

# Mozilla 项目缺陷修复追踪关系研究

张宇霞

(北京大学高可信软件技术教育部重点实验室 北京 100871)

**摘要** 软件可追踪性为软件工程的很多活动提供了非常重要的支持,如变更影响分析、回归测试、版本控制以及一致性检验等。在开源软件项目中,缺陷数据和版本数据之间的缺陷修复关联是一种重要的软件追踪关系。研究开源软件项目缺陷和版本两种制品间的缺陷修复关联,选取大型开源项目 Mozilla 作为研究对象,在深入了解所获得数据的整体分布情况后,针对 Mozilla 项目的产品 Firefox 浏览器,利用 Fellegi-Sunter 模型挖掘缺陷数据与 commit 数据之间的缺陷修复关联并建立二者之间的追踪关系,最后对挖掘出的缺陷修复追踪关系进行结果分析。该项工作为开源项目制品间追踪关系的研究提供了经验参考。

**关键词** 可追踪性,缺陷修复,版本控制,数据挖掘,Fellegi-Sunter 模型

**中图分类号** TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.04.005

## Study on Bug-fixed Traceability of Mozilla Project

ZHANG Yu-xia

(Key Lab of High Confidence Software Technologies, Ministry of Education, Peking University, Beijing 100871, China)

**Abstract** Software traceability provides important supports for many activities of the software engineering, such as changing impact analysis, regression analysis, version control and so on. In the open source software projects, the bug-fixed relationship between bugs and commits is a significant traceability. In this paper, we studied the relationship between bugs and version products in the open source software projects and choosed the large open source software project Mozilla as research subject. After having an in-depth understanding of the overall distribution of Mozilla's bug-related data, we used Fellegi-Sunter model to mine the association between bugs data and commits data, then we built and analyzed the bug-fixed traceability in the Firefox browser. The result of this study provides a reference to the research of traceability in the open source software.

**Keywords** Traceability, Bug-fixed, Version control, Data mining, Fellegi-Sunter model

## 1 引言

软件可追踪性是指将一种软件制品同其他制品相关联,并在项目的整个开发阶段维护这种关联,最后将关联应用到软件开发过程中的一种能力<sup>[1]</sup>。早在 20 世纪 70 年代,人们已经发现建立软件制品间的追踪关系可以有效提高软件质量并降低维护成本<sup>[2]</sup>。针对开源软件追踪关系的研究目前还处于起步阶段,这与开源软件特殊的开发和维护过程紧密相关。对于一个大型开源软件项目,其开发和维护是由开源社区成员共同完成的。这些社区成员分散在世界各地,其开发语言、习惯及个人代码风格等差异很大,并且开源项目的数据主要以项目参与者之间的沟通讨论形式分散在各个网页中,因此构建相对完整的追踪关系比较困难。建立开源项目缺陷和版本制品之间的追踪关系对发现制约项目发展的瓶颈及异常现象、指导软件测试工作以及预测缺陷数和合理分配资源等都具有重要意义<sup>[3]</sup>。

本文旨在深入了解 Mozilla 项目缺陷,同时在代码相关数

据的基础上,分析挖掘二者之间的关联关系并构建缺陷修复追踪关系。主要贡献有:1)对 Mozilla 项目历史缺陷报告和版本相关数据进行了统计分析,为建立缺陷修复追踪关系提供了数据支持;2)实现了一种结合正则表达式与 Fellegi-Sunter (FS)模型的缺陷修复追踪关系构造方法,并用 FS 模型对初步建立的关联关系进行检测验证以提高追踪关系的准确性。

本文第 2 节介绍 Mozilla 项目相关数据以及缺陷修复追踪关系;第 3 节阐述基于 Fellegi-Sunter 模型的缺陷修复追踪关系的构建方法;第 4 节利用该方法构建 Mozilla 项目 Firefox 产品的缺陷修复追踪关系并分析验证结果;第 5 节讨论了相关工作;最后总结全文。

