Pós-Graduação em Ciência da Computação

"A Bug Report Analysis and Search Tool"

By

Yguaratã Cerqueira Cavalcanti

M.Sc. Dissertation

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

RECIFE, JULY/2009

Universidade Federal de Pernambuco
Centro de Informática
Pós-graduação em Ciência da Computação

Yguaratã Cerqueira Cavalcanti

"A Bug Report Analysis and Search Tool"

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

A M.Sc. Dissertation presented to the Federal University of Pernambuco in partial fulfillment of the requirements for the degree of M.Sc. in Computer Science.

Advisor: *Silvio Romero de Lemos Meira*
Co-Advisor: *Eduardo Santana de Almeida*

RECIFE, JULY/2009

I dedicate this dissertation to myself and all my family, friends and professors who gave me all necessary support to get here.

Acknowledgements

I would like to thank and dedicate this dissertation to the following people: ...

I open my eyes each morning I rise, to find a true thought, know that it's real, I'm lucky to breathe, I'm lucky to feel, I'm glad to wake up, I'm glad to be here, with all of this world, and all of it's pain, all of it's lies, and all of it's flipped down, I still feel a sense of freedom, so glad I'm around,

It's my freedom, can't take it from me, i know it, it won't change, but we need some understanding, I know we'll be all right.

—S.O.JA. (Open My Eyes)

Resumo

Manutenção e evolução de *software* são atividades caracterizadas pelo seu enorme custo e baixa velocidade de execução. Não obstante, elas são atividades inevitáveis para garantir a qualidade do *software* – quase todo *software* bem sucedido estimula os usuários a fazer pedidos de mudanças e melhorias. Sommerville é ainda mais enfático e diz que mudanças em projetos de *software* são um fato. Além disso, diferentes estudos têm afirmado ao longo dos anos que as atividades de manutenção e evolução de *software* são as mais caras do ciclo de desenvolvimento, sendo responsável por cerca de até 90% dos custos.

Todas essas peculiaridades da fase de manutenção e evolução de *software* leva o mundo acadêmico e industrial a investigar constantemente novas soluções para reduzir os custos dessas atividades. Neste contexto, Gerência de Configuração de Software (GCS) é um conjunto de atividades e normas para a gestão da evolução e manutenção de *software*; GCS define como são registradas e processadas todas as modificações, o impacto das mesmas em todo o sistema, dentre outros procedimentos. Para todas estas tarefas de GCM existem diferentes ferramentas de auxílio, tais como sistemas de controle de versão e *bug trackers*. No entanto, alguns problemas podem surgir devido ao uso das mesmas, como por exemplo o problema de atribuição automática de responsável por um *bug report* e o problema de duplicação de *bug reports*.

Neste sentido, esta dissertação investiga o problema de duplicação de *bug reports* resultante da utilização de *bug trackers* em projetos de desenvolvimento de *software*. Tal problema é caracterizado pela submissão de dois ou mais *bug reports* que descrevem o mesmo problema referente a um *software*, tendo como principais conseqüências a sobrecarga de trabalho na busca e análise de *bug reports*, e o mal aproveitamento do tempo destinado a essa atividade.

Palavras-chave: relatos de bug, gerenciadores de relatos de bug, relatos de bug duplicados, requisição de mudança, experimento, estudo de caracterização, ferramenta, busca

Abstract

Software maintenance and evolution are characterised by their huge cost and slow speed of implementation. Yet they are inevitable activities – almost all software that is useful and successful stimulates user-generated requests for change and improvements. Sommerville is even more emphatic and says that software changes is a fact of life for large software systems. In addition, a set of studies has stated along the years that software maintenance and evolution is the most expensive phase of software development, taking up to 90% of the total costs.

All those characteristics from software maintenance lead the academia and industry to constantly investigate new solutions to reduce costs in such phase. In this context, Software Configuration Management (SCM) is a set of activities and standards for managing and evolving software; SCM defines how to record and process proposed system changes, how to relate these to system components, among other procedures. For all these tasks it has been proposed different tools, such as version control systems and bug trackers. However, some issues may arise due to these tools usage, such as the dynamic assignment of a developer to a bug report or the bug report duplication problem.

