

O objetivo de seção é introduzir ao leitor na disciplina de Manutenção de Software, em especial quanto aos tipos de manutenção descritos na literatura e a utilização de uma ferramenta para o seu gerenciamento

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 preço final [Kaur & Singh, 2015], as atividades relacionadas a manter e evoluir software têm sua importância considerada tanto pela comunidade científica quanto pela indústria.

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 valor necessário para manter o software [Herrin, 1985, Hirota et al., 1994]. Apesar da evolução das metodologias 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 1.1 onde é possível verificar a evolução dos gastos com manutenção de software como fração do preço final do produto.

Uma vez que o software entra em operação, anomalias são descobertas, mudanças ocorrem do ambiente de operação e novos requisitos são solicitados pelo usuário. Todas estas demandas devem ser solucionadas na fase de Manutenção que inicia com entrega do sistema, entretanto, alguns autores defendem que certas atividades iniciam bem antes da entrega do produto.

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].



Figura 1.1: Evolução da manutenção de software como percentual do custo total. Extraído de [Engelbertink & Vogt, 2010]

Verificamos na literatura uma discussão sobre a diferença entre manutenção e evolução de software. Percebe-se ainda que pesquisadores e profissionais utilizam evolução como o substituto preferido para manutenção [Bennett & Rajlich, 2000]. Todavia, não está no escopo desta dissertação discutir e apresentar as diferenças entre os conceitos. Neste sentido, utilizamos os termos *manter* e *evoluir* software de forma intercambiáveis.

As manutenções em software podem ser divididas em *Corretiva*, *Adaptativa*, *Perfeccionista* e *Preventiva* [Lientz & Swanson, 1980, IEEE, 1990]. A ISO 14764 discute os quatro tipos de manutenções e propõe que exista um elemento comum denominado *Requisição de Mudança* que representa as características comuns a todas aqueles tipos de manutenção.

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 *Ferramentas de Gerenciamento de Requisição de Mudança - FGMR*, 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. A literatura não define 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. 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 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 software que podem ser classificados como FGRM's. Também são listados serviços da Internet que oferecem funcionalidades presentes nas FGRM's na forma de Software como Serviço [Fox et al., 2013].

Ferramentas		Serviços da Internet	
Bugzilla	https://www.bugzilla.org/	SourceForge	https://sourceforge.net/
MantisBT	https://www.mantisbt.org/	Launchpad	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]

1.1 Motivação

O objetivo desta seção é apresentar a motivação do estudo. Em síntese, tenta responder a seguinte pergunta: Por que dentro do contexto da manutenção de software estudar as Ferramentas de Gerenciamento de Requisição de Mudança é IMPORTANTE?

Diante da maior presença de software em todos os setores da sociedade existe um interesse por parte da academia e da indústria no desenvolvimento de processos, técnicas e *ferramentas* que reduzam o esforço e o custo das tarefas de desenvolvimento e manutenção de software. Nesta 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 em alguns casos são necessários que que 60% dos desenvolvedores fique dedicados à tarefas de manutenção de sistemas [Zhang, 2003].

Em certos projetos de software, especialmente durante as etapas de desenvolvimento e teste, se faz necessário uma ferramenta para gerenciar as Requisições de Mudança por conta do volume e da grande quantidade de pessoas que necessitam de um local para inserir os erros encontrados [Serrano & Ciordia, 2005]. Este tipo de ferramenta vem sendo utilizada em projetos de código aberto (Apache, Linux, Open Office) bem como em organizações públicas e privadas (NASA, IBM).

Não obstante, alguns estudos demonstram que as FGRM's desempenham um papel além de gerenciar os pedidos de manutenção software. Avaliando o controle de demandas como um processo social, Bertram e outros [Bertram et al., 2010a] 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, de recursos ou pedidos de informação, mas também atua como um ponto focal para a comunicação e coordenação de 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.

No trabalho de Breu e outros [Breu et al., 2010a] 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 defeitos registrados em uma FGRM de dois projetos de software livre foi possível verificar que o uso da ferramenta propiciou que os usuários desempenhassem 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.

Um outro importante benefício da utilização das FGRM é que as mudanças no software podem ser rapidamente identificada e reportada para os desenvolvedores [Anvik et al., 2005]. Além disso, eles podem ajudar a estimar o custo do software, na análise de impacto, planejamento, rastreabilidade, descoberta do conhecimento [Cavalcanti et al., 2013].

Contudo, no escopo de utilização das FGRM's diversos desafios se apresentam: duplicação RM's, pedidos de modificação abertos inadvertidamente, grande volume de RM's que devem ser atribuídas aos desenvolvedores, erros descrito de forma incompleta, análise de impacto das RM's e RM's atribuídas de maneira incorreta [Cavalcanti et al., 2014]. Diante de tantos problemas e desafios é importante entender como estas ferramentas vêm sendo utilizadas bem como analisar o que está sendo proposta na literatura com objetivo de melhorar as funcionalidades oferecidas por elas.

1.2 Problema

OBJETIVO: Apresentar o problema que esta dissertação pretende resolver. O problema deverá ser definido claramente ou deverão ser apresentadas provas da importância do mesmo dentro do escopo da Engenharia de 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 FGRM's 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. Contudo, muitas daquelas ferramentas são meramente melhores interfaces para um banco de dados que armazena todos os bugs reportados [Zimmermann et al., 2009a].

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. 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 pode ser visto no trabalho proposto por Baysal & Holme [Baysal & Holmes, 2012] no qual desenvolvedores que utilizam o Bugzilla¹ relatam a dificuldade em manter uma compreensão global das RM's em que eles estão envolvidos. Segundo os participantes 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 problema que é potencializado pela ausência de certas funcionalidades nas FGRM são as RM's que acabam sendo relatadas de forma insatisfatória. Nesta situação os usuários acabam sendo questionados a inserir maiores detalhes que muitas vezes eles não tem conhecimento. Por outro lado, verifica-se uma frustração por parte dos desenvolvedores que acabam desapontados sobre a qualidade do que foi reportado [Just et al., 2008].

Com o objetivo de melhorar as FGRM's, que no contexto do trabalho recebem o nome de issue tracking system, Zimmermann e outros discute [Zimmermann et al., 2009a] quatro dimensões de melhorias deste tipo de ferramenta, conforme descrito a seguir e esquematizado na Figura 1.2:

Informação Estas melhorias focam diretamente na informação fornecida pelo reportador da RM. Com ajuda da FGRM o responsável por descreve um bug, por exemplo, poderia ser motivado a coletar mais informações sobre o problema. O sistema poderia verificar a validade e consistência daquilo que foi repassado pelo usuário.

Processo Melhorias com foco no processo visam dar suporte à administração de atividades relacionadas à solução de RM. Por exemplo, a triagem de RM, poderia ser automatizada visando acelerar o processo. Um outro exemplo de melhoria poderia ocorrer no aumento do entendimento do progresso realizado em cada RM ou

¹<https://www.bugzilla.org>

mesmo fornecer ao usuário afetado uma estimativa de em quanto tempo a sua requisição será solucionada.

Usuário Nesta dimensão estão incluídos tanto os usuário que relatam as RM's quanto os desenvolvedores responsável por solucioná-la. Os reportadores podem ser educados de qual informação fornecer e como coletá-la. Os desenvolvedores também podem beneficiar de um treinamento similar em qual informação esperar e como esta informação pode ser utilizada para solucionar a RM.

Ferramenta As melhorias centradas na ferramenta são realizadas nas funcionalidades fornecidas pelas FGRM. Elas podem reduzir a complexidade da coleta e fornecimento das informações necessárias para solucionar o RM. Por exemplo, as FGRM poderiam ser configuradas para automática localizar a pilha de erro (stack trace) e adicioná-la ao erro reportado. A ferramenta poderia simplificar o processo de reprodução do erro mediante a simplificação do processo de capturas de telas. Estes exemplos visam ajudar com a coleta das informações necessárias pelos desenvolvedores para corrigir o bug, por exemplo.

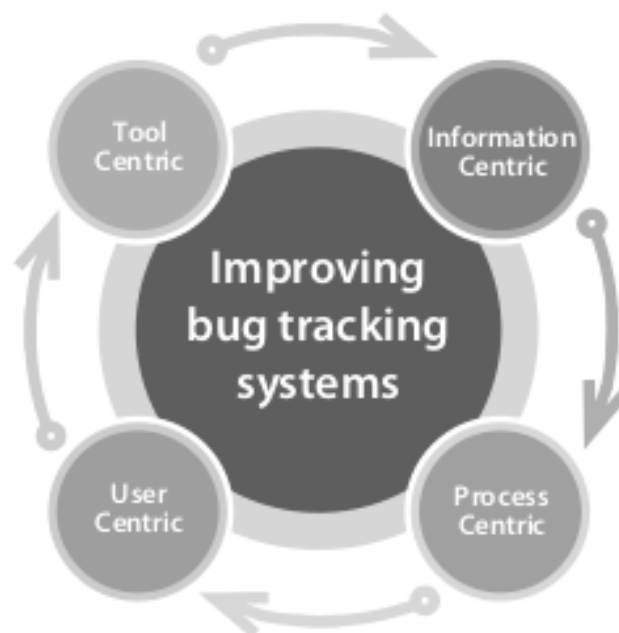


Figura 1.2: Dimensões de melhoria das FGRM's. Adaptado de [Zimmermann et al., 2005]

Neste estudo estamos especialmente interessados em analisar e propor melhorias relativas ao domínio da *Ferramenta*. Ao bem do nosso conhecimento é reduzido o

número de trabalhos que avaliem de forma sistemática as funcionalidades oferecidas pelas FGRM ao mesmo tempo que faça relação com que vêm sendo proposto na literatura sobre o assunto. De maneira similar número de estudo que avaliam a opinião dos profissionais envolvidos em manutenção de software sobre o que é ofertado pelas FGRM.

Além disso, os estudos anteriormente propostos não discutem o fato que 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, seria importante que ferramentas que dão suporte à manutenção, tal como as FGRM's, evoluíssem 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, o que pode ser observado considerando as diversas extensões (plugins) propostas na literatura [Rocha et al., 2015, Thung et al., 2014b, Kononenko et al., 2014].

1.3 Objetivos

Conforme exposto o distanciamento entre as necessidades dos profissionais envolvidos em manutenção de software e as funcionalidades oferecidas pelas FGRM resulta em diversos problemas. Neste contexto, este trabalho de dissertação investiga e contribui 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.

Neste contexto, elaboramos um estudo sobre as Ferramentas de Gerenciamento de Requisição de Mudança (FGRM) com os seguintes objetivos:

- (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 melhorias ou novas funcionalidades para as FGRM.

1.4 Visão Geral do Estudo

OBJETIVO: Apresentar de forma sucinta o trabalho realizado nesta dissertação com o objetivo de apresentar uma solução para o problema declarado na seção anterior.

A fim de alcançarmos os objetivos descritos na seção anterior, um conjunto de melhorias nas funcionalidades das FGRM's foi proposto. As melhorias resultaram de três estudos empíricos: um mapeamento sistemático da literatura, apresentado no Capítulo 3, uma caracterização das funcionalidades das FGRM, discutida no Capítulo ??; e uma pesquisa com profissionais, apresentada no Capítulo 5.

Mediante o mapeamento sistemático obtivemos e avaliamos o estado da arte sobre novas funcionalidades bem como melhorias no escopo das FGRM. A partir do estudo foi possível propor quatro esquemas de classificação: por tipo de problema, por suporte ao papel desempenhado na manutenção de software, por técnicas de Recuperação da Informação utilizada e por ferramenta estendida.

De maneira similar, através da caracterização das funcionalidade de algumas FGRM's código aberto ou disponíveis comercialmente e escolhidas mediante uma pesquisa com profissionais identificamos o estado da prática deste tipo de ferramenta.

Incluir os resultados da caracterização das ferramentas de forma sucinta.

Com base dois estudos anteriores conduzimos uma pesquisa com profissionais envolvidos em manutenção de software onde pedimos que avaliassem os requisitos funcionais e não funcionais que poderiam melhorar as FGRM já existentes. O questionário também quis saber a opinião dos profissionais sobre a relevância das propostas de melhorias existente na literatura em sua rotina de trabalho.

Incluir os resultado de forma sucinta da pesquisa com profissionais

1.5 Metodologia de Pesquisa

OBJETIVO: Em linhas gerais apresenta como o problema descrito na seção anterior foi resolvido mediante a apresentação da metodologia científica utilizada

A metodologia de pesquisa utilizada neste estudo é baseada em uma abordagem multi-método [Hesse-Biber, 2010]. Este tipo de desenho combina dois ou mais métodos quantitativo (ou qualitativo) em um único estudo. Um estudo que faça uso de um survey e um experimento é um exemplo deste tipo de enfoque [Hesse-Biber, 2010].

Alguns estudos descrevem quatro tipos de desenho em abordagens multi-método: embutido (embedded), exploratória, triangulada e explanatória [Creswell & Clark, 2007]. Neste estudo, utilizamos uma abordagem de triangulação no qual consolidamos os resultados de diferentes métodos, considerando,

A medida que o mapeamento for concluído irei adicionar os resultados obtidos.

contudo, que a mesma questão de pesquisa foi investigada em cada um deles. A utilização de um desenho triangular no trabalho melhora as conclusões e completude do estudo, trazendo maior credibilidade para os achados da pesquisa [Hesse-Biber, 2010].

As etapas do trabalho, que compõem a abordagem multi-método estão listadas a seguir:

- (i) Mapeamento Sistemático da Literatura [Petersen et al., 2008]
- (i) Caracterização das Ferramentas de Gerenciamento de Requisição de Mudança (FGRM)
- (i) Pesquisa (Survey) com os desenvolvedores [Wohlin et al., 2012]

1.6 Contribuições do Estudo

Aguardar o andamento do estudo para melhor discutir as contribuições efetivas do mesmo

1.7 Organização do Trabalho

Vamos aguardar o desenvolvimento deste trabalho para definirmos a estrutura do texto.

Capítulo 2

Manutenção de Software: Uma Visão Geral

Uma tendência natural do software é evoluir a fim de atender aos novos requisitos e alterações do 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). A primeira diz que 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 sobre o tema Yu & Mishra [Yu & Mishra, 2013] examinaram de forma empírica as Leis de Lehman em relação a evolução da qualidade do software. O estudo demonstrou que com base na métrica proposta que a qualidade de um produto de software declinará ao menos que uma reestruturação é realizada.

Conforme exposto, a mudança em um produto de software é inevitável. Desta forma, é importante a existência de uma área de estudo preocupada com o gerenciamento e controle destas mudanças. Dentro do escopo da Engenharia de Software esta tarefa fica a cargo da Manutenção de Software. Nas próximas seções discutimos os conceitos básicos que mostram onde e como a Manutenção se encaixa dentro da Engenharia de Software. São apresentados os conceitos que fazem da Manutenção de Software uma disciplina distinta.

2.1 Conceitos Fundamentais

Esta 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 suporte a um produto de software. Entretanto, encontramos na literatura outras definições mais elaboradas sobre a área.

Manutenção de Software é definida pela IEEE 1219 [ISO/IEEE, 1998]- Padrão para a Manutenção de Software, como a modificação de um produto de software após a sua entrega com o objetivo de corrigir falhas, melhorar o desempenho ou outros atributos com a finalidade de adaptar o software as modificações ambientais. O padrão cita a ocorrência de atividade de manutenção antes da entrega propriamente dita, contudo, de forma bastante superficial.

Posteriormente a IEEE/EIA 12207 - Padrão para o Processo de Ciclo de Vida do Software [?] retrata a manutenção como um dos principais processos no ciclo de vida do software. Em seu texto a manutenção é vista como atividade de modificação do código e da documentação associada e ocorre devido a algum problema ou necessidade de melhoria [Society et al., 2014]. Por outro lado a ISO/IEC 14764 - Padrão para Manutenção de Software [ISO/IEC, 2006] enfatiza aspectos pré-entrega da manutenção, como por exemplo o planejamento.

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]. A ISO/IEC 9126 - 01 [ISO/IEC, 2001] define a Manutenibilidade como uma característica de qualidade do processo de Manutenção.

Apesar das diversas definições para Manutenção de Software é possível identificar dois aspectos comuns: manter e evoluir. O conceito de evolução de software carece de uma definição padrão na literatura, contudo, pesquisadores e profissionais utilizam o termo como substituto preferido para manutenção [Bennett & Rajlich, 2000]. Embora exista o entendimento que os processos de manutenção e evolução possuem características distintas, não está no escopo desta dissertação discutir e apresentar tais diferenças. Neste sentido, utilizamos os termos *manter* e *evoluir* software de forma intercambiáveis. Caso em algum contexto haja a necessidade de diferenciação, ela será discutida.

A Manutenção é necessária para garantir que o software seja capaz de satisfazer os requisitos dos usuários. Neste sentido, a atividade de manter software pode ser vista como um desenvolvimento contínuo, sobretudo, pelo fato que alguns sistemas nunca estão completos e continuam a evoluir.

2.2 Requisição de Mudança

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 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. A Preventiva 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 propõe que exista um elemento comum denominado Requisição de Mudança que representa as características comuns a todas aqueles tipos de manutenção. A Figura 2.1 exibe a classificação das RM's conforme discutido pela ISO.

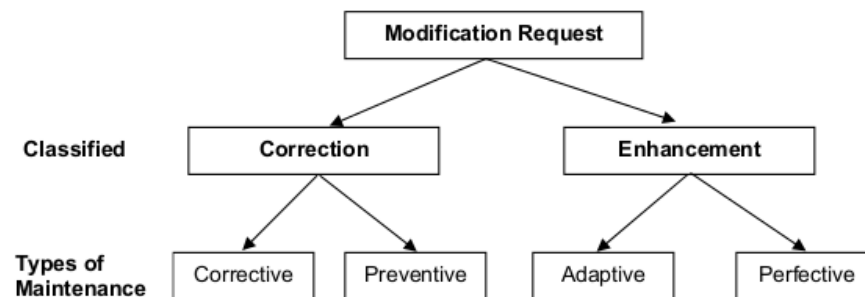


Figura 2.1: Tipos de manutenção segundo a norma ISO/IEC 14764. Extraído de [ISO/IEC, 2006]

A ISO/IEC 14764 classifica as manutenções adaptativas e perfectivas como melhorias e agrupa as manutenções corretivas e preventivas em uma única categoria de correção, conforme exibido na Tabela 2.1. A manutenção preventiva é mais frequentemente realizada em produtos de software onde atributos de segurança são mais críticos.

	Correção	Melhoria
Pró-ativa	Preventiva	Perfectiva
Reativa	Corretiva	Adaptativa

Tabela 2.1: Categorias da Requisição de Mudanças. Adaptado de SWE-BOK [442, 2004]

Em síntese, apesar das diferentes nomenclaturas existentes na literatura (demanda, bug, defeito, bilhete, tíquete, requisição de modificação, relato de problema) uma Requisição de Mudança representa o relato, independente de sua estrutura, que têm por objetivo gerar a manutenção ou evolução do software.

2.3 O processo de Manutenção de Software

Um Processo de Software é o conjunto de atividades, métodos, práticas e transformações utilizadas para desenvolvê-lo ou mantê-lo bem como seus artefatos associados [Paulk et al., 1993]. Independente do contexto em que a manutenção ocorra é importante que o processo esteja bem definido. Existe na literatura a proposição de alguns modelos do processo de manutenção de software, especialmente baseado em uma visão tradicional no qual desenvolvimento e manutenção possuem uma clara separação. Recentemente os métodos propostos pelos agilistas vêm sendo utilizados para manter software. Esta tendência surge da demanda crescente por serviços de manutenção com um retorno mais rápido para o usuário.

Nas próximas seções apresentamos alguns modelos encontrados na literatura na perspectiva tradicional, ao mesmo tempo descrevemos propostas do uso da metodologia dos agilistas na manutenção de software.

2.3.1 Manutenção de Software Tradicional

Em resumo, um processo de manutenção de software descreve as atividades e suas respectivas entradas e saídas. Alguns modelos são descritos nos padrões IEEE 1219 e ISO/IEC 14764. O processo especificado no Padrão para Manutenção de Software (IEEE- 1219) indica que as atividade de manutenção de software iniciem após a entrega do produto de software. O padrão também discute itens para fins de planejamento da manutenção. As atividades que compõe o processo são apresentas na Figura 2.2.