## 2 Mozilla 项目数据及缺陷修复追踪关系

### 2.1 Mozilla 项目及其相关数据

Mozilla 项目是 NetScape 公司在 1998 年启动的,同年 3 月源代码被公开,成为开源项目。该项目的主要目的是开发并完善 NetScape 公司主打的浏览器,并提供基于 Mozilla

框架的开发工具来维护应用程序套件。目前, Mozilla 已经成为一个有十几年历史的大型开源项目, 其代表性产品有 Firefox, Thunderbird, SeaMonkey 等。

Bugzilla 是 Mozilla 中的缺陷追踪系统。本文所研究的 Mozilla 项目缺陷数据来自 Bugzilla, 主要包括以下内容: 缺陷标识 (BugID)、标题 (Title)、状态 (Status)、来源 (Product)、报告时间 (Reported\_time)、报告者 (Reporter)、重要性 (Importance) 等。

Mercurial 为 Mozilla 项目目前使用的版本控制系统。本文研究所需的 commit 数据来自 Mercurial, 主要包括以下信息: commit 编号 (changeset)、提交者 (Author)、提交时间 (Time)、提交描述 (Summary) 等。

## 2.2 缺陷修复追踪关系的依据

缺陷报告和版本数据作为软件项目的两种制品, 存在多种关联关系。缺陷修复追踪关系本质上是指缺陷与版本控制系统中修复该缺陷的代码提交之间的关联关系。软件从一个版本到下一个版本的演化过程中, 开发团队所做的工作不止是修复缺陷, 还包括实现新功能、修正稳定性及安全性问题等。所以缺陷与版本库中的代码提交之间的关系有多种, 如: 1) 缺陷在 commit 的 summary 中被提及, commit 所属的本次代码提交刚好是用来修复该缺陷的; 2) 已修复缺陷虽被 commit 提到, 但二者并非修复关联; 3) 部分缺陷虽多次被 commit 提到, 但缺陷本身并未被修复; 4) 一部分缺陷已经被修复, 但由于关键信息缺失而无法直接建立修复关联。

本文主要是建立情形 1) 和情形 4) 中隐含的缺陷修复追踪关系, 并剔除情形 2) 和情形 3) 中的“伪”缺陷修复关系。

## 3 基于 Fellegi-Sunter 模型的缺陷修复追踪关系的构建

### 3.1 Fellegi-Sunter 模型简介

Fellegi-Sunter (FS) 模型是用来链接相关记录的规范化数学模型, 1969 年首次由 Fellegi 和 Sunter 提出, 目前已被广泛应用于解决记录链接问题<sup>[4]</sup>。在应用 FS 模型构建缺陷修复追踪关系之前, 首先要将 FS 模型针对 Mozilla 数据集做适应性改变, 包括识别缺陷和 commit 记录中代表意义相同的字段, 确定匹配对并规定相应的匹配法则, 以及构造训练集以获取匹配标准值等。

FS 模型可以将属于不同数据集但本身存在关联关系的记录挖掘出来并建立链接。在开源软件领域, 已有一些研究, 如文献[5]将 FS 模型应用到缺陷与版本库数据间的关联关系挖掘中。本文在此基础上, 将 FS 模型与正则表达式结合起来, 共同构建缺陷修复追踪关系。

### 3.2 缺陷修复追踪关系的构建方法

本文通过对 Mozilla 项目相关数据的研究得出 BugID 可以唯一标识一个缺陷报告, 同理, changeset 也可唯一标识一次代码提交。所以将具有修复关系的缺陷报告的 BugID 和 commit 的 changeset 关联起来, 就可以唯一确立这种缺陷修复追踪关系。本文建立缺陷修复追踪关系的工作分为如下 5 个步骤。