In this sense, this dissertation investigates the problem of bug report duplication emerged by the use of bug trackers on software development projects. The problem of bug report duplication is characterized by the submission of two or more bug reports that describe the same software issue, and the main consequence of this problem is the overhead of rework when managing these bug reports.

Keywords: bug reports, bug trackers, bug report duplication, change request, tool experiment, bug report duplication characterization study, bug report search and analysis tool

Contents

List of Figures

List of Tables

Acronyms

AJAX	Asynchronous JavaScript and XML
BAST	Bug Report Analysis and Search Tool
BTT	Bug Report Tracker Tool
BRN	Bug Report Network
CCB	Change Control Board
C.E.S.A.R.	Recife Center For Advanced Studies and Systems C.E.S.A.R. (http://www.cesar.org.br) is a CMMi level 3 company with around 700 employees
FR	Functional Requirement
GQM	Goal Question Metric
LOC	Lines of Code
NFR	Non-Functional Requirement
NLP	Natural Language Processing
ORM	Object-Relational Mapper
RiSE	Reuse in Software Engineering Group http://www.rise.com.br
SCM	Software Configuration Management
SD	Standard Deviation
TF-IDF	Term Frequency-Inverse Document Frequency
UFPE	Federal University of Pernambuco
VSM	Vector Space Model
WAD	Work as Design
XP	eXtreme Programming

1 Introduction

Um passo à frente e você não está mais no mesmo lugar
One step forward and you are not in the same place
—CHICO SCIENCE (Um Passeio No Mundo Livre, Afrociberdelia)

Software maintenance and evolution are characterised by their huge cost and slow speed of implementation. However they are inevitable activities – almost all software that is useful and successful stimulates user-generated requests for change and improvements (Bennett and Rajlich, 2000). Sommerville (Sommerville, 2007) is even more emphatic and says that software changes is a fact of life for large software systems. In addition, a set of studies (Huff, 1990; Moad, 1990; Eastwood, 1993; Erlikh, 2000) has stated along the years that software maintenance and evolution is the most expensive phase of software development, taking up to 90% of the total costs.

All of these characteristics from software maintenance leaded the academia and industry to investigate constantly new solutions to reduce costs in such phase. In this context, Software Configuration Management (SCM) is a set of activities and standards for managing and evolving software, defining how to record and process the proposed system changes, how to relate these to system components, among other procedures. For all these tasks, it has proposed different tools, such as version control systems and bug trackers (Sommerville, 2007). However, some issues may arise due to these tools usage. In this work, the focus are the issues from bug trackers, as it will be discussed along this dissertation.

The remainder of this chapter describes the focus of this dissertation and starts by presenting its motivation in Section 1.1 and a clear definition of the problem in Section 1.2. An overview of the proposed solution is presented in Section 1.3, while

Section 1.4 describes some related aspects that are not directly addressed by this work. Section 1.5 presents the main contributions and, finally, Section **??** describes how this dissertation is organized.

