# 数据驱动的软件缺陷预测研究综述

李 勇[1,2],黄志球[1],王 勇[1],房丙午[1]
(1. 南京航空航天大学计算机科学与技术学院,江苏南京 211106;
2. 新疆师范大学网络信息安全与舆情分析重点实验室,新疆乌鲁木齐 830054)

**摘 要:** 数据驱动的软件缺陷预测是提高软件测试效率、保证软件可靠性的重要途径之一,近几年已成为实证软件工程的研究热点.首先介绍了数据驱动软件缺陷预测的研究背景;然后总结了已有软件缺陷数据属性度量方法的特点,并按照软件开发中缺陷预测的使用场景,以数据来源为主线从基于版本内数据、跨版本数据和跨项目数据实现缺陷预测三个方面对近10年(2005~2015)已有的研究工作进行分类归纳和比较;最后对该领域未来的研究趋势进行了展望.

**关键词:** 软件缺陷预测;数据驱动;软件度量;机器学习
**中图分类号:** TP311.5 **文献标识码:** A **文章编号:** 0372-2112 (2017)04-0982-07
**电子学报URL:** http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2017.04.030

# Survey on Data Driven Software Defects Prediction

LI Yong[1,2],HUANG Zhi-qiu[1],WANG Yong[1],FANG Bing-wu[1]
(1. *College of Computer Science and Technology,Nanjing University of Aeronautics and Astronautics,Nanjing,Jiangsu* 211106,*China*;
2. *Key Laboratory of Network Information Security and Public Opinion Analysis,Xinjiang Normal University,Urumqi,Xinjiang* 830054,*China*)

**Abstract:** Data driven software defects prediction is considered as an effective means for the optimization of quality assurance activities. This paper tries to survey the studies of data driven software defects prediction from 2005 to 2015. First, it briefly introduces the research background in this field. Then, it summarizes and analyzes the software modules attribute metrics in detail, and thoroughly surveys the state-of-the-art methods of data driven software defects prediction. The discussed topics include within-release, cross-release and cross-project defects prediction. Finally, the potential research directions in the field are discussed.

**Key words:** software defects prediction;data driven;software metrics;machine learning

## 1 引言

随着计算机软件应用领域的不断扩大,软件的开发和运行环境更加多样化,其规模和重要性也在急剧上升.由于系统对软件的强烈依赖,软件在运行中一旦失效有可能导致严重的后果,有时甚至是致命的.由软件失效的机理可知,导致软件不可靠的根本原因是软件中存在缺陷[1].通过软件缺陷预测技术对软件模块的缺陷分布进行预测,可以及时发现和修复缺陷,充分利用有限资源提高软件产品的质量,对软件的可靠性保证具有重要意义[2].

软件缺陷预测是实证软件工程的一个重要方向,已成为近年来的热门研究领域.纵观该方向的发展历史,通过对软件开发过程中积累的历史数据进行属性度量,采用统计或机器学习技术构建模型,实现数据驱动的软件缺陷预测是学术界和工业界关注的热点[3].研究人员已经对该方向涉及的相关技术和方法开展了大量的研究工作,取得了丰硕的成果,有力地促进了软件工程技术的发展.本文全面调查了近10年内(2005~2015)在期刊和会议中发表的相关文献,以数据来源为主线对已有的研究工作进行分类回顾、总结比较和展望.

## 2 数据驱动的软件缺陷预测相关背景

在数据驱动的软件缺陷预测研究中,通常将实现

收稿日期:2015-10-19;修回日期:2016-08-01;责任编辑:李勇锋
基金项目:国家自然科学基金(No.61562087,No.61272083);江苏省普通高校研究生科研创新计划(No.CXLX13_160);中央高校基本科研业务费专项项目

属性度量和缺陷预测的程序单元称为软件模块,其任务是从软件开发中积累的历史数据中学习预测模型,实现目标软件模块的缺陷倾向、缺陷严重度或缺陷数量分布等预测. 按照在软件工程中不同的使用场景,图1给出数据驱动软件缺陷预测的流程框架,包括四个步骤:①从软件工程数据仓库中选择适合于目标项目的历史数据;②对所选择历史数据进行属性度量和缺陷标注;③选择相应的机器学习或统计分析方法实现模型构建;④实现目标软件模块的缺陷预测,并对模型性能进行评价以实现模型的选择和改进.