De maneira relacionada, na ISO/IEC 14764 as atividades que compõe o processo são similares aquelas propostas na IEEE- 1219, exceto pelo fato que elas são agregadas de uma forma diferente. O processo descrito na ISO/IEC 14764 são exibidas na Figura 2.3

As atividades de manutenção propostas na ISO/IEC 14764 são detalhadas em tarefas conforme apresentadas a seguir:

- Implementação do Processo
- Análise e Modificação do Problema
- Aceitação e Revisão da Manutenção
- Migração
- Aposentadoria do Software

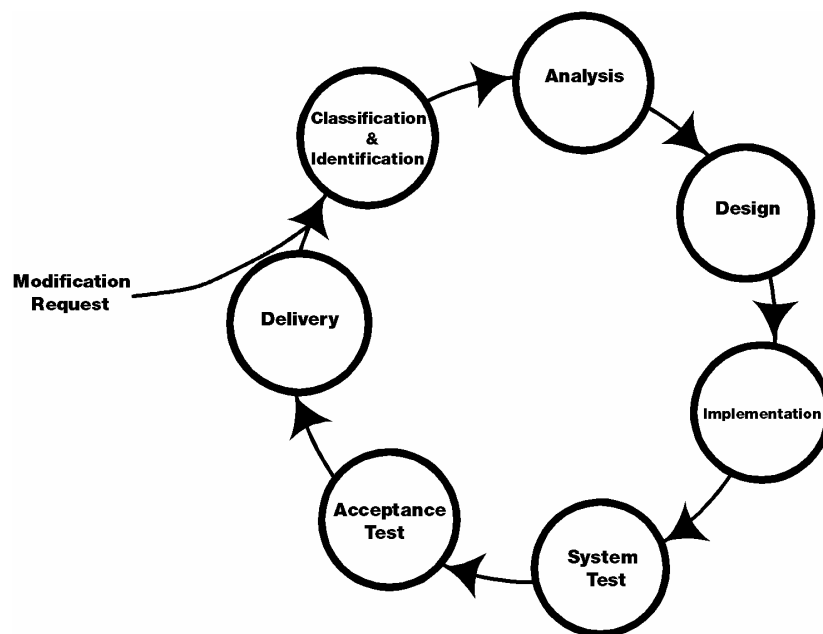


Figura 2.2: IEEE 1219 - Processo de Manutenção de Software

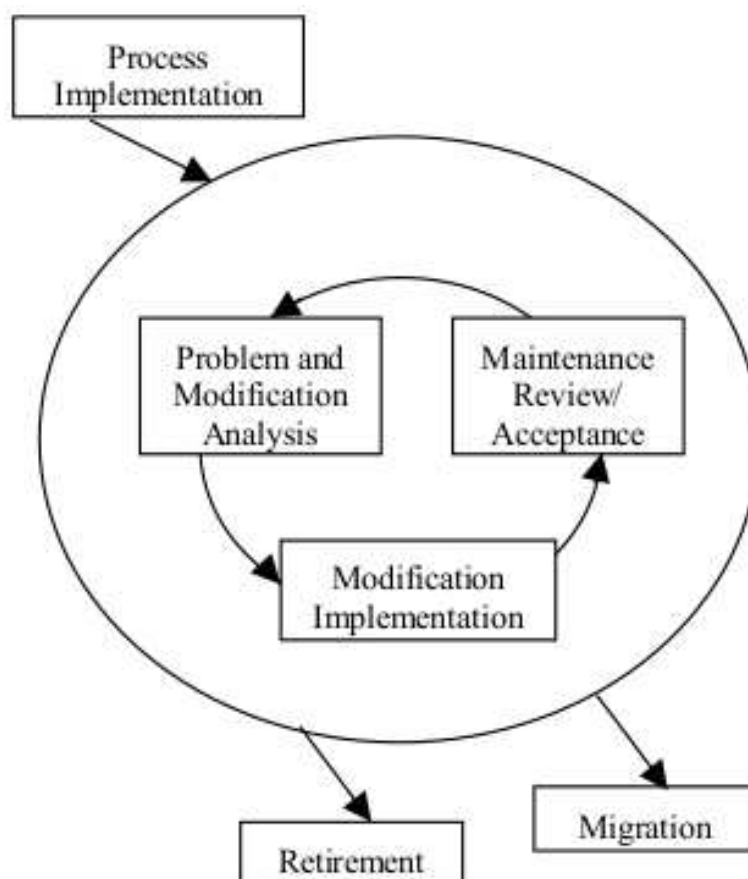


Figura 2.3: ISO/IEC 14764 Processo de Manutenção de Software

É possível notar que algumas atividades realizadas durante a manutenção de software são similares a outras presentes no desenvolvimento de software, como por exemplo, análise de desempenho, codificação, teste e documentação. Outra atividade comum à manter e desenvolver software é o gerenciamento dos requisitos. Nas duas situações os profissionais responsáveis por controlar os requisitos devem atualizar documentação por conta de alterações ocorridas no código fonte. Por outro lado certas atividades estão vinculadas apenas ao contexto da manutenção de software. Dentre elas podemos destacar:

Suporte ao Usuário Função de ajuda ao usuário final que podem disparar ações tais como avaliação e priorização das Requisições de Mudanças

Análise de Impacto

Suporte ao Uso de Software Ajuda aos usuários final com relação a determinada requisição de informação (regras de negócio, validação e requisição de relatório)

Acordo de Nível de Serviço Definição de um nível mínimo de qualidade que a atividade de manutenção deve ser realizada.

2.3.2 Manutenção de Software com Método dos Agilistas

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

Alguns resultados demonstram certa dificuldade para implantação da metodologia dos agilista na manutenção de software [Svensson & Host, 2005]. Um dos possíveis problemas é a necessidade de adequação das práticas da organização de modo que a se adequar as necessidades do time de desenvolvimento. Estudo apresentam resultados

relativos à melhorias no aprendizado e produtividade da equipe mediante o aumento da moral, encorajamento e confiança entre os desenvolvedores, o que propicia uma alta motivação durante o processo de manutenção de software [Choudhari & Suman, 2014].

2.4 Ferramentas de Gerenciamento de Requisições de Mudança (FGRM)

Dentro da disciplina de Gerenciamento da Configuração do Software a atividade de controle de configuração é responsável por gerenciar mudanças ocorridas durante o ciclo de vida de um produto de software. Tais ações incluem determinar quais alterações serão feitas, definir a autoridade responsável por autorizar certos tipos de mudança e aprovar desvios relativos aos requisitos iniciais do projeto [442, 2004]. De uma forma mais ou menos estruturada este tipo de processo ocorre em diferente tipos de projeto de software, seja ele dentro de um processo de manutenção tradicional ou mesmo naqueles que utilizam os métodos propostos pelos agilistas.

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.

No últimos anos alguns estudos discutem o fato que as FGRM's não apenas ajudam as organizações gerenciar, atribuir, controlar, resolver e arquivar Requisições de Mudança. Em alguns casos, este tipo de ferramenta se tornou o ponto focal para comunicação e coordenação para diversas partes interessadas, dentro e além da equipe de manutenção [Bertram et al., 2010a]. As FGRM's também servem como um repositório central para monitorar o progresso da RM, solicitar informações adicionais da pessoa

responsável por redigir a requisição e o ponto de discussão para potenciais soluções um bug [Zimmermann et al., 2009a].

Em projetos de código aberto, as FGRM são uma importante parte de como o time interage com comunidade de usuários. Como consequência é possível observar o fenômeno da participação dos usuários no processo de solução da RM: eles não apenas submetem a RM, mas também participam na discussão de como resolvê-la. Desta forma, o usuário final ajuda tomar as decisões sobre a direção futura do produto de software [Breu et al., 2010b].

Conforme exposto as FGRM desempenham um papel que vai além de gerenciar as Requisições de Mudança. Neste sentido, é importante estudar este tipo de software em busca de como melhorá-las de modo a atender as diversas necessidades dos seus usuários. Contudo, é importante avaliar as novas funcionalidades propostas na literatura ou ainda mesmo a melhoria das já existentes. Uma possível forma de melhoria é através do uso de extensões. Na próxima seção discutiremos esta propriedade de algumas FGRM's que permitem a inclusão e modificação de funcionalidades e comportamentos da ferramenta segundo as necessidades do usuário.

2.4.1 Extensões em FGRM

Em determinados domínios de aplicação é interessante desenvolver produtos de software com uma arquitetura que permita o sistema se adaptar às mudanças nos requisitos. Existe naturalmente a possibilidade de incluir as novas funcionalidades dentro das já existentes no software, todavia, verificamos que sistemas que permitem extensões apresentam os seguintes benefícios:

- Extensibilidade: o software pode se dinamicamente estendido mediante a inclusão de novas características.
- Desenvolvimento em Paralelo: Desde de que as funcionalidades podem ser implementadas como componentes separados, eles podem ser desenvolvidos em paralelo por times diferentes
- Simplicidade: uma extensão tipicamente tem uma única funcionalidade, desta forma os desenvolvedores possuem o único foco.

No escopo deste trabalho, uma extensão é um componente de software que adiciona uma característica ou comportamento específico para um programa de computador¹. Cabe-nos ressaltar que o nosso escopo de extensão inclui aquelas que não es-

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

tão acopladas ao código de determinada FGRM. Por exemplo, a funcionalidade de atribuição de uma requisição de mudança ao desenvolvedor é inerente às FGRM, segundo o nosso entendimento uma proposta de melhoria desta funcionalidade mediante uma atribuição automatizada, por exemplo, será analisada como extensão mesmo que ela não esteja efetivamente funcionando em alguma FGRM. O nosso foco é analisar as possíveis melhorias nas funcionalidades oferecidas pelas FGRM e não estamos especialmente interessados em analisar e discutir a facilidade que o conjunto destas ferramentas oferecem para a implementação de novas funcionalidades mediante uma extensão.

Incluir alguma referência de trabalho que propõe atribuição automática de RM

Verificamos na literatura alguns estudos em que as soluções propostas já se tornaram extensões de determinadas FGRM. Como pode ser observado no Mapeamento Sistemático realizado, a implementação da proposta do estudo em extensão de ferramenta não é o padrão observado.

A extensão *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. De maneira similar *NextBug* [Rocha et al., 2015] também é uma extensão para o Bugzilla que recomenda novos bugs para um desenvolvedor baseado no defeito que ele esteja tratando atualmente. Em ambos os casos a extensão foi implementada utilizando técnicas de Recuperação da Informação.

Os softwares que utilizam módulos de extensão têm aspectos de desenvolvimento e de manutenção potencialmente distintos daqueles sem esta característica. Este trabalho de mestrado faz uma contribuição na direção de uma melhor compreensão deste contexto a partir da análise de aspectos específicos das FGRM's.

Resultado parcial, devemos posteriormente reavaliar esta frase

Capítulo 3

Mapeamento Sistemático da Literatura

3.1 Introdução

Os sistemas de software possuem a tendência de evoluir ao longo do tempo e se adaptar ao seu ambiente de trabalhos. Devido a isso, o aprimoramento de software, a codificação e a adaptação a novos requisitos se tornam mais complexos. A literatura tradicional em Engenharia de Software (ES) descreve que as atividades de manutenção começam assim que os primeiros artefatos de software são construídos e entregue [Pigoski, 1996]. Além dessa dificuldade, a evolução do software é demorada, tediosa e muitas vezes compreende até 75% dos custos do desenvolvimento do software [Liu et al., 2012]. No entanto, as atividades relativas à manutenção de software não devem ser evitadas nem adiados caso a intenção é que um sistema de software permaneça útil por um longo tempo. Geralmente as modificações em software são solicitadas através de uma Requisição de Mudança (RM). Nos projetos de software as RM's são gerenciadas por um sistema de informação ao qual denominamos Ferramenta de Gerenciamento de Requisições de Mudança. Este tipo de ferramenta foram originalmente concebidos para acompanhar os pedidos de modificação, contudo, acabaram evoluindo para um ponto central onde as diversas partes interessadas, tais como gerentes, desenvolvedores e clientes coordenam atividades e se comunicam entre si [Bertram et al., 2010b] [8]. Neste contexto, este tipo de ferramenta desempenha um papel fundamental no processo de Manutenção de Software.

As FGRM's fornecem aos desenvolvedores o gerenciamento de correções de defeitos e aprimoramentos dos artefatos de software. Além disso, também permitem

a implementação de novos recursos, seja no desenvolvimento ou aprimoramento do software. Um outro benefício importante do uso deste tipo de ferramenta é que as mudanças podem ser rapidamente identificados e relatados aos desenvolvedores. Além disso, eles também podem ajudar com a estimativa do custo de desenvolvimento de software, análise de impacto, rastreabilidade, planejamento, descoberta de conhecimento e compreensão de software [Cavalcanti et al., 2013].

No entanto, todos estes benefícios não vêm de graça tendo em vista que uma série de questões surgem do gerenciamento das Requisições de Mudança: RM's duplicadas que são abertas inadvertidamente; o grande número de RM's que devem ser atribuídas aos desenvolvedores; RM's que não foram relatadas da forma mais correta; análise de impacto de determinada RM e etc [Cavalcanti et al., 2014]. Neste sentido, é importante analisar o estado da arte sobre as melhorias das funcionalidades das FGRM's para solucionar os problemas descritos anteriormente e diversos outros relacionados com a gestão das RM's.

Neste capítulo realizamos um Mapeamento Sistemático para identificar as pesquisas existentes no campo de melhorias das funcionalidades fornecidas pelas FGRM's. A partir de um conjunto de 64 artigos realizamos a classificação em quatro dimensões de melhoria conforme proposto por Zimmermann e outros [Zimmermann et al., 2009a]. Naquele estudo o foco foi melhorar as FGRM's visando aumentar a completude do que é relatado nas RM's. Especificamente, o trabalho consistiu em melhorar os sistemas de rastreamento de Bugs de quatro maneiras: foco na ferramenta; foco na informação; foco no processo; foco no usuário. Os estudos primários também foram classificados pelo papel desempenhado no processo de manutenção de software. A partir de um conjunto de papéis existentes no processo de manter de software [Polo et al., 1999b].

As contribuições deste estudo estão relacionadas com o fornecimento de uma visão abrangente sobre o estado da arte das melhorias das funcionalidades existentes ou a proposição de novas. Através deste estudo, os pesquisadores encontrarão lições aprendidas, questões abertas e compreender o caminho a seguir no que refere ao processo de investigar FGRM's. Por outra, os profissionais devotados com desenvolvimento e manutenção de software poderão encontrar técnicas para o seu trabalho diário, que podem ser utilizados para melhorar seus próprios processos e ferramentas.

O restante deste trabalho está organizado da seguinte forma. Na Seção 3.2 descrevemos as questões de pesquisa, apresentamos a metodologia de pesquisa e como os estudos classificados são explicados. Na Seção 4.4 relatamos os resultados do estudo com relação às dimensões de melhoria e o papéis na manutenção de software. Em seguida, na Seção 4.5, discutimos os principais achados e as questões em aberto sobre

melhorias nas funcionalidades das FGRM's, bem como os aspectos relacionados ao estado da arte. As ameaças à validade e ao trabalho relacionado são discutidas nas Seções ?? e 3.5, respectivamente. Um resumo do capítulo é realizado na Seção ??.

3.2 Metodologia de Pesquisa

Um *Mapeamento Sistemático da Literatura*, também conhecido como Estudo de Escopo (Scoping Studies), tem como seus 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]. A pesquisa e a prática baseadas em evidências foram desenvolvidas inicialmente na medicina porque a pesquisa indicou que a opinião de especialistas baseada em conselhos médicos não era tão confiável quanto o conselho baseado na acumulação de resultados de experimentos científicos. Na caso da Engenharia de Software objetivo deste tipo de trabalho é fornecer os meios pelos quais as melhores evidências atuais da pesquisa podem ser integradas com a experiência prática e os valores humanos no processo de tomada de decisão relativo ao desenvolvimento e manutenção de software.

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 computador¹.

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

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 papéis 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 2010²
 - Artigos escritos em língua inglesa e portuguesa
 - Artigos disponíveis com texto completo
- Critérios de Exclusão
 - 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

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

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

3.2.3 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

³<https://www.jabref.org/>

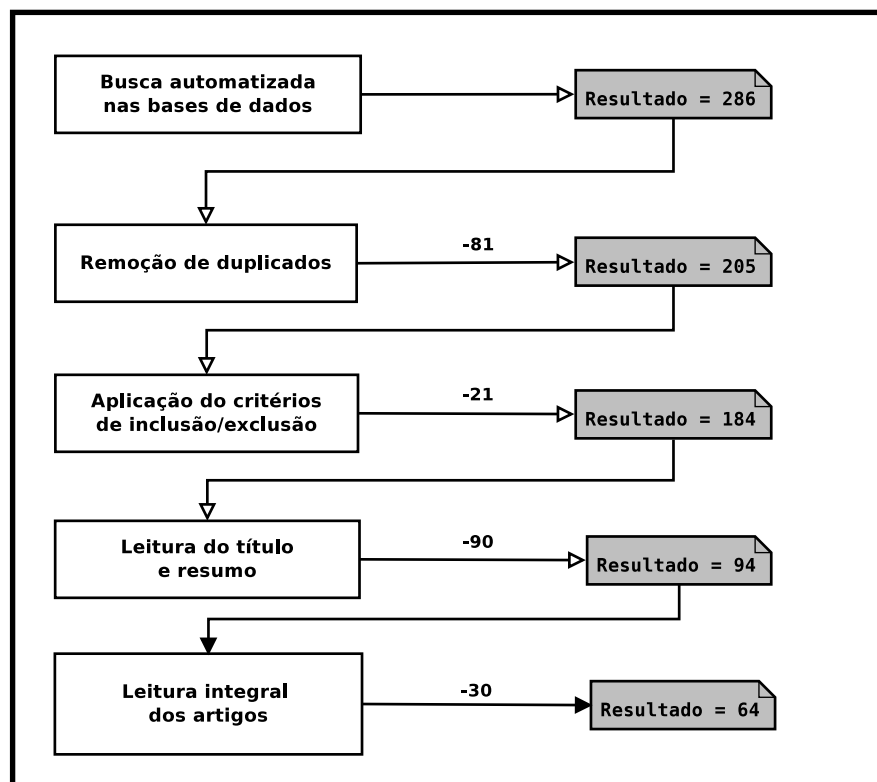


Figura 3.1: Número de artigos incluídos durante o processo de seleção dos estudos. Baseado em [Petersen et al., 2015]

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

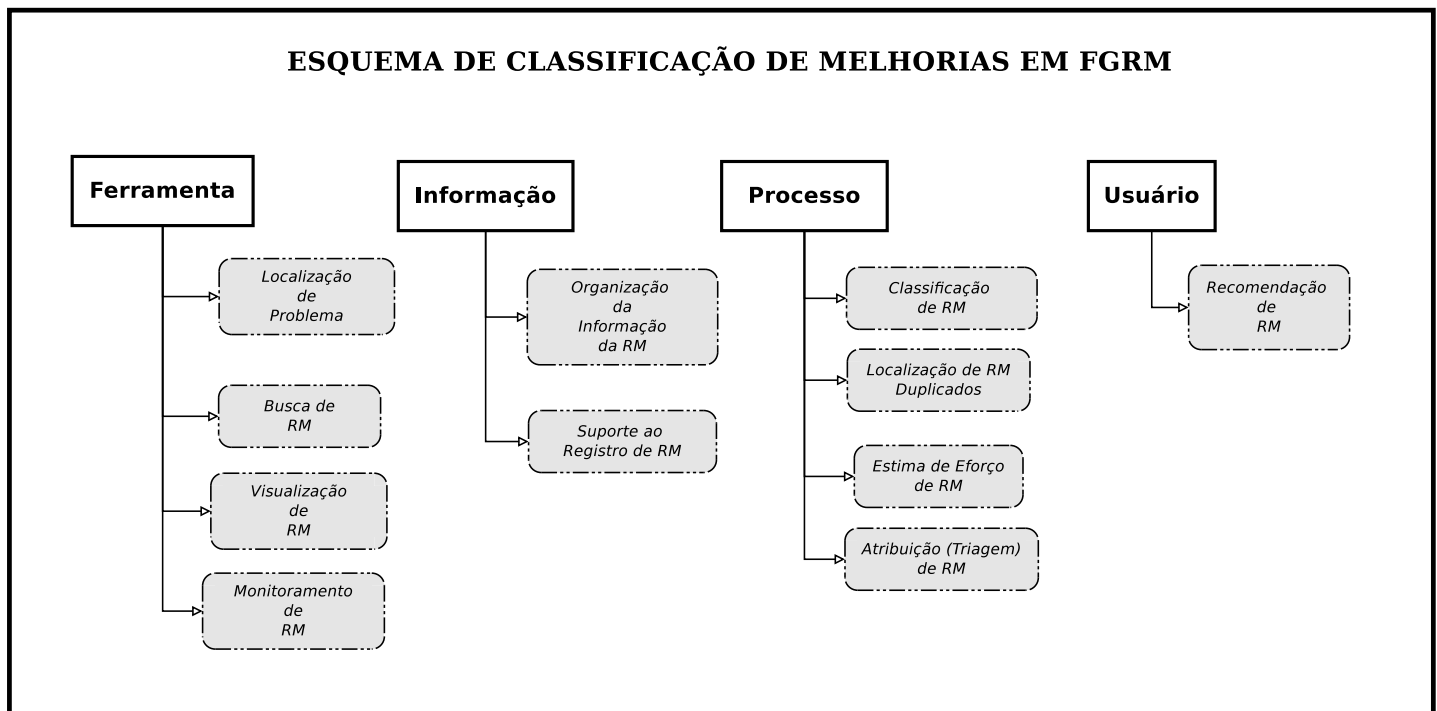


Figura 3.2: Esquema de classificação das melhorias propostas na literatura. Os retângulos representam as dimensões de melhorias e os polígonos de cantos arredondados representam as melhorias.

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 análise 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 3.5.

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 papéis 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 papéis a extensão pretender dar suporte, ou seja, relatar se existem papéis 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 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 o total de artigos que utilizam alguma 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.