(1) FS 模型针对 Mozilla 项目做适应性改变

将 FS 模型应用到 Mozilla 项目的缺陷和 commit 数据集中需要做的主要工作包括匹配对查找、匹配规则制定、训练集构造以及匹配标准值获取。

(2) 初步建立缺陷修复追踪关系

部分 commit 记录的 summary 字段包含缺陷 BugID 信息, 该信息对于建立缺陷修复追踪关系具有重要意义。本文利用正则表达式将 summary 字段中的 BugID 提取出来, 通过查找缺陷数据集上相应的缺陷报告初步建立缺陷修复追踪关系。

(3) 利用 FS 模型验证初步建立的缺陷修复追踪关系

利用 FS 模型对初步建立的缺陷修复追踪关系计算匹配结果并将其与匹配标准值进行比较, 若匹配结果不小于标准值则保留, 否则舍弃。

(4) 利用 FS 模型挖掘缺陷修复追踪关系

选择任一未建立追踪关系的缺陷报告, 并利用 FS 模型计算其与可能存在追踪关系的 commit 数据集的匹配结果 (记为 FS 和), 最后选取最大的 FS 和与匹配标准值进行比较来决定追踪关系是否建立。

(5) 人工验证已建立的缺陷修复追踪关系

人工浏览 Bugzilla 和 Mercurial 系统页面, 对已建立的缺陷修复追踪关系进行验证<sup>[7]</sup>。验证成功准则有: 1) 在缺陷报告的讨论区是否有开发人员指出修复该缺陷的 commit 的 changeset; 2) 在代码提交界面是否指出所修复缺陷; 3) commit 代表的代码是否解决了相应的缺陷。

## 4 缺陷修复追踪关系的建立与结果分析应用

### 4.1 数据集的选取以及数据预处理

#### 4.1.1 数据集的选取

本文选取在 Firefox 版本 4 到版本 5 演化期间 Bugzilla 中的缺陷报告以及 Mercurial 上的 commit 数据作为基础研究数据集, 尝试挖掘并建立二者的缺陷修复追踪关系。

#### 4.1.2 数据预处理

基础数据集的预处理主要包括脏数据处理与关键值提取。

(1) 脏数据处理

为保证所建立缺陷修复追踪关系的质量, 尽可能选取缺陷报告信息填写相对完整的数据进行研究分析, 所以舍弃掉关键字段为 unspecified 的缺陷报告。事实上, 即使缺陷报告数据相对完整, 依然会存在脏数据, 如缺陷提交时间与其存在版本的发布时间冲突、版本信息错误等。对于这样的误差数据, 因其数据量相对较少, 本文通过访问相关网站进行了人工更正<sup>[7]</sup>。

(2) 关键值提取

本文利用正则表达式匹配法提取 commit 记录中 summary 字段包含的 BugID, 关键代码如下:

```
Pattern pattern=Pattern.compile("[0-9]{6}");
```

```
//匹配模式为六位连续数字
```

```
Matcher matcher=pattern.matcher(commit[i][4]);
```

```
//commit 记录中的 summary
while (matcher.find())
//判断 summary 字段是否有 6 位连续数字
bugid=matcher.group(0);
//提取 BugID
```

### 4.2 建立缺陷修复追踪关系

建立缺陷修复追踪关系包含以下 4 个步骤。

#### 4.2.1 部署 FS 模型到 Mozilla 项目相关数据集

基于前期 Mozilla 项目相关数据的研究,选取缺陷数据和 commit 记录中实际意义相同的字段作为匹配对,共计 6 组,如表 1 所列。

表 1 基于 Mozilla 项目缺陷和版本数据集所选的匹配字段

匹配对	匹配变量(Bugzilla)		匹配变量(Mercurial)	
MR1	LA	Last Comment Author	CA	Commit Author
MR2	LT	Last Comment Time	CT	Commit Time
MR3	SA	Second Comment Author	CA	Commit Author
MR4	ST	Second Comment Time	CT	Commit Time
MR5	FT	Fixed Time	CT	Commit Time
MR6	TL	Title	LS	Log Summary

