面向单个文件的个性化缺陷预测方法

陈 恒 刘文广 高东静 彭 鑫 赵文耘

(复旦大学软件学院 上海 201203) (上海市数据科学重点实验室(复旦大学) 上海 201203)

摘 要 现有的缺陷预测方法大多数是面向项目或个人的,这些方法或没有区分文件之间和开发人员之间的差异性,或只区分了开发人员的差异性。然而,在软件开发中,开发人员之间和代码文件之间的差异性是同时存在的,而且这些差异性都可能会影响缺陷建模或预测的结果。因此,如果缺陷预测方法忽视这些差异性或忽视其中任意一种,针对整个项目或某个开发人员建立缺陷预测模型均可能会影响预测准确性。针对此问题,提出了一种面向单个文件的个性化缺陷预测方法,即将每个开发人员修改每个代码文件的历史数据都作为单独的数据集,建立对应的缺陷模型,并将之用来预测对应开发人员修改对应文件的缺陷情况。通过实验初步确认了在单个文件的个人缺陷数据充分的情况下该方法能够有效地提高缺陷预测的准确性。

关键词 缺陷预测,个性化,源代码,提交

中图法分类号 TP311.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.04.020

Personalized Defect Prediction for Individual Source Files

CHEN Heng LIU Wen-guang GAO Dong-jing PENG Xin ZHAO Wen-yun

(School of Software, Fudan University, Shanghai 201203, China)

(Shanghai Key Laboratory of Data Science, Fudan University, Shanghai 201203, China)

Abstract Most defect prediction methods are project-oriented or developer-oriented. These methods do not distinguish the differences between source files and between developers, or only distinguish the differences between developers. However, there exist differences between developers and between source files in development process, and both differences may have an effect on defect prediction model and result. Therefore, if these two kinds of differences or either one kind of differences are ignored, and defect prediction models based on either the whole project or a single developer's development history are built, the prediction accuracy may be affected. To solve this problem, this paper proposed a personalized defect prediction method for individual source files, that is, we regarded the historical data of each file modified by each developer as an independent dataset, built a corresponding defect prediction model and used it to predict the defect possibility for the corresponding file modified by the corresponding developer. Experiments show the proposed method can improve the prediction accuracy with sufficient personal defect data.

Keywords Defect prediction, Personalization, Source file, Committing

1 引言

随着计算机科学技术的应用越来越广泛,软件规模和复杂度显著增长,在软件开发的过程中,软件缺陷更加容易出现。软件缺陷是评估软件质量的重要指标之一,能否保证软件质量决定了人们在相关领域的活动能否顺利开展。

针对这个问题,不少学者在缺陷预测领域进行了大量研究,缺陷预测技术能够很好地辅助相关人员了解软件缺陷的情况,以评估软件质量。缺陷预测是利用统计学、机器学习等领域的方法(如线性回归、神经网络等)对代码度量数据、历史版本中的缺陷信息和缺陷库中的信息进行综合分析,统计、学习出代码文件中缺陷发生的规律或模型,然后利用规律或模型对后续开发过程中代码文件可能出现的缺陷进行预测。

现有的缺陷预测技术大都是针对整个项目的开发历史进

到稿日期:2015-11-30 返修日期:2016-03-04 本文受国家自然科学基金(61370079),国家高技术研究发展计划(863)(2013AA01A605)资助。

陈 恒(1992—),男,硕士生,主要研究方向为软件维护与演化,E-mail:hengchen15@fudan.edu.cn;刘文广(1991—),男,硕士生,主要研究方向为软件维护与演化;高东静(1992—),女,硕士生,主要研究方向为软件维护与演化;彭 鑫(1979—),男,博士,教授,CCF高级会员,主要研究方向为软件维护、自适应软件、软件复用、产品线等,E-mail:pengxin@fudan.edu.cn;赵文耘(1964—),男,博士,教授,CCF高级会员,主要研究方向为软件工程、企业应用集成、电子商务等。

行分析预测的。研究表明，开发人员的不同代码风格、提交频率和开发经验等都可能会对缺陷预测模型产生影响（开发人员差异性）。基于此结论，Tian J 等人[1]提出了一种个性化的缺陷预测方法。然而，即使是同一个开发人员，在修改不同的代码文件时，由于这些文件具有不同的结构、嵌套深度和代码复杂度等（文件差异性），也可能会呈现出不同的软件缺陷规律。因此，只考虑开发人员差异而忽略代码文件差异性的缺陷预测方法可能会影响缺陷预测的精度。