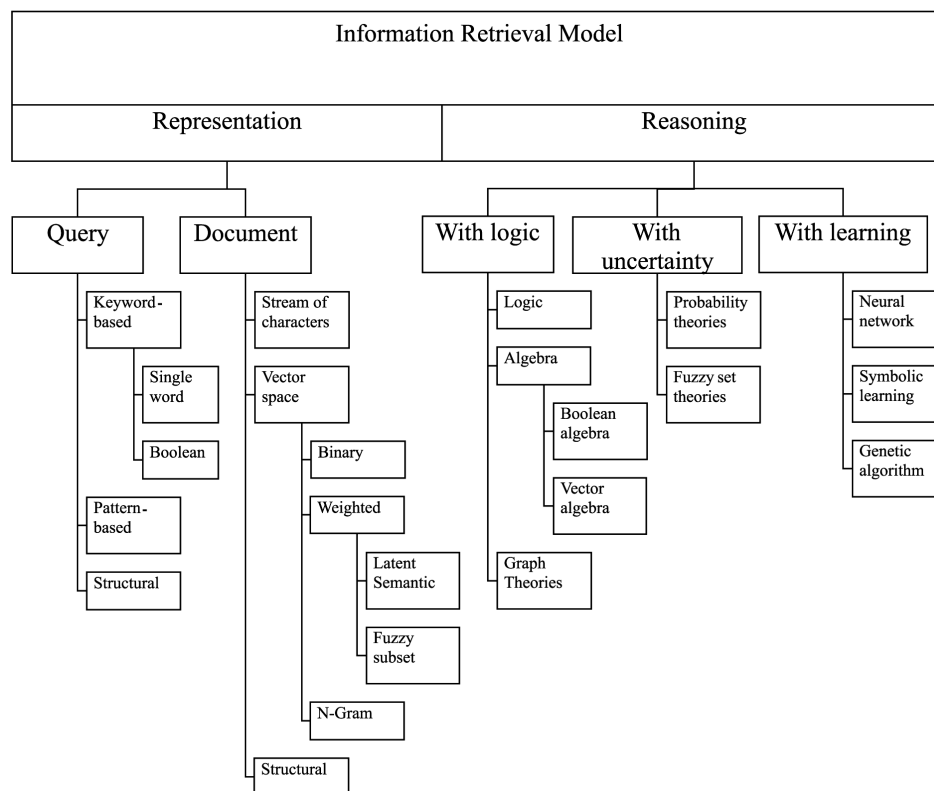


Figura 3.3: Taxonomia vertical para Modelos de Recuperação da Informação. Adaptado de [Cerulo & Canfora, 2004]

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 realizamos 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 exhibe 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 com uma análise da frequência de publicação relativo à melhoria das funcionalidades das ferramentas. Posteriormente apresentamos os resultados pela classificação por problemas na Manutenção de Software, o qual dividimos os estudos por área e tópico de pesquisa. Seguimos com a análise dos estudos 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 Frequência das Publicações

A Figura 3.4 exhibe o número de estudos primários identificados entre os anos 2010@-@2016, período de referência utilizado no mapeamento. Dentre os estudos escolhidos no período em questão verificamos que em 2010 foram publicados cinco estudos sobre o assunto [Sun et al., 2010, Gegick et al., 2010, Song et al., 2010a, Nagwani & Verma, 2010a, Zimmermann et al., 2010]. Posteriormente verificamos um acréscimo no número de estudos no qual um significativo aumento pode ser observado entre os anos de 2012@-@2014. O crescente número de estudos publicados sobre melhorias nas FGRM pode indicar que esta área vem sendo considerada altamente relevante pela comunidade acadêmica em Engenharia de Software.

3.3.2 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 melhorias nas funcionalidades das FGRM estão sendo proposta na literatura.

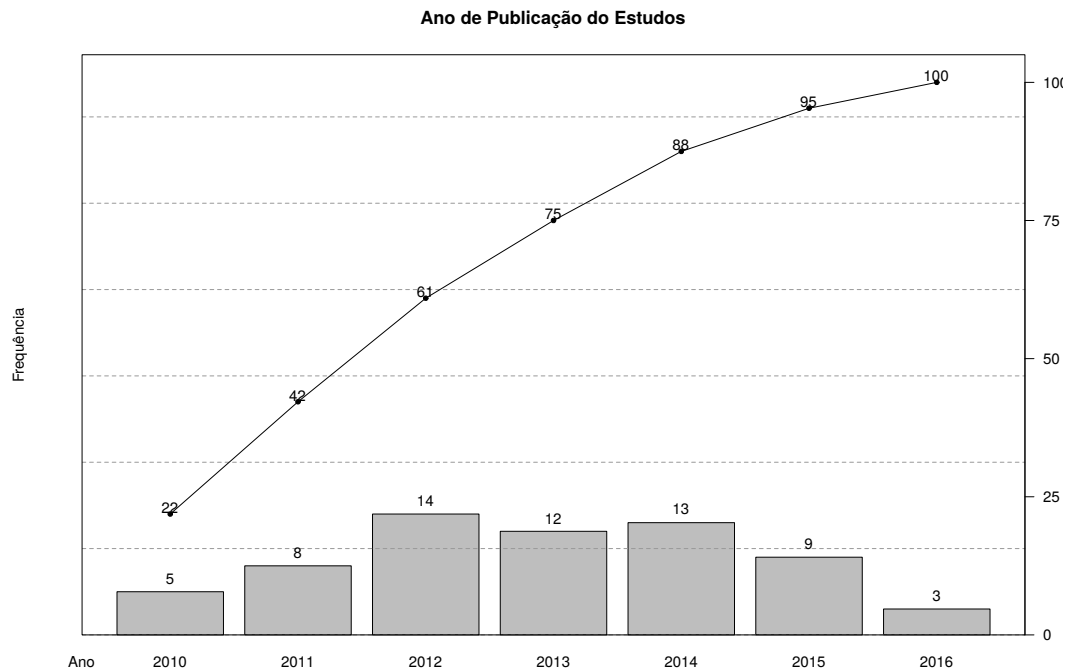


Figura 3.4: Número de estudos primários por ano de publicação.

Com base no mapeamento sistemático realizado e nas dimensões de melhoria discutidas por Zimmermann e outros [Zimmermann et al., 2009b] construímos o esquema de classificação dos apresentado na Figura 3.5

Adicionalmente a tabela 3.2 exibe a distribuição dos estudos pela dimensão de melhoria e o seu respetivo tópico. De uma certa maneira o número de estudos pode indicar a maturidade da pesquisa naquele tipo de assunto. Contudo, não representa diretamente a importância do tópico no estudo sobre melhorias em FGRM's.

3.3.2.1 Melhorias Propostas na Dimensão Ferramenta

Este ramo do esquema de classificação de classificação proposto foi criada para agrupar os estudos que discutem melhorias na dimensão Ferramenta. Conforme anteriormente exposto, esta classe acomoda estudos em tópicos como Localização do Problema e Visualização de RM's.

Localização do Problema Os estudos incluídos neste tópico de classificação estão devotados ao problema de localizar a origem de um problema de software com base dos dados da RM. Trata-se do processo de estatisticamente localizar um bug utilizando os dados das RM's em conjunto com o código fonte [Hovemeyer & Pugh, 2004].

Tabela 3.2: Lista de artigos de acordo com o esquema de classificação

Dimensão de Melhoria	Tópico	Estudos	Total
Processo	Localização de RM Duplicados	[Alipour et al., 2013, Banerjee et al., 2012, Hindle et al., 2016, Koopaei & Hamou-Lhadj, 2015] [White et al., 2015a, Prifti et al., 2011, Song et al., 2010b, Sun et al., 2010, Sun et al., 2011] [Sun et al., 2011, Thung et al., 2014a, Tian et al., 2012a, Tomašev et al., 2013, Lerch & Mezini, 2013]	13
Processo	Atribuição (Triagem) de RM	[Banitaan & Alenezi, 2013, Hosseini et al., 2012, Hu et al., 2014, Naguib et al., 2013] [Nagwani & Verma, 2012, Shokripour et al., 2012, Tian et al., 2015, Valdivia Garcia & Shihab, 2014] [Wu et al., 2011, Xuan et al., 2012, Zanetti et al., 2013, Zhang et al., 2014]	12
Processo	Classificação de RM	[Behl et al., 2014, Chawla & Singh, 2015, Gegick et al., 2010, Izquierdo et al., 2015] [Kochhar et al., 2014, Nagwani et al., 2013, Netto et al., 2010] [Somasundaram & Murphy, 2012, Tian et al., 2013, Zhang & Lee, 2011]	10
Ferramenta	Localização do Problema	[Bangcharoensap et al., 2012, Corley et al., 2011, Nguyen et al., 2012] [Thung et al., 2014b, Wong et al., 2014] [Romo & Capiluppi, 2015, Thung et al., 2013]	7
Informação	Suporte ao Registro da RM	[Bettenburg et al., 2008a, Correa et al., 2013, Moran et al., 2015, Moran, 2015] [Tu & Zhang, 2014, White et al., 2015b, Kaiser & Passonneau, 2011]	7
Processo	Estima de Esforço da RM	[Bhattacharya & Neamtiu, 2011, Nagwani & Verma, 2010b, Thung et al., 2012b] [Vijayakumar & Bhuvaneswari, 2014, Xia et al., 2015]	5
Ferramenta	Visualização de RM	[Dal Sasse & Lanza, 2013, Dal Sasso & Lanza, 2014, Hora et al., 2012, Takama & Kurosawa, 2013]	4
Informação	Organização da Informação da RM	[Mani et al., 2012, Ootom et al., 2016]	2
Usuário	Recomendação de RM	[Malheiros et al., 2012, Wang & Sarma, 2011]	2
Ferramenta	Busca de RM	[Liu & Tan, 2014]	1
Ferramenta	Monitoramento de RM	[Aggarwal et al., 2014]	1

ESQUEMA DE CLASSIFICAÇÃO DE MELHORIAS EM FGRM

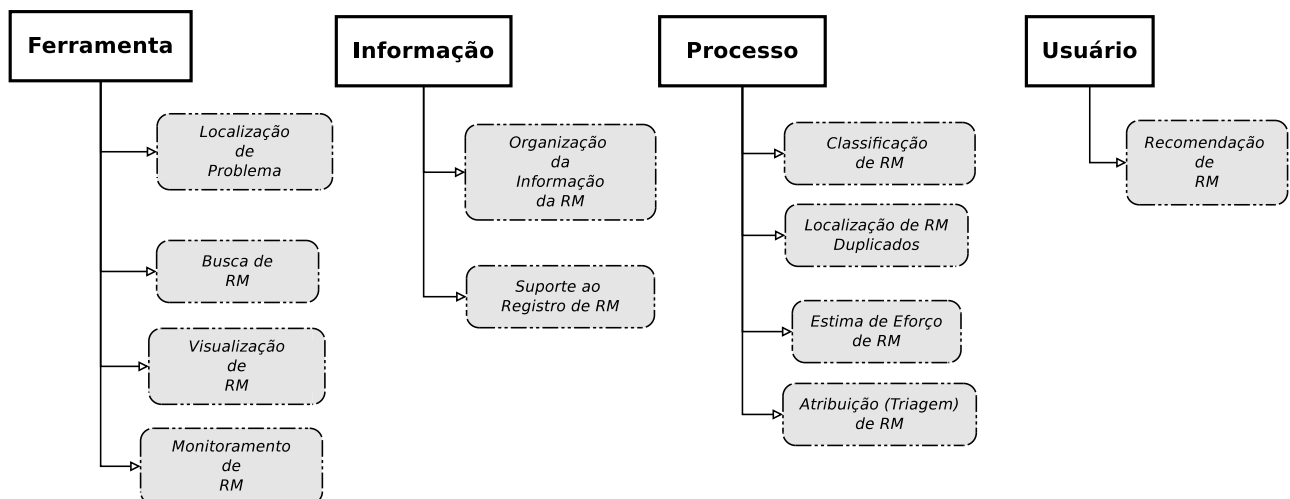


Figura 3.5: Esquema de classificação das melhorias propostas na literatura. Os retângulos representam as dimensões de melhorias e os polígonos de cantos arredondados representam as melhorias.

A tarefa de encontrar a origem de um falha de software é complexa de consome muito tempo. Em estudo Lucia e outros relataram que entre 84 a 93% dos problemas em software afetam apenas 1 - 2 arquivos de código-fonte [Thung et al., 2012a]. Contudo não é fácil identificar esses poucos arquivos dos milhares de arquivos de código-fonte. Esta situação realça que localizar a origem de um problema (buggy files) é uma tarefa árdua [Thung et al., 2014b].

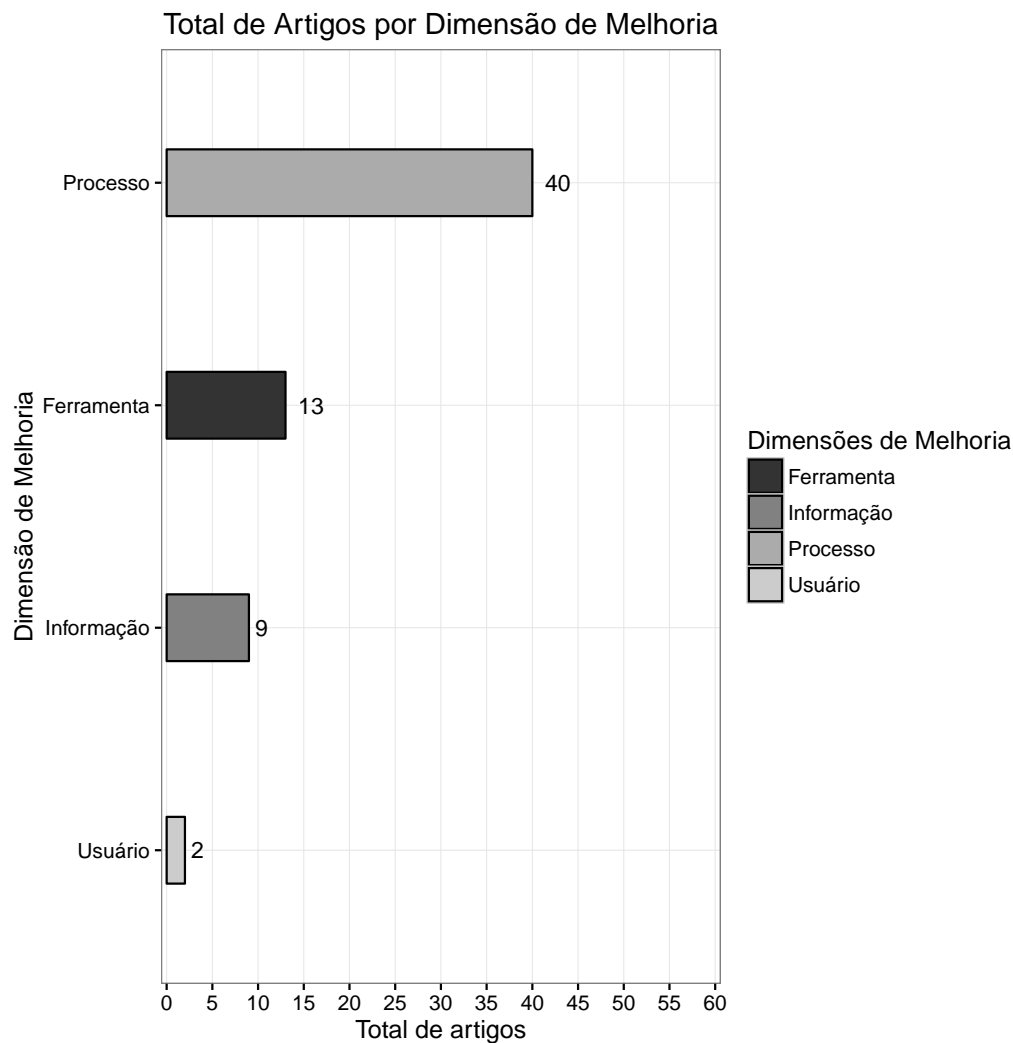


Figura 3.6: Total de artigos por dimensão de melhoria

Neste contexto, pesquisadores vem propondo abordagens baseadas em Recuperação da Informação para localizar falhas com base no que está descritos nos relatos de bugs. Nessas abordagens existe a tentativa de encontrar entre o código fonte do sistema um subconjunto de arquivos de fonte que estão diretamente relacionados à solução do problema reportado [Wong et al., 2014].

Com objetivo de melhorar a eficiência da Localização do Problema diversas informações contidas nas RM's estão sendo utilizadas. As diversas abordagens propostas utilizam informações como pilha-erros (stack-trace) [Wong et al., 2014], descrição e campos estruturados das RM's [Thung et al., 2014b], o históricos de versões [Bangcharoensap et al., 2012, Corley et al., 2011, Romo & Capiluppi, 2015].

Alguns dos estudos propostos foram incluídos em ferramentas largamente utilizadas no mercado utilizando as propriedades de extensão do soft-

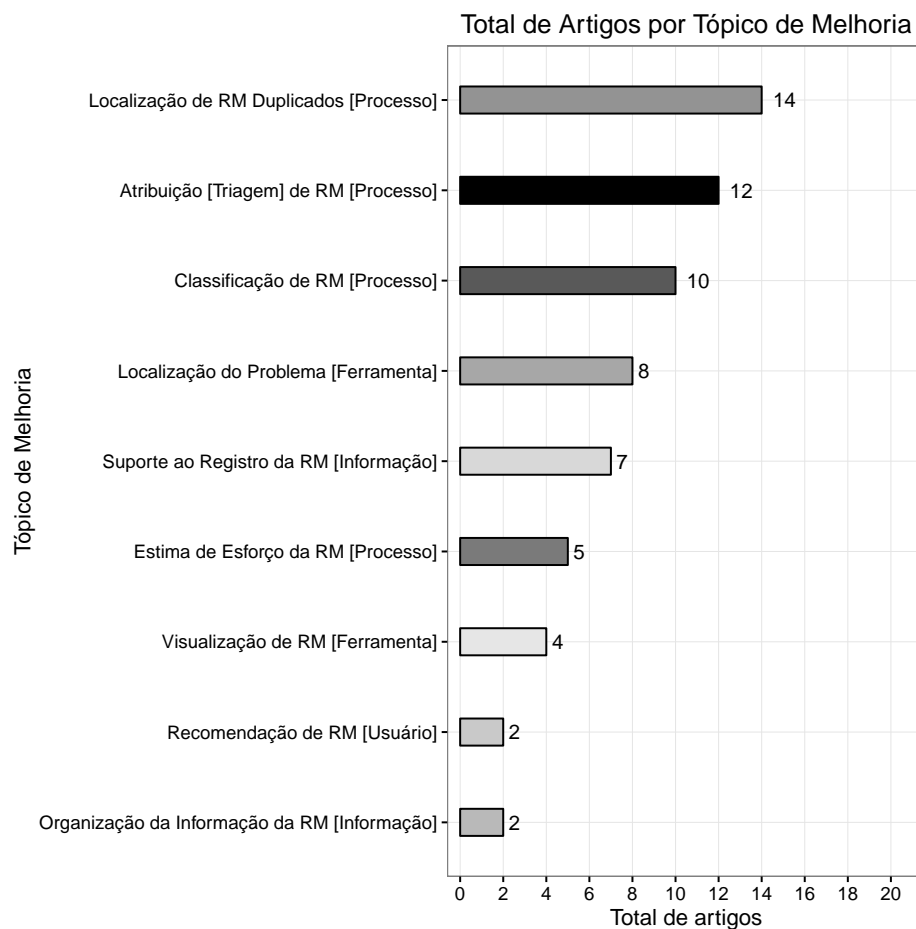


Figura 3.7: Total de artigos por tópico de melhoria

ware [Thung et al., 2014b, Corley et al., 2011]. Os autores argumentam que pelo fato de nenhuma das técnicas propostas na literatura estarem integradas às FGRM existe uma dificuldade de adoção da prática da localização automática de problemas pelos desenvolvedores. Neste sentido, eles argumentam sobre a necessidade da melhoria das funcionalidades das FGRM em especial com melhoria ou inclusão de funcionalidades nas FGRM's utilizando o que vem sendo proposto na literatura.

Visualização de RM Os estudos classificados neste tópicos propõe melhorar a visualização das informações contidas em uma RM. A tomada de decisão deve estar subsidiada por informações corretas. Este fato não é diferente na manutenção e desenvolvimento de software. Contudo, uma fonte de informação recebeu menos atenção do que o código fonte: os bugs no sistema. Pouco se sabe sobre o comportamento evolutivo, o tempo de vida, distribuição e estabilidade dos problemas reportados nas FGRM [Hora et al., 2012]. Este problema é reforçado pela forma como as FGRM's armazenam os dados das RM's. Em geral, esses exibem informações sobre as RM's de

forma textual, o que não é apenas complicado para para navegar, mas também dificulta a compreensão das intrincadas peças de informação que giram em torno dos problemas de software [Dal Sasso & Lanza, 2014].

Com o objetivos de apresentar novas forma de visualizar os dados de uma RM novos conceitos estão sendo propostos. No estudo de Hora e outros [Hora et al., 2012] é apresentado o conceito de Mapa de Bugs (Bugs Maps) que é a ligação de uma RM com diversos outros artefatos de software como por exemplo o histórico de versões. No estudo de Lanza e Dal Sasso [Dal Sasso & Lanza, 2014] o conceito de hiperligação entre documentos é utilizado para permitir a navegação entre os diversões artefatos que estão relacionados a um RM.