1.1 Motivation

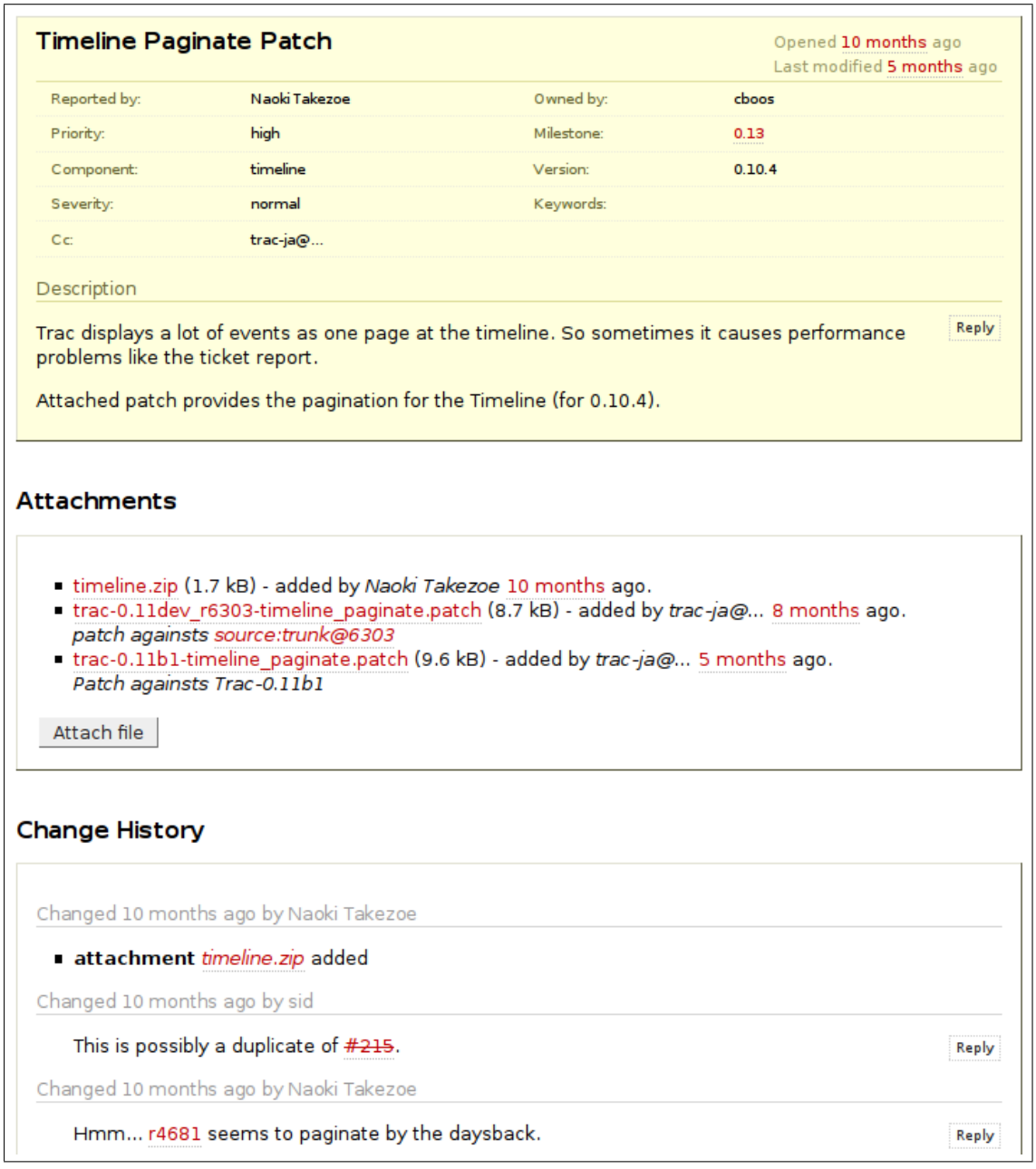
Aiming to improve change management processes, some organizations have used specific systems (generally called *bug-trackers*) to manage, store and handle change requests (also known as *bug reports*). A bug report is defined as a software artifact that describes some defect, enhancement, change request, or an issue in general, that is submitted to a bug tracker; generally, bug report submitters are developers, users, or testers. Such systems are useful because changes to be made in a software can be quickly identified and submitted to the appropriate people (Anvik *et al.*, 2005).

Moreover, the use of bug trackers helps to monitor the software evolution, because bug reports are recorded in a database as well as people involved in a particular bug report are recorded. Thus, changes and their respective responsible can be easily found. Organizations also use such systems to guide the development of software, thus any task to be undertaken in the software development process must be registered and monitored through a bug-tracker. In addition, the historical data of these systems can be used as history and documentation for the software. Examples of such systems are Bugzilla (`http://www.bugzilla.org`), Mantis (`http://www.mantisbt.org`) and Trac (`http://trac.edgewall.org`).