为了解决现有缺陷预测技术没有同时考虑开发人员差异性和文件差异性的问题，本文提出了一种面向单个文件的个性化缺陷预测方法。该方法同时考虑了开发人员和代码文件差异性可能对缺陷模式产生的影响，对不同的个人和文件建立单独的预测模型。实验表明，该方法在个人缺陷数据充足的条件下能够提高缺陷预测的精度。

本文第 2 节介绍了相关工作；第 3 节介绍了特征抽取和缺陷引入识别方法；第 4 节介绍了所提的缺陷预测方法；第 5 节介绍了实验设计；第 6 节对实验结果进行了分析；第 7 节对所提方法的适用条件和实际应用场景进行了讨论；最后总结全文，并对未来工作进行了展望。

2　相关工作

目前已有大量的学者对缺陷预测进行了相关研究。王青等人[2]对静态和动态缺陷预测技术进行了总结，总结的缺陷预测技术主要针对整个项目开发流程；原子等人[3]采用静态分析和自然语言处理技术对整个项目的开发过程进行了细粒度（语句级）的缺陷预测；Premraj R 等人[4]使用软件实体网络相关的度量（如出入度等）分布进行了数值型和存在性预测；Chulani S[5]提出了 COQUALMO 缺陷密度预测模型的流程。也有一些学者对缺陷或代码元素进行聚类后再进行缺陷预测[6-7]。涂亚明等人[8]对开发流程中的系统测试阶段的缺陷预测模型进行了分析。为了提高缺陷预测的性能，也有一些学者针对属性选择、过滤方法进行了研究[9-10]。然而这些工作及其他一些缺陷预测的工作[11-12]主要是面向项目的，同时也没有严格区分开发人员和代码文件之间的差异性。

然而，开发人员有自己的代码风格，不同的软件开发人员甚至不同文件的缺陷引入模式及分布规律都可能有较大差别，因此可能需要对不同开发人员和代码文件区别对待。Tian J 等人[1]通过将每个开发人员的历史缺陷数据作为单独的数据集建立了缺陷预测模型，并证明了区分对待开发人员的必要性。然而该方法并未区分不同的代码文件，并且针对的是开发过程中的某个时间段而不是每次提交。文献[13]利用项目开发历史上某一段时间内的所有开发人员修改所有文件的提交是否引入了缺陷的数据建立 SVM 模型，运用该模型预测未来提交的文件是否会引入缺陷。本文则同时将开发人员和代码文件分开，以提交为粒度，对每个开发人员修改的每个文件均建立一个缺陷模型，并且通过实验验证了接此区分的意义。

3　缺陷特征抽取和缺陷引入识别

面向单个文件的个性化缺陷预测方法以修改作为基本单元进行缺陷预测。一次修改指的是软件版本控制系统的一次提交（Commit）中一个文件被修改的代码行[1]。本文的特征抽取方法借鉴了 Kim S 等在文献[13]中的工作。每次代码修改是否引入缺陷的识别方法则采用文献[14]中提出的 SZZ 算法。本节内容主要有：抽取历史修改记录（3.1 节）、特征抽取（3.2 节）和缺陷引入识别-SZZ 算法（3.3 节）。

3.1　抽取历史修改记录

首先，从软件配置管理（Software Configuration Management，SCM）系统中抽取源代码的修改历史，抽取的内容包括提交日志（Commit Log）、开发人员和提交日期等；然后，这些修改信息通过 3.2 节的处理可以将代码文件的一次修改转化为一个特征值的集合；最后，利用 SZZ 算法对每次修改进行标记。

3.2　特征抽取

本文按照文献[13]的方法从提交元数据、提交日志和代码文件 3 个数据源挖掘每次修改的特征。

3.2.1　提交元数据特征

在一些研究中，累计缺陷数量和累计修改次数被广泛地用作缺陷预测的特征[14-16]，本文也将这些过程度量用于所提方法中。此外，通过挖掘提交日志信息，本文将开发人员提交的时间（1－7 表示周一至周日，1－24 表示提交小时）、提交日志的字符串长度、该次提交修改的代码行数（即同一个文件在前后两个版本中被修改（包括增加和删除）的代码行数）和提交后版本的文件代码总行数作为特征。