根据不同匹配对的性质,制定相应的匹配准则。如果是人员匹配,则要求变量值完全相等(LA=CA,SA=CA);如果是时间匹配,基于开发者在提交缺陷修复代码之后并不会立即去 Bugzilla 评论并更改相应缺陷,本文选取 24 小时作为可以容忍的偏差范围;对于 MR6 匹配对,本文认为 Title 中的单词如果有 30%出现在 commit 记录的 summary 中,则匹配成功,30%的文本相似度是通过统计分析训练集相关字段匹配结果得出的。

本文选取 100 对来自 Firefox 浏览器的缺陷报告和 commit 记录作为训练集,这些记录对的缺陷修复追踪关系已经过人工验证。基于以上 6 组匹配对,统计训练集中各个匹配对的成功率,如图 1 所示。比较训练集中 100 对具有修复关联的缺陷报告和相应 commit 记录的 6 个匹配对的结果,匹配成功记为 1,失败记为 0,并将 6 个匹配项的结果求和,记为 FS 和。统计结果显示,有 100%的缺陷修复关联 FS 和大于或等于 1,有 75%的 FS 和大于或等于 3。在之后利用 FS 模型验证建立缺陷修复关联时,采用 FS 和为 3 作为检验标准,因为 0.25 的误差在统计学上是可以接受的。

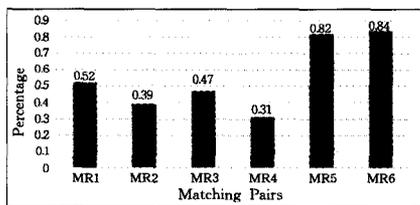


图 1 训练集的 FS 模型匹配对的结果分布

#### 4.2.2 初步建立缺陷修复追踪关系并利用 FS 模型验证

统计 Mozilla 项目在 Firefox 版本 4 到版本 5 演化期间存储在 Mercurial 系统上的 71682 条 commit 记录,发现 summary 字段带有 BugID 信息的 commit 共计 59403 条。针对此类 commit 记录,利用正则表达式提取 BugID,并与同时间段 Firefox 被修复的缺陷报告进行关联,初步建立 890 条缺陷修复追踪关系,再利用 FS 模型对其进行验证,计算 6 组匹配对

的结果并求 FS 和。结果有 615 条追踪关系的 FS 和不少于 3,将这一部分缺陷修复追踪关系保留下来。

#### 4.2.3 利用 FS 模型挖掘缺陷修复追踪关系

针对在 Firefox 版本 4 到版本 5 演化期间未建立缺陷修复追踪关系的已修复缺陷报告,依次选择缺陷被修复时间并查找 commit 记录中提交时间与缺陷修复时间偏差在 24 小时之内的全部 commit 记录,构造每个缺陷的可能关联集;然后利用 FS 模型计算缺陷与其可能关联集中任一 commit 记录的 FS 和;最后选取 FS 和最大的 commit 记录与 FS 标准值进行比较,若不小于 3,则在该缺陷报告和 FS 和最大的 commit 记录之间建立缺陷修复追踪关系。通过统计和计算,本文共建立了 73 条缺陷修复追踪关系。

#### 4.2.4 验证 FS 模型挖掘出的缺陷修复追踪关系

通过前述步骤,共建立了 688 条缺陷修复追踪关系,其中有 255 条缺陷修复追踪关系 FS 和为 4,占全部的 37.06%,比例最大;FS 和为 6 的追踪关系共 77 条,占比 11.19%,所占比例最小。然后,采用分层抽样的方法按照 5:1 的比例对 FS 和值为 3,4,5,6 的追踪关系进行人工检验,估计准确率。对 FS 和为 3 的追踪关系,选择 42 组缺陷修复追踪关系,通过人工浏览 Bugzilla 和 Mercurial 主页<sup>[7]</sup>检验是否正确,发现在这 42 组记录对中,有 38 组记录的缺陷修复关联关系确实存在,准确率为 90.43%。同理,FS 和为 4,5,6 的追踪关系准确率分别为 92.32%,95.64%和 99.99%。此外,对后期利用 FS 模型挖掘出的 73 条追踪关系进行了人工验证,结果如表 2 所列。对于这一部分追踪关系,commit 的 summary 字段并没有显示所修复缺陷的信息,因此准确率相对较低。

