面向Issue跟踪系统的变更请求报告关闭可能性预测

熊文军[1] 张 璇[1,2] 王 旭[3] 李 彤[1,2] 尹春林[1]
(云南大学软件学院 昆明 650091)[1] (云南省软件工程重点实验室 昆明 650091)[2]
(云南大学经济学院 昆明 650091)[3]

摘 要 在Issue跟踪系统中存在大量长期未关闭的变更请求报告,增加了开发者不断点击和阅读这些报告的可能性,严重影响了软件需求管理任务的实施和用户的反馈体验。准确和及时地预测这些报告关闭的可能性或重要性可以提高软件维护任务的质量。定义若干衡量变更请求报告特征的指标,选择在训练数据集上预测效果最佳的指标构建Logistic回归预测模型。使用提出的方法对20个SourceForge项目构成的测试数据集进行实验,得到平均查全率为94%和平均伪正率为14%的结果。实验结果表明,提出的方法能在测试数据集上取得很好的预测性能;关闭状态的变更请求报告所占的百分比或数量大小并不影响模型的性能;变更请求报告具有的某些特征可用于预测其在下一版本中得到关闭的可能性。

关键词 变更请求报告,软件需求,缺陷报告,报告优先级
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.11.022

Prediction on Closed-probability of Change Request Report for Issue Tracking System

XIONG Wen-jun[1] ZHANG Xuan[1,2] WANG Xu[3] LI Tong[1,2] YIN Chun-lin[1]
(School of Software, Yunnan University, Kunming 650091, China)[1]
(Key Laboratory of Software Engineering of Yunnan Province, Kunming 650091, China)[2]
(School of Economics, Yunnan University, Kunming 650091, China)[3]

Abstract There are lots of change request reports without being closed for a long time in the Issue tracking system, which increases the likelihood of developers to click and read the reports again and again. It seriously affects the implementation of management task of software requirements and the feedback experiences of users. The accurate and instant prediction of closed-probability or importance of these reports can improve the quality of the task of software maintenance. Several metrics were defined to measure the feature of change request, and Logistic regression prediction model was built by using these best predictive metrics on datasets for training. Then experiments which applied proposed method were performed on datasets of testing which contain 20 SourceForge projects, and achieved a result that average recall of 95% and average FPR (False Positive Rate) of 14%. Analysis of experimental result shows that the proposed method can achieve a good prediction performance on datasets of testing, and closed-percentage or size of change requests report doesn't affect the performance of the model, and some features of change request report can be used to predict its closed-probability in the next version.

Keywords Change request report, Software requirement, Bug report, Prioritization of report

1 引言

在软件项目开发和维护过程中,需求的管理和确定是一项耗费时间和精力的工作。软件维护任务中,软件组织常使用Issue跟踪系统(Issue Tracking System,缺陷跟踪系统)(如Bugzilla,JIRA等)来支持变更请求的反馈。变更请求报告(Change Request Report)可以是缺陷报告(Bugs Report)、功能请求报告(Feature Requests Report)、代码补丁报告(Patches Report)等。在Issue跟踪系统中,用户和开发者可以提出他们在使用软件系统时遇到的问题、合理的改进意见以及

到稿日期:2016-10-21 返修日期:2016-12-04 本文受国家自然科学基金项目(61502413,61262025,61379032,61262024),云南省科技计划项目(2016FB106),云南省教育厅科学研究基金(2015Z020,2013A056),云南省软件工程重点实验室开放基金(2015SE202),云南省创新团队"数据驱动的软件工程创新团队"项目,云南大学高水平创新团队"软件工程创新团队"专项项目,云南大学"中青年骨干教师培养计划"专项项目云南大学人文社科基金(13YNUHSS007)资助。

熊文军(1988-),男,硕士生,主要研究方向为软件仓库挖掘、软件需求预测;**张 璇**(1978-),女,博士,副教授,主要研究方向为需求工程、软件过程、可信软件,E-mail:zhxuan@ynu.edu.cn;**王 旭**(1976-),男,博士,讲师,主要研究方向为金融安全、计量经济学;**李 彤**(1963-),男,博士,教授,主要研究方向为软件过程、形式化方法、软件工程;**尹春林**(1991-),男,硕士生,主要研究方向为软件工程、软件演化、数据挖掘。

对变更请求报告的评论[1]。

通常变更请求报告的生命周期是：提交者提交反映变更请求的报告，报告的状态为 Open；用户和开发者对变更请求报告进行投票和评论以决定是否处理；决定处理报告后，阅读报告的分派者(Triager)并将其分派给合适的开发者，此时状态由 Open 标记为 Open-Resolved；开发者实现或修复变更请求；开发者或用户验证后，将报告的状态由 Open-Resolved 标记为 Closed 或 Closed-Resolved；若报告在关闭之后又出现了新问题，则需重新打开变更请求报告，将变更请求报告再次标记为 Open 状态。因此，变更请求报告一般有两种状态：Open 状态和 Closed 状态。常见的 Resolved 标记有 Accepted，Rejected，Fixed，Duplicate，Wont-fix，Invalid，Out-of-date 等。

监控和管理变更请求报告往往是一项具有挑战性的任务。缺陷跟踪系统中每天都有大量新增的变更请求，这些报告往往在有限的时间和资源的情况下很难得到处理[2]。Anvik[3]对 Mozilla 项目部署的 Bugzilla 缺陷跟踪系统的分析指出，Mozilla 平均每天收到 300 个缺陷报告(变更请求报告的一种)。如果开发者阅读这些没有优先级次序的报告会导致很多重要的变更请求得不到及时处理。Zhang 等[4]对 Apache 和 Eclipse 的缺陷报告进行了统计分析，发现分别有 10.72%和 14.94%的缺陷报告需要进一步处理。Weiss 和 Entry[5]对 SourceForge 上的软件项目进行了统计分析，发现绝大多数开源项目开发者或管理者只有 1 个。在 Issue 跟踪系统中还存在很多解决方案标记为 Resolved，且开发者已说明在某个版本的实现或修复报告中所描述的变更请求，然而，此报告的状态在很长时间后还未标记为 Colsed。如图 1 所示，变更请求报告已标记 Resolved，图 2 所示的评论表明，开发者表示已修复图 1 中所描述的变更请求，但是此报告在 1 年半后仍然处于打开状态。理想情况是每一个变更请求报告都会得到关闭。然而在实际情况中，如果没有 Issue 跟踪系统的提示，开发者可能会忘记关闭一些已经 Resolved、实现或修复的变更请求。如果用户提交的变更请求报告没有得到及时处理和关闭，开发者和用户会一次次点击和阅读处于 Open 状态的变更请求报告，这会让繁忙的开发者进行没有必要的返工。Issue 跟踪系统中存在的大量长期未关闭的变更请求报告亦会降低用户的满意度和反馈积极性。