由上述流程可知,度量属性和模型算法的选择决定着数据驱动缺陷预测模型的性能[4]. 但在不同的软件缺陷预测实现场景中影响模型性能的关键因素有所差异. 例如在当前版本数据驱动的预测中,选择合适的软件产品属性"刻画"软件模块特征后,如何选择算法实现高性能的模型是问题的关键;在跨版本数据驱动的预测中,软件过程属性更为直接地反映了缺陷的产生过程;而在跨项目数据驱动的预测中,模型的迁移方法则是影响最终性能的关键因素. 本文在总结常用软件缺陷数据属性度量方法的基础上,从基于版本内数据、跨版本数据和跨项目数据三个场景中实现模型的关键技术出发,对已有的研究进行综述.

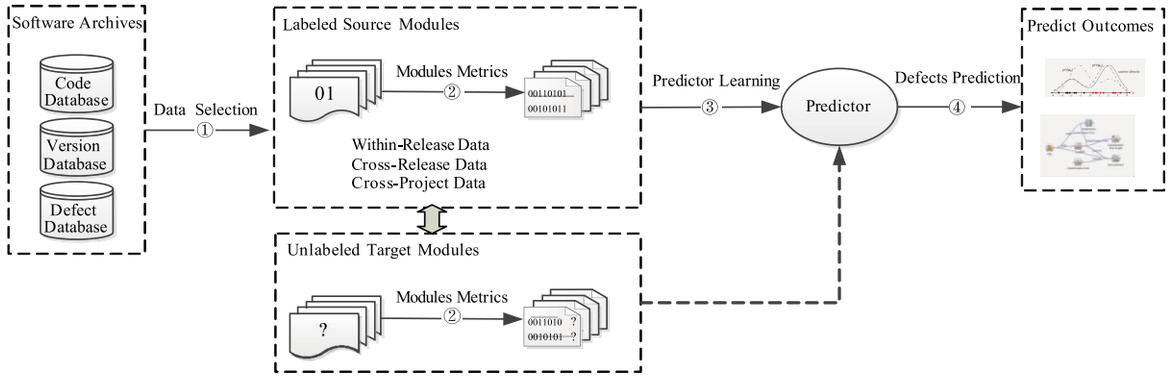


图1 数据驱动的软件缺陷预测流程图

## 3 软件缺陷数据属性度量方法

在软件开发过程中积累的历史数据包括软件代码、版本控制和缺陷追踪日志等数据. 实现数据驱动缺陷预测的前提是认为软件缺陷与历史数据属性之间存在某种关系,发现这些关系也成为软件度量发展的一个目标. 现有的软件缺陷数据度量方法可分为软件产品属性度量和软件过程属性度量两大类,如表1所示.

**表1 软件缺陷数据属性度量方法**

| 类型 | 子类型 | 属性度量 | 概述 | 代表性文献 |
|---|---|---|---|---|
| Product Metrics | Traditional Metrics | LOC | 代码行度量 | [7],[8] |
| | | Halstead | 操作数和操作符度量 | [5],[7],[8] |
| | | McCabe | 代码复杂度度量 | [6],[7],[8] |
| | OO Metrics | CK;MOOD;QMOOD;Martin | 面向对象度量 | [8],[9],[10],[11] |
| Process Metrics | Change Metrics | Code Churn | 代码变换度量 | [12],[13],[14] |
| | | Revisions Count | 代码变更次数度量 | [15] |
| | | Pre-version Defects Count | 前序版本缺陷数度量 | [16],[17] |
| | Developer Metrics | Code Developers | 代码开发者度量 | [18],[19] |