Each bug report is stored with a variety of fields of free text and custom fields defined according to the necessity of each project. In Trac, for example, it is defined fields for summary and detailed description of a bug report. In the same bug report it can also be recorded information about software version, dependencies with other bug reports (duplicate bug reports, for example), the person who will be assigned to the bug report, among other information. Moreover, during the life cycle of a bug report, comments can be inserted to help solving it. Figure 1.1 shows an example of a bug report from Trac.

Some challenges have emerged through the use of bug trackers, among them, we can cite: dynamic assignment of bug reports (Anvik *et al.*, 2006), change impact analysis and effort estimation (Song *et al.*, 2006), quality of bug report descriptions (Ko *et al.*, 2006), software evolution and traceability (Sandusky *et al.*, 2004), and duplicate bug reports detection (Hiew, 2006). Each one of these issues are briefly described as follows:

- **Dynamic assignment** of bug reports is to detect (automatic or semi-automatically) the best developer suited to solve a problem reported in a bug report;



Timeline Paginate Patch

Opened 10 months ago
Last modified 5 months ago

Reported by:	Naoki Takezoe	Owned by:	cboos
Priority:	high	Milestone:	0.13
Component:	timeline	Version:	0.10.4
Severity:	normal	Keywords:	
Cc:	trac-ja@...		

Description

Reply

Trac displays a lot of events as one page at the timeline. So sometimes it causes performance problems like the ticket report.

Attached patch provides the pagination for the Timeline (for 0.10.4).

Attachments

- timeline.zip (1.7 kB) - added by *Naoki Takezoe* 10 months ago.
- trac-0.11dev_r6303-timeline_paginate.patch (8.7 kB) - added by *trac-ja@...* 8 months ago.
  patch against source:trunk@6303
- trac-0.11b1-timeline_paginate.patch (9.6 kB) - added by *trac-ja@...* 5 months ago.
  Patch against Trac-0.11b1

Attach file

Change History

Changed 10 months ago by Naoki Takezoe

- **attachment** *timeline.zip* added

Changed 10 months ago by sid

This is possibly a duplicate of ~~#215~~.

Reply

Changed 10 months ago by Naoki Takezoe

Hmm... r4681 seems to paginate by the daysback.

Reply

Figure 1.1 Example of a bug report

- **Change impact analysis** and **effort estimation** focus on calculating the impact of a bug report in a project and calculating the necessary effort to solve it;
- **Quality of bug report descriptions** is to ensure that the submitted bug reports are properly described;
- **Software evolution traceability** is concerned with the understanding what drives the changes performed in the software along the time; and
- **Duplicate bug reports detection** consists in avoiding the submission of bug reports that describe an already submitted issue.

The focus of this work is trying to avoid duplicate bug reports submission. The problem of bug reports duplication is better explained and characterized in Chapter 4, through a study which examines the factors that cause it and how it impacts on the software development. Furthermore, the other challenges are further detailed on Chapter 3, where it is described the state-of-the-art of mining bug report repositories.

1.2 Problem Statement

The goal of this dissertation can be stated as follows:

> *This work investigates the problem of bug report duplication emerged by bug trackers, characterizing it empirically to understand its causes and consequences, and provides a tool for search and analysis of bug reports to reduce the effort spent on such tasks.*

1.3 Overview of the Proposed Solution

In order to reduce the effects of the bug report duplication problem, it was developed the Bug Report Analysis and Search Tool (BAST). The remainder of this section describes the context where it was developed and the outline of the proposal.

1.3.1 Context

This dissertation is part of the Reuse in Software Engineering Group (RiSE) (Almeida *et al.*, 2004), formerly called RiSE Project, whose goal is to develop a robust framework for software reuse in order to enable the adoption of a reuse program. However, it is

influenced by a series of areas, such as software measurement, architecture, quality, environments and tools, and so on, in order to achieve its goal. The influence areas are depicted in Figure 1.2.