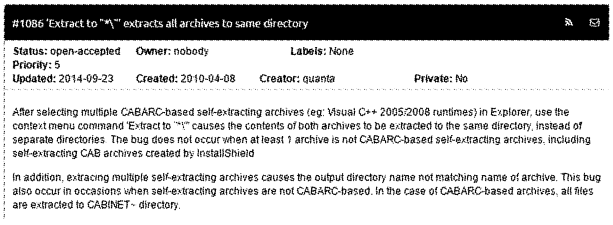


图 1　变更请求报告已被接受

图 2　开发者表示已修复缺陷

变更请求报告是部署具有 Issue 跟踪系统的软件项目不断演化的动力。通过变更请求报告，开发者可以识别和解决软件系统的缺陷、增加新的功能以及接受提交者提交的源代码补丁，进而提高软件系统的质量[6]。若开发者决定接受变更请求报告中反映的软件需求，则该变更请求就进入下一版本的规划中。因此拒绝、接受或不处理变更需求报告，都会造成后续开发过程中软件需求数目的变化，即需求易变性(Requirements Volatility)。在传统软件工程中，需求易变性指用例模型(Use Case Model，UCM)数目的变动情况[7]。变更请求报告与 UCM 表达的软件需求(UCM 是需求的抽象)类似。因此，Issue 跟踪系统中处于打开状态的变更请求报告存在关闭的可能性会导致下一版本软件需求数目的变动。研究表明，需求数目的变动对软件项目过程的实施性能具有重要的影响[8-11]。Costello 和 Liu[12]指出项目管理人员可以识别关键的需求，并在进一步分析需求易变性的原因时合理分配资源。

面对 Issue 跟踪系统中存在的大量没有处理的变更请求报告会对软件过程实施产生需求易变性(需求数目的不确定)，从而对软件过程实施产生影响这一挑战，本文提出了一种能够预测变更请求报告关闭可能性的方法。

本文第 2 节介绍提出的基于 Logistic 回归的预测方法框架；第 3 节以 SourceForge 项目数据为案例介绍该方法的应用方式，并对实验结果进行分析；第 4 节对第 3 节的实验结果进行讨论，并指出不足；第 5 节介绍本文的相关工作；最后总结，并对未来工作进行展望。

2　构建变更请求报告关闭可能性预测模型

本文研究目的是使用变更请求报告的特征构建一个能够预测其关闭可能性的预测模型。

2.1　定义预测指标

在构建预测模型时，选取满足以下能够衡量变更请求报告的特征来定义预测指标：

(1)指标能够反映变更请求报告内容的复杂程度；

(2)指标能够反映变更请求报告随着时间不断变化的特征；

(3)指标能够反映利益者参与变更请求报告讨论的程度。

Menzies 和 Marcus[13]在预测缺陷报告严重等级时指出，相对于有结构报告文本，无结构缺陷报告文本的预测结果可能更好。有结构报告文本指对变更请求报告采取自然语言文本预处理的加工技术，包括分词、停用词移除，词根还原以及使用 TF-IDF 和信息增益。Loconsole 和 Börstler[7]使用 UCM 的大小(Size)特征来预测需求变更的数目。Hooimeijer 和 Weimer[14]在分析两个反映同一个缺陷但不同的报告内容时指出，拥有特征的数目不同会影响它们的最终状态和达到最终状态的时间。这两个报告特征的主要差别是：描述的严重性(Minor vs. Normal)、主机操作系统(Windows XP vs. Windows NT)、评论的数目(Few vs. Many)、附件(Screen vs. Nothing)以及描述的质量(Potentially Misleading vs. Direct)。

受需求易变性、缺陷预测、缺陷优先级排序、缺陷报告质

量等相关研究的启发，本文定义了在 Issue 跟踪系统中比较容易获取的衡量变更请求报告的 12 个指标（见表 1）。衡量变更请求报告复杂度的指标包括描述标题（Title Content）及报告描述内容（Report Description）的相关指标。Zhang[15] 对 Eclipse 和 NASA 数据集进行了实验分析，发现简单的复杂度指标，比如代码行（Line of Code，LOC），可用于预测具有较多缺陷的组件。与度量软件代码复杂度大小的 LOC 指标类似，我们定义了衡量变更请求报告复杂度的指标，包括衡量标题长度的 NTITLELEN、标题单词数的 NTITLEWORD、内容长度的 NREQLEN、内容单词数的 NREQWORD 以及内容行数的 NREQLINE。

表 1 定义的指标

Category	Metric	Description
Title Content	NTITLELEN	变更请求报告标题的长度（字符数）
	NTITLEWORD	变更请求报告标题的单词数目
Report Description	NREQLEN	变更请求报告的长度（字符数）
	NREQWORD	变更请求报告的单词数目
	NREQLINE	变更请求报告的行数
	NREQHREF	变更请求报告所包含的链接数目
	NREQATTACH	变更请求报告包含的附件数目
	NUMPRETAG	变更请求报告包含的带有预格式化文本的数目
Evolution	NREQPOSTS	变更请求报告具有的评论数目
	NREQOWNER	开发者参与变更请求报告讨论的次数
	REQWAITDAY	变更请求报告的等待生存时间/天
Status	inLABEL	变更请求报告的状态（Open：−1；Closed：1）

Bottenburg 等[16-17] 对 3 个开源项目（Apache，Eclipse，Mozilla）的开发者进行了缺陷报告质量的相关调查。其调查结果表明，开发者认为最常用到的缺陷报告信息包括重新生成缺陷的步骤、期待的行为以及堆栈信息。我们定义了预格式化文本数 NUM-PRETAG来衡量报告含有预格式化的嵌入文本信息，带有预格式化的文本往往包含源代码、程序执行堆栈等信息。Issue 跟踪系统中的变更请求报告来源于互联网的环境，报告常用网页链接来说明变更请求。我们认为衡量变更请求的链接数目也很重要，因此定义了链接数目 NREQHREF 来统计变更请求内容含有链接的数目。提交者可以对报告附加诸如屏幕截图、失败测试案例等附件，因此定义附件数目 NREQATTACH 来衡量报告含有附件的情况。

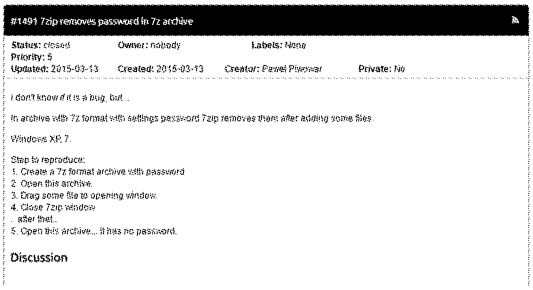
一旦变更请求报告被提交，相关的开发者或有权限的用户都可以对报告进行评论，以讨论不同解决方案或详细了解变更请求反映的需求。我们定义了评论数目 NREQPOSTS 来衡量报告的利益相关者参与讨论沟通的情况，以及使用开发者参与讨论次数 NREQOWNER 来衡量关键利益相关者（开发者）参与讨论的情况。定义报告等待生存时间 REQWAITDAY 来衡量报告的生命周期，具体表示如下：

$$REQWAITDAY=\begin{cases} ClosedDate-OpenDate & (1)\\ GetData/LatestResolvedDate-OpenDate & (2)\end{cases}$$