### 3.1 软件产品属性度量

(1)传统代码属性度量

在软件缺陷预测中,传统代码属性度量一般指软件代码的规模和复杂性度量. 常用的规模属性度量有LOC和Halstead[5],复杂性属性度量有McCabe[6]. 文献[7]采用信息增益方法获取传统代码属性的缺陷预测最优属性子集,发现常用的传统代码属性用于缺陷预测时都是有效的,但在不同项目中最优属性有所差异. 文献[8]指出传统代码属性在软件缺陷预测中具有局限性,一般适用于版本发布前或规模较小的软件系统.

(2)面向对象属性度量

随着面向对象开发技术的流行,常见的面向对象软件度量也被用于软件缺陷预测,如 CK, MOOD, QMOOD 和 Martin 等度量方法[8]. 各种度量方法均基于面向对象的封装、继承和多态等特性提供了多个属性,其区别在于所关注的度量粒度和侧重属性不同. 文献[9]选择规模较小的软件数据进行实验,发现 CK 度量的性能要优于传统的代码度量. 文献[10]和文献[11]分别对 CK, MOOD, QMOOD 和 Martin 度量的缺陷预测性能进行比较,实验表明在类粒度层次 CK 度量的性能最好,而在包粒度层次 Martin 度量的预测性能要优于其他度量方法.

### 3.2 软件过程属性度量

(1)代码变更度量

代码变换(Code Churn)是常见的代码变更度量方法,用于捕获变更过程中代码结构变化的相关属性,包括代码行变换、复杂度变换和类结构变换三种类型. 文献[12]验证了代码行变换度量在缺陷预测中的有效性,并表明代码行的增加和修改属性要优于删除属性. 文献[13]对多种度量方法进行比较,发现代码复杂度变换与软件缺陷的相关度最高;文献[14]提出"类基本设计变化"度量,实验表明在缺陷预测中要优于传统产品度量方法.

也有研究提出使用代码变更次数度量和前序版本缺陷数度量用于软件缺陷预测. 文献[15]假设每次代码变更都有可能引入新的缺陷,提出 Change bursts 度量,实验表明该方法在软件开发的早期阶段实现缺陷预测时要优于传统复杂性度量和代码变换度量. 文献[16]认为前序版本中缺陷数量较多的模块在变更后仍然存在缺陷的可能性较大. 但文献[17]指出软件变更的目标之一就是进行缺陷的修复,前序版本缺陷数度量并不适用于软件模块变更后的缺陷预测.

(2)代码开发者度量

代码开发者度量基于开发者之间的交互数据以及开发者的编码日志数据计算获取. 文献[18]提出新增开发者的数量与软件缺陷之间存在正相关性,但文献[19]用大量实验表明在缺陷预测中增加开发者属性后对预测性能的提高有限,并且特定开发者属性与缺陷的相关度较低,在考虑度量成本后不建议选择开发者度量用于缺陷预测.

通过上述分析可知,在软件产品属性度量中,面向对象属性度量的预测性能要优于传统代码属性度量,传统代码属性度量一般适用于方法级模块或规模较小软件的缺陷预测. 而在软件过程度量中,代码变换度量的性能较为稳定,其他过程度量则受开发和使用环境影响较大.

## 4 数据驱动的软件缺陷预测实现方法

### 4.1 版本内数据驱动的缺陷预测方法

由于当前开发版本已有的历史数据可以较好的反映版本内未完成软件代码的缺陷特征,当前开发版本有一定的历史数据积累后,已有研究中通常采用软件产品度量属性,选择合适的机器学习算法构建预测模型.