Por outro lado, verificamos o artigo proposto por Takama e Kurosawa [Takama & Kurosawa, 2013] onde a aplicação de tecnologias de visualização de informação foi empregada para o monitoramento das informações contidas nas RM's. A atualização das informações de uma RM quando gerenciada por uma FGRM são estruturadas como um fluxo de texto. Contudo, é difícil para as partes interessadas monitorar uma RM o tempo todo. Neste contexto, a solução proposta pelo autores visa suportar o monitoramento das RM apresentando ao usuário mediante animações as atualizações ocorridas em determinada RM.

Por conta da natureza das melhorias propostas neste tópico de pesquisa, verificamos que diversos estudos foram prototipados em ferramentas. Desta forma, é possível avaliar as propostas através das ferramentas como o bugMaps [Hora et al., 2012], In* Bug [Dal Sasso & Lanza, 2014].

3.3.2.2 Melhorias Propostas na Dimensão Informação

Neste ramo do esquema de classificação foram acomodados os estudos que se propõe em melhorar a informação que as partes interessadas registram em uma RM. Ela é composta de dois principais tópicos: no primeiro temos o conjunto de funcionalidades que dão suporte ao registro de uma RM antes que ela seja armazenada na base de dados de uma FGRM; o segundo tópico se dedica aos artigos que visam organizar a informação já registrada em uma RM, de modo a facilitar o entendimento por parte dos desenvolvedores e demais profissionais envolvidos na manutenção de software.

Suporte ao Registro da RM Durante o processo de correção de um problema de software verifica-se que a reprodução manual dos bugs é demorada e tediosa. Os mantenedores de software rotineiramente tentam reproduzir problemas não confirmados usando as informações contidas nas RM's que muitas vezes estão incomple-

tos [White et al., 2015b]. Para complementar os dados necessários à resolução do problema o desenvolvedor deve solicitar ao responsável pelo relato da RM as informações necessárias. Os relatos contidos nas RM's podem conter informações valiosas que podem ser utilizadas para melhorar a qualidade da informação contidas em novos relatórios de problemas de software. Esta melhoria da qualidade pode implicar na redução do custo do processo de garantia de qualidade bem como aumentar a confiabilidade a confiabilidade do software com a redução gradativa de bugs [Tu & Zhang, 2014].

A pesquisa visando a melhoria da qualidade da informação fornecida nas RM's começa com estudos visando mensurar de alguma forma os relatos realizadas pelos usuários. A determinação do que seria um bom relato de um problema de software foi obtido mediante uma pesquisa (survey) com profissionais devotadas à manutenção de software [Bettenburg et al., 2008a]. Em outro estudo os autores propõem as métricas que posteriormente serão utilizadas para avaliar o relato que compõe a RM [Tu & Zhang, 2014].

Um segundo nicho de estudos nesta área está relacionado ao suporte na reprodução do erro do software. Estes estudos incluem tanto em registrar o conjunto de ações que resultaram no erro [White et al., 2015b], quanto em autocompletar as informações que compõe o relato do problema [Moran et al., 2015]. Um ponto em comum deste dois estudos é que eles foram desenvolvido para o ambiente de desenvolvimento móvel, especial para o sistema Android ⁴. Uma possível justificativa para este foco em aplicações móveis pode estar relacionado à inerente dificuldade em registrar um problema de software naquele ambiente de software. Neste sentido, a fim de ajudar aos mantenedores a obter relatos de maior qualidade a melhoria da usabilidade na forma de reportar os defeitos é importante.

Muitos dos estudos realizados resultaram em ferramentas com a finalidade de realizar uma prova de conceito no tocante a dar suporte ao usuário em fornecer um relato de boa qualidade [Tu & Zhang, 2014, Bettenburg et al., 2008a, Kaiser & Passonneau, 2011, White et al., 2015b, Moran et al., 2015]. Conforme anteriormente exposto, o desenvolvimento deste protótipos é importante para fomentar a pesquisa sobre a melhorias das FGRM.

Organização da Informação da RM Em alguns casos não é possível aumentar a qualidade da informação fornecida em um relato de uma RM antes que ela seja armazenada em seu respectivo repositório. Nestas situações uma abordagem adotada é organizar de uma maneira previamente definidas as informações contidas em uma RM.

⁴<https://www.android.com/>

Durante o processo de análise de uma RM, em especial para aquelas de caráter corretiva, existe a tendência dos desenvolvedores em procurarem por problemas semelhantes que foram resolvidos no passado. No entanto, em diversas situações o desenvolvedor ainda precisa examinar manualmente o conteúdo dos bugs recomendados que podem variar em tamanho e complexidade [Mani et al., 2012]. Neste contexto, o resumo (sumarização) automático de RM's que tenham relação com problema em análise é uma maneira de reduzir a quantidade de dados que um desenvolvedor pode precisar analisar. A ferramenta denominada AUSUM [Mani et al., 2012] propõe uma abordagem, utilizando técnicas não supervisionadas de IR, de criar este resumo automático de um conjunto de relato de problemas.

3.3.2.3 Melhorias Propostas na Dimensão Processo

Localização de RM Duplicadas O processo de localização de RM's Duplicadas consiste em avaliar se determinado relato já foi realizado em algum outro momento. Quando uma RM duplicada é identificada ele deve ser vinculada a outra que na literatura da área é denominada como RM Mestre. Geralmente a Mestre é aquela que foi incluída no repositório de erros em data anterior. Alguns estudos revelam que entre 10% e 30% das RM's podem ser classificadas com duplicatas, o que causa um substancial impacto nas atividades de manutenção de software [Anvik et al., 2005, Cavalcanti et al., 2013, Runeson et al., 2007]. Por conta do grande número de RM's duplicadas se faz necessário designar pessoas para manualmente analisar as Requisições de Mudanças que chegam com objetivo de evitar que as duplicatas cheguem aos desenvolvedores [Anvik et al., 2005].

O processo de identificação de RM's duplicadas requer um prévio conhecimento do conjunto de relatos existentes previamente no projeto ou mediante a busca manual em toda base de dados da FGRM [Banerjee et al., 2012, Lerch & Mezini, 2013, Hindle et al., 2016]. Ambas as estratégias consomem tempo e não garantem que falso positivos possam ocorrer [Kaushik & Tahvildari, 2012]. Os falso positivos podem ainda acarretar em caso de relevantes problemas do software relatados pelo usuários serem desconsideradas.

A abordagem adotada da literatura para tratar o problema das RM's duplicadas podem ser divididas em dois tipos [Kaushik & Tahvildari, 2012, Tian et al., 2012b]: *(i)* remoção de duplicatas, *ii* identificação de duplicatas, que agrupa grande parte da pesquisa no assunto [Cavalcanti et al., 2014]. No primeiro tipo, o objetivo é evitar que RM duplicadas entrem na base de dados de uma FGRM e desta forma evitar o esforço e o tempo extra necessário para identificá-la posteriormente.

Por outro lado, o segundo tipo não se importa se duplicados entram na base de dado de RM's. Não obstante, o objetivo é sugerir uma lista de possíveis duplicatas durante o processo de registro de uma nova RM e possivelmente agrupá-los. Um ponto importante para se ressaltar é que este segundo tipo se baseia na premissa que registrar um mesmo problema por mais de uma vez nem sempre é problemático tendo em vista que pode fornecer informações adicionais que podem ser úteis [Bettenburg et al., 2008b]. No entanto, alguns estudos compensam o custo de sua recuperação e permanência na base de dados de RM's [Davidson et al., 2011]. Neste sentido, é importante que novas abordagens tentem equilibrar este dois tipos de abordagem de modo a evitar o tempo extra para análise de uma RM bem como apoiar os desenvolvedores com informações adicionais [Lerch & Mezini, 2013, Thung et al., 2014a].

Atribuição (Triagem) de RM O processo de atribuição de RM, também conhecido como *triagem*, possui como principal objetivo encontrar o desenvolvedor mais capacitado para manipular uma dada RM [Cavalcanti et al., 2014]. Existe a premissa que a escolha do desenvolvedor apropriado é crucial para obter um menor tempo para resolução de determinada RM [Di Lucca et al., 2002]. Estudos também discutem que o processo de atribuição deve considerar fatores tais como a carga de trabalho do desenvolvedor a prioridade da RM [Aljarah et al., 2011].

Classificação da RM Independentemente do tipo e tamanho de um projeto é sempre importante determinar qual tipo manutenção deverá ser realizada tomando como base o relato de uma RM. Este processo consiste de forma resumida em classificar uma requisição com base em algum esquema de classificação previamente definido. Este esquema pode ser classificar em *corretiva*, *perfectiva*, *preventiva* ou *adaptativa* [442, 2004] ou mesmo como *resolver um problema de software* ou *implementar uma melhoria*. A diversa de categorias em determinado esquema de classificação pode tornar complexo a tarefa, tendo em vista que em muitos casos não é fácil determinar os limites entre os tipos [Antoniol et al., 2008]. Por exemplo, a uma classificação incorreta de um defeito como melhoria pode acarretar em atrasos no projeto ou mesmo fazendo que uma RM receba menos atenção do que ela mereça [Cavalcanti et al., 2014].

Estimativa de Esforço da RM A gestão do custo e esforço de um projeto de manutenção de software passa pelo controle do esforço necessário ao cumprimento de suas RM's. Os estudos que medem o esforço de uma RM utilizam em geral três forma para medir este esforço [Cavalcanti et al., 2014]: determinar o tempo para solucionar

novas RM's; determinar os artefatos que são impactados por determinada RM; prever o número de novas RM's que poderão fazer parte do projeto.

No primeiro tipo de estudo a preocupação é estimar o tempo necessário para tratar a mudança solicitada em determinada requisição. Naturalmente, existe uma complexidade produzir uma estimativa precisa por conta das diferentes atividades envolvidas em conjunto aos diferentes níveis de capacitação que o responsável pela execução das tarefas pode ter [Xia et al., 2015]. Apesar da inerente imprecisão deste tipo de trabalho é importante salientar que estimar o tempo de solução de uma RM é importante para o gerenciamento do projeto porque ajudar alocar recurso de forma mais eficiente [Bhattacharya & Neamtiu, 2011] e melhorar a previsão do custo necessário para o lançamento de futuras versões do sistema [Vijayakumar & Bhuvaneswari, 2014].

No segundo grupo temos os artigos que tentam identificar previamente o conjunto de artefatos que serão impactados pela tarefa de manutenção [Nagwani & Verma, 2010b].

Incluir citação para o artigo Impact analysis by mining e Identifying the start impact set of a

. A literatura sobre análise de impacto é bastante abrangente e pode envolver os estudos de artefatos tais como documentos de requisitos e arquiteturais dos software, código fonte, registros (logs) de teste e assim por diante [Cavalcanti et al., 2014]. Neste sentido, estamos focados em estudos onde a RM são o ponto de partida para a análise de impacto.

O último grupo de trabalho se dedica em prever o número de RM que possivelmente serão relatadas em futuras versões do sistema. De forma similar ao primeiro grupo este tipo de estudo visa contribuir com o planejamento das atividades de manutenção e evolução. A predição do que será relatado inclui tanto novas requisições de problemas que não existiam em versões anteriores, por exemplo,

Inserir citação de Software defect association mining and defects e Mining software code repositories and bug database

, bem como aqueles que serão reabertos, ou seja, problemas que não foram solucionados anteriormente mesmo as suas RM's dizendo o contrário [Xia et al., 2015]

3.3.2.4 Melhorias Propostas na Dimensão Usuário

Recomendação de RM Os estudos contidos neste tópico possuem foco em dar suporte à novos desenvolvedores na redução da curva de aprendizagem quando eles pretendem ingressar em um novo projeto. Por exemplo, quando um novo desenvolvedor entra na equipe seria interessante que ele resolvesse as RM's que tivessem um menor nível de dificuldade. Posteriormente, quando o desenvolvedor ganhasse experiência,

poderia aumentar o grau de dificuldade relacionado à RM que ele deve tratar. Este tipo de processo ocorre com certa frequência em projetos de código aberto, onde a contribuição de desenvolvedores fora do projeto é fundamental. No entanto, encontrar um bug apropriado e uma correção apropriada para esse bug requer uma boa compreensão do projeto, que não é trivial [Wang & Sarma, 2011].

Em alguns projetos, um membro experiente da equipe, geralmente ensina os recém-chegados o que eles precisam para fazer completar suas tarefas necessárias à conclusão de uma RM. Todavia, alocar um membro experiente de uma equipe para ensinar um recém-chegado durante um longo tempo nem sempre é possível ou desejável, porque o mentor poderia ser mais útil fazendo tarefas mais importantes [Malheiros et al., 2012].

Para facilitar a inclusão de novos desenvolvedores alguns estudos vêm se dedicando em desenvolver sistemas de recomendação de RM's [Malheiros et al., 2012, Wang & Sarma, 2011]. Estes sistemas de recomendação podem ajudar o recém-chegado a solucionar uma RM mediante a apresentação de outras de código fonte potencialmente relevante que o ajudará na solução da RM do qual ficou responsável [Malheiros et al., 2012].

O segundo tipo de abordagem pode ser vista como ambiente de exploração do repositório de RM's. Este tipo de funcionalidade permite que permita que novos desenvolvedores pesquisem descrições das requisições que possam ser do seu interesse bem como dos artefatos relacionados àquela RM (por exemplo, arquivos relacionados, desenvolvedores contribuintes, registros de comunicação) [Wang & Sarma, 2011].

Com base nos estudos que compõem esta categoria, verificamos que modelos de IR vêm sendo utilizados para possibilitar a recomendação de RM. Neste contexto, técnicas bem conhecidas na literatura tais como VSM [Wang & Sarma, 2011] e o modelo estatístico PPM [Malheiros et al., 2012].

3.3.3 Classificação por Melhorias com Suporte à Papéis da Manutenção de Software

Verificar os artigos com N/A e não especificado

Nesta seção apresentamos as classificações dos estudos pelo tipo de papel na manutenção de software ao qual a funcionalidade visa dar suporte. Conforme exposto anteriormente a definição dos compõem o processo de manter de software foi desenvolvido conforme os trabalhos realizados por Polo e outros [Polo et al., 1999b] e Ihara e outros [Ihara et al., 2009]. Com estas modificações é possível acoplar os papéis utilizados neste estudo tanto aos processos adotados na indústria e em projetos de código aberto. A Figura 3.8 exibe o total de artigos em comparação com o papel ao qual

funcionalidade proposta visa dar suporte. Como pode ser observado verificamos um maior número de estudos para os papéis de Agendador e Desenvolvedor. Na Figura [?] verificamos uma prevalência de estudos com foco na dimensão Processo, em especial nos tópicos “Localização de RM Duplicados” e “Atribuição [Triagem] de RM”, o que é natural tendo em vista que há um mapeamento o papel desempenhado na manutenção com as atividades desempenhada por aquele papel. Neste sentido, não é de se estranhar a prevalência de estudos para aqueles funções vinculada ao processo de manutenção de software.

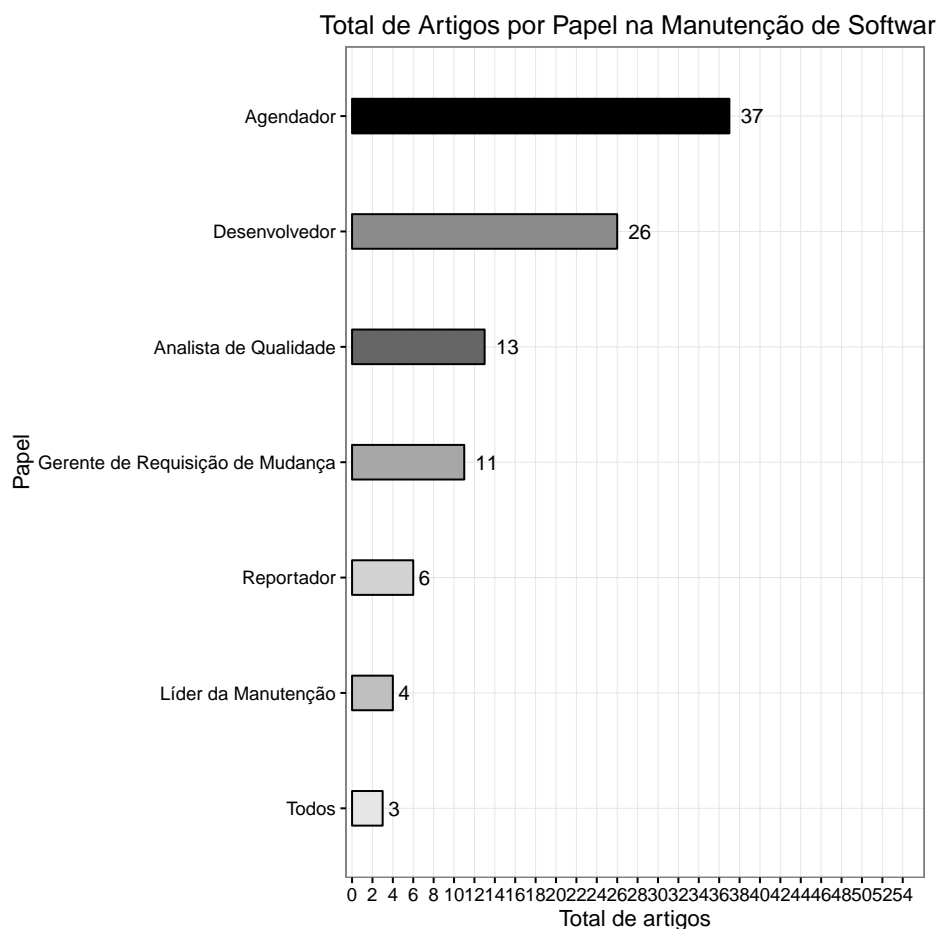


Figura 3.8: Total de artigos por papel na manutenção de software

Agendador Os estudos classificados neste tópico visam dar suporte as atividades de Agendador. Esta função tem como principal objetivo atribuição das RM's para o desenvolver mais apto [Banitaan & Alenezi, 2013]. O processo de atribuição de RM's deve ser realizado em acordo com a carga de trabalho do desenvolvedor e com a prioridade que foi atribuída à RM [Chawla & Singh, 2015].

Neste contexto, os estudos têm focado em apresentar soluções de atribuição automática [Banitaan & Alenezi, 2013, Shokripour et al., 2012, ?, Naguib et al., 2013, Zhang et al., 2014, Zanetti et al., 2013]; classificação automatizada [Gegick et al., 2010, Liu & Tan, 2014, Behl et al., 2014, Chawla & Singh, 2015, Tian et al., 2015]; visualização da fila de RM's [Izquierdo et al., 2015]; clusterização das requisições [Liu & Tan, 2014]; determinação do tempo necessário à conclusão da RM (time do fix) [Hosseini et al., 2012, Bhattacharya & Neamtiu, 2011]; sumarização das informações contidas na RM [Mani et al., 2012]; determinação de RM'S duplicadas [Sun et al., 2011, Kaiser & Passonneau, 2011].

Apesar da lista de artigos apresentada em cada tópico não ser exaustiva, os resultados demonstram um foco maior no suporte à atribuição e categorização das RM's apresentado soluções automatizadas para estas atividades. Existe possivelmente uma crença melhorar a produtividade do processo de manutenção de software reduzindo o esforço de encontrar o desenvolvedor mais apto.

Desenvolvedor Ao Desenvolvedor cabe aplicar as ações que irão produzir o resultado solicitado/esperado na RM@. Os estudos nesta categoria deveriam suportar atividades tais como codificação, depuração e testes.

No suporte ao desenvolvedor verificamos estudos que propõe à atribuição de RM's ao conjunto de desenvolvedor, em contrapartida da tradicional atribuição ao único programador [Banitaan & Alenezi, 2013], visando da minimizar o problemas decorrentes à propriedade de código e propiciar um maior nivelamento de informações entre os membros da equipe. Não obstante, o maior grupo de estudos nesta categoria estão relacionados na ajuda ao desenvolvedor de vincular um determinado problema do software à sua efetiva origem, ou seja, ao código fonte [Corley et al., 2011, Wong et al., 2014, Thung et al., 2014b, Nguyen et al., 2012, Thung et al., 2013, Romo & Capiluppi, 2015]. Neste mesma categoria verificamos estudos que dão suporte ao desenvolvedor em classificar à RM que lhe foi atribuída, em especial aquelas que estão relacionadas às questões de segurança do sistema [Gegick et al., 2010] ou aquelas RM's que possam impedir a resolução de outras (blocking-bugs) [Valdivia Garcia & Shihab, 2014]

Analista de Qualidade Cabe ao Analista de qualidade avaliar se uma RM foi solucionada por um Desenvolvedor afim de verificar se a RM foi corretamente resolvida. Neste sentido, melhorias em FGRM's que visam facilitar as atividades deste papel podem estar relacionadas ao processo de teste de software.

De maneira similar ao que ocorre na categoria de Desenvolvedor verificamos uma prevalência dos estudos visam determinar uma ligação entre um problema de software o código fonte [Corley et al., 2011, Wong et al., 2014, Thung et al., 2014b, Nguyen et al., 2012, Thung et al., 2013, Romo & Capiluppi, 2015]. Verificamos ainda estudos que tentam prever a probabilidade que determinada RM será reaberta [Xia et al., 2015], o que pode ajudar ao Analista de Qualidade na priorização das requisições com alta possibilidade de reabertura

Gerente de Requisição de Mudança O papel que representa esta categoria está vinculado à gestão do processo de manutenção de software, em especial por decidir se uma Requisição de Mudança será aceita ou rejeitada. Neste contexto, melhorias relacionadas à classificação quanto ao nível de segurança [Gegick et al., 2010, Zhang & Lee, 2011, Valdivia Garcia & Shihab, 2014], identificação de duplicados [Hindle et al., 2016, Sun et al., 2010, Alipour et al., 2013, Banerjee et al., 2012].