3.2.2　提交日志特征

版本管理中每一条提交日志都有一定的意义。例如，Eclipse Core 项目的某条提交日志（“HEAD-Remove empty block warnings”）体现的意义是这次提交去除了空代码块的警告。为了简便，本文借鉴了词袋（Bag of Words，BOW）模型[17]的思想来挖掘提交日志的特征，以体现提交日志的意义。具体的处理步骤：首先将提交日志信息中的所有字母转化为小写字母；然后去除里面的特殊符号，按照空格将一长串信息分割成一个个单词；最后统计每个单词的频数作为缺陷预测的特征。对“HEAD-Remove empty block warnings”这条日志信息的处理结果是特征集{head=1，remove=1，empty=1，block=1，warnings=1}。

3.2.3　代码文件特征

对源代码文件的处理与对提交日志特征（3.2.2 节）的处理类似，文件可以看成是包含了更多信息且更长的字符串。但因为代码自身的特点，代码文件与提交日志特征抽取的词袋模型处理方法还存在以下差异：1）除了源文件中的关键字、描述符、数字和注释等外，本文还抽取代码中的操作符（如＋＋，&&. 等）作为特征；2）对代码的标志符命名（按驼峰命名、下划线等）进行分解，例如“gitChangeLineExtractor”会被分解为

"git","change","line"和"extractor" 4 个词;3)代码文件中所有被空格或分号分割的词及对应的频数都被视为特征。

一次修改包括了一个文件的前后两个版本,对两个版本的文件分别抽取特征之后,本文仅仅将两个文件特征集的增量(修改后的版本比修改前的版本增加的特征)和减量(修改后的版本比修改前的版本减少的特征)作为特征,这样做主要出于两个考虑:1)一个文件两个版本的差值更能描述修改这一行为;2)如果将源文件所有的词都作为特征,部分文件的特征会很多,导致建模和预测的效率低下。表 1 列出了这些特征。

表 1 特征集

特征数据源	描述	示例特征
提交元数据	提交特征	提交时间,代码长度等
提交日志	日志文本意义	fix,remove,add 等
代码文件	文件差值	&&,double,else 等

3.3 缺陷引入识别-SZZ 算法

本文应用 SZZ 缺陷引入识别算法(Bug-introducing Change Identification Algorithm)来识别每次修改是否引入缺陷,并将对应的文件修改标记为引入缺陷(Buggy)和不引入缺陷(Clean)。缺陷引入识别算法是由 Sliwerski J,Zimmermann T 和 Zellert A 在文献[14]中提出的。

首先,SZZ 算法通过搜索提交日志信息,判断其中是否存在某些关键字可以确定某次提交是否修复了缺陷。例如,在 Eclipse Core,Eclipse Debug 和 Eclipse UI 项目中,主要搜索以下两类关键字:1)搜索例如"Fixed"或者"Bug"这类关键字[18];2)搜索缺陷报告的索引,例如"#225500"[14,19]。此外,本文还对某次提交是否真正修复缺陷进行了人工分析,剔除了误报的情况。

然后,SZZ 算法使用 diff 工具检测该提交中所有文件修改的代码行,每个文件对应的修改前的代码行被认为是缺陷存在的位置。

最后,SZZ 算法利用版本库中的追踪功能(Git 为 blame,SVN 为 annotate)在版本历史中回溯检测出最近添加或最后修改这行代码的提交,并认为这次提交引入了缺陷,将其标记为引入缺陷(Buggy)。

4 本文的缺陷预测方法

本节主要介绍本文提出的面向单个文件的个性化缺陷预测方法,包括建模的分类器和评估分类效果的评价指标。

4.1 面向单个文件的个性化缺陷预测方法

在项目开发中,每个开发人员会修改多个代码文件,每个文件也会被多个开发人员修改。为了消除不同人员和不同文件的差异对缺陷预测造成的影响,本文将每个开发人员对每个文件的历史修改均作为独立的数据集,利用分类算法,建立对应缺陷预测模型,并用缺陷模型预测后续开发过程中对应开发人员修改对应文件的代码提交。