其中，式(1)的条件为：变更请求报告的状态为 Closed；式(2)的条件为：变更请求报告的状态为 Open。

若变更请求报告已经到 Open-Resolved 步骤，则使用最新一次标记 Open-Resolved 的时间。因为 Issue 跟踪系统中往往会有如图 1 和图 2 所示的应该关闭但没有关闭的变更请求报告。

Issue 跟踪系统中存在一些状态为关闭，但没有关闭时间记录的变更请求报告，如图 3 所示。这些报告往往从 Open 到 Closed 时或者提交者在提交报告时状态相关字段部分直接选了 Closed，对于这部分报告，其 REQWAITDAY 指标需要做特殊处理。本文在抓取数据时确定指标 REQWAITDAY 的方法是：如果报告的状态为 Open 且没有 Resolved，REQWAITDAY 的值为获取数据的时间与打开报告的时间之间相差的天数；如果报告的状态为 Open-Resolved，REQWAITDAY 的值为最后一次 Resolved 时间与打开报告的时间之间相差的天数；如果报告的状态为 Closed，且可以找到关闭时间记录，指标 REQWAITDAY 的值为 Closed 时间与打开报告的时间之间相差的天数；如果报告的状态为 Closed，但没有 Closed 时间记录（见图 3），此时 REQWAITDAY 赋值为 −999。预测这些未按正确流程关闭的变更请求应具有的真实状态也是有意义的。开发者同样可以对这种变更请求报告进行评论，讨论解决方案或提供意见反馈。


#1491 7zip removes password in 7z archive

Status: closed Owner: nobody Labels: None
Priority: 5
Updated: 2015-03-13 Created: 2015-03-13 Creator: Pawel Piwowar Private: No

I don't know if it is a Bug, but ..

In archive with 7z format with settings password 7zip removes them after adding some files.

Windows XP, 7.

Step to reproduce:
1. Create a 7z format archive with password
2. Open this archive.
3. Drag some file to opening window.
4. Close 7zip window
.. after that..
5. Open this archive... it has no password.

Discussion

图 3 缺少关闭时间的变更请求

Issue 跟踪系统往往把 Open 和 Closed 相关的报告放在不同的导航类别下，我们可以轻易获得变更请求报告所处的状态，定义 inLABEL 来量化变更请求报告的状态，Open 为 −1，Closed 为 1。

2.2 预测方法和预测指标选择

在本文研究中，需要预测一个变更请求报告是否会被关闭。选择 Logistic 回归（Logistic Regression）作为构建预测模型的方法。Logistic 回归使用可以量化（如报告的单词数）的布尔值或类别的组合因素来预测事件发生的概率，例如变更请求报告是否会被关闭。Logistic 回归与其他诸如线性回归的回归技术不同，它可以用于分析自变量和因变量之间是否存在某种函数式的依赖。Logistic 回归没有假定自变量和因变量之间存在线性的关系。假设变更请求报告发生 Closed 情况的概率为 p，则其不发生（即 Open）的概率为 $1-p$，发生与不发生的概率之比为 $p/(1-p)$，取对数得 $\ln(\frac{p}{1-p})$，记为 $Logit(p)$，Logistic 回归模型的方程如下：

$$Logit(p)=\ln(\frac{p}{1-p})=\beta_0+\beta_1x_1+\beta_2x_2+\cdots+\beta_nx_n \quad (3)$$

其中，β_0 为常数项，β_0，β_1，…，β_n 为回归系数，x_n 表示选入到模型的预测指标。Logistic 回归模型提供变更请求报告关闭可能性的概率值。因此需要确定一个[0，1]区间的 Cut-off 点来决定变更请求报告得到关闭的可能性的高低，关闭可能性高

的预测为 Closed 状态，关闭可能性低的预测为 Open 状态。本文设置 Logistic 模型的 Cut-off 点为一般社会统计学软件常用的 0.5，即概率小于 0.5 时把变更请求报告分类为打开(Open)状态，大于 0.5 时分类为关闭(Closed)状态。

2.1 节中定义了 12 个衡量变更请求报告特征的指标。将在训练数据集上表现最佳的指标作为 Logistic 回归模型的输入变量。使用逐步回归[18](Stepwise Regression)模型选择的技术来选择在训练数据集上表现最佳的指标。在逐步回归中，有一种“向后回归(Backward Regression)”的策略。这种方法先将可能对因变量产生影响的自变量都纳入模型，然后逐步地从中剔除对因变量没有影响的自变量，直至所留在模型中的自变量都不能被剔除为止(通过设置置信水平)。

在进行逐步回归筛选指标后，我们在训练数据集上得到了几个表现最佳的预测指标。为找到一个通用的预测指标集合，将训练数据集含有项目总数的中位数作为分界线来确定哪些指标可以作为最终的预测指标集合。我们使用逐步回归训练了 N 个模型，计算逐步回归筛选后得到的每个指标在 N 个模型中出现的总次数，并最终确定在 $N/2$ 及其以上的模型中都表现最佳的指标为最终的预测指标集合。

2.3　预测模型性能综合评价指标

在数据集分布不平衡的预测模型中，精确率(Accuracy)不具有对结果的表征作用[19]。鉴于在预测变更请求报告关闭可能性时，其得到关闭的可能性会随着变更请求报告的属性特征不断演化而发生变化，因此不使用准确率(Precision)来评估模型。本文使用查全率(Recall)、伪正率(False Positive Rate)这两个指标来评价模型的性能，它们可以通过如表 2 所列的混淆矩阵计算得到。这两个指标在预测模型领域中得到了广泛应用[20-21]。

表 2　混淆矩阵

分类		预测	
		Open	Closed
实际	Open	TN=True Negative	FP=False Positive
	Close	FN=False Negative	TP=True Positive

查全率(Recall)，又称为召回率，是变更请求报告被正确预测为 Closed 的数目与总的实际为 Closed 的数目的比值，其计算公式如下：

$$Recall=TP/(TP+FN) \tag{4}$$

伪正率(False Positive Rate，FPR)是变更请求报告被错误预测为 Closed 的数目与总的实际为 Open 的数目的比值，其定义为：

$$FPR=FP/(FP+TN) \tag{5}$$

3　实验分析

3.1　数据集划分

本文将部署在 SourceForge 网站的开源软件项目作为研究数据来源。通常软件组织使用的 Issue 跟踪系统提供的变更请求报告的提交和管理过程是类似的，因此本文提出的方法很容易扩展到使用 Bugzilla，JIRA 等其他 Issue 跟踪系统的软件项目上。SourceForge 提供的 Issue 跟踪系统(Tickets)允许开发者和用户提交和管理缺陷、代码补丁请求、功能请求、支持请求等变更请求报告。SourceForge 提供的 Issue 跟踪系统的变更请求报告中 Open 状态的标记为蓝色，Closed 状态的标记为红色，且两个状态的变更请求报告分别在不同的类别下，我们可以很容易地获取报告的状态。