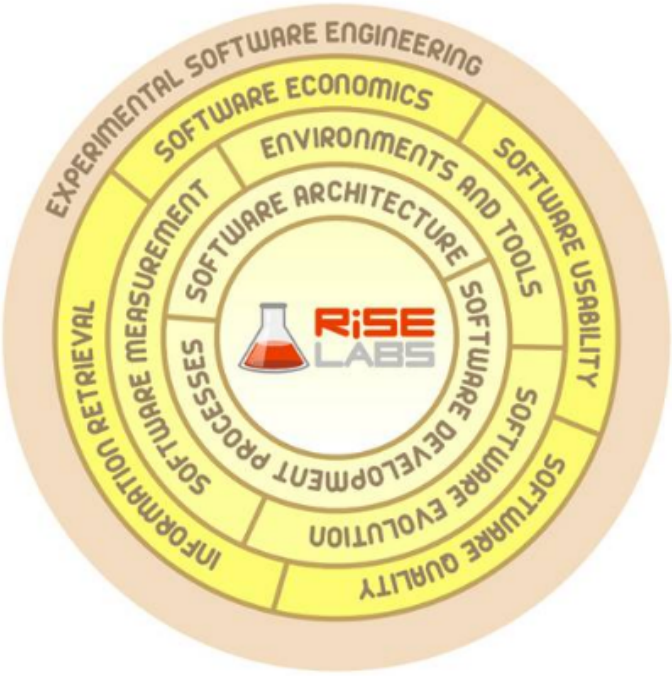


Figure 1.2 RiSE Labs Influences

Based on these areas, the RiSE Labs is divided in several projects, as shown in Figure 1.3. As it can be seen, this framework embraces several different projects related to software reuse and software engineering. They are:

- **RiSE Framework:** Involves reuse processes (Almeida *et al.*, 2004; Nascimento, 2008), component certification (Alvaro *et al.*, 2006) and reuse adoption process (Garcia *et al.*, 2008).
- **RiSE Tools:** Research focused on software reuse tools, such as the Admire Environment (Mascena, 2006), the Basic Asset Retrieval Tool (B.A.R.T) (Santos *et al.*, 2006), which was enhanced with folksonomy mechanisms (Vanderlei *et al.*, 2007), semantic layer (Durao, 2008), facets (Mendes, 2008) and data mining (Martins *et al.*, 2008), and the Legacy InFormation retrieval Tool (LIFT) (Brito, 2007);
- **RiPLE:** Development of a methodology for Software Product Lines (Filho *et al.*, 2008);

- **SOPLE:** Development of a methodology for Software Product Lines based on services;
- **MATRIX:** Investigates the area of measurement in reuse and its impact on quality and productivity;
- **BTT:** Research focused on tools for detection of duplicate bug reports, such as in Cavalcanti *et al.* (2008). Thus, this work is part of the BTT research group;
- **Exploratory Research:** Investigates new research directions in software engineering and its impact on reuse;
- **CX-Ray:** Focused on understanding the Recife Center For Advanced Studies and Systems (C.E.S.A.R.), and its processes and practices in software development.

This dissertation is part of the Bug Report Tracker Tool (BTT) project and its goal is to provide a tool for search and analysis of bug reports with the objective of avoiding duplicate bug reports submission. This work was conducted inside a group for software reuse research, because the bug report duplication problem is more prone to appear when different parts of software are being reused by different projects. A common case where software reuse implies the submission of duplicate bug reports is when the concept of Software Product Lines (Pohl *et al.*, 2005) approach is used to develop software (Runeson *et al.*, 2007). In this context, different software projects share the same basis components, and if some of these components are defective they will affect all software that use these components, thus increasing the possibility of duplicate bug reports submission.

Figure 1.3 RiSE Labs Projects

1.3.2 Outline of the Proposal

The proposed solution consists in a Web based application that enables people involved with bug report search and analysis to perform such tasks more effectively. Although bug report tracking process involves a complete cycle of finding errors, reporting them, validating, fixing the problems and, finally, releasing the changes, the proposed solution aims to assess only the reporting phase. However, the benefits of improving the reporting phase of bug tracking can be reflected to the other phases also, since the time that is saved in the reporting phase can be used to perform the tasks involved in other phases.