例如,在表 2 中,pmulet 和 oliviert 都修改过 Main. java 和 QualifiedNameReference. java 文件,本文方法将对这些开发人员和文件交叉建立 4 个缺陷预测模型。

表 2 交叉组合建模示例

作者\文件	Main. java	QualifiedNameReference. java
pmulet	Model_p_M	Model_p_Q
oliviert	Model_o_M	Model_o_Q

4.2 分类器

本文使用朴素贝叶斯、SMO 支持向量机和 C4.5 决策树的分类器对缺陷数据建立缺陷模型。

朴素贝叶斯分类器(Naive Bayes Classifier,NBC)是一种基于贝叶斯定理独立性假设的概率分类器,其被广泛应用于二元分类问题。NBC 假设属性之间相互独立,具有较小的误差率和稳定的分类效率。该分类器的优点在于需要的参数很少,对缺失的数据不敏感,算法比较简单。

支持向量机(Support Vector Machine,SVM)是一种监督式学习方法,通常用于模式识别、统计分类和回归分析。SVM 是一个能够将不同样本在特征空间上分隔的超平面,它的学习策略是使间隔最大化,一般可以转化为二次规划问题进行求解。SMO 算法是一种二次规划的优化算法,在数据稀疏的情况下性能优势明显。

决策树是以"条件分支树"的形式进行分类的方法。树中的节点表示对象,路径代表属性选择方法,叶结点对应着某个实例按照其属性从根节点到该叶节点进行选择得到的最终类别。C4.5 是一种构建决策树的算法。在节点划分时,C4.5 选取具有最高信息增益且尚未被用于划分的属性作为划分标准。C4.5 算法的优点在于产生的分类规则易于理解,准确率较高。

4.3 评价指标

为了评估预测效果,本文选用了准确率(Precision)、召回率(Recall)和 F1-Measure 这 3 个评价指标。

对于 Buggy 的修改,在所有的预测结果中,记真阳性的个数为 n_{tp},假阴性的个数为 n_{fn},真阴性的个数为 n_{tn},假阳性的个数为 n_{fp},有以下两个指标:

$$Presicion_{(b)}=\frac{n_{tp}}{n_{tp}+n_{fp}} \tag{1}$$

这个指标表征了所有被预测为会引入缺陷的修改中确实会引入缺陷的比例。

$$Recall_{(b)}=\frac{n_{tp}}{n_{tp}+n_{fn}} \tag{2}$$

这个指标表征了所有会引入缺陷的修改中被预测到会引入缺陷的比例。

同样地,对于 Clean 的修改,也有这两个指标:$Presicion_{(c)}$ 和 $Recall_{(c)}$。本文将综合这两类评价指标来评估对所有修改的预测效果。记同一个开发人员修改同一个文件的总次数为 n,其中 Buggy 的修改次数记为 n_b,Clean 的修改次数记为 n_c,满足 $n=n_b+n_c$。本文用以下 3 个评价指标来评估总体预测效果:

$$Precision=Presicion_{(b)}*\frac{n_b}{n}+Presicion_{(c)}*\frac{n_c}{n} \quad (3)$$

$$Recall=Recall_{(b)}*\frac{n_b}{n}+Recall_{(c)}*\frac{n_c}{n} \quad (4)$$

为了综合考虑准备率和召回率，本文也将 F1-Measure 作为评价指标。F1-Measure 是根据准确率和召回率给出的一个综合的评价指标，可以看作准确率和召回率间的一个平衡点。

$$F1\text{-}Measure=\frac{2*Precision*Recall}{Precision+Recall} \quad (5)$$

5　实验设计

为了验证面向单个文件的个性化缺陷预测方法的效果，本文设计了以下实验。

5.1　实验对象

本文的实验对象是 Eclipse 的 3 个开源项目 Core，Debug 和 UI，表 3 列出了 3 个项目的概况。这 3 个项目都具有时间跨度久、开发人员多、文件总数多和提交次数多的特点。在开发的过程中，大量的修改信息和缺陷信息被记录在提交日志和缺陷报告中。通过对缺陷信息的挖掘、修改特征的抽取、缺陷引入的识别，本文应用 3 个分类器建立针对开发人员和代码文件的缺陷预测模型，并选用十折交叉验证方法计算评价指标值，通过实验验证本文提出的面向单个文件的个性化缺陷预测方法在这 3 个项目上的预测效果。