O estudos que fazem parte desta categoria destacam que um considerável conhecimento sobre o projeto é necessário bem como a capacidade de negociação com os desenvolvedores e demais partes interessadas são importantes para desempenho de papel. Todavia, tendo em vista o esforço e tempo gasto por esta tarefa, especialmente quando realizada manualmente, seria importante que as FGRM automatizasse algumas destas atividades.

Reportador Conforme discutido anteriormente os dados contidos nas RM'S são fundamentais em em diversas abordagens de melhorias da funcionalidades das FGRM@. Esta relevância ainda é maior nos estudos que fazem uso de técnicas de Recuperação da Informação. Neste sentido é importante que as FGRM deem suporte ao Reportador que, na maioria, é primeiro a registrar as informações que serão necessárias à solução da RM@.

Muitos dos estudos que fazem parte desta categoria partem da premissa que melhorar a qualidade dos dados na RM é o ponto de partida para tratar outros problemas relacionados ao processo de manutenção de software [Moran et al., 2015, Moran, 2015, Bettenburg et al., 2008a]. Neste sentido verificamos estudos para autocompletar as informações fornecidas pelo Reportador [Moran et al., 2015], suporte a reprodução do problema [Moran, 2015]; análise da qualidade da informação fornecida [Bettenburg et al., 2008a, Tu & Zhang, 2014]. Esta categoria também contempla um estudo que visa detectar que determinada RM já foi registrada durante o processo de escrita da mesma [Thung et al., 2014a].

Líder da Manutenção De forma similar ao Gerente de Requisição de Mudança as melhorias de funcionalidades proposta nesta categoria estão vinculadas à gestão de manter software. Conforme o esquema de classificação de papéis utilizado neste estudo, o Líder da Manutenção têm por responsabilidade definir os padrões e procedimentos que compõe o processo de manutenção que será utilizado. Para ajudar nesta tarefa alguns estudo vêm propondo melhorar a alocação de Tarefas do processo de resolução das Requisições de Mudanças [Netto et al., 2010]. Outros estudos visam mensurar o esforço necessário para solucionar determinada RM [Vijayakumar & Bhuvaneswari, 2014, Nagwani & Verma, 2010b], o que podem ajudar ao Líder no planejamento de liberações de novas versões do sistema que está sendo mantido.

Todos Esta categoria abarca os estudos para o qual a melhoria proposta possui impacto positivo para todos os papéis envolvidos na manutenção de software. A definição que o foco da melhoria é geral decorre do que foi dito como objetivos dos autores dos estudos que fazem parte desta categoria ou ainda por não ser possível determinar uma atividade específica sendo beneficiada.

Conforme pode ser observado, os estudos estão relacionados principalmente com a melhoria da visualização das informações contidas nas RM's [Hora et al., 2012, Takama & Kurosawa, 2013, Dal Sasso & Lanza, 2014]. As melhorias podem estar vinculadas a questões de usabilidade das ferramentas, como por exemplo a navegabilidade entre as RM's [Dal Sasso & Lanza, 2014].

3.3.4 Ferramentais Estendidas

Conforme verificamos nas seções anteriores diversos estudos vêm sendo propostos na literatura com o objetivo de melhorias atividades relacionadas à manutenção de software. Não obstante, os nossos resultados demonstraram, tomando como base os estudos que fazem parte deste mapeamento, que grande parte das melhorias propostas não foram implementadas nas FGRM de modo a ser utilizadas ou avaliadas pelos profissionais envolvidas em manutenção de software.

A Figura 3.9 apresenta a quantidade de estudos que efetivamente foram transformadas em extensões de funcionalidades de uma FGRM@. Do total de 64 estudos que foram utilizados neste mapeamento apenas fazem parte das funcionalidade de uma FGRM@.

reavaliar a quantidade de estudos

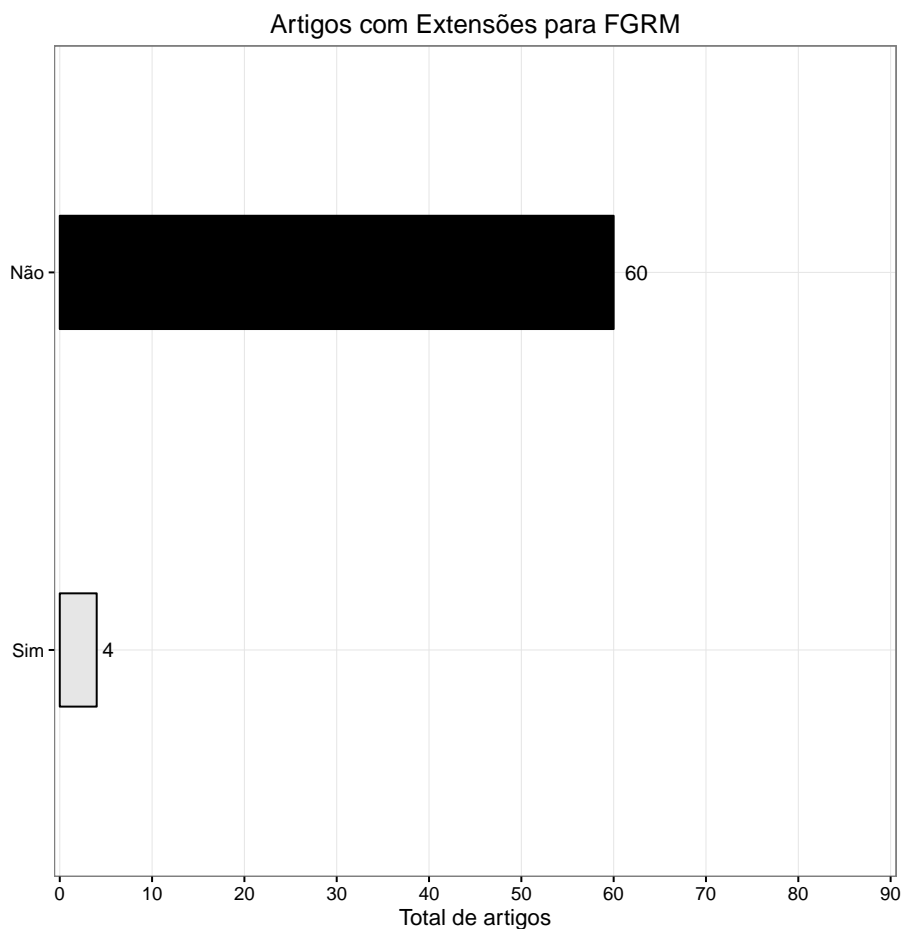


Figura 3.9: Total de estudos que são extensões para FGRM

Cabe ressaltar que no escopo de determinado estudo pode não estar prevista a efetiva transformação da melhoria proposta de modo a ser utilizada efetivamente pelo seu público-alvo, como por exemplo, a criação ou melhoria de uma funcionalidade de determinada FGRM@. Ademais, não está no escopo deste estudo avaliar ou discutir a facilidade que as FGRM possuem para criar novas funcionalidades ou melhorias as já existentes. Contudo, o nosso entendimento é que como um maior número de melhorias proposta na literatura sendo utilizadas pelos profissionais envolvidos em manutenção de software poderia melhorar a qualidade das soluções propostas mediante a redução da diferença entre o estado-da-prática com o estado-da-arte.

3.4 Limitações e Ameaças à Validade

Alguns dos procedimentos adotados neste trabalho não acompanharam exatamente as diretrizes existentes na literatura para condução de um Mapeamento Sistemático, em

aqueles propostos por Kitchenham e outros [Keele, 2007]. Um único investigador selecionou os estudos candidatos e este mesmo revisor teve a responsabilidade de analisar os artigos que seriam incluídos ou excluídos.

O mapeamento realizado neste estudo utilizou o método de aplicação de setenças de busca nas bases de dados selecionadas para coletar os estudos primários. Outros estudos utilizam além da estratégia descrita fazem de uso de uma técnica conhecida como “bola de neve” (snowballing) [Wohlin, 2014] onde as referências dos estudos primários podem ser usadas para o compor o conjunto de artigos do mapeamento. Neste sentido, ao usarmos uma única estratégia podemos ter perdido estudos relevantes e, portanto, subestimar a extensão dos resultados encontrados. Em particular, por termos optamos por escolher artigo apenas em língua inglesa também pode ter havido faltado artigos publicados em revistas e conferências nacionais. Assim, nossos resultados devem ser considerados apenas com base em artigos em inglês contidos nas bases de dados escolhidas e em especial publicados nas principais conferências da área de Engenharia de Software.

O fato de um único pesquisador ter sido o responsável pela análise dos estudos pode significar ponto significa que alguns dos dados coletados podem ser errôneas. O processo de seleção e validação dos estudos primários pode levar a problemas de extração e agregação das informações quando há um grande número de artigos ou os dados são complexos [Keele, 2007]. No entanto, neste estudo terciário, houve poucos estudos primários e os dados extraídos eram relativamente objetivos. Desta forma, não esperamos erros de extração. Cabe ressaltar que apesar do processo de validação ter sido executado por um único pesquisador, os critérios de qualidade foram avaliados independentemente por dois pesquisadores, desta forma minimizando a inclusão de estudos cuja qualidade comprometa os resultados.

No tocante as questões de pesquisa é possível que as perguntas de pesquisa definidas possam não abranger completamente o campo de pesquisa sobre as funcionalidades das FGRM's. No entanto, algumas discussões com membros do projeto e especialistas em Manutenção de Software foram realizadas para validar as perguntas. Assim, mesmo que não tenhamos considerado o melhor conjunto de questões, tentamos abordar as questões mais frequentes e abertas no campo, tanto do ponto de vista do praticante como do investigador.

Como as bases de dados digitais não funcionam com regras de pesquisa compatíveis entre si, todas as sequências de pesquisa foram adaptadas e calibradas para cada banco de dados digital. No entanto, não conhecemos todas as regras que as bases de dados digitais utilizam para procurar um documento. Neste sentido, a forma que as setenças de busca foram estruturadas podem não ser a mais otimizada para seleção

do maior número de estudos relevantes para o estudo.

3.5 Trabalhos Relacionados

Os autores Kagdi e outros [Kagdi et al., 2012] realizaram uma revisão da literatura sobre abordagens para mineração de repositórios de relatos de problema de software. No contexto daquele trabalho este tipo de repositório pode ser comparada à uma FGRM. O resultado foi uma taxonomia baseada em quatro dimensões: o tipo de repositório extraído (o que), o propósito (por que), o método proposto (como) e o método de avaliação (qualidade). No entanto, sua taxonomia não fornece um entendimento extensivo sobre as investigações em repositórios de RM. Em primeiro lugar, embora pesquisassem muito trabalho e fizessem uma taxonomia consistente, isso foi feito já em 2006 e desde então muitos outros trabalhos foram publicados, considerando novos tópicos e abordagens. Em segundo lugar, de acordo com seus critérios de exclusão para estudos, eles estavam muito preocupados com estudos que abordavam mudanças evolutivas de artefatos de software investigando múltiplos repositórios de software. Como consequência, muitos estudos que usaram dados de um repositório único estavam além de seu escopo.

Por outro lado, o estudo realizado neste trabalho aumentou o escopo para as funcionalidades oferecidas pelas FGRM possibilitando uma visão mais abrangente do estado da arte deste tipo de estudo. As dimensões propostas por Kagdi [Kagdi et al., 2012]. Uma outra diferença com o trabalho de Kagdi [Kagdi et al., 2012] é que sua taxonomia considera as técnicas e métodos para mineração de repositórios de software como o foco principal do seu estudo, por lado este trabalho considera as FGRM, sobre o prima de suas funcionalidades, como entidades de primeira classe.

Capítulo 4

Caracterização das Ferramentas de Gerenciamento de Requisição de Mudança

4.1 Introdução

Quando uma empresa ou projeto software de código aberto decide adotar uma Ferramenta de Gerenciamento de Requisições de Mudança - FGRM um desafio é encontrar aquela que melhor atenda suas necessidades. Um possível critério de seleção é o conjunto de funcionalidades oferecidas pelo sistema. Outros critérios podem envolver o custo ou o suporte a falhas da ferramenta. De maneira relacionada, o pesquisador que estuda propostas de melhorias para as FGRM's pode estar está interessado em analisar o conjunto de funções comuns que caracterizam este tipo de software.

O numero de FGRM's disponíveis atualmente é bastante elevado. Em uma inspeção inicial, verificamos a existência de mais de 50 ferramentas fornecidas comercialmente ou em código aberto ¹. Apesar das diversas opções disponíveis, ao bem do nosso conhecimento, desconhecemos estudos que avaliem sistematicamente as funcionalidades oferecidas por este tipo de ferramenta a fim de compará-las. Entendemos que a partir de um conjunto compartilhado de funções/comportamento seja possível caracterizar as FGRM's, ao mesmo tempo possibilita avaliar a contribuição de novas funcionalidades propostas na literatura, conforme discutido no Capítulo 3. Para alcançarmos este objetivo, realizamos um estudo exploratório visando determinar as principais funcionalidades presentes nas FGRM's. Um estudo exploratório está preocupado com a

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

análise do objeto em sua configuração natural e deixando que as descobertas surjam da própria observação [Wohlin et al., 2012]. Neste tipo de estudo nenhuma hipótese é previamente definida.

O trabalho descrito neste capítulo consistiu na leitura da documentação disponível na Internet de algumas FGRM's de modo a sistematizar as funcionalidades oferecidas por cada ferramenta. As funções foram coletadas e organizadas utilizando a técnica de Cartões de Classificação (Sorting Cards) [Zimmermann et al., 2009b, Rugg & McGeorge, 2005]. Devido ao alto volume de ferramentas disponíveis e ao esforço necessário para analisar a documentação de todas elas, optamos por realizar este estudo com um conjunto mínimo escolhido com a ajuda de profissionais envolvidos em manutenção de software. Através de um levantamento (survey) onde os profissionais responderam dentre as ferramentas apresentadas quais eram as mais representativas dentro do domínio de aplicação das FGRM's. A representatividade neste contexto não está no número de projetos que utiliza determinada ferramenta, mas pelas características que determinado software possui que o torna diferenciável dentro do seu domínio.

Este capítulo está organizado da seguinte forma: na Seção 4.2 discutimos os objetos deste capítulo; na Seção 4.3 apresentamos o método utilizado na condução do estudo, em especial a técnica de Cartões de Ordenação e o levantamento realizado com os profissionais para escolher as ferramenta do estudo.

4.2 Objetivo do Capítulo

O objetivo inicial deste capítulo é apresentar e discutir as principais funcionalidades das FGRM's que dão suporte ao desenvolvimento e manutenção de software. Tomando como ponto de partida um conjunto de sistemas definidos como os mais relevantes por profissionais envolvidos em manutenção e desenvolvimento de software. Em um segundo momento, o foco foi caracterizar este tipo de ferramenta tomando como base as funcionalidades oferecidas pelos softwares. Conforme já exposto, a literatura em Manutenção de Software apresenta diferentes nomenclaturas para este tipo de ferramenta (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), sem, contudo, se preocupar em diferenciá-las.

Acreditamos que o resultado deste estudo permitirá compreender melhor este tipo de software tomando como base o conjunto de funções que eles oferecem aos seus usuários. Também será possível propor novas funcionalidades ou melhorias das existentes tendo em vista a possibilidade de determinar o conjunto mínimo de com-

portamentos deste tipo de ferramenta. Uma outra contribuição é a criação de uma taxonomia com base nas funcionalidades oferecidas.

4.3 Metodologia

A fim de determinarmos o conjunto das principais funcionalidades das FGRM's que dão suporte à manutenção e desenvolvimento de software realizamos um estudo exploratório dividido em três etapas que estão listadas a seguir. O resultado obtido em etapa foi utilizado para subsidiar as atividades do etapa posterior. O início de uma nova fase do trabalho era precedido de uma avaliação geral afim de verificar possíveis inconsistências e avaliação das lições aprendidas.

- (i) Seleção das Ferramentas
- (ii) Inspeção da Documentação
- (iii) Agrupamento das Funcionalidades

4.3.1 Seleção das Ferramentas

A primeira etapa consistiu da definição das ferramentas que seriam utilizadas no estudo. A partir de uma pesquisa na Internet obtivemos um conjunto inicial de 50 ferramentas ² que podem ser visualizadas no Anexo B. Devido ao esforço necessário e a dificuldade de realizar a análise em cada uma daquelas ferramentas, optamos por escolher um subconjunto de sistemas que fossem mais representativos, tomando como base a opinião de profissionais envolvidos em manutenção e desenvolvimento de software. A representatividade neste caso corresponde a opinião do profissional sobre notoriedade que a ferramenta possui dentro do seu domínio de aplicação em comparação com as demais que lhe foram apresentadas ou outras do qual tenha prévio conhecimento.

A opinião dos profissionais foi obtida mediante a realização de uma pesquisa (survey [Wohlin et al., 2012]) realizada com o uso de um formulário eletrônico. O formulário foi estruturado em duas partes principais: a formação de base do participante (background) e a avaliação das ferramentas. Na primeira parte estávamos interessados em conhecer as características do respondente. Esta informação é relevante tendo em vista que, como descreveremos a seguir, o questionário foi replicado em três grupos distintos de profissionais. Neste sentido, foi possível realizar análises sobre como é formado cada um dos grupos que participaram deste estudo. Na segunda parte da

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

pesquisa apresentamos as ferramentas e foi solicitado aos participantes que avaliassem a relevância de cada uma delas através de questões de múltipla escolha. As opções de respostas foram estruturadas em escala do tipo Likert [Robbins & Heiberger, 2011].

Antes da efetiva aplicação do questionário no público-alvo do estudo, o documento foi validado em um processo de três etapas. Na primeira parte foi solicitado a dois pesquisadores experientes da área de Engenharia de Software que avaliassem o formulário. A partir das sugestões obtidas dos pesquisadores foram realizadas adequações no documento. Após a alteração uma nova versão do formulário foi enviada para dois profissionais envolvidos diretamente em manutenção de software. O critério utilizado para seleção dos profissionais foi o tempo dedicado à tarefa de manter software, que no caso dos desenvolvedores escolhidos era em média de 10 anos. O formulário foi modificado com as sugestões dos profissionais finalizando a segunda etapa de validação. A última etapa consistiu na realização de um piloto com dez profissionais envolvidos em manutenção de uma empresa pública de informática. Os profissionais tiveram que preencher o questionário, contudo, foram adicionadas questões as quais era possível inserir sugestões de melhoria. O resultado deste processo de validação é o questionário presente no Anexo C. Como o público-alvo do questionário poderia incluir desenvolvedores de diferentes nacionalidades foi construído em uma versão em língua inglesa do formulário.

A população de interesse deste levantamento é o conjunto de profissionais envolvidos em manutenção de software. Naturalmente é difícil definir o tamanho e características desta população de modo a mensurar uma amostra significativa. Neste sentido, visando minimizar enviesamento deste estudo, o questionário foi replicado em três grupos:

Grupo 01: Profissionais de empresa pública e privada de desenvolvimento e manutenção de software.

Grupo 02: Profissionais que contribuem em projetos de código aberto

Grupo 03: Profissionais que participam de grupos de interesse sobre desenvolvimento e manutenção de software em uma rede social profissional (LinkedIn) ou aqueles que participam de discussão sobre este assunto em uma rede social (Stack Overflow).

Os participantes que compõe cada grupo foram escolhidos conforme critérios que são detalhados no Capítulo 5. A reutilização desta base de desenvolvedores se deu por conta de ambos os estudos compartilharem a mesma população de interesse, podendo

neste caso compartilhar a mesma amostra. Ademais, como o questionário descrito nesta seção foi realizada antes daquele contido no Capítulo 5, as lições aprendidas no primeiro serviram para melhorar o processo de desenho e execução do segundo.

Com base nos dados obtidos da pesquisa com os profissionais, as FGRM's foram classificadas como "*ferramentas*" e "*serviços da internet*". O primeiro grupo representa os softwares que são capazes de serem implantados na infraestrutura do seu cliente e permite algum grau de personalização de pelo menos um dos componentes, como por exemplo, o banco de dados utilizado. No segundo grupo estão os software que ofertam a gerência das RM's mediante uma arquitetura do tipo Software como Serviço (Software as Service) [Fox et al., 2013], onde certos tipos de alterações no comportamento do software são mais restritas. Acreditamos que ao escolher ferramentas dos dois tipos descritos anteriormente iremos cobrir uma grande parte do domínio de aplicação das FGRM's. Optamos por escolher *06 ferramentas* para o estudo, sendo três de cada um dos grupos. Neste sentido, foram escolhidas as três ferramentas mais relevantes para cada grupo com base nos dados da pesquisa com os profissionais.