1.4 Out of Scope

- **Quality of search results.** The proposed solution uses a well-known model (Vector Space Model (Salton *et al.*, 1975)) to represent documents and perform searches that better meets our necessity, however it is out of the scope of this work to analyze how efficient is the model. Some discussion involving the efficiency of this model can be found in the work of Salton *et al.* (1975);

- **Impact on other phases of bug tracking process.** Our solution concerns with the reporting phase from bug tracking process. Thus, we are interested on how this phase can be improved by the proposed solution. In this way, it is out of scope the analysis and improvement of other phases;

- **Type of users.** Initially, the subjects of this work can be developers, testers or other stakeholders with some technical background in software development, specially using bug trackers. Thus, it is out of scope to provide a tool that supports all types of users.

1.5 Statement of the Contributions

As a result of the work presented in this dissertation, the following contributions can be highlighted:

A characterization of the bug report duplication problem. It was conducted an extensive study about the duplication problem in order to confirm its existence, and potential causes for bug report duplication.

An analysis of the state-of-the-art for mining bug report repositories. It presents an overview of the work found in the literature that have mined specifically bug report repositories, for all diverse purposes.

A solution for bug reports duplication. It specifies and implements a solution based on *Text Mining* and *Keyword search* techniques (Baeza-Yates and Ribeiro-Neto, 1999), with the objective of to reduce the effects of the bug report duplication problem.

Two empirical studies to validate the proposed solution. This dissertation also presents a case study performed in a real environment for software development and test, and an experiment performed with 18 subjects comparing BAST with a baseline tool.

In addition to the contribution mentioned, some papers presenting the findings of this dissertation were produced:

- Cavalcanti, Y. C., Martins, A. C., de Almeida, E. S., and de Lemos Meira, S. R. (2008a). Avoiding Duplicate CR reports in Open Source Software Projects. In *The 9th International Free Software Forum (IFSF'08)*, Porto Alegre, Brazil.

- Cavalcanti, Y. C., de Almeida, E. S., da Cunha, C. E. A., Pinto, E. R., and Meira, S. R. L. (2008b). The Bug Report Duplication Problem: A Characterization Study. Technical report, C.E.S.A.R and Federal University of Pernambuco.

Bibliography

Almeida, E. S., Alvaro, A., Lucrédio, D., Garcia, V. C., and Meira, S. R. L. (2004). Rise project: Towards a robust framework for software reuse. In *IEEE International Conference on Information Reuse and Integration (IRI)*, pages 48–53, Las Vegas, NV, USA.

Alvaro, A., Almeida, E. S., and Meira, S. L. (2006). A software component quality model: A preliminary evaluation. In *Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO'06)*, pages 28–37, Washington, DC, USA. IEEE Computer Society.

Anvik, J., Hiew, L., and Murphy, G. C. (2005). Coping with an open bug repository. In *Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange*, pages 35–39, New York, NY, USA. ACM Press.

Anvik, J., Hiew, L., and Murphy, G. C. (2006). Who should fix this bug? In *Proceeding of the 28th International Conference on Software Engineering (ICSE'06)*, pages 361–370, New York, NY, USA. ACM Press.

Baeza-Yates, R. A. and Ribeiro-Neto, B. A. (1999). *Modern Information Retrieval*. ACM Press / Addison-Wesley.

Bennett, K. H. and Rajlich, V. T. (2000). Software maintenance and evolution: a roadmap. In *Proceedings of the Conference on The Future of Software Engineering (ICSE'00)*, pages 73–87, New York, NY, USA. ACM Press.

Brito, K. S. (2007). *LIFT: A Legacy InFormation retrieval Tool*. Master's thesis, Federal University of Pernambuco, Recife, Pernambuco, Brazil.

Cavalcanti, Y. C., Martins, A. C., Almeida, E. S., and Meira, S. R. L. (2008). Avoiding duplicate cr reports in open source software projects. In *The 9th International Free Software Forum (IFSF'08)*, Porto Alegre, Brazil.

Durao, F. A. (2008). *Semantic Layer Applied to a Source Code Search Engine*. Master's thesis, Federal University of Pernambuco, Recife, Pernambuco, Brazil.

Eastwood, A. (1993). Firm fires shots at legacy systems. *Computing Canada*, **19**(2), 17.