表 3　实验对象概况

项目	提交数	时间跨度	开发人员数	文件数
Core	21027	2001.6—2014.5	62	9470
Debug	9226	2001.5—2014.5	42	3806
UI	26622	2001.5—2014.5	63	17566

5.2　对照组实验

为了验证本文提出的缺陷预测方法可以提高预测的精度，本文设置了两个对照组：1）将多个开发人员对同一个文件的历史修改作为数据集，建立缺陷预测模型；2）将同一个开发人员对多个文件的历史修改作为数据集，建立缺陷预测模型。将本文的缺陷预测方法的结果和对照组的预测结果进行对比分析，可以验证本文方法可以提高缺陷预测的精度。

6　实验结果分析

本节展示了在 Core，Debug 和 UI 3 个 Eclipse 开源项目上的实验结果，并与对照组的预测结果进行了对比。

6.1　面向单个文件的个性化缺陷预测结果分析

按照随机抽样的原则，本文分别在 3 个项目中随机抽取了开发人员和代码文件的多个分组进行实验，并且排除了一个开发人员对一个文件的修改次数过少的情况。表 4 列出了本文方法实验结果的概况，对于每个项目，均采用 3 个分类器分别建立缺陷预测模型；表 4 中的评价指标是多个分组指标的平均值。

表 4　实验结果概况

项目	分类器	准确率	召回率	F1-Measure
Core	NBC	0.706	0.623	0.662
	SMO	0.744	0.748	0.746
	C4.5	0.763	0.744	0.753
Debug	NBC	0.707	0.629	0.665
	SMO	0.640	0.678	0.658
	C4.5	0.621	0.625	0.623
UI	NBC	0.731	0.709	0.720
	SMO	0.759	0.742	0.750
	C4.5	0.703	0.702	0.702

从表 4 的实验结果中可以看到，在不同项目和不同分类器上，本文方法都有较理想的准确率和召回率，能够实现一定精度的缺陷预测。

6.2　对照组结果分析

对照组 1 中，本文将多个开发人员对同一个文件的历史修改作为数据集，建立预测模型进行缺陷预测；对照组 2 中，本文将同一个开发人员对多个文件的历史修改作为数据集，建立统一的预测模型进行缺陷预测。为了保证控制变量，本文方法和对照组均采用 NBC 分类器建立预测模型。表 5 和表 6 分别展示了在 UI 项目上本文方法与对照组 1 和对照组 2 的实验结果。为了简化表格，用 file1，file2，…表示不同的文件。

表 5 和表 6 中，权重是该行的开发人员对代码文件的修改次数占该表中所有开发人员对该文件修改总次数的比值，因此只有前 5 行数据有权重。第 1—5 行是本文缺陷预测方法的实验结果，第 6 行是以上结果的加权平均数，第 7 行是对照组 1 或对照组 2 的实验结果。

表 5　本文方法和对照组 1 的实验结果

开发人员	代码文件	准确率	召回率	F1-Measure	权重
kmaetzel	file1	0.766	0.748	0.757	0.302
dmegert	file1	0.762	0.760	0.761	0.374
teicher	file1	0.677	0.679	0.678	0.191
cknaus	file1	0.652	0.650	0.651	0.072
dbaeumer	file1	0.559	0.588	0.573	0.061
加权平均	—	0.727	0.722	0.725	—
对照组 1	—	0.676	0.673	0.674	—

从表 5 中可以看到，第 1—3 行的各项指标均优于对照组 1，第 4—5 行的各项指标虽然不如对照组 1，但是相差不大且权重较小；从表 6 中可以看到，第 1—2，4—5 行的各项指标均优于对照组 1，第 3 行的各项指标虽然不如对照组 2，但是也相差不大且权重较小。综合表 5 和表 6，本文方法对这几组实验结果的加权平均数优于对照组 1 和对照组 2，在准确率、召回率和 F1-Measur 方面均有 5%左右的提升。