表 2 FS 挖掘出的 73 条追踪关系验证

FS 和	追踪关系数	正确追踪数	错误追踪数	准确率/%
4	47	11	36	23.40
5	21	8	13	38.10
6	5	4	1	80.00

### 4.3 缺陷修复追踪关系结果的分析应用

在 Firefox 版本 4 到版本 5 演化期间共修复了 1237 个缺陷。本文利用 FS 模型共建立 688 条追踪关系,其中 615 条追踪关系相关的 commit 记录中含有所修复缺陷的 BugID,追踪关系准确率较高,均在 90%以上;剩余 73 条追踪关系的准确率相对较低,下一步将对对此进行更深入的研究以提高准确率。

已经建立的缺陷修复追踪关系可以有以下几方面的应用:1)通过缺陷修复追踪关系迅速定位修复该缺陷的代码提交,有助于解决由缺陷修复工作引入的新问题;2)评估缺陷报告对软件版本库的影响程度;3)对与 commit 关联的缺陷进行特征分析,识别缺陷追踪系统中质量较高的缺陷报告,进而指导缺陷修复工作;4)通过关联 commit 的提交时间获取更准确的缺陷修复时间,可用于评估缺陷生命周期以及预测缺陷平均修复时长等<sup>[3]</sup>。

## 5 相关工作

近年来,开源领域追踪关系的自动化构建已形成一个 (下转第 55 页)

- Automated Technology for Verification and Analysis. Springer, 2013;178-92.
- [3] SALA N G, ETCHEVERS X, DE PALMA N, et al. Verification of a Self-configuration Protocol for Distributed Applications in the Cloud [M]// Assurances for Self-Adaptive Systems. Springer, 2013;60-79.
- [4] ETCHEVERS X, SALA N G, BOYER F, et al. Reliable self-deployment of cloud applications[C]//Proceedings of the 29th Annual ACM Symposium on Applied Computing. ACM, 2014.
- [5] ETCHEVERS X, COUPAYE T, BOYER F, et al. Self-Configuration of Distributed Applications in the Cloud[C]//2011 IEEE International Conference on Proceedings of the Cloud Computing (CLOUD). 2011;4-9.
- [6] OASIS. Topology and Orchestration Specification for Cloud Applications Simple Profile in YAML Version 1.0[S]. 2015.
- [7] Puppet[OL]. <https://puppetlabs.com>.
- [8] Consul[OL]. <https://www.consul.io>.
- [9] BREITENBERGER U, BINZ T, K PES K, et al. Combining Declarative and Imperative Cloud Application Provisioning based on TOSCA [C]//IC2E IEEE. 2014.

(上接第23页)

比较热门的研究方向。这方面比较有代表性的研究是 Bachmann 等人提出的正则表达式提取法<sup>[8]</sup>。该方法使用正则表达式提取 commit 记录中的缺陷标识来建立追踪关系,进而通过专家分析等方式验证追踪关系的正确性。实验证明,这种追踪关系建立方法可以提高追踪关系的准确率,但不适用于那些 commit 记录中缺省了所修复缺陷信息的情况。基于正则表达式构建追踪关系的研究还有很多<sup>[9-11]</sup>。

Ashish Sureka 等人<sup>[5]</sup>首次将 Fellegi-Sunter 模型应用到开源领域追踪关系的建立中。该研究证明 Fellegi-Sunter 模型可以有效建立开源软件缺陷修复追踪关系。本文在此基础上做了一些改进,首先引入正则表达式来提取关键数据,初步建立追踪关系,再利用 Fellegi-Sunter 模型对初步建立的追踪关系进行筛选,最后采用该模型全面挖掘可能的缺陷修复关联并建立追踪关系。相比 Ashish Sureka 等人的研究,本文引进正则表达式提取关键信息,并利用 FS 模型对初步建立的追踪关系进行验证筛选是本文的创新之处,且文中所提方法提高了缺陷修复追踪关系的质量。这种缺陷修复追踪关系构建方法为研究开源项目制品间的追踪关系提供了经验参考<sup>[12]</sup>。