为找到一个通用的预测指标集合，我们选择分布于 SourceForge 不同类别下的软件项目。我们并未使用文献[22-24]使用的 SourceForge 公共数据集：FLOSSMole[25] 和 SRDA[26]。其原因是 FLOSSMole 只有一些宽泛量化的统计，比如在统计变更请求报告时，只有报告的数目，没有具体的报告内容；并且 FLOSSMole 和 SARD 分别于 2009 年和 2014 年之后就不再提供更新了。因此我们编写了脚本程序来获取本文的研究数据。

为了找到合适的数据集样本以及能够实现标准化的数据收集工作，我们限制了 SourceForge 软件项目的样本数量。首先，统计 SourceForge 首页 10 个类别的每个类别中下载次数最多(Weekly Downloads，2016 年 4 月)的前 50 个项目(SourceForge 每个类别的搜索结果每页有 25 个项目，统计前 2 页共 50 个项目)，统计共计 500 个项目；然后剔除那些没有使用 Issue 跟踪系统的项目；接着编写了 Python 脚本来收集变更请求报告数据。在 2016 年 6 月使用脚本爬取数据，然后将数据导入到 MongoDB 数据库中。我们抓取的变更请求报告的数据指标矩阵示例如表 3 所列，其中 URL 字段唯一标识变更请求报告。

表 3　变更请求报告指标矩阵

Project ID	URL	NREQWORD	NREQHREF	…	NREQLEN	inLABEL
keepass	sourceforge.net/p/keepass/bugs/1530	182	1	…	1141	−1
keepass	sourceforge.net/p/keepass/bugs/1532	161	0	…	1142	1

完成了数据收集的工作后，使用如下的筛选条件确定原始数据集：

(1)软件项目具有 Issue 跟踪系统；

(2)Issue 跟踪系统中有 500 个以上的变更请求报告；

(3)Issue 跟踪系统中的提交者数目在 50 人以上(提交者为 Anonymous 的不计数)；

(4)变更请求报告关闭的百分比在 4%～96%之间(确保数据集含有 Open 和 Closed 两种状态)。

确定了原始数据集后，选择每个软件类别报告数目最多的 4 个项目作为我们的研究数据集。至此确定了 40 个项目作为研究数据集(SourceForge 共有 10 个类别入口)。具体过程如下：

(1)对 40 个项目计算状态为 Closed 的百分比；

(2)选择每个类别中 Closed 百分比最高的 2 个项目加入训练数据集；

(3)选择每个类别中在步骤(2)后剩下的 2 个项目加入测试数据集。

最终确定的训练数据集如表 4 所列，测试数据集如表 5 所列。

表 4 训练数据集

Category	Project ID	提交者数量	数据量	Closed 量	Closed/%	数据集
Audio&Video	megui	289	902	858	95	Train
	dvdstyler	319	789	410	52	Train
Business & Enterprise	texniccenter	602	2206	1904	86	Train
	keepass	1515	3716	3158	85	Train
Communications	googlesyncmod	523	961	790	82	Train
	davmail	437	754	537	71	Train
Development	winmerge	1092	6467	5505	85	Train
	pmd	739	1925	1448	75	Train
Home & Education	gnuplot	638	2925	2653	91	Train
	jmol	147	840	744	89	Train
Games	mumble	1028	2602	2026	78	Train
	fceultra	118	544	407	75	Train
Graphics	flightgear	69	1863	1237	66	Train
	jfreechart	678	1858	1187	64	Train
Science & Engineering	jtds	534	968	833	86	Train
	maxima	537	3381	2569	76	Train
Security & Utilities	ipcop	828	1372	1232	90	Train
	passwordsafe	644	2131	1558	73	Train
System Administration	nsis	602	1935	1531	79	Train
	net-snmp	1294	4165	3294	79	Train
Total	—	—	42304	33881	—	—

表 5 测试数据集

Category	Project ID	提交者数量	数据量	Closed 量	Closed/%	数据集
Audio&Video	smplayer	691	1453	639	44	Test
	infrarecorder	141	869	181	21	Test
Business & Enterprise	odf-converter	106	3433	2497	73	Test
	freemind	1012	2283	863	38	Test
Communications	mobac	311	585	380	65	Test
	aresgalaxy	92	524	19	4	Test
Development	mingw	1460	3559	2366	66	Test
	kompozer	387	1052	256	24	Test
Home & Education	gimp-print	628	1421	961	68	Test
	crengine	236	576	127	22	Test
Games	dosbox	379	802	548	68	Test
	desmume	439	1071	640	60	Test
Graphics	sweethome3d	713	1406	854	61	Test
	meshlab	316	624	154	25	Test
Science & Engineering	avogadro	129	1061	712	67	Test
	librecad	221	815	537	66	Test
Security & Utilities	awstats	1419	3519	2485	71	Test
	clamwin	289	771	436	57	Test
System Administration	webadmin	1549	5550	3860	70	Test
	sevenzip	1386	3214	391	12	Test
Total	—	—	34588	18906	—	—

3.2 选择预测指标与构建模型

使用 SPSS 23 对每一个表 4 所列训练数据集中的项目执行逐步回归来筛选预测指标。为了得到尽可能多的预测指标，选择 SPSS 提供的“Backward:LR”策略。该策略使用基于最大似然估计的向后逐步回归方法，先将所有的指标变量放入方程中(inLABEL 作为目标变量)，然后依据似然比(Likelihood Ratio)检验的结果剔除不具有统计学意义的指标变量。在统计学上一般认为 p 值小于 0.05(即置信区间为 95%)的变量具有较好的显著性水平，此时无效假说不成立，说明预测指标变量具有统计学意义，能够显著地区分变更请求报告的打开状态和关闭状态。设定 SPSS 软件的置信区间为 95%。在表 4 所列的 20 个训练项目中得到表现最佳的预测指标，结果如表 6 所列，其中“√”表示在项目上筛选得到的指标。

统计每一个指标在 20 个项目中出现的总次数。选择出现次数在训练数据集的项目总数中位数(20/2=10)及其以上的指标作为最佳预测指标，结果如图 4 所示。图 4 中，我们在 20 个训练项目上筛选得到了表现最佳的预测指标集合：报告标题长度 NTITLELEN、报告内容行数 NREQLINE、报告含有附件数 NREQATTACH、开发者参与讨论次数 NREQOWNER、报告等待生存时间 REQWAITDAY。其中 REQWAITDAY 衡量了报告不断变化的报告生命周期，NREQOWNER 衡量了关键利益相关者(开发者)参与报告的讨论情况，其他 3 个指标刻画了报告标题和详细描述内容的复杂程度。