表 6　本文方法和对照组 2 的实验结果

开发人员	代码文件	准确率	召回率	F1-Measure	权重
maeschli	file1	0.822	0.767	0.794	0.242
maeschli	file2	0.713	0.699	0.706	0.176
maeschli	file3	0.655	0.652	0.653	0.197
maeschli	file4	0.733	0.712	0.722	0.149
maeschli	file5	0.695	0.669	0.682	0.236
加权平均	—	0.727	0.701	0.714	—
对照组 2	—	0.669	0.651	0.660	—

此外，本文还在其他数据集上做了类似的实验，如在 De-

bug 项目中：1) darin，jburns，darins 和 lbourlier 都修改过 JDIThread. java 这个文件，按照对照组 1 的方法，准确率是 0. 676，召回率是 0. 674，F1-Measure 是 0. 675；若采用本文方法，准确率是 0. 697，召回率是 0. 726，F1-Measure 是 0. 711。2)darin 修改过 JDIThread. java，JavaRuntime. java，JDIDebug-Target. java，JavaBreakpoint. java，JDIModelPresentation. java 和 JDIStackFrame. java 这些文件，按照对照组 2 的方法，准确率是 0. 670，召回率是 0. 668，F1-Measure 是 0. 669；若采用本文方法，准确率是 0. 709，召回率是 0. 702，F1-Measure 是 0. 705。

综合以上分析可以得到，本文提出的面向单个文件的个性化缺陷预测方法确实优于对照组 1 和对照组 2 使用的缺陷预测方法。

7 讨论

缺陷预测领域的普遍问题是可用的历史缺陷数据太少。不少学者开始关注跨项目的缺陷预测，研究多个项目间的缺陷预测方法，利用其他项目的数据进行迁移学习，运用大数据等前沿技术进行缺陷分析与预测。如 Zimmermann T 等人[20]对跨项目的缺陷预测进行了经验研究；Nam J 等人[21]则提出了跨项目的缺陷预测方法。然而现有的跨项目缺陷预测方法的效果并不如传统方法的效果好，其根本原因就是项目之间的差异性较大。即使是在项目内部，也可能存在各种差异。因此，一些学者针对这些差异性提出了改进后的缺陷预测方法，如个性化缺陷预测方法[1]、对代码元素按模块或类簇划分后进行缺陷预测[6-7]等。

本文方法对缺陷数据进一步细化，同时考虑开发人员和代码文件的差异性，针对某个开发人员修改某个代码文件的历史数据进行缺陷预测。适合本文方法的条件是单个文件的个人缺陷数据充足，即该代码文件经常被某个开发人员修改，这也是符合实际开发情况的。软件项目的利益相关者往往更加关注那些经常被修改的代码文件，这是因为经常被修改的代码文件更容易被引入缺陷。此外，项目中的代码文件在某个时间段一般也具有所有者(某一时间段经常修改这个代码文件的开发人员)。因此，本文方法具有现实意义。

本文实验初步说明了所提方法在缺陷数据充分的情况下能够提高缺陷预测的效果。然而在针对单个文件的个人缺陷历史数据不充足的情况下，本文方法的预测效果可能并不理想。此时可以将本文所提方法与面向项目的缺陷预测方法相结合，根据缺陷数据质量提供多层次的缺陷预测结果。

软件工程是一个与软件开发实践密切结合的研究领域，该领域注重在实践中发现问题并进行研究，进而提出一套可以解决该问题的理论，理论成熟之后可以应用到实践中以提高软件开发的效率，增强软件的可维护性，保证软件质量。

本文的工作动机与软件工程领域的研究目标相符合，旨在优化软件开发实践过程。在将本文方法应用到实践方面，一个可行的方法是在集成开发环境(如 Eclipse)里安装相应的工具或者插件，通过对开发人员的历史修改信息进行收集和分析，建立缺陷预测模型。在未来的开发过程中，对软件产品中潜在的缺陷进行预测，通过窗口提示帮助开发人员更深入地了解软件风险，及时调整软件测试、代码审查等策略，确保能够及时发现和修复软件缺陷，从而保证软件质量。

结束语 软件缺陷预测在软件维护中具有重要的作用。现有工作或同时忽略开发人员差异性和文件差异性，或忽略了文件差异性。然而，忽略开发人员差异性和文件差异性可能会对缺陷预测的结果产生影响。针对此问题，本文提出了一种面向单个文件的个性化缺陷预测方法，并通过实验证明了该方法能够提升缺陷预测的效果。