Erlikh, L. (2000). Leveraging legacy system dollars for e-business. *IT Professional*, **2**(3), 17–23.

Filho, E. D. S., Cavalcanti, R. O., Neiva, D. F. S., Oliveira, T. H. B., Lisboa, L. B., Almeida, E. S., and Meira, S. R. L. (2008). Evaluating domain design approaches using systematic review. In R. Morrison, D. Balasubramaniam, and K. E. Falkner, editors, *2nd European Conference on Software Architecture (ECSA'08)*, volume 5292 of *Lecture Notes in Computer Science*, pages 50–65. Springer.

Garcia, V. C., Lisboa, L. B., ao, F. A. D., Almeida, E. S., and Meira, S. R. L. (2008). A lightweight technology change management approach to facilitating reuse adoption. In *2nd Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS'08)*, Porto Alegre, Brazil.

Hiew, L. (2006). *Assisted Detection of Duplicate Bug Reports*. Master's thesis, The University of British Columbia.

Huff, F. (1990). Information systems maintenance. *The Business Quarterly*, (55), 30–32.

Ko, A. J., Myers, B. A., and Chau, D. H. (2006). A linguistic analysis of how people describe software problems. In *Proceedings of the Visual Languages and Human-Centric Computing (VLHCC'06)*, pages 127–134, Washington, DC, USA. IEEE Computer Science.

Martins, A. C., Garcia, V. C., Almeida, E. S., and Meira, S. R. L. (2008). Enhancing components search in a reuse environment using discovered knowledge techniques. In *2nd Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS'08)*, Porto Alegre, Brazil.

Mascena, J. C. C. P. (2006). *ADMIRE: Asset Development Metric-based Integrated Reuse Environment*. Master's thesis, Federal Uniersity of Pernambuco, Recife, Pernambuco, Brazil.

Mendes, R. C. (2008). *Search and Retrieval of Reusable Source Code using Faceted Classification Approach*. Master's thesis, Federal University of Pernambuco, Recife, Pernambuco, Brazil.

Moad, J. (1990). Maintaining the competitive edge. *Datamation*, **4**(36), 61–62.

Nascimento, L. M. (2008). *Core Assets Development in SPL - Towards a Practical Approach for the Mobile Game Domain*. Master's thesis, Federal University of Pernambuco, Recife, Pernambuco, Brazil.

Pohl, K., Böckle, G., and van der Linden, F. (2005). *Software Product Line Engineering: Foundations, Principles, and Techniques*.

Runeson, P., Alexandersson, M., and Nyholm, O. (2007). Detection of duplicate defect reports using natural language processing. In *Proceedings of the 29th International Conference on Software Engineering (ICSE'07)*, pages 499–510. IEEE Computer Science Press.

Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Commun. ACM*, **18**(11), 613–620.

Sandusky, R. J., Gasser, L., and Ripoche, G. (2004). Bug report networks: Varieties, strategies, and impacts in a f/oss development community. In *Proceedings of the 1st International Workshop on Mining Software Repositories (MSR'04)*, pages 80–84, University of Waterloo, Waterloo.

Santos, E. C. R., ao, F. A. D., Martins, A. C., Mendes, R., Melo, C., Garcia, V. C., Almeida, E. S., and Meira, S. R. L. (2006). Towards an effective context-aware proactive asset search and retrieval tool. In *6th Workshop on Component-Based Development (WDBC'06)*, pages 105–112, Recife, Pernambuco, Brazil.

Sommerville, I. (2007). *Software Engineering*. Addison Wesley, 8 edition.

Song, Q., Shepperd, M. J., Cartwright, M., and Mair, C. (2006). Software defect association mining and defect correction effort prediction. *IEEE Transactions on Software Engineering*, **32**(2), 69–82.

Vanderlei, T. A., ao, F. A. D., Martins, A. C., Garcia, V. C., Almeida, E. S., and Meira, S. R. L. (2007). A cooperative classification mechanism for search and retrieval software components. In *Proceedings of the 2007 ACM symposium on Applied computing (SAC'07)*, pages 866–871, New York, NY, USA. ACM.