**结束语** 在开源软件领域,软件制品可追踪性研究还处于起步阶段。本文在了解了开源社区软件开发过程、缺陷追踪系统以及版本控制系统等相关知识之后,对 Mozilla 项目 Bugzilla 缺陷追踪系统和 Mercurial 版本控制系统中的相关数据进行了深入的分析研究,并选取 Mozilla 项目中最大的软件产品 Firefox 浏览器的缺陷数据和代码提交记录作为基础研究数据集进行缺陷修复追踪关系的挖掘和建立。

本文采用 Fellegi-Sunter 模型对在 Firefox 浏览器版本 4 到版本 5 演化期间的缺陷修复关联关系进行挖掘,找到约 55.61% 的缺陷修复追踪关系。建立开源项目缺陷和版本制品之间的缺陷修复追踪关系对于评估项目发展进度、发现制约项目发展的瓶颈以及开发中的异常现象、指导软件测试工作、提高缺陷发现率、改善软件质量管理版本更新以及预测 Bug 数和分派资源等具有重要的意义。本文对于 Firefox 浏览器缺陷修复追踪关系的建立方法可以为开源项目其他追踪关系的建立提供一定的经验参考。此外,在 Firefox 浏览器版本 4 到版本 5 演化期间,剩余 44.39% 的缺陷修复关联没有

找到,还需要更加深入的研究。

**致谢** 本文工作是在北京大学软件研究所教授金芝老老师和周明辉老师的指导下完成的,在此致以诚挚的感谢!

### 参考文献

- [1] CoEST: Center of excellence for software traceability[OL]. <http://www.CoEST.org>.
- [2] BISSYANDE T F, THUNG F, WANG S, et al. Empirical Evaluation of Bug Linking[C]//European Conference on Software Maintenance & Reengineering. 2013;89-98.
- [3] D'AMBROS M, LANZA M, ROBBES R. Evaluating defect prediction approaches: a benchmark and an extensive comparison [J]. Empirical Software Engineering, 2012, 17(4/5):531-577.
- [4] FELLEGI I P, SUNTER A B. A Theory for Record Linkage [J]. Journal of the American Statistical Association, 1969, 64(328):1183-1210.
- [5] SUREKA A, LAL S, AGARWAL L. Applying Fellegi-Sunter (FS) Model for Traceability Link Recovery between Bug Databases and Version Archives[C]//2011 18th Asia Pacific Software Engineering Conference (APSEC). IEEE, 2011;146-153.
- [6] BETTENBURG N, WEISS C, JUST S, et al. What Makes a Good Bug Report? Revision 1.1[J]. Fse, 2008, 36(5):618-643.
- [7] Bugzilla official website[OL]. <http://www.bugzilla.mozilla.org>.
- [8] BACHMANN A, BERNSTEIN A. Data retrieval, processing and linking for software process data analysis; Technical Report IFI-2009. 0003[R]. Department of Informatics, University of Zurich, May 2009.
- [9] SCHRÖTER A, ZIMMERMANN T, PREMRAJ R, et al. If your bug database could talk[J]. Proceedings of International Symposium on Empirical Software Engineering, 2006, 7(5):18-20.
- [10] SLIWERSKI J, ZIMMERMANN T, ZELLER A. When do changes induce fixes?[J]. ACM Sigsoft Software Engineering Notes, 2005, 30(1):1-5.
- [11] ZIMMERMANN T, PREMRAJ R, ZELLER A. Predicting Defects for Eclipse[C]//Proc International Workshop on Predictor Models in Software Engineering. 2007;9.
- [12] SHIHAB E, IHARA A, KAMEI Y, et al. Studying re-opened bugs in open source software[J]. Empirical Software Engineering, 2013, 18(5):1005-1042.