表 6　SPSS 逐步回归选择得到的指标

Project ID	NTIT-LELEN	NTIT-LEWORD	NREQ-LEN	NREQ-WORD	NREQ-LINE	NREQ-HREF	NREQ-ATTACH	NUMP-RETAG	NREQ-POSTS	NREQ-OWNER	REQWA-ITDAY
megui		√	√		√	√	√	√			
dvdstyler				√	√		√			√	√
texnic-center											√
keepass	√		√	√	√				√	√	√
google-syncmod		√		√		√	√			√	√
davmail		√	√		√	√	√	√			
winmerge	√		√		√	√	√		√		√
pmd	√	√							√		√
gnuplot		√		√	√	√		√		√	√
jmol	√		√		√		√				√
mumble	√	√								√	√
fceultra	√	√							√	√	√
flightgear								√	√		√
jfreechart	√		√		√		√		√	√	√
jtds							√			√	√
maxima	√		√	√	√	√		√	√	√	√
ipcop			√	√	√		√			√	√
pasword-safe	√			√	√					√	√
nsis		√	√		√		√		√		√
net-snmp	√			√	√	√		√		√	√
Total	10	8	9	8	13	7	10	3	8	12	18

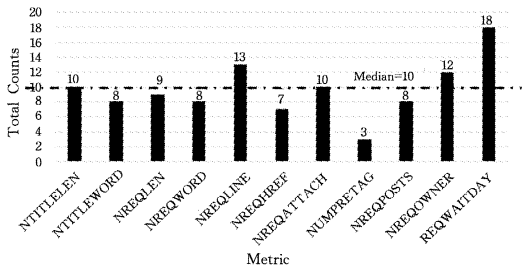


图 4　预测指标筛选结果

这 5 个指标是变更请求报告在不同方面的度量,它们之间不存在相关性,因此它们可以作为 Logistic 模型的输入变量。使用筛选得到的最佳预测指标集合来构建预测模型。

$$Logit(p)=\ln\left(\frac{p}{1-p}\right)$$
$$=\beta_0+\beta_1*NTITLELEN+\beta_2*NREQLINE+$$
$$\beta_3*NREQATTACH+\beta_4*NREQOWNER+$$
$$\beta_5*REQWAITDAY \quad (6)$$

其中,β_0为常数项,β_1,β_2,β_3,β_4,β_5为回归系数。

使用 SPSS 23 对表 5 中的每一个测试项目分别构建式(6)所描述的 Logistic 回归预测模型(共得到 20 个模型)。构建模型时,选择的 Cut-off 点为 0.5,即变更请求报告的 Closed 概率在 0.5 以下的预测为 Open,Closed 概率在 0.5 及其以上预测为 Closed。

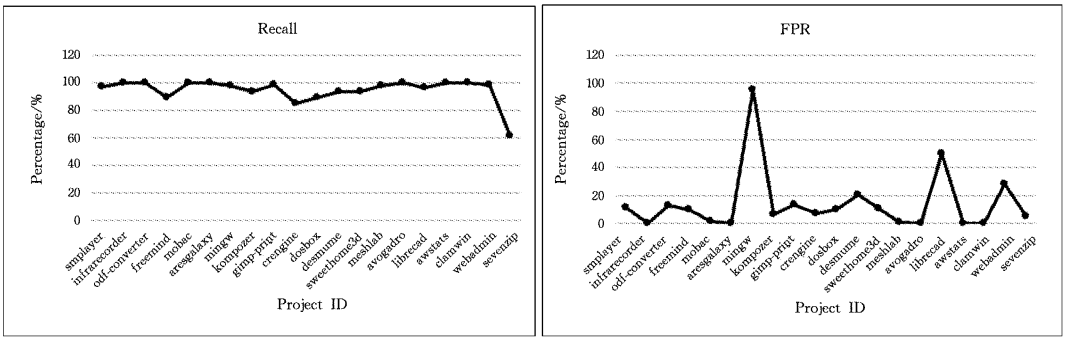


图 5　评估指标的值分布

表 7　评估指标性能

Project ID	提交者数量	数据量	Closed/%	TN	FN	FP	TP	Recall/%	FPR/%
smplayer	691	1453	44	724	19	96	622	97	12
infrarecorder	141	869	21	688	0	0	181	100	0
odf-converter	106	3433	73	817	9	119	1581	99	13
freemind	1012	2283	38	1275	93	145	770	89	10
mobac	311	585	65	203	1	4	379	100	2
aresgalaxy	92	524	4	505	0	0	19	100	0
mingw	1460	3559	66	53	48	1090	2319	98	95
kompozer	387	1052	24	741	17	55	239	93	7
gimp-print	628	1421	68	398	18	63	943	98	14
crengine	236	576	22	416	19	34	108	85	8

（续表）

Project ID	提交者数量	数据量	Closed/%	TN	FN	FP	TP	Recall/%	FPR/%
dosbox	379	802	68	1275	93	145	770	89	10
desmume	439	1071	60	342	42	90	623	94	21
sweethome3d	713	1406	61	495	59	58	794	93	10
meshlab	316	624	25	467	4	3	150	97	1
avogadro	129	1061	67	351	0	0	711	100	0
librecad	221	815	66	139	20	139	518	96	50
awstats	1419	3519	71	1321	5	3	2130	100	0
clamwin	289	771	57	335	0	0	436	100	0
webadmin	1549	5550	70	1211	60	481	3802	98	28
sevenzip	1386	3214	12	2677	150	145	241	62	5
平均	595	1729	49	722	33	134	867	94	14

表 8 librecad 的预测结果

Project ID	URL	inLABEL	标记状态	关闭概率	预测值	预测结果
librecad	https://sourceforge.net/p/librecad/bugs/485/	−1	open-fixed	0.823	1	√
librecad	https://sourceforge.net/p/librecad/bugs/476/	−1	open-accepted	0.818	1	√
librecad	https://sourceforge.net/p/librecad/bugs/464/	−1	open-fixed	0.709	1	√
librecad	https://sourceforge.net/p/librecad/bugs/456/	−1	open-fixed	0.815	1	√
librecad	https://sourceforge.net/p/librecad/bugs/449/	−1	open-fixed	0.271	−1	×
librecad	https://sourceforge.net/p/librecad/bugs/440/	−1	open-fixed	0.305	−1	×

3.3 实验结果

在 3.2 节中，我们在测试数据集上构建了 20 个预测模型，记录每一个预测模型在 2.3 节介绍的混淆矩阵中的 TN，FN，FP，TP 值。然后根据式(4)、式(5)分别计算查全率和伪正率。图 5 给出了 20 个测试项目上的综合评价指标值的分布情况，表 7 列出了 20 个测试项目的详细实验结果。由此可以得出以下结论：

(1)对于综合评价指标查全率，除 sevenzip 为 62%性能较差外，所提方法在其他 19 个测试项目上的查全率皆在 85%及以上。20 个测试项目中有 19 个具有较高的查全率，说明提出的方法能够很好地识别和预测 Closed 状态的变更请求报告。