Para seleção das ferramentas utilizados a formula apresentada na Equação 4.1. Atribuímos a métrica r_i que representa a relevância de determinada ferramenta f_i . A métrica é calculada somando a frequência de cada um dos graus de relevância apresentado na Tabela 4.1 multiplicado pela pelo grau de relevância descrito na mesma tabela.

$$r_i = \sum_{i=1}^n f_i \times w_j \quad (4.1)$$

A documentação de algumas ferramentas, em especial aquelas que adotam uma arquitetura cliente/servidor e necessitam de um certo grau de administração, dividem as funcionalidades do software entre aquelas com foco no usuário final e administradores. Nestes casos optamos por coletar as funcionalidades cujo o foco seja o usuário da FGRM, tendo vista que eles, profissionais devotadas às atividade de manutenção de software, estarem entre o público-alvo desta dissertação.

j	Grau de Relevância	Peso(w_j)
1	Não conheço a ferramenta	1
2	Nada relevante	2
3	Pouco relevante	3
4	Pouco relevante	4
5	Muito relevante	5

Tabela 4.1: Graus de Relevância

4.3.2 Inspeção da Documentação

Nesta etapa do trabalho realizamos a leitura do material disponível na Internet para cada uma das ferramentas que foram selecionadas na etapa anterior. Entre estes materiais utilizamos manuais do usuário e do desenvolvedor e notas de lançamento. Para cada uma das FGRM's optamos por estudar a última versão estável do software a fim de analisarmos o que há de mais novo disponível aos usuários. A Tabela 4.2 apresenta as ferramentas analisadas e o elo de ligação para cada 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 esteja mais atualizada.

Nome da Ferramenta	Elo de Ligação
Bugzilla	\protect\T1\textbracelefthttps://www.bugzilla.org/features/}
Github Issue Tracking System	\protect\T1\textbracelefthttps://github.com/blog/411-github-issue-tracker}
Github Issue Tracking System	\protect\T1\textbracelefthttps://github.com/features}
Github Issue Tracking System	\protect\T1\textbracelefthttps://guides.github.com/features/issues/}
Gitlab Issue Tracking System	\protect\T1\textbracelefthttp://docs.gitlab.com/ce/user/project/labels.html}
Gitlab Issue Tracking System	\protect\T1\textbracelefthttps://about.gitlab.com/2016/08/22/announcing-the-gitlab-issue-board/}
Gitlab Issue Tracking System	\protect\T1\textbracelefthttps://about.gitlab.com/solutions/issueboard/}
Gitlab Issue Tracking System	\protect\T1\textbracelefthttps://docs.gitlab.com/ee/user/project/description_templates.html}
Gitlab Issue Tracking System	\protect\T1\textbracelefthttps://docs.gitlab.com/ee/user/project/issues/automatic_issue_closing.html}
Gitlab Issue Tracking System	\protect\T1\textbracelefthttps://docs.gitlab.com/ee/workflow/issue_weight.html}
Gitlab Issue Tracking System	\protect\T1\textbracelefthttps://docs.gitlab.com/ee/workflow/milestones.html}
Gitlab Issue Tracking System	\protect\T1\textbracelefthttps://docs.gitlab.com/ee/workflow/time_tracking.html}
JIRA	\protect\T1\textbracelefthttps://br.atlassian.com/software/jira/features}
MantisBT	\protect\T1\textbracelefthttps://www.mantisbt.org/wiki/doku.php/mantisbt:features}
Redmine	\protect\T1\textbracelefthttp://www.redmine.org/projects/redmine/wiki/Features}

Tabela 4.2: Ferramentas e sus respectivas documentação

Os dados obtidos da leitura do material disponíveis para cada ferramenta foram sistematizados por meio de técnica denominada *Cartões de Classificação - Sorting Cards*. Cartões de Classificação é um técnica de elicitación de conhecimento de baixo custo e com foco no usuário largamente utilizada em arquitetura informacional para criar modelos mentais e derivar taxonomias da entrada utilizada [Just et al., 2008]. Ela envolve a categorização de um conjunto de cartões em grupos distintos de acordo com algum critério previamente definido [McGee & Greer, 2009]. O estudo de Maiden e outros [Maiden & Rugg, 1996] sugere que a técnica de Cartões de Classificação é uma das mais úteis para aquisição de conhecimento de dados, em contraste ao conhecimento de comportamento ou de processo.

Existem três principais fases dentro do processo de classificação dos cartões: (1) preparação, no qual participantes e o conteúdo dos cartões são selecionados; (2) execução, onde o cartões são organizados em grupo significativos com um título que o descreve; e por fim, (3) análise, no qual os cartões são sistematizados para formar hierarquias mais abstratas que são usadas para deduzir temas. No processo tradicional de Cartões Ordenados cada declaração realizada por um participante resulta na criação de

exatamente um único cartão [Just et al., 2008]. Contudo, no nosso caso, foi realizada a divisão da documentação da ferramenta por cada funcionalidade encontrada. Neste sentido, cada funcionalidade obtida mediante a inspeção da documentação foi mapeada em único cartão.

Os cartões foram organizados de modo que continham o nome e a versão da ferramenta analisada; a URL da documentação utilizada; o nome da funcionalidade coletada, que consiste de uma descrição breve conforme existente na documentação; descrição detalhada da funcionalidade, cujo objetivo é facilitar o processo de agrupamento que será descrito na próxima seção. O Anexo apresenta um formulário que representa os cartões utilizados neste estudo.

Inserir anexo com o formulário que representa os cartões

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

Análise Compartilhada Em um segundo momento tanto o autor 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 representar as funções descritas na coluna “nomenclatura original”.

4.4 Resultados

4.4.1 Ferramentas Escolhidas

4.4.2 Categorização das Ferramentas

Após a inspeção da documentação e validação dos dados obtivemos um total de 123 cartões. Nós sistematizamos os cartões manualmente tendo em vista que não existem

Tabela 4.3: Ferramentas utilizados no estudo

Ferramenta	Classificação	Versão	URL
Bugzilla	Ferramenta	5.0.3	https://www.bugzilla.org
Mantis Bug Tracker	Ferramenta	1.3.2	https://www.mantisbt.org
Redmine	Ferramenta	3.3.1	http://www.redmine.org/
JIRA Software	Serviço	7.2.4	https://br.atlassian.com/software/jira
Github Issue Tracking System	Serviço	-	https://github.com/
Gitlab Issue Tracking System	Serviço	-	https://gitlab.com/

ferramentas ou métodos capazes de automatizar o processo de construção de hierarquias. Tendo em vista que nosso objetivo é derivar tópicos a partir do conjunto inicial de cartões, optamos por realizar um *ordenamento aberto* dos cartões. Naquele tipo de abordagem, os grupos são estabelecidos durante o processo de Classificação os cartões em oposição a outra forma de utilização da técnica onde a sistematização dos cartões ocorre com base em grupos pré-determinados. Ao final do processo obtivemos os seguintes tópicos listados a seguir. Nas próximas seções apresentamos as funcionalidades que compõe cada um deles.

1. Busca e Duplicados
2. Extensão de Funcionalidades
3. Gerenciamento da Informação
4. Gerenciamento de Artefatos
5. Internacionalização da Ferramenta
6. Processo de Trabalho
7. Segurança da Informação
8. Suporte ao Trabalho do Desenvolvedor
9. Triagem de RM's
10. Visualização e Monitoramento de RM's

4.4.2.1 Busca e Duplicados

Este tópico foi criado para agrupar as funcionalidades relacionadas a busca de RM's e a localização de duplicados.

4.4.2.2 Extensão de Funcionalidades

As funcionalidades que compõem este grupo têm por objetivo extensor o conjunto de funcionalidades oferecidas através de uma arquitetura de plugins ou mediante o suporte de API's³.

4.4.2.3 Gerenciamento da Informação

Este tópico contempla as funcionalidade que se dedicam ao armazenamento e consistência das informações contidas na FGRM.

4.4.2.4 Gerenciamento de Artefatos

O processo de manutenção de software pode consumir ou gerar diversos artefatos estudos de artefatos tais como documentos de requisitos e arquiteturais dos software, código fonte, registros (logs) de teste e assim por diante [Cavalcanti et al., 2013]. Em alguns contextos, devido ao volume de artefato gerados, é importante que a FGRM dê suporte para armazenamento e recuperação deste ativos do processo de software.

4.4.2.5 Internacionalização da Ferramenta

Neste tópicos estão as características das FGRM que ajudam no desenvolvimento e/ou adaptação de um produto, em geral softwares de computadores, para uma língua e cultura de um país.

4.4.2.6 Processo de Trabalho

Este ramo do esquema de classificação proposto foi criada para agrupar as funcionalidades que visão dar suporte ao processo de manter software.

4.4.2.7 Segurança da Informação

Neste grupo estão as características de uma FGRM que diretamente relacionada com proteção de um conjunto de informações, no sentido de preservar o valor que possuem para um indivíduo ou uma organização.

³https://en.wikipedia.org/wiki/Application_programming_interface

4.4.2.8 Suporte ao Trabalho do Desenvolvedor

Este tópico contém ideias sobre como o trabalho para desenvolvedores pode ser reduzido, seja por melhor suporte de ferramentas (por exemplo, configuração automática de espaços de trabalho) ou relatórios de bugs melhores com mais dados.

4.4.2.9 Triagem de RM's

Este tópico descreve comentários sobre o processo de triagem de RM's. O processo de atribuição de RM, também conhecido como triagem, possui como principal objetivo encontrar o desenvolvedor mais capacitado para manipular uma dada RM.

4.4.2.10 Visualização e Monitoramento de RM's

Em diversos contextos, devido ao volume das RM's, é importante que as partes interessadas na manutenção de software, possam visualizar e monitorar a situação das requisições que estão analisadas em determinado período.

4.5 Discussão

Classificar envolve categorização, e há uma literatura sofisticada sobre categorização, taxonomia e semântica, todas as quais são potencialmente relevantes [Rugg & McGeorge, 2005]

As técnicas de triagem são uma parte inestimável Engenheiro de conhecimento ou kit de ferramentas do engenheiro de requisitos. Eles são simples de usar e combinam a flexibilidade de uso com um formalismo de representação altamente formalizado, preenchendo a lacuna entre as técnicas qualitativas e quantitativas.

4.6 Ameças à Validade

A generalidade dos resultados é muitas vezes sacrificada pela riqueza e Complexidade, refletindo um conflito inerente entre E validade externa [19]. Dada a disparidade entre Taxonomias de mudanças publicadas e o debate entre os Participantes deste estudo, pode-se argumentar que a mudança Classificação é, por natureza, uma avaliação subjetiva.

Motivado, no entanto, pelo potencial de melhoria Exigências mudança visibilidade e gestão, modelagem fontes de mudança é uma iniciativa digna. A abordagem colaborativa adotada aqui levou a um modelo internamente utilizável e reflete a recomen-

dação de Sjoberg [19] de “formular um escopo relativamente estreito para começar e, em seguida, estendê-lo gradualmente”. Portanto, enquanto não há reivindicações de validade externa pode ser feita, Foram extraídas de um Variedade de estudos empiricamente baseados, é plausível Os resultados se aplicam a contextos além dos limites deste estudo. As construções iniciais são fornecidas aqui, juntamente com a descrição dos métodos de tal forma que deve ser possível replicar este estudo. Dada a natureza colaborativa desta pesquisa, e sua aplicabilidade imediata, ela tem um alto nível de relevância.

4.7 Resumo do Capítulo

Neste artigo apresentamos os resultados de uma análise quantitativa sobre o feedback recebido por 175 colaboradores e usuários dos projetos APACHE, ECLIPSE, MOZILLA. O resultado desta discussão é uma lista de sete recomendações para o projeto de novos sistemas de rastreamento de bugs. Em nosso trabalho futuro, iremos construir protótipos de projeto para tais sistemas e testar sua usabilidade.

Capítulo 5

Pesquisa com Profissionais

5.1 Introdução

Uma pesquisa baseada em questionários, conhecido na literatura como *Survey*, é uma abordagem de coleta e análise de dados na qual os participantes respondem a perguntas ou declarações que foram desenvolvidas antecipadamente [Kasunic, 2005]. Este tipo de pesquisa dividem-se em duas grandes categorias: questionários auto-administrados e entrevistas [Kasunic, 2005]. O primeiro tipo é aquele que a a maioria das pessoas pensa quando falamos em “pesquisa de opinião”, no qual geralmente recebemos por meio de correio eletrônico ou estão disponíveis através de páginas da Internet. As entrevistas apresentam as mesmas característica dos questionários auto-administrados, estando a principal diferença na profundidade que as perguntas são apresentadas aos participantes. Neste estudo utilizamos um questionário auto-administrado como instrumento de coleta dos dados.

Em uma pesquisa baseada em questionário os participantes responder a perguntas ou afirmações que foram que foram desenvolvidas com antecedência. Quando conduzida adequadamente, esse tipo de pesquisa permite que os pesquisadores generalizem as crenças e opiniões de uma população mediante os dados coletados de um subconjunto do público-alvo (amostra). No trabalho conduzido por Kasunic [Kasunic, 2005] são apresentada um conjunto de etapas a serem seguidas no processo de condução de um survey:

1. Identificar os objetivos da pesquisa
2. Identificar e caracterizar o público-alvo
3. Elaborar o plano de amostragem

4. Elaborar e escrever um questionário
5. Aplicar questionário de teste ou piloto
6. Distribuir o questionário
7. Analisar os resultados e escrever o relatório

Em uma série de artigos [Pfleeger & Kitchenham, 2001, Pfleeger & Kitchenham, 2002], Kitchenham e Pfleeger discutem os princípios da pesquisa com questionário no âmbito da Engenharia de Software (ES), tendo como foco principal as etapas 3, 4, 6 e 7 descritas no estudos feito por Kasunic [Kasunic, 2005]. Os autores apresentam conceitos básicos de estatística para discutir algumas questões relativas à população e amostra. Nestes mesmos estudos os autores investigaram o desenho de quatro surveys na área de ES e concluem que em apenas um deles foi composto por uma amostra significativa de uma população. Ao final eles salientam a necessidade que este tipo estudo científico utilize uma metodologia de modo a reduzir qualquer viés no tocante à amostra utilizada.

Com o objetivo de coletar os aspectos mais importantes das funcionalidades oferecidas pelas Ferramentas de Gerenciamento de Requisições de Mudança (FGRM) do ponto de vista dos profissionais ligados à manutenção de software foi realizada uma pesquisa com profissionais (survey). O planejamento e o desenho da pesquisa seguiu as diretrizes propostas nos trabalhos de Wohlin [Wohlin et al., 2012] e Kasunic [Kasunic, 2005]. Em especial, no tocante a definição da população e da amostra de interesse utilizamos o arcabouço (framework) proposto por De Mello e outros [de Mello et al., 2015, de Mello et al., 2014].

A população da pesquisa proposta é a comunidade envolvida com o processo de manutenção de software e que faça uso de FGRM's. Desta utilizamos como amostra os profissionais que estão envolvidos nos projetos Python¹, Eclipse Foundation², Mantis Bug Tracker³ que contemplam os profissionais vinculados a algum projeto de código aberto. Por outro lado, visando alcançar os profissionais que trabalham em empresas privadas, utilizamos profissionais que fazem parte das rede sociais Stack Overflow e LinkedIn e que façam parte de discussões na rede de assuntos relacionados à manutenção de software. A pesquisa foi replicada em uma empresa pública de software do qual o autor possui vínculo. Maiores detalhes sobre o processo de escolha das amostras serão discutidas posteriormente.

¹<http://bugs.python.org/>

²<https://bugs.eclipse.org/bugs/>

³<https://mantisbt.org/>

A importância deste tipo de trabalho está na possibilidade de avaliar se as pesquisas relativas a evolução das funcionalidades das FGRM's 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.

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's, bem como a adoção das metodologias propostos pelos agilistas no processo de manutenção de software.

Para uma melhor apresentar a finalidade desta parte da dissertação estruturamos o objetivo conforme propõe a metodologia GQM (Goal, Question e Metric)[Basili et al., 1994], *o propósito deste estudo avaliar as funcionalidade oferecidas e as extensões propostas nas literatura ponto de vista dos profissionais envolvidos em manutenção de software no contexto de projetos de software de código aberto e uma empresas privadas 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 a opinião dos profissionais envolvidos em Manutenção de Software com relação as funcionalidades oferecidas atualmente pelas FGRM?

Questão 02 Na visão dos profissionais envolvidos em Manutenção de Software quais das extensões propostas na literatura teriam maior relevância em suas atividades atuais?

Questão 03 Como as práticas propostas pelos agilistas estão sendo utilizadas especialmente no processo de manutenção de software?

Questão 04 Como as FGRM's podem ajudar aos times devotados à manutenção de software na prática adotada pelos agilistas?

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 e Metodologia da Pesquisa com Profissionais

5.3.1 Conceitos Básicos

Estudos primários em Engenharia de Software (SE) são muitas vezes conduzidos em amostras estabelecidas por conveniência (Pickard et al., 1998, Sjøberg et al., 2005, Dybå et al., 2007). Este cenário é especialmente crítico para os levantamentos em grande escala (Kasunic 2005), nos quais são aplicados esforços consideráveis no recrutamento e recolha de dados dos participantes (Conradi et al., 2005), mas a generalização dos resultados é limitada, mesmo quando as características de outros inquéritos são claramente descritas e Repetidos em seus ensaios.

Um desafio no estabelecimento de amostras representativas nos inquéritos SE inclui a identificação de fontes relevantes e disponíveis a partir das quais podem ser estabelecidas estruturas de amostragem. Portanto, os pesquisadores da SE são tentados a explorar fontes alternativas tipicamente disponíveis na Web para ampliar o tamanho das amostras, como as redes sociais (de Mello e Travassos, 2013). No entanto, o uso ad hoc dessas tecnologias Web por si só não é suficiente para evoluir o cenário de amostragem em relação aos levantamentos SE, uma vez que o tamanho das amostras é apenas uma das questões que dificultam a generalização dos resultados das pesquisas SE.

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.

Conforme discutido anteriormente, a população de interesse desta pesquisa é composta por profissionais envolvidos em desenvolvimento e manutenção de software. Para sistematizar o processo de escolha da amostragem utilizamos o arcabouço (framework) conceitual proposto por de Mello e outros [de Mello et al., 2014] que está representado na Figura 5.1. O modelo proposto inclui além dos conceitos estatísticos tradicionalmente utilizados em pesquisas por questionário (surveys), tais como público-alvo, população, amostragem e unidade de observação [Thompson, 2012], introduz um conjunto de novos conceitos que visam apoiar o processo de escolha da amostragem em questionários, especialmente em ES. Os conceitos que os autores propuserem incluem: *Fonte de Amostragem*, *Unidade de Pesquisa*, *Plano de Pesquisa* e *Estratégia de Amostragem*.

ER1 Uma Fonte de Amostragem não deve representar intencionalmente um subconjunto segregado do público-alvo, ou seja, dado um público-alvo “X”, não seria adequado pesquisar por unidades da fonte intencionalmente desenhada para compor um subconjunto específico de “X”.

ER2 Uma Fonte de Amostragem não deve apresentar qualquer viés em incluir na sua base de dados determinados subconjuntos do público-alvo. Critérios desiguais para inclusão de Unidades de Pesquisa significam oportunidades desiguais para oportunidades de amostragem.

ER3 Todas as Unidades de Pesquisa das amostras e suas Unidades de Observação devem identificados por de forma única.

ER4 Todas as amostragem de determinada Unidade de Pesquisa devem ser acessíveis. Se houver unidades de pesquisa ocultas, não é possível contextualizar a população.

Os autores outros requisitos que são definidos como desejáveis que estão relacionados com amostragem, clareza e integridade da amostra. Os critérios adicionais e exemplos de avaliação de Fontes de Amostragem utilizados os requisitos discutidos podem ser encontrado no estudo proposto por de Mello e outros [de Mello et al., 2014].

5.3.2 Metodologia

No caso desta pesquisa, o público-alvo é composto por profissionais devotados em desenvolvimento e manutenção de software. O perfil do participante da população de interesse é bastante geral, uma vez que um conjunto de características e práticas deste tipo de profissional é bastante diversa e pode depender de questões como processo de software utilizado, linguagem de programação utilizada, tipo de projeto no qual está envolvido, dentre outras. Assim, todos os profissionais que trabalham em projetos de software, sejam estes projetos de código aberto ou de empresas privadas, podem potencialmente contribuir com esta investigação. É importante destacar que cabe ao Plano de Pesquisa avaliar quando a relevância do participante utilizando, por exemplo, o nível de experiência do respondente como critério de inclusão. As subseções a seguir descrevem a estratégia de recrutamento projetada para a pesquisa com profissionais realizada neste estudo.

5.3.2.1 Fonte de Amostragem, Unidade de Pesquisa e População

Para a realização deste estudo foi estabelecida 05 diferentes Fontes de Amostragem conforme exibido na Tabela ???. Estas fontes foram selecionadas de modo a incluir profis-

sionais que estão se dedicam a projetos de código aberto (Python, Eclipse, MantisBT) e aqueles que possivelmente façam parte de empresas privadas de desenvolvimento e manutenção de software. Para o primeiro grupo, escolhemos projetos de código aberto que tivessem pelo menos 5 anos de existência, possuem uma comunidade bem estabelecida e permitam acesso aos dados de suas FGRM's. Além disso, escolhemos projetos que utilizem diferentes FGRM's como o objetivo de abarcar diferentes experiências de uso deste tipo de ferramenta. Para alcançarmos os profissionais que trabalham em empresas privadas de desenvolvimento de software utilizamos as redes sociais LinkedIn e Stack Overflow. Tais redes foram selecionadas especialmente por conta de sua cobertura, que contam com e um mais de 6 milhões de usuários para o caso do Stack Overflow ⁴ e mais de 10 milhões de profissionais de TI espalhados pelo mundo, quando falamos do LinkeIn [de Mello et al., 2014].