Appendices

.1 Time sheet

ID	*Start date*	*Start time*	*End date*	*End time*	*Is it a duplicate?*
1	/ /	:	/ /	:	[ ] Yes. [ ] No. ID:
2	/ /	:	/ /	:	[ ] Yes. [ ] No. ID:
3	/ /	:	/ /	:	[ ] Yes. [ ] No. ID:
4	/ /	:	/ /	:	[ ] Yes. [ ] No. ID:
5	/ /	:	/ /	:	[ ] Yes. [ ] No. ID:
6	/ /	:	/ /	:	[ ] Yes. [ ] No. ID:
7	/ /	:	/ /	:	[ ] Yes. [ ] No. ID:
8	/ /	:	/ /	:	[ ] Yes. [ ] No. ID:
9	/ /	:	/ /	:	[ ] Yes. [ ] No. ID:
10	/ /	:	/ /	:	[ ] Yes. [ ] No. ID:
...					
23	/ /	:	/ /	:	[ ] Yes. [ ] No. ID:
24	/ /	:	/ /	:	[ ] Yes. [ ] No. ID:
25	/ /	:	/ /	:	[ ] Yes. [ ] No. ID:
26	/ /	:	/ /	:	[ ] Yes. [ ] No. ID:
27	/ /	:	/ /	:	[ ] Yes. [ ] No. ID:
28	/ /	:	/ /	:	[ ] Yes. [ ] No. ID:
29	/ /	:	/ /	:	[ ] Yes. [ ] No. ID:
30	/ /	:	/ /	:	[ ] Yes. [ ] No. ID:
31	/ /	:	/ /	:	[ ] Yes. [ ] No. ID:
32	/ /	:	/ /	:	[ ] Yes. [ ] No. ID:

Table 1 Time sheet used in the study.

.2 Questionnaire for Subjects Profile

Questionnaire for Subjects Profile
How many years since graduation? [ ] years.
How many projects do you have participated according to the following categories? [ ] Low complexity. [ ] Medium complexity. [ ] High complexity.
What were the roles that you played in the projects cited before (developer, configuration manager, tester…)?
How do you define your experience with bug-trackers? [ ] I never used them before. [ ] I used them in every project i participated. I used them in [ ] projects.
Do you have used any of the following bug-trackers? [ ] Bugzilla. In: [ ] industry [ ] academia [ ] Trac. In: [ ] industry [ ] academia [ ] Mantis. In: [ ] industry [ ] academia [ ] Jyra. In: [ ] industry [ ] academia [ ] BSD Bug-tracker. In: [ ] industry [ ] academia [ ] Other:
Have you performed any analysis of Firefox bug-reports before? [ ] Yes. [ ] No.

Table 2 Questionnaire for bug-report submitters.

.3 Form for Qualitative Analysis

Questionnaire for Qualitative Analysis
Did you use any of the search filters provided by BAST? [ ] Yes. [ ] No.
Is there any search filter you think it must be present in BAST? [ ] Yes. [ ] No. Cite them:
Did you have any problem with the search filters usage? [ ] Yes. [ ] No. Cite them:
Did you use the ordering features of BAST? [ ] Yes. [ ] No.
Did you have any problem with ordering features? [ ] Yes. [ ] No. Cite them:
Do you think there is any other important information that must be present in the list of search results? [ ] Yes. [ ] No. Cite them:
Did you have any problem to visualize the details from some bug-report? [ ] Yes. [ ] No. Cite them:
Do you believe the way bug-reports details are presented was helpful to perform the analysis? [ ] Yes. [ ] No.
Was the recommendation of related bug-reports, presented in the bug-report details, useful for the analysis? [ ] Yes. [ ] No.
Is there any other information concerning bug-reports details you believe it should be present or emphasized? [ ] Yes. [ ] No. Cite them:
Did you use the help provided by BAST? [ ] Yes. [ ] No.
Did you found any other problem/enhacement/defect that was not mentioned before? Cite them.
Please, write down any suggestion you think might would be useful.

Table 3 Questionnaire for qualitative analysis.