(2)对于综合评价指标伪正率，大多数测试项目的伪正率值分布在 0%～20%之间，但 mingw 的伪正率高达到 95%。这说明该项目有大量 Open 状态的变更请求报告需要进一步处理。表 8 列出了伪正率为 50%的 librecad 部分预测结果，可以看到 librecad 预测为“伪正”的变更请求报告的标记状态和其应当具有的真实状态进行的分析比较。理论上，标记为 Open-Accepted 或 Open-Fixed 的报告在一段时间后应当得到关闭，人工阅读这些报告的评论发现，这些变更请求已经修复或完成，但在 1 年以后这些报告仍未关闭。与预测结果对比后发现，预测模型能够较为准确地预测报告的真实状态，即伪正率指标是有意义的。

(3)结论(1)和结论(2)表明提出的方法具有较高的预测性能。

(4)测试项目关闭状态变更请求报告所占百分比的大小并不影响其预测模型的性能。用 SPSS 23 计算表 7 中“Closed”列数据与“Recall”列和“FPR”列的 Pearson 相关系数，结果为 0.424(p 值为0.062，双尾)和 0.355(p 值为0.125，双尾)，这两个相关性系数的绝对值都小于0.5，且 p 值大于 0.05，这说明测试项目中关闭状态的变更请求报告所占的百分比大小并不显著影响预测模型的性能。

(5)测试项目变更请求报告数目的大小并不影响预测模型的性能。与结论(4)类似，使用 SPSS 23 计算表 7 中“数据量”列数据与“Recall”列和“FPR”列的 Pearson 相关系数，结果为−0.118(p 值为 0.618，双尾)和 0.372(p 值为 0.106，双尾)，这两个相关性系数的绝对值也都小于0.5，且 p 值大于 0.05，这说明测试项目中变更请求报告数目的大小并不显著影响预测模型的性能。

(6)变更请求报告具有的某些特征可用于构建其在下一版本中得到关闭可能性的预测模型。提出的方法基于衡量变更请求报告特征的指标构建 Logistic 回归模型。使用本文提出的方法筛选得到的 5 个最佳预测指标在 20 个测试项目上构建的预测模型得到了 19 个查全率在 85%及以上的模型。这说明变更请求报告具有的某些特征可以显著地区分 Open 和 Closed 的变更请求，构建的预测模型有较高的查全率。

4 讨论与不足

开发者需要更多地关注这些实际为 Open 状态却被预测为 Closed 的变更请求报告。由于 Logistic 回归模型的输入变量报告等待生存时间 REQWAITDAY 和开发者参与讨论次数 NREQOWNER 会随着报告的演化而不断变化，现在为 Open 状态的变更请求报告在将来也有可能会关闭。在缺陷预测时，伪正率表示没有缺陷的组件被错误地预测成了有缺陷。在预测变更请求报告的关闭可能性时，伪正率表示现在 Open 状态的变更请求在下一版本中可能会关闭。伪正率的预测可以让开发者了解 Issue 跟踪系统需要进一步处理变更请求报告的工作量，进而分配合理的时间、精力、金钱等资源。

在 3.3 节中，我们在 20 个测试项目上得到了平均查全率 94%和平均伪正率 14%的结果。实验结果说明，这 20 个测试项目中的 19 个项目都具有较高的查全率，另外一个较低的也达到 62%。这说明所提方法构建的模型具有较好的预测性能；本文选择的测试数据集项目平均有 14%的变更报告需要进一步优先处理。

本文的案例数据来源于全球最大的在线开源软件开发仓库 SourceForge，本文方法能够很容易地应用到大量部署在

SourceForge 上的开源软件项目。我们定义了的 12 个指标适用于任何使用自然语言书写的变更请求报告或软件需求。软件项目组织可以快速应用本文的方法筛选适合的预测指标，然后构建预测模型。据我们所知，本文第一次提供了预测变更请求报告关闭可能性的方法框架。我们定义了在 Issue 跟踪系统中较容易获取的 12 个指标，提供了筛选通用指标的方法，且使用 Logistic 回归构建的预测模型具有较高的预测性能。

但本文提出的方法也存在一些不足之处：

(1)20 个测试项目有 5 个项目没有预测到可能关闭的变更请求。这 5 个项目的查全率全部为 100%，即全部 Closed 的变更请求都被正确分类。但这 5 个项目没有预测到任何一个未来可能关闭的变更请求报告。当然随着变更请求报告的演化，这种情况可能会发生改变。这说明模型有改进的空间，可以根据需要调整 Cut-off 点，并结合自然语言处理技术构建变更请求报告的单词向量空间，以进行更为精确的预测。

(2)定义的指标未能衡量变量请求报告和评论内容的词汇与文本特征。本文未定义指标来衡量报告文本具有的音节数目、平均单词长度等词汇特征。本文也未定义指标来衡量变更请求报告的讨论对话包含的解决方案、详细变更请求询问以及解答问题疑惑等特征。

(3)本文筛选的最佳预测指标集合使用的数据样本较小。在寻找一个通用的预测指标集合时，只使用了 20 个 SourceForge 不同类别的项目。

(4)本文未对各个指标和变更请求报告关闭可能性的具体影响进行分析。在构建 Logistic 回归方程时得到的系数是有意义的，系数的符号可以解释为指标对变更请求报告得到关闭可能性的相关方向(促进、抑制、没有影响)，每一个系数值的大小粗略地说明了具体指标对变更请求报告得到关闭的可能性的影响，Hosmer 和 HemeShow[36] 详细说明了如何将这些系数转化为具体的概率。

(5)本文未进一步确认预测为“伪正”的变更请求报告应该具有的状态。表 8 列出了测试数据集 librecad 的部分预测结果，本文未对预测为“伪正”的变更请求报告进行完整的展示和分析。对这些预测为“伪正”的变更请求报告及其评论进行逐个人工阅读判断，将变更请求报告应当具有的实际状态与预测的状态进行比较，可以更为准确地评价预测模型的性能。

5　相关工作

5.1　缺陷报告优先级预测

过去的十年间，研究者们提出了很多基于数据挖掘和机器学习技术实现缺陷报告(变更请求报告的一种)优先级排序的方法。Uddin 等[27] 详细深入地总结了对缺陷报告优先级排序的相关研究。研究者使用缺陷报告的文本和类别特征，采用机器学习算法和数据挖掘技术对这些报告进行分类，实现了报告的优先级排序。研究者往往还使用自然语言处理技术，利用缺陷报告的文本和类别信息构建特征向量，然后使用支持向量机[28-29]、朴素贝叶斯[30]、决策树[31-32] 等机器学习技术实现缺陷报告的分类和排序。