Fonte de Amostragem	URL	Membros
Python	https://bugs.python.org/	~19 K
Eclipse Foundation	https://bugs.eclipse.org/bugs/	~2.6 K
MantisBT	http://www.mantisbt.org/bugs/my_view_page.php	~40 K
LinkeIn	https://www.linkedin.com/	~347 M
Stack Overflow	http://stackoverflow.com/	~6.6 M

Tabela 5.1: Fontes de Amostragem utilizadas no estudo.

No escopo deste estudo a lista de Requisições de Mudanças no projetos de código aberto serão a Unidade de Pesquisa a ser considerada. Desde que o LinkedIn permite realizar pesquisa abrangente de “grupo de interesse”, serão estes as Unidades de Pesquisa. No caso da rede social Stack Overflow foi considerada cada discussão proposta por um usuário (thread) foi considerada como a Unidade de Pesquisa. Em todas as Fontes de Amostragem foram coletados os seguintes atributos: Nome do Participante, E-mail do Participante, Data de Interação e Tipo de Interação. No caso do LinkedIn utilizamos um métrica adicional da própria rede social conhecida como reputação ⁵ que é uma medida aproximada de quanto a comunidade poderia confiar em você. A medida é calculada quando o usuário com base nas ações do usuário e em como a comunidade avalia tais ações. Neste trabalho a métrica foi utilizada para verificar a frequência de participação de determinado usuário em discussões sobre manutenção ou desenvolvimento de software. Em resumo, podemos considerar que as pessoas que fazem parte dos projetos de código aberto, os participantes de grupo de interesse do LinkeIn e das discussões do Stack Overflow que compõe a população a ser considerada neste estudo.

⁴<http://stackexchange.com/sites>

⁵<http://stackoverflow.com/help/whats-reputation>

5.3.2.2 Unidade de Observação e Unidade de Análise

Nesta pesquisa, a Unidade de Observação e a Unidade de Análise são a mesma entidade (indivíduo) e cada membro distinto de cada grupo é potencialmente considerado uma unidade válida a ser amostrada. Os seguintes atributos devem ser coletados de cada um:

- Nome do Participante
- E-mail do Participante
- Data de Interação
- Tipo de Interação

O Tipo de Interação representa a ação que o participante realizou na Fonte de Amostragem, por exemplo relatar uma RM, finalizar uma RM, responder a uma pergunta e etc. Estes atributos foram utilizados para avaliar se determinada Unidade de Observação (indivíduo) será incluído na Fonte de Amostragem. Outros atributos foram coletados através do questionário de pesquisa (instrumento de medição): localização geográfica, tempo de experiência, nome da função desempenhada, principais atribuições, dentre outros.

5.3.2.3 Plano de Pesquisa

De modo a construir as Unidade de Pesquisa que foram utilizadas neste estudo, utilizamos três estratégias distintas cada uma relacionada as características da Fonte de Amostragem utilizada. Para as fontes que foram coletadas de projetos de código aberto (Python, Eclipse e MantisBT) utilizamos os registros históricos das RM's ocorridos nos últimos 05 anos. Além disso registro a frequência com o qual um participante teve contato com o projeto. Esta última métrica nos permitiu selecionar os participantes que tenham um mínimo de interação com projeto.

No caso do LinkedIn realizamos a busca por grupos de interesse utilizando as sentenças de busca descritas na Figura 5.2. Um conjunto similar de sentenças de busca foi utilizado no mapeamento sistemático descrito no Capítulo 3.

Para a rede social Stack Overflow utilizamos a busca oferecida pelo próprio site ⁶ de modo a encontrar discussões que se encaixam com as sentenças apresentadas na Figura 5.2. Neste contexto, visando restringir a seleção de grupos de participantes que estejam vinculados à desenvolvimento e manutenção de software realizamos a exclusão dos dados de participantes que:

⁶<http://data.stackexchange.com/>


```
("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 prioritization" OR  
"bug fix" OR "bug assignment" OR  
"bug reassignment" OR "bug triage" OR  
"duplicate bug" OR "reopened bug" OR  
"bug impact" OR "bug localization" OR  
"bug prediction" OR "bug risk" OR  
"bugseverity" OR "bug classification")
```

Figura 5.2: Sentenças utilizadas para escolhas dos grupos do LinkedIn e de discussões no Stack Overflow

- Proíbem expressamente a utilização dos seus dados, especialmente do seu endereço eletrônico, para a realização de estudos;
- A Fonte de Amostragem ao qual pertence não possui um mínimo de 05 anos de registros
- Para os grupos do LinkedIn e Stack Overflow, aqueles que restringem explicitamente a mensagem individual entre seus membros;
- Utilizam uma língua diferente do inglês, tendo em vista que o idioma inglês é padrão em fóruns internacionais e apenas existiam uma versão em inglês e português para o questionário utilizados.

5.3.2.4 Estratégia de amostragem

Apesar das Fontes de Amostras terem sido extraídos de diferentes locais, pode ocorrer que uma sobreposição de participantes, ou seja, um mesmo participante estar em duas ou mais fontes. Para minimizar a possibilidade da duplicação de participação de uma mesma unidade de observação realizamos um desenho de amostragem conhecido como Agrupada. Uma Amostragem agrupada pode ser aplicada quando grupos homogêneos (clusters) compostos por unidades distintas podem ser identificados numa população. Como consequência, devido a essa similaridade, apenas um subconjunto desta população pode ser utilizada como amostra de forma aleatória aleatoriamente sem perda significativa de confiança [Thompson, 2012]. Assim, a amostragem em agrupamentos é comumente aplicada em pesquisas em larga escala nas quais os pesquisadores têm restrições operacionais para recrutar e coletar dados [Roberts et al., 2004].

5.3.2.5 Extração de Dados

Para extrair os atributos necessários de cada participante desta pesquisa utilizamos três estratégias distintas. No caso da rede social Stack Overflow utilizamos uma ferramenta da web oficial que permite compartilhar, consultar e analisar os dados de todos os sites da rede Stack Exchange ⁷. A ferramenta possibilita a utilização da linguagem SQL para acesso aos dados e apresenta o respectivo modelo de dados para facilitar a consulta. A Figura [?]. A ferramenta permite a extração no formato CSV (Comma Separated Values) o qual foi inserido em um banco de dados para posterior aplicação das regras de inclusão e exclusão.

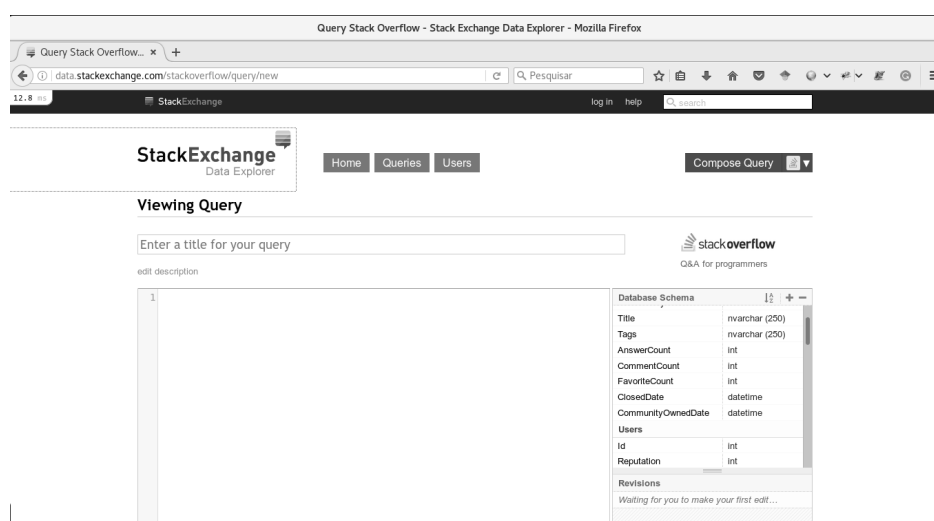


Figura 5.3: Ferramenta de coleta de dados da rede Stack Overflow

Para os projetos de código aberto (Python, Eclipse e MantisBT) foi desenvolvido um Web Crawler para coletar as informações dos participantes. Um Web Crawler (rastreador web) é um programa de computador que navega pela World Wide Web de uma forma metódica e automatizada. A partir de uma lista de Requisições de Mudança previamente coletadas a ferramenta coleta os dados dos participantes a partir do histórico de modificações de determinada RM. A Figura [?] apresenta o histórico de registros de uma RM do projeto Python onde os dados dos participantes podem ser visualizados nos quadros inseridos. A ferramenta utiliza uma marca HTML `<a>` e seu valor de classe (título, ou seja, nome de membro) para coletar os dados. De modo similar ao que foi realizado com os dados do Stack Overflow as informações extraídas foram armazenadas em um banco de dados para posterior aplicação de critérios de inclusão.

⁷<http://data.stackexchange.com/stackoverflow>

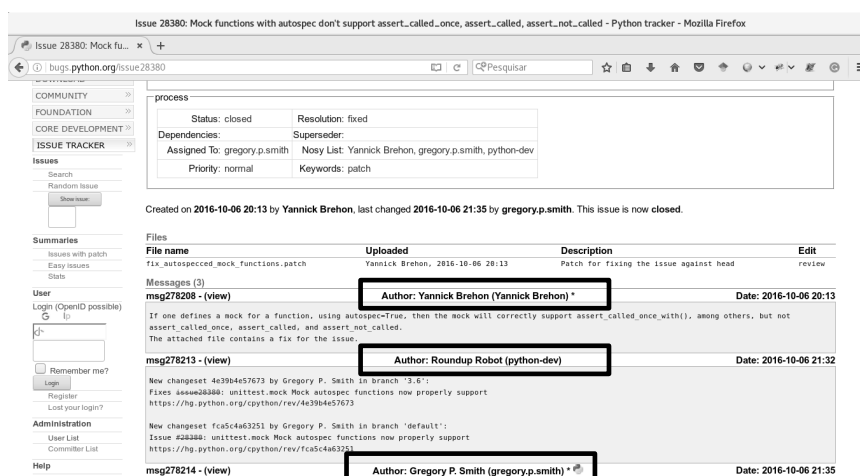


Figura 5.4: Histórico de relatos de uma RM do projeto Python

No caso da Fonte de Amostragem do LinkedIn utilizamos a busca já existente na ferramenta utilizando as sentenças de busca exibidas na Figura 5.2. Para extrair os atributos necessários de cada foi utilizada uma extensão do Firefox chamada iMacros⁸. Este plugin torna possível selecionar valores de tags HTML e salvá-los como arquivo CSV (Comma Separated Values). Além disso ele permite o envio de mensagem aos membros de forma automatizada.

Os dados coletados nas diferentes Fontes de Amostragem foi inicialmente extraídos e salvos no formato CSV. Todavia, o arquivo CSV contém muitos dados indesejados e artefatos de texto que devem ser removidos. Neste caso, foi necessário a realização de um processo de limpeza e transformação das informações antes do envio propriamente dito aos participantes.

5.3.2.6 Questionário

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

⁸<http://imacros.net/>

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 - PRODABEL⁹
- (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.

5.3.2.7 Envio da Mensagem

A fim de viabilizar e mitigar os riscos operacionais do envio manual de mensagens ao participantes foi desenvolvido um processo automatizado de envio de mensagens ao participantes. Excetuando o caso do grupos do LinkedIn, onde a mensagem foi enviada utilizado uma funcionalidade própria da ferramentas, o demais participantes foram convidados mediante correio eletrônico. O processo de envio seguiu uma política que consiste em enviar uma mensagem ao participante com base em um modelo. Após um um prazo de dois dias uma mensagem de lembrete. As mensagens foram preenchidas (uma a uma) e enviadas através de correio eletrônico para cada um dos participantes com base no seguinte modelo:

⁹<http://www.prodabel.pbh.gov.br>

Dear {{ nome do participante }}

I'm Vagner Clementino (homepages.dcc.ufmg.br/~vagnercs), Master Student at Federal University of Minas Gerais, Brazil. I'm conducting a research, supervised by ProfRodolfo Resende - homepages.dcc.ufmg.br/~rodolfo concerned with improvements in Issue Tracking System. As part of them, we planned and executed a survey aiming at to reach a large-scale population of researchers/practitioners interested on to improve the features of the Issue Tracking Systems. Based on your area of interest, we kindly invite you to take part in the following survey:

{{url do formulario}}

You was chosen because your relevant participation/contribution in {{nome da fonte de amostragem}}- {{url da fonte de amostragem}}. Your opinion is essential to strength our findings. Please, help us accordingly your possibilities by answering this survey until January 29th. As soon as we conclude data analysis, we will share the results with all participants and the software engineering community. If you have already fulfilled this questionnaire, please ignore this email.

Thanks in advance, Vagner Clementino

5.3.3 População, Amostra e Respostas

5.4 Resultados

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]

5.4.1 Perfil dos Participantes

5.5 Discussão

5.6 Ameças à Validade

5.6.0.1 Ameças Internas

5.6.0.2 Ameças 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

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

- [442, 2004] (2004). Guide to the software engineering body of knowledge 2004 version. *SWEBOK 2004 Guide to the Software Engineering Body of Knowledge*.
- [Aggarwal et al., 2014] Aggarwal, A.; Waghmare, G. & Sureka, A. (2014). Mining issue tracking systems using topic models for trend analysis, corpus exploration, and understanding evolution. Em *Proceedings of the 3rd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering*, RAISE 2014, pp. 52--58, New York, NY, USA. ACM.
- [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.
- [Aljarah et al., 2011] Aljarah, I.; Banitaan, S.; Abufardeh, S.; Jin, W. & Salem, S. (2011). Selecting discriminating terms for bug assignment: a formal analysis. Em *Proceedings of the 7th International Conference on Predictive Models in Software Engineering*, p. 12. ACM.
- [Antoniol et al., 2008] Antoniol, G.; Ayari, K.; Di Penta, M.; Khomh, F. & Guéhéneuc, Y.-G. (2008). Is it a bug or an enhancement?: a text-based approach to classify change requests. Em *Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds*, p. 23. ACM.
- [Anvik et al., 2005] Anvik, J.; Hiew, L. & Murphy, G. C. (2005). Coping with an open bug repository. Em *Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange*, pp. 35--39. ACM.
- [Baeza-Yates et al., 1999] Baeza-Yates, R.; Ribeiro-Neto, B. et al. (1999). *Modern Information Retrieval*, volume 463. ACM press New York.

- [Banerjee et al., 2012] Banerjee, S.; Cukic, B. & Adjeroh, D. (2012). Automated duplicate bug report classification using subsequence matching. Em *High-Assurance Systems Engineering (HASE), 2012 IEEE 14th International Symposium on*, pp. 74–81. IEEE.
- [Bangcharoensap et al., 2012] Bangcharoensap, P.; Ihara, A.; Kamei, Y. & Matsumoto, K.-i. (2012). Locating source code to be fixed based on initial bug reports—a case study on the eclipse project. Em *Empirical Software Engineering in Practice (IWESEP), 2012 Fourth International Workshop on*, pp. 10–15. IEEE.
- [Banitaan & Alenezi, 2013] Banitaan, S. & Alenezi, M. (2013). Decoba: Utilizing developers communities in bug assignment. Em *Machine Learning and Applications (ICMLA), 2013 12th International Conference on*, volume 2, pp. 66–71. IEEE.
- [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 mozilla 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, Reliability, and Information Technology (ICROIT), 2014 International Conference on*, pp. 294–299. IEEE.
- [Bennett & Rajlich, 2000] Bennett, K. H. & Rajlich, V. T. (2000). Software maintenance and evolution: A roadmap. Em *Proceedings of the Conference on The Future of Software Engineering, ICSE '00*, pp. 73–87, New York, NY, USA. ACM.
- [Bertram et al., 2010a] Bertram, D.; Volda, A.; Greenberg, S. & Walker, R. (2010a). Communication, collaboration, and bugs: The social nature of issue tracking in small, colocated teams. Em *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work, CSCW '10*, pp. 291–300, New York, NY, USA. ACM.

- [Bertram et al., 2010b] Bertram, D.; Voida, A.; Greenberg, S. & Walker, R. (2010b). 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*, pp. 291--300. ACM.
- [Bettenburg et al., 2008a] Bettenburg, N.; Just, S.; Schröter, A.; Weiss, C.; Premraj, R. & Zimmermann, T. (2008a). What makes a good bug report? Em *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pp. 308--318. ACM.
- [Bettenburg et al., 2008b] Bettenburg, N.; Premraj, R.; Zimmermann, T. & Kim, S. (2008b). Duplicate bug reports considered harmful... really? Em *Software maintenance, 2008. ICSM 2008. IEEE international conference on*, pp. 337--345. IEEE.
- [Bhattacharya & Neamtiu, 2011] Bhattacharya, P. & Neamtiu, I. (2011). Bug-fix time prediction models: can we do better? Em *Proceedings of the 8th Working Conference on Mining Software Repositories*, pp. 207--210. ACM.
- [Breu et al., 2010a] Breu, S.; Premraj, R.; Sillito, J. & Zimmermann, T. (2010a). 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.
- [Breu et al., 2010b] Breu, S.; Premraj, R.; Sillito, J. & Zimmermann, T. (2010b). Information needs in bug reports: improving cooperation between developers and users. Em *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, pp. 301--310. 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.
- [Cavalcanti et al., 2013] Cavalcanti, Y. C.; Neto, P. A. d. M. S.; Lucrédio, D.; Vale, T.; de Almeida, E. S. & de Lemos Meira, S. R. (2013). The bug report duplication problem: an exploratory study. *Software Quality Journal*, 21(1):39--66.
- [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.
- [Choudhari & Suman, 2014] Choudhari, J. & Suman, U. (2014). Extended iterative maintenance life cycle using extreme programming. *SIGSOFT Softw. Eng. Notes*, 39(1):1--12. ISSN 0163-5948.
- [Corley et al., 2011] Corley, C. S.; Kraft, N. A.; Etzkorn, L. H. & Lukins, S. K. (2011). Recovering traceability links between source code and fixed bugs via patch analysis. Em *Proceedings of the 6th International Workshop on Traceability in Emerging Forms of Software Engineering*, pp. 31--37. ACM.
- [Correa et al., 2013] Correa, D.; Lal, S.; Saini, A. & Sureka, A. (2013). Samekana: A browser extension for including relevant web links in issue tracking system discussion forum. Em *2013 20th Asia-Pacific Software Engineering Conference (APSEC)*, volume 1, pp. 25--33. IEEE.
- [Creswell & Clark, 2007] Creswell, J. W. & Clark, V. L. P. (2007). Designing and conducting mixed methods research.
- [Dal Sasso & Lanza, 2013] Dal Sasso, T. & Lanza, M. (2013). A closer look at bugs. Em *Software Visualization (VISSEFT), 2013 First IEEE Working Conference on*, pp. 1--4. IEEE.
- [Dal Sasso & Lanza, 2014] Dal Sasso, T. & Lanza, M. (2014). In* bug: Visual analytics of bug repositories. Em *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on*, pp. 415--419. IEEE.
- [Davidson et al., 2011] Davidson, J. L.; Mohan, N. & Jensen, C. (2011). Coping with duplicate bug reports in free/open source software projects. Em *VL/HCC*, volume 11, pp. 101--108.
- [de Mello et al., 2014] de Mello, R. M.; da Silva, P. C.; Runeson, P. & Travassos, G. H. (2014). Towards a framework to support large scale sampling in software engineering surveys. Em *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, p. 48. ACM.
- [de Mello et al., 2015] de Mello, R. M.; Da Silva, P. C. & Travassos, G. H. (2015). Investigating probabilistic sampling approaches for large-scale surveys in software engineering. *Journal of Software Engineering Research and Development*, 3(1):8.

- [Devulapally, 2015] Devulapally, G. K. (2015). Agile in the context of Software Maintainability.
- [Di Lucca et al., 2002] Di Lucca, G. A.; Di Penta, M. & Gradara, S. (2002). An approach to classify software maintenance requests. Em *Software Maintenance, 2002. Proceedings. International Conference on*, pp. 93--102. IEEE.
- [Dybå et al., 2007] Dybå, T.; Dingsøy, 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. *Omnex White Paper*. Accessed.
- [Fox et al., 2013] Fox, A.; Patterson, D. A. & Joseph, S. (2013). *Engineering software as a service: an agile approach using cloud computing*. Strawberry Canyon LLC.
- [Gegick et al., 2010] Gegick, M.; Rotella, P. & Xie, T. (2010). Identifying security bug reports via text mining: An industrial case study. Em *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, pp. 11--20. IEEE.
- [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.
- [Hesse-Biber, 2010] Hesse-Biber, S. N. (2010). *Mixed methods research: Merging theory with practice*. Guilford Press.
- [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.