软件缺陷预测是一个需要继续深入探索的领域。在本文的基础之上，作者希望在以下两个方面作进一步的研究：

1)对开发人员修改代码文件这一行为进行更深入的挖掘，特别是从代码结构方面挖掘出一些特征以丰富原来的特征集，特征描述越准确，缺陷预测的精度越高；

2)词袋模型的假设是对于一个文本的，忽略了其词序和语法、句法，将文本仅仅看作是一个词的集合，这就否认了词之间的相关性。这个假设虽然在处理过程中带来了极大的方便，但忽视了实际场景中属性之间的关联性，进一步分析这些关联并将其应用于缺陷建模，可以进一步优化预测结果。

参 考 文 献

[1] TIAN J, LIN T, KIM S. Personalized defect prediction[C]// 2013 IEEE/ACM International Conference on Automated Software Engineering. IEEE Computer Society, 2013: 279-289

[2] WANG Q, WU S J, LI M S. Software Defect Prediction [J]. Journal of Software, 2008, 19(7): 1565-1580. (in Chinese)
王青，伍书剑，李明树. 软件缺陷预测技术[J]. 软件学报，2008，19(7)：1565-1580.

[3] YUAN Z, YU L L, LIU C. Bug prediction method for fine-grained source code changes[J]. Journal of Software, 2014, 25(11): 2499-2517. (in Chinese)
原子，于莉莉，刘超. 面向细粒度源代码变更的缺陷预测方法[J]. 软件学报，2014，25(11)：2499-2517.

[4] PREMRAJ R, HERZIG K. Network Versus Code Metrics to Predict Defects: A Replication Study[C]// Proceedings of the 2011 International Symposium on Empirical Software Engineering and Measurement. IEEE Computer Society, 2011: 215-224

[5] CHULANI S. Constructive Quality Modeling for Defect Density Prediction: COQUALMO[R]. Center for Software Engineering, IBM Research, 1999.

[6] LEI T. Software Defect prevention based on defect classification and defect prediction[J]. Computer Engineering and Design, 2013, 34(1): 215-220. (in Chinese)
雷挺. 基于缺陷分类和缺陷预测的软件缺陷预防[J]. 计算机工程与设计，2013，34(1)：215-220.

[7] PAN S, TAN X, PENG X, et al. Improving Software Defect Prediction by Combining the Information of Class and Package[J].

Journal of Frontiers of Computer Science and Technology, 2012, 6(2): 109-117. (in Chinese)
潘森，谭曦，彭鑫，等. 综合包级和类级度量的软件缺陷预测方法[J]. 计算机科学与探索，2012，6(2)：109-117.

[8] TU Y M, MAO J P, YU J, et al. Analysis of Software Defect Prediction Model of System Testing Process[J]. Journal of Computer Research and Development, 2010, 47(Suppl.): 108-112. (in Chinese)
涂亚明，毛军鹏，余静，等. 系统测试阶段的软件缺陷预测模型分析[J]. 计算机研究与发展，2010，47(Suppl.)：108-112.

[9] WANG P, JIN C, GE H H. Mutual information-based feature selection approach for software defect prediction[J]. Journal of Computer Applications, 2012, 32(6): 1738-1740. (in Chinese)
王培，金聪，葛贺贺. 面向软件缺陷预测的互信息属性选择方法[J]. 计算机应用，2012，32(6)：1738-1740.

[10] SHIVAJI S, WHITEHEAD E J, AKELLA R, et al. Reducing Features to Improve Bug Prediction[C]// 2009 IEEE/ACM International Conference on Automated Software Engineering. IEEE Computer Society, 2009: 600-604

[11] XU G C, LIU X Z, HU L, et al. Software reliability assessment models incorporating software defect correlation[J]. Journal of Software, 2011, 22(3): 439-450. (in Chinese)
徐高潮，刘新忠，胡亮，等. 引入关联缺陷的软件可靠性评估模型[J]. 软件学报，2011，22(3)：439-450

[12] JIANG H Y, ZONG M, LIU X Y. Research of Software Defect Prediction Model Based on ACO-SVM[J]. Chinese Journal of Computers, 2011, 34(6): 1148-1154. (in Chinese)
姜慧研，宗茂，刘相莹. 基于 ACO-SVM 的软件缺陷预测模型的研究[J]. 计算机学报，2011，34(6)：1148-1154

[13] KIM S, WHITEHEAD E J, ZHANG Y. Classifying Software Changes: Clean or Buggy? [J]. IEEE Transactions Software Engineering, 2008, 34(2): 181-196.