实现缺陷报告优先级预测的方式主要有 3 种：根据提交者或报告分派者确定的优先级(Priority)、缺陷报告解决时间(Resolved Time)的长短以及提交者或报告分派者确定的严重程度(Severity)。Kanwal 和 Maqbool[33] 发现将缺陷报告的类别和文本特征相结合可在支持向量机上得到最高的精确率。他们将 Eclipse 缺陷报告作为数据集，选择具有 P1－P5 优先级且报告标记为关闭(Closed)或已解决(Resolved)的报告数据集进行训练和预测。他们的方法将大量的缺陷报告分类到同一个类别中，即在同一个类别中的缺陷报告具有相同的优先级。Hooimeijier 和 Weimer[14] 通过判断缺陷报告是否在给定的时间段内得到解决，将缺陷报告分类为“Cheap”和“Expensive”。他们提出的方法将缺陷报告的生命周期时间(Lifetime)作为分类时“Expensive”程度的代理(Proxy)。如果报告在一个 Cut-off 点之前解决，则这些报告需要花费的时间和资源较少，分类为“Cheap”，否则分类为“Expensive”。这种方法的难点在于每个项目具有不同的 Cut-off 点(即天数)，且没有得到报告的重要性程度。本方法得到的关闭概率越高，变更请求报告越重要。与 Hooimeijier 和 Weimer 类似，Giger 等[20] 通过分析缺陷报告的属性与修复时间的关系，试图将报告分为“Fast”和“Slow”两类。他们通过研究发现，使用报告提交后的信息(比如评论数目)可以提高预测模型的性能 5%～10%。我们使用的预测模型也使用了报告提交后的信息，但把报告的生命周期或等待的时间作为预测因子，认为生命周期也是报告演化的一部分。与使用优先级作为分类代理类似，Jianhong 等[34]、Menzies 和 Marcus[13] 提出的方法根据报告的严重程度将缺陷报告分为不同的等级。这种按照严重程度分类的方法往往只适合于缺陷报告，不适合于新增功能请求、源代码补丁请求等变更请求，并且并不是所有的缺陷报告都是反映缺陷相关的变更请求的。Herzig 等[35] 指出，Issue 跟踪系统中很多缺陷报告存在误分类，人们往往混淆了缺陷(Bugs)和功能(Features)。本文所提方法不止适用于缺陷报告，也适用于新增功能报告和代码补丁报告，即不区分报告的类型，统称为变更请求报告。现有的变更请求报告优先级排序的研究往往只适用于缺陷报告，本文所提方法可用于 Issue 跟踪系统支持的所有使用自然语言书写的变更请求报告。

本文方法可用于预测变更请求报告的重要性，与使用优先级、解决时间、严重程度作为报告重要性程度的“代理”类似。本文使用报告的状态作为报告重要性的代理，关闭可能性越高的报告越重要。已有的预测报告重要性的方法仅仅是对报告的优先级进行排序，决定了处理顺序的优先级。与预测报告重要性不同，预测报告关闭可能性让软件项目组织了解在下一版本开发中可能关闭的变更请求，统计需求工作量，为软件开发过程活动提供指导，并即时关闭 Issue 跟踪系统应当关闭的报告，以避免不必要的返工并提高用户反馈的积极性和满意度。

5.2 需求易变性的预测

软件需求的变动会增加软件开发的花费和时间安排的延期。需求的变动包括需求内容的变动以及需求数目的变化。需求易变性(Requirements Volatility)指的是需求文档随着时间变化的总量[8]。能力成熟度模型集成[37](Capability Maturity Model Integration,CMMI)的能力等级 4 或 5 要求软件企业能够以量化的方式管理其需求变化(Requirement Change)。量化和预测需求的变化可以帮助软件实践者提高需求管理活动的能力。预测需求的易变性可以帮助项目管理人员在项目风险最小化和建立更稳定过程时采取适当的行动[7]。Loconsole 和 Börstler[7]使用瑞典某软件公司的用例模型(Use Case Model,UCM)作为研究数据来源,收集软件项目需求文档所有版本的大小和变更数据,探究 UCM 大小(Size)的测量指标(行数、单词数、Use Case 数目、Actor 数目)与总变更数目之间的关系。然后计算需求文档大小的每一个测量指标与需求文档变更大小之间的 Spearman 相关系数。他们的结果表明,在预测基于用例的需求文档变更数目时,需求文档大小的测量指标是一个很好的指标,越大的用例需求越容易发生变更。在其随后的工作中[8],他们对需求易变性进行了单变量和多变量回归分析。结果发现,需求文档的行数(Lines)在预测需求易变性时是最为显著的指标。他们定义的指标比较常见并且可以用于任何文本形式书写的用例文档,并且“行数”这个指标可以用于任何文本形式的需求。Issue 跟踪系统中的变更请求往往是用自然语言书写的,没有传统需求工程中的用例。受其工作的启发,我们定义了 12 个指标来刻画 Issue 跟踪系统中常见的变更请求报告的标题、详细描述内容的复杂度、利益相关者参与讨论的情况以及变更报告等待的时间。定义的指标比较容易获取且常见,可用于绝大多数(如 Bugzilla,JIRA)Issue 跟踪系统。

Shi 等[21]提出了一种能够基于软件变更历史来预测未来可能发生变更需求的方法。他们定义了衡量软件制品演化历史的度量指标,提供了训练预测模型的过程框架。其与本文研究目的不同,基于的数据也不同。他们定义的指标衡量的是历史版本之间需求变化的信息,衡量的是在时间维度上软件需求的变化情况,例如需求变化的次数,两次需求变更之间的时间距离等;本文方法衡量了每一个变更请求报告的特征,即软件需求方面的内容。他们的方法基于软件需求历史变更信息(需求的增加、删除、修改、没有变化)构建需求演化矩阵,进而计算定义的 6 个指标;本文方法直接使用从 Issue 跟踪系统中收集到的指标,定义的指标也直接通过抓取数据来自动收集计算。

如果同时联合 Issue 跟踪系统的变更请求报告以及变更请求演化的历史信息,将可以更好地进行变更请求的预测及影响分析。在预测得到可能发生状态改变的变更请求后,可以为软件过程活动的改善提供支持,如升级计划或通过定义新的活动来消除变更的原因,这样变更请求的实现可以转化为软件过程改善的策略。

据我们所知,目前还没有面向 Issue 跟踪系统的变更请求报告关闭可能性预测的相关研究。

结束语 Issue 跟踪系统中存在着大量长期未关闭的变更请求报告,本文提出预测这些变更请求报告得到关闭可能性的方法。通过定义 12 个衡量变更请求报告特征的指标来构建预测模型。为了找到一个通用的预测指标集合,本文选择了 20 个 SourceForge 项目作为训练数据集,使用逐步回归技术,筛选得到在大多数项目上预测效果最佳的 5 个指标。

使用筛选得到的 5 个指标对 20 个 SourceForge 测试项目(与训练数据集的项目完全不一样)构建 Logistic 回归模型。由实验结果可知,所提方法在 19 个项目上得到了查全率在 85%及以上的较高预测性能。我们在训练数据集上筛选得到的预测效果最佳的指标集合迁移到新的测试项目时具有较高的预测性能的原因在于,我们在筛选预测指标时使用了足够多且分布在不同类别下的软件项目。我们构建预测模型时使用了 5 个预测指标:报告标题长度 NTITLELEN、报告的行数 NREQLINE、报告含有的附件数 NREQATTACH、开发者参与讨论的次数 NREQOWNER、报告的等待生存时间 REQWAITDAY。

变更请求报告是不断演化的。量化和预测这些处于 Open 状态且不断演化的报告在下一版本中得到关闭的可能性,可以为开发者提示这些报告的重要性,使其关注优先级比较高的报告,辅助开发者分派变更请求报告,以及改善现有的 Issue 跟踪系统设计。这在 SourceForge,GitHub,Apache 等部署有 Issue 跟踪系统的开源软件项目开发实践中具有广泛的应用意义。预测变更请求报告关闭的可能性还可应用于预测软件开发和维护过程中任务的需求工作量,为改善软件过程活动提供指导。

下一步的研究工作将把变更请求报告关闭可能性的预测结果应用于软件过程改进的推荐。

参 考 文 献

[1] ANVIK J,HIEW L,MURPHY G C. Who should fix this bug? [C]// Proceedings of the 28th International Conference on Software Engineering. ACM,2006:361-370.