- [Hora et al., 2012] Hora, A.; Anquetil, N.; Ducasse, S.; Bhatti, M.; Couto, C.; Valente, M. T. & Martins, J. (2012). Bug maps: A tool for the visual exploration and analysis of bugs. Em *Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on*, pp. 523--526. 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.
- [Hovemeyer & Pugh, 2004] Hovemeyer, D. & Pugh, W. (2004). Finding bugs is easy. *SIGPLAN Not.*, 39(12):92--106. ISSN 0362-1340.
- [Hu et al., 2014] Hu, H.; Zhang, H.; Xuan, J. & Sun, W. (2014). Effective bug triage based on historical bug-fix information. Em *2014 IEEE 25th International Symposium on Software Reliability Engineering*, pp. 122--132. 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, 2001] ISO/IEC (2001). *ISO/IEC 9126. Software engineering – Product quality*. ISO/IEC.
- [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.
- [ISO/IEEE, 1998] ISO/IEEE (1998). Ieee standard for software maintenance.
- [Izquierdo et al., 2015] Izquierdo, J. L. C.; Cosentino, V.; Rolandi, B.; Bergel, A. & Cabot, J. (2015). Gila: Github label analyzer. Em *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pp. 479--483. IEEE.

- [Just et al., 2008] Just, S.; Premraj, R. & Zimmermann, T. (2008). Towards the next generation of bug tracking systems. Em *2008 IEEE Symposium on Visual Languages and Human-Centric Computing*, pp. 82--85. IEEE.
- [Kagdi et al., 2012] Kagdi, H.; Gethers, M.; Poshyvanyk, D. & Hammad, M. (2012). Assigning change requests to software developers. *Journal of Software: Evolution and Process*, 24(1):3--33.
- [Kaiser & Passonneau, 2011] Kaiser, L. W. B. X. G. & Passonneau, R. (2011). Bug-miner: Software reliability analysis via data mining of bug reports. *delta*, 12(10):09-0500.
- [Kasunic, 2005] Kasunic, M. (2005). Designing an effective survey. Relatório técnico, DTIC Document.
- [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).
- [Kaushik & Tahvildari, 2012] Kaushik, N. & Tahvildari, L. (2012). A comparative study of the performance of ir models on duplicate bug detection. Em *Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on*, pp. 159--168. IEEE.
- [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.
- [Kochhar et al., 2014] Kochhar, P. S.; Thung, F. & Lo, D. (2014). Automatic fine-grained issue report reclassification. Em *Engineering of Complex Computer Systems (ICECCS), 2014 19th International Conference on*, pp. 126--135. IEEE.
- [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.

- [Koopaei & Hamou-Lhadj, 2015] Koopaei, N. E. & Hamou-Lhadj, A. (2015). Crashautomata: an approach for the detection of duplicate crash reports based on generalizable automata. Em *Proceedings of the 25th Annual International Conference on Computer Science and Software Engineering*, pp. 201--210. IBM Corp.
- [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.
- [Lerch & Mezini, 2013] Lerch, J. & Mezini, M. (2013). Finding duplicates of your yet unwritten bug report. Em *Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on*, pp. 69--78. IEEE.
- [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.
- [Liu et al., 2012] Liu, H.; Ma, Z.; Shao, W. & Niu, Z. (2012). Schedule of bad smell detection and resolution: A new way to save effort. *IEEE Transactions on Software Engineering*, 38(1):220--235.
- [Liu & Tan, 2014] Liu, K. & Tan, H. B. K. (2014). Faceted bug report search with topic model. Em *Computer Software and Applications Conference (COMPSAC), 2014 IEEE 38th Annual*, pp. 123--128. IEEE.
- [Maiden & Rugg, 1996] Maiden, N. A. & Rugg, G. (1996). Acre: selecting methods for requirements acquisition. *Software Engineering Journal*, 11(3):183--192.
- [Malheiros et al., 2012] Malheiros, Y.; Moraes, A.; Trindade, C. & Meira, S. (2012). A source code recommender system to support newcomers. Em *2012 IEEE 36th Annual Computer Software and Applications Conference*, pp. 19--24. IEEE.
- [Mani et al., 2012] Mani, S.; Catherine, R.; Sinha, V. S. & Dubey, A. (2012). Ausum: approach for unsupervised bug report summarization. Em *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, p. 11. ACM.

- [McGee & Greer, 2009] McGee, S. & Greer, D. (2009). A software requirements change source taxonomy. Em *Software Engineering Advances, 2009. ICSEA'09. Fourth International Conference on*, pp. 51--58. IEEE.
- [Moran, 2015] Moran, K. (2015). Enhancing android application bug reporting. Em *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pp. 1045--1047. ACM.
- [Moran et al., 2015] Moran, K.; Linares-Vásquez, M.; Bernal-Cárdenas, C. & Poshyanyk, D. (2015). Auto-completing bug reports for android applications. Em *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pp. 673--686. ACM.
- [Naguib et al., 2013] Naguib, H.; Narayan, N.; Brügge, B. & Helal, D. (2013). Bug report assignee recommendation using activity profiles. Em *Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on*, pp. 22--30. IEEE.
- [Nagwani et al., 2013] Nagwani, N.; Verma, S. & Mehta, K. K. (2013). Generating taxonomic terms for software bug classification by utilizing topic models based on latent dirichlet allocation. Em *ICT and Knowledge Engineering (ICT&KE), 2013 11th International Conference on*, pp. 1--5. IEEE.
- [Nagwani & Verma, 2010a] Nagwani, N. K. & Verma, S. (2010a). Predictive data mining model for software bug estimation using average weighted similarity. Em *Advance Computing Conference (IACC), 2010 IEEE 2nd International*, pp. 373--378. IEEE.
- [Nagwani & Verma, 2010b] Nagwani, N. K. & Verma, S. (2010b). Predictive data mining model for software bug estimation using average weighted similarity. Em *Advance Computing Conference (IACC), 2010 IEEE 2nd International*, pp. 373--378. IEEE.
- [Nagwani & Verma, 2012] Nagwani, N. K. & Verma, S. (2012). Predicting expert developers for newly reported bugs using frequent terms similarities of bug attributes. Em *2011 Ninth International Conference on ICT and Knowledge Engineering*, pp. 113--117. IEEE.
- [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.

- [Nguyen et al., 2012] Nguyen, A. T.; Nguyen, T. T.; Nguyen, H. A. & Nguyen, T. N. (2012). Multi-layered approach for recovering links between bug reports and fixes. Em *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, FSE '12, pp. 63:1--63:11, New York, NY, USA. ACM.
- [Otoom et al., 2016] Otoom, A. F.; Al-Shdaifat, D.; Hammad, M. & Abdallah, E. E. (2016). Severity prediction of software bugs. Em *2016 7th International Conference on Information and Communication Systems (ICICS)*, pp. 92--95. IEEE.
- [Paulk et al., 1993] Paulk, M. C.; Weber, C. V.; Garcia, S. M.; Chrissis, M. B. C. & Bush, M. (1993). Key practices of the capability maturity model version 1.1.
- [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.
- [Pfleeger & Kitchenham, 2001] Pfleeger, S. L. & Kitchenham, B. A. (2001). Principles of survey research: part 1: turning lemons into lemonade. *ACM SIGSOFT Software Engineering Notes*, 26(6):16--18.
- [Pfleeger & Kitchenham, 2002] Pfleeger, S. L. & Kitchenham, B. A. (2002). Principles of survey research part 2: designing a survey. *Software Engineering Notes*, 27(1):18--20.
- [Pigoski, 1996] Pigoski, T. M. (1996). *Practical software maintenance: best practices for managing your software investment*. Wiley Publishing.
- [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.

- [Prifti et al., 2011] Prifti, T.; Banerjee, S. & Cukic, B. (2011). Detecting bug duplicate reports through local references. Em *Proceedings of the 7th International Conference on Predictive Models in Software Engineering*, p. 8. ACM.
- [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.
- [Roberts et al., 2004] Roberts, L.; Lafta, R.; Garfield, R.; Khudhairi, J. & Burnham, G. (2004). Mortality before and after the 2003 invasion of iraq: cluster sample survey. *The Lancet*, 364(9448):1857--1864.
- [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).
- [Romo & Capiluppi, 2015] Romo, B. A. & Capiluppi, A. (2015). Towards an automation of the traceability of bugs from development logs: A study based on open source software. Em *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*, EASE '15, pp. 33:1--33:6, New York, NY, USA. ACM.
- [Rugg & McGeorge, 2005] Rugg, G. & McGeorge, P. (2005). The sorting techniques: a tutorial paper on card sorts, picture sorts and item sorts. *Expert Systems*, 22(3):94-107.
- [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.
- [Serrano & Ciordia, 2005] Serrano, N. & Ciordia, I. (2005). Bugzilla, itracker, and other bug trackers. *IEEE Software*, 22(2):11--13. ISSN 0740-7459.
- [Shokripour et al., 2012] Shokripour, R.; Kasirun, Z. M.; Zamani, S. & Anvik, J. (2012). Automatic bug assignment using information extraction methods. Em *Advanced Computer Science Applications and Technologies (ACSAT), 2012 International Conference on*, pp. 144--149. IEEE.

- [Society et al., 2014] Society, I. C.; Bourque, P. & Fairley, R. E. (2014). *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*. IEEE Computer Society Press, Los Alamitos, CA, USA, 3rd edição. ISBN 0769551661, 9780769551661.
- [Soltan & Mostafa, 2016] Soltan, H. & Mostafa, S. (2016). Leanness and Agility within Maintenance Process. (January 2014).
- [Somasundaram & Murphy, 2012] Somasundaram, K. & Murphy, G. C. (2012). Automatic categorization of bug reports using latent dirichlet allocation. Em *Proceedings of the 5th India software engineering conference*, pp. 125--130. ACM.
- [Song et al., 2010a] Song, Y.; Wang, X.; Xie, T.; Zhang, L. & Mei, H. (2010a). Jdf: detecting duplicate bug reports in jazz. Em *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*, pp. 315--316. ACM.
- [Song et al., 2010b] Song, Y.; Wang, X.; Xie, T.; Zhang, L. & Mei, H. (2010b). Jdf: detecting duplicate bug reports in jazz. Em *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*, pp. 315--316. ACM.
- [Sun et al., 2011] Sun, C.; Lo, D.; Khoo, S.-C. & Jiang, J. (2011). Towards more accurate retrieval of duplicate bug reports. Em *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*, pp. 253--262. IEEE Computer Society.
- [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.

- [Thompson, 2012] Thompson, S. (2012). *Sampling*. CourseSmart. Wiley. ISBN 9781118162941.
- [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*, pp. 871--874. 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.
- [Thung et al., 2013] Thung, F.; Lo, D. & Jiang, L. (2013). Automatic recovery of root causes from bug-fixing changes. Em *2013 20th Working Conference on Reverse Engineering (WCRE)*, pp. 92--101. IEEE.
- [Thung et al., 2012a] Thung, F.; Lo, D.; Jiang, L. et al. (2012a). Are faults localizable? Em *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on*, pp. 74--77. IEEE.
- [Thung et al., 2012b] Thung, F.; Lo, D.; Jiang, L.; Rahman, F.; Devanbu, P. T. et al. (2012b). When would this bug get reported? Em *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, pp. 420--429. IEEE.
- [Tian et al., 2013] Tian, Y.; Lo, D. & Sun, C. (2013). Drone: Predicting priority of reported bugs by multi-factor analysis.
- [Tian et al., 2015] Tian, Y.; Lo, D.; Xia, X. & Sun, C. (2015). Automated prediction of bug report priority using multi-factor analysis. *Empirical Software Engineering*, 20(5):1354--1383.
- [Tian et al., 2012a] Tian, Y.; Sun, C. & Lo, D. (2012a). Improved duplicate bug report identification. Em *Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on*, pp. 385--390. IEEE.
- [Tian et al., 2012b] Tian, Y.; Sun, C. & Lo, D. (2012b). Improved duplicate bug report identification. Em *Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on*, pp. 385--390. IEEE.

- [Tomašev et al., 2013] Tomašev, N.; Leban, G. & Mladenić, D. (2013). Exploiting hubs for self-adaptive secondary re-ranking in bug report duplicate detection. Em *Information Technology Interfaces (ITI), Proceedings of the ITI 2013 35th International Conference on*, pp. 131--136. IEEE.
- [Tu & Zhang, 2014] Tu, F. & Zhang, F. (2014). Measuring the quality of issue tracking data. Em *Proceedings of the 6th Asia-Pacific Symposium on Internetware on Internetware*, pp. 76--79. ACM.
- [Valdivia Garcia & Shihab, 2014] Valdivia Garcia, H. & Shihab, E. (2014). Characterizing and predicting blocking bugs in open source projects. Em *Proceedings of the 11th working conference on mining software repositories*, pp. 72--81. ACM.
- [Vijayakumar & Bhuvaneswari, 2014] Vijayakumar, K. & Bhuvaneswari, V. (2014). How much effort needed to fix the bug? a data mining approach for effort estimation and analysing of bug report attributes in firefox. Em *Intelligent Computing Applications (ICICA), 2014 International Conference on*, pp. 335--339. IEEE.
- [Wang & Sarma, 2011] Wang, J. & Sarma, A. (2011). Which bug should i fix: helping new developers onboard a new project. Em *Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering*, pp. 76--79. ACM.
- [White et al., 2015a] White, M.; Linares-Vásquez, M.; Johnson, P.; Bernal-Cárdenas, C. & Poshyvanyk, D. (2015a). Generating reproducible and replayable bug reports from android application crashes. Em *2015 IEEE 23rd International Conference on Program Comprehension*, pp. 48--59. IEEE.
- [White et al., 2015b] White, M.; Linares-Vásquez, M.; Johnson, P.; Bernal-Cárdenas, C. & Poshyvanyk, D. (2015b). Generating reproducible and replayable bug reports from android application crashes. Em *2015 IEEE 23rd International Conference on Program Comprehension*, pp. 48--59. IEEE.
- [Wohlin, 2014] Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. Em *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, p. 38. 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.

- [Wong et al., 2014] Wong, C.-P.; Xiong, Y.; Zhang, H.; Hao, D.; Zhang, L. & Mei, H. (2014). Boosting bug-report-oriented fault localization with segmentation and stack-trace analysis. Em *ICSME*, pp. 181--190. Citeseer.
- [Wu et al., 2011] Wu, W.; Zhang, W.; Yang, Y. & Wang, Q. (2011). Drex: Developer recommendation with k-nearest-neighbor search and expertise ranking. Em *2011 18th Asia-Pacific Software Engineering Conference*, pp. 389--396. IEEE.
- [Xia et al., 2015] Xia, X.; Lo, D.; Shihab, E.; Wang, X. & Zhou, B. (2015). Automatic, high accuracy prediction of reopened bugs. *Automated Software Engineering*, 22(1):75--109.
- [Xuan et al., 2012] Xuan, J.; Jiang, H.; Ren, Z. & Zou, W. (2012). Developer prioritization in bug repositories. Em *2012 34th International Conference on Software Engineering (ICSE)*, pp. 25--35. IEEE.
- [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.
- [Zanetti et al., 2013] Zanetti, M. S.; Scholtes, I.; Tessone, C. J. & Schweitzer, F. (2013). Categorizing bugs with social networks: a case study on four open source software communities. Em *Proceedings of the 2013 International Conference on Software Engineering*, pp. 1032--1041. IEEE Press.
- [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.
- [Zhang et al., 2014] Zhang, W.; Han, G. & Wang, Q. (2014). Butter: An approach to bug triage with topic modeling and heterogeneous network analysis. Em *Cloud Computing and Big Data (CCBD), 2014 International Conference on*, pp. 62--69. IEEE.

- [Zimmermann et al., 2010] Zimmermann, T.; Premraj, R.; Bettenburg, N.; Just, S.; Schroter, A. & Weiss, C. (2010). What makes a good bug report? *IEEE Transactions on Software Engineering*, 36(5):618--643.
- [Zimmermann et al., 2009a] Zimmermann, T.; Premraj, R.; Sillito, J. & Breu, S. (2009a). Improving bug tracking systems. Em *ICSE Companion*, pp. 247--250. Citeseer.
- [Zimmermann et al., 2009b] Zimmermann, T.; Premraj, R.; Sillito, J. & Breu, S. (2009b). Improving bug tracking systems. Em *Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on*, pp. 247--250.
- [Zimmermann et al., 2005] Zimmermann, T.; Zeller, A.; Weissgerber, P. & Diehl, S. (2005). Mining version histories to guide software changes. *Software Engineering, IEEE Transactions on*, 31(6):429--445.

Apêndice A

Instrumentos do Mapeamento Sistemático

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

Base de Dados	Sentença 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 prioritization" OR "bug fix" OR "bug assignment" OR "bug reassignment" 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 prioritization" OR "Document Title":"bug fix" OR "Document Title":"bug assignment" OR "Document Title":"bug reassignment" 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 prioritization" OR "bug fix" OR "bug assignment" OR "bug reassignment" 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 prioritization" OR "bug fix" OR "bug assignment" OR "bug reassignment" 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"))

Apêndice B

Lista de Ferramenta de Gerenciamento de Requisição de mudanças

Tabela B.1: Lista de Ferramentas

Nome	Licença	Lançamento
Apache Bloodhound	Apache License	2012
Assembla Tickets	Proprietary, hosted. Available for f	2008
Axosoft	Proprietary, Saas	2002
BMC Remedy Action Request System	Proprietary	1992
Bontq	Proprietary, hosted.	2010
Brimir	AGPL	2013
Bugzilla	MPL	1998
Debbugs	GPL	1994
FogBugz	Saas	2000
Fossil SCM	BSD	2006
FusionForge	GPLv2	2009
Gestionnaire libre de parc informatique	GPLv2	2003
GNATS	GPL	1992
Google Code Hosting	Proprietary, hosted; available for	2004
HP Quality Center	Proprietary	1995
IBM Rational ClearQuest	Proprietary	1998
IBM Rational Team Concert	Proprietary	2008
JIRA	Proprietary. Free community licen	2002
Kayako SupportSuite	Proprietary, some parts GPL	2001
Launchpad	AGPL	2004
Liberum Help Desk	GPL	2000
MantisBT	GPL	2000
Microsoft Dynamics CRM	Proprietary, Commercial	2003
org-mode	GPL	2003
Open-source Ticket Request System	AGPL	2002
Pivotal Tracker	Proprietary, free version for public	2008
Plain Ticket	Proprietary, online, hosted.	2011
Planbox	Proprietary, free version	2009
QuickBase	Proprietary	2000
Redmine	GPLv2	2006
Request Tracker	GPLv2	1999
Roundup	MIT license (ZPL v 2.0 for the tem	2001
StarTeam	Proprietary	2011
Supportworks	Proprietary	1994
SysAid	Proprietary	2002
Targetprocess	Proprietary	2005
Team Foundation Server	Proprietary, Commercial	2005
Twproject	Proprietary, some parts LGPL	2003
TechExcel's DevTrack	Proprietary	1997
TestTrack	Proprietary	1996
The Bug Genie	Mozilla Public License 1.1	2003
Trac Bug Tracking System	New BSD	2006
TrackerSuite.Net	Proprietary	2006
Tuleap	GPLv2	2011
Usersnap Bug Tracking System	Proprietary	2013
Web Help Desk	Proprietary	1999
Wrike Project management software	Proprietary, hosted	2006
YouTrack	Proprietary, stand-alone and hoste	2009
Zoho BugTracker	Proprietary	2011

Apêndice C

Formulário Aplicado para Seleção de Ferramentas

Tabela C.1: Lista de Ferramentas

Nome	Licença	Lançamento
Apache Bloodhound	Apache License	2012
Assembla Tickets	Proprietary, hosted. Available for f	2008
Axosoft	Proprietary, Saas	2002
BMC Remedy Action Request System	Proprietary	1992
Bontq	Proprietary, hosted.	2010
Brimir	AGPL	2013
Bugzilla	MPL	1998
Debbugs	GPL	1994
FogBugz	Saas	2000
Fossil SCM	BSD	2006
FusionForge	GPLv2	2009
Gestionnaire libre de parc informatique	GPLv2	2003
GNATS	GPL	1992
Google Code Hosting	Proprietary, hosted; available for	2004
HP Quality Center	Proprietary	1995
IBM Rational ClearQuest	Proprietary	1998
IBM Rational Team Concert	Proprietary	2008
JIRA	Proprietary. Free community licen	2002
Kayako SupportSuite	Proprietary, some parts GPL	2001
Launchpad	AGPL	2004
Liberum Help Desk	GPL	2000
MantisBT	GPL	2000
Microsoft Dynamics CRM	Proprietary, Commercial	2003
org-mode	GPL	2003
Open-source Ticket Request System	AGPL	2002
Pivotal Tracker	Proprietary, free version for public	2008
Plain Ticket	Proprietary, online, hosted.	2011
Planbox	Proprietary, free version	2009
QuickBase	Proprietary	2000
Redmine	GPLv2	2006
Request Tracker	GPLv2	1999
Roundup	MIT license (ZPL v 2.0 for the tem	2001
StarTeam	Proprietary	2011
Supportworks	Proprietary	1994
SysAid	Proprietary	2002
Targetprocess	Proprietary	2005
Team Foundation Server	Proprietary, Commercial	2005
Twproject	Proprietary, some parts LGPL	2003
TechExcel's DevTrack	Proprietary	1997
TestTrack	Proprietary	1996
The Bug Genie	Mozilla Public License 1.1	2003
Trac Bug Tracking System	New BSD	2006
TrackerSuite.Net	Proprietary	2006
Tuleap	GPLv2	2011
Usersnap Bug Tracking System	Proprietary	2013
Web Help Desk	Proprietary	1999
Wrike Project management software	Proprietary, hosted	2006
YouTrack	Proprietary, stand-alone and hosted	2009
Zoho BugTracker	Proprietary	2011