[14] SLIWERSKI J, ZIMMERMANN T, ZELLER A. When do changes induce fixes?[J]. ACM Sigsoft Software Engineering Notes, 2005, 30(1): 1-5.

[15] SEBASTIANI F. Machine learning in automated text categorization[J]. ACM Computing Surveys, 2002, 34(2): 1-47

[16] OSTRAND T J, WEYUKER E J, BELL R M. Where the bugs are[J]. AcmSigsoft Software Engineering Notes, 2004, 29(4): 86-96.

[17] SCOTT S, MATWIN S. Feature Engineering for Text Classification[C]// International Conference on ICML. 1999: 379—388.

[18] MOCKUS A, VOTTA L G. Identifying Reasons for Software Changes Using Historic Databases[C]// Proceedings of IEEE International Conference on Software Maintenance. 2000: 120-130

[19] CUBRANIC D, MURPHY G C. Hipikat: recommending pertinent software development artifacts[C]// Proceedings of 25th International Conference on Software Engineering. 2003: 408-418

[20] ZIMMERMANN T, NAGAPPAN N, GALL H, et al. Cross-project Defect Prediction A Large Scale Experiment on Data vs. Domain vs. Process[C]// European Software Engineering Conference. 2009: 91-100.

[21] NAM J, PAN S J, KIM S. Transfer defect learning[C]// International Conference on Software Engineering. 2013: 382-339.

(上接第 84 页)

UbuntuKylin 操作系统质量度量最终结果：基本都浮动在 80 左右，属于优良级别。该度量结果与广大用户对 UbuntuKylin 操作系统的评价相符，一定程度上反映了用户的使用体验度，也说明了模型的合理性及度量算法的有效性。

结束语　通过分析混源软件的质量特性，提出了混源软件质量模型。然后，以混源软件质量模型为基础，设计了一套实施度量的算法体系，并实现了 MiSP 度量工具。最后，对 UbuntuKylin 操作系统进行了探索性的度量实验。下一步工作将深入优化度量模型，规范度量指标，丰富度量平台。

参 考 文 献

[1] ISO/IEC. 2011 Systems and Software Quality Requirements and Evaluation: ISO/IEC 25010[S]. 2011.

[2] PETRINJA E, SILLITTI A, SUCCI G. Comparing OpenBRR, QSOS, and OMM Assessment Models[C]// Open Source Software: New Horizons. 2010: 224-238.

[3] HAALAND K, GROVEN A K, GLOTT R, et al. Free/Libre Open Source Quality Models — a comparison between two approaches[OL]// http://publications.nr.no/FLOS34workshop.final.pdf.

[4] PETRINJA E, NAMBAKAM R, SILLITTI A. Introducing the OpenSource Maturity Model[C]// Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development. IEEE Computer Society, 2009: 37-41.

[5] CHEN Y, HU C J, WU T. Open Source Software Maturity Assessment-Part One[J]. Information Technology & Standardizaton, 2011(9): 62-67. (in Chinese)
陈越，胡昌军，吴桐. 开放源代码软件成熟度评估(上)[J]. 信息技术与标准，2011(9)：62-67.

[6] NI G N. Independence and Controllability is the premise to enhance security internet[J]. Business Culture, 2014(30): 42-45. (in Chinese)
倪光南. 自主可控是增强网络安全的前提[J]. 商业文化，2014，(30)：42-45.

[7] 《运筹学》教材编写组. 运筹学(第 4 版)[M]. 北京：清华大学出版社，2012：522-527.

[8] WANG J, CHEN Y X, GU B, et al. An approach to measuring and grading software trust for spacecraft software[J]. Science China Press, 2015, 45(2): 221-228. (in Chinese).
王婧，陈仪香，顾斌，等. 航天嵌入式软件可信性度量方法及应用研究[J]. 中国科学：技术科学，2015，45(2)：221-228.