[2] ČUBRANIĆ D. Automatic bug triage using text categorization [C]// Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering(SEKE 2004). 2004.

[3] ANVIK J. Assisting bug report triage through recommendation [D]. Vancouver:University of British Columbia,2007.

[4] ZHANG J,WANG X,HAO D,et al. A survey on bug-report analysis[J]. Science China Information Sciences,2015,58(2):1-24.

[5] WEISS D,ENTRY B E X. A large crawl and quantitative analysis of open source projects hosted on sourceforge[OL]. http://core. ac. uk/display/24543641.

[6] RAYMOND E. The cathedral and the bazaar[J]. Knowledge,Technology & Policy,1999,12(3):23-49.

[7] LOCONSOLE A,BÖRSTLER J. An Industrial Case Study on Requirements Volatility Measures[C]// APSEC. 2005:249-256.

[8] LOCONSOLE A, BÖRSTLER J. A Correlational Study on Four Size Measures as Predictors of Requirements Volatility[J]. Submitted to Journal of Software Measurement, 2007: 166-170.

[9] PFAHL D, LEBSANFT K. Using simulation to analyse the impact of software requirement volatility on project performance [J]. Information and Software Technology, 2000, 42(14): 1001-1008.

[10] STARK G E, OMAN P, SKILLICORN A, et al. An examination of the effects of requirements changes on software maintenance releases [J]. Journal of Software Maintenance, 1999, 11(5): 293-309.

[11] ZOWGHI D, NURMULIANI N. A study of the impact of requirements volatility on software project performance [C]// Ninth Asia-Pacific Software Engineering Conference. IEEE, 2002: 3-11.

[12] ALI M J. Metrics for requirements engineering[OL]. http://www8. cs. umu. se/education/examina/Rapporter/JaveedAli. pdf.

[13] MENZIES T, MARCUS A. Automated severity assessment of software defect reports[C]// IEEE International Conference on Software Maintenance(ICSM 2008). IEEE, 2008: 346-355.

[14] HOOIMEIJER P, WEIMER W. Modeling bug report quality [C]// Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering. ACM, 2007: 34-43.

[15] ZHANG H. An investigation of the relationships between lines of code and defects[C]// IEEE International Conference on Software Maintenance(ICSM 2009). IEEE, 2009: 274-283.

[16] BETTENBURG N, JUST S, SCHRÖTER A, et al. Quality of bug reports in Eclipse[C]// Proceedings of the 2007 OOPSLA Workshop on Eclipse Technology EXchange. ACM, 2007: 21-25.

[17] BETTENBURG N, JUST S, SCHRÖTER A, et al. What makes a good bug report? [C]// Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering. ACM, 2008: 308-318.

[18] CHAMBERS J M, HASTIE T J. Statistical models in S[M]. CRC Press, Inc., 1991.

[19] TAN P N. Introduction to data mining[M]. Pearson Education India, 2006.

[20] GIGER E, PINZGER M, GALL H. Predicting the fix time of bugs[C]// Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering. ACM, 2010: 52-56.

[21] SHI L, WANG Q, LI M. Learning from evolution history to predict future requirement changes[C]// 2013 21st IEEE International Requirements Engineering Conference (RE). IEEE, 2013: 135-144.

[22] SCHWEIK C M, ENGLISH R, HAIRE S. Factors leading to success or abandonment of open source commons: An empirical analysis of Sourceforge. net projects[J]. Fatigue & Fracture of Engineering Materials & Structures, 2009, 32(7): 567-572.

[23] GAROUSI V, LEITCH J. IssuePlayer: An extensible framework for visual assessment of issue management in software development projects[J]. Journal of Visual Languages & Computing, 2010, 21(3): 121-135.

[24] ABDOU T, GROGONO P, KAMTHAN P. Managing Corrective Actions to Closure in Open Source Software Test Process[C]// SEKE. 2013: 306-311.

[25] HOWISON J, CONKLIN M, CROWSTON K. FLOSSmole: A collaborative repository for FLOSS research data and analyses [J]. International Journal of Information Technology and Web Engineering (IJITWE), 2006, 1(3): 17-26.

[26] MADEY G. The sourceforge research data archive (SRDA) [D]. South Bend: University of Notre Dame, 2010.

[27] UDDIN J, GHAZALI R, DERIS M M, et al. A survey on bug prioritization[J]. Artificial Intelligence Review, 2016, 47(2): 145-180.

[28] KANWAL J, MAQBOOL O. Managing open bug repositories through bug report prioritization using SVMs[C]// Proceedings of the International Conference on Open-Source Systems and Technologies. Lahore, Pakistan, 2010.

[29] CHATURVEDI K K, SINGH V B. Determining bug severity using machine learning techniques[C]// 2012 CSI Sixth International Conference on Software Engineering (CONSEG). IEEE, 2012: 1-6.

[30] LAMKANFI A, DEMEYER S, GIGER E, et al. Predicting the severity of a reported bug[C]// 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010). IEEE, 2010: 1-10.

[31] VALDIVIA GARCIA H, SHIHAB E. Characterizing and predicting blocking bugs in open source projects[C]// Proceedings of the 11th Working Conference on Mining Software Repositories. ACM, 2014: 72-81.

[32] ALENEZI M, BANITAAN S. Bug Reports Prioritization: Which Features and Classifier to Use?[C]// 2013 12th International Conference on Machine Learning and Applications (ICMLA). IEEE, 2013: 112-116.

[33] KANWAL J, MAQBOOL O. Bug prioritization to facilitate bug report triage[J]. Journal of Computer Science and Technology, 2012, 27(2): 397-412.

[34] JIANHONG Z, SANDHU P S, RANI S. A Neural network based approach for modeling of severity of defects in function based software systems[C]// 2010 International Conference On Electronics and Information Engineering (ICEIE). IEEE, 2010: 568-575.

[35] HERZIG K, JUST S, ZELLER A. It's not a bug, it's a feature: how misclassification impacts bug prediction[C]// Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, 2013: 392-401.

[36] HOSMER JR D W, LEMESHOW S. Applied logistic regression [M]. John Wiley & Sons, 2004.

[37] TEAM C P. Capability Maturity Model® Intergration(CMMI), Version 1. 1—Staged Representation[J]. Software Engineering Institute Carnegie Mellon University, 2002(1): 31-37.