(1)基于监督方式的缺陷预测

当前开发版本的历史数据已完成软件模块的缺陷标注时,可以采用监督方式的算法构建预测模型. 机器学习中常见的监督算法都已经被用来实现模型,如决策树,朴素贝叶斯,支持向量机和神经网络等算法[20]. 集成学习技术作为一种提高模型泛化性能的有效途径,也被广泛应用在模型构建中. 特别是对于软件缺陷预测中的类不平衡性问题,集成学习提供了较好的解决方案,如文献[21]提出的 AdaBoost. NC 算法在类不平衡严重的应用中取得了理想的预测性能.

已有研究中对于监督方式构建的模型通常采用交叉验证的方式实现预测结果的评价,在特定的应用中可以获得不错的预测结果. 但由于软件项目类型、算法参数设置等不同,不存在性能最好的通用模型构建算法. 本文认为未来的研究应该从缺陷预测模型构建中存在的特性问题(如类不平衡性、代价敏感和模型实现成本等)出发结合不同应用的特点探索相应的算法方案.

(2)基于无监督和半监督方式的缺陷预测

考虑到对当前版本历史数据进行标注时代价较高等问题,有研究提出了无监督和半监督方式的模型方法. 无监督方式的基本思想是将软件模块进行聚类后,以聚类簇为单位进行缺陷标注从而实现预测. 文献[22]采用 X-means 算法实现软件模块的聚类簇,然后由领域专家进行聚类簇的缺陷标注. 也有研究提出通过属性阈值实现聚类簇的自动标注[23],但由于数据噪声等问题容易使得阈值偏差较大而导致预测性能的降低[24]. 当只有部分历史数据完成缺陷标注后,可以采用半监督的模型构建方式,如文献[25]提出的 CoForest 算法和 ACoForest 算法,在实验中获得了理想的预测结果.

基于无监督和半监督方式的模型方法对数据噪声比较敏感,但该类方法的研究可以用于实现软件缺陷数据的自动或半自动标注,对于降低软件模块的标注代价具有一定意义.

### 4.2 跨版本数据驱动的缺陷预测方法

在软件产品生命周期内,软件版本是不断演化和发展的. 采用前序版本的历史数据实现后序版本中软

件模块的缺陷预测,不仅可以有效降低软件版本迭代的开发成本、提高新版本软件的可靠性,而且可以用于理解缺陷产生的原因,改进软件开发过程. 在跨版本数据驱动的缺陷预测研究中,已有文献主要关注于跨版本历史数据属性度量的选择,采用统计分析方法实现预测.

(1)基于软件产品度量的跨版本预测

基于软件产品度量实现跨版本的缺陷预测集中在较早期的文献中,这类研究认为前序版本代码的产品属性可以用于刻画后序版本中的软件缺陷,通常采用统计分析的方法对产品属性和后序版本缺陷之间的相关性进行验证或实现预测.

文献[11]基于多种面向对象度量采用逻辑回归算法实现跨版本的缺陷预测,发现在迭代次数较多或敏捷开发中常用的面向对象度量无法获得理想的预测结果. 文献[26]指出软件产品度量在跨版本预测中具有滞后性,容易导致后续版本中具有相似属性但在版本迭代中已经完成缺陷修复的软件模块被再次预测为存在缺陷,特别是规模较大而缺陷密度较低的软件模块.

(2)基于软件过程度量的跨版本预测

由于跨版本间的代码变更是导致软件缺陷产生的主要原因,而软件过程度量属性是在版本迭代或者代码变更中获取,较直接的反映了缺陷的产生过程. 因此大部分研究认为软件过程度量比软件产品度量更适合于跨版本的缺陷预测.

文献[27]基于软件代码行变更属性实现连续版本的缺陷密度预测,获得了理想的预测性能. 文献[14]基于类结构变更属性分别实现跨版本软件模块的缺陷倾向、缺陷数量和缺陷密度预测,实验表明类结构变更度属性可以显著提高模型的预测性能. 文献[28]选择代码开发者度量用于跨版本的缺陷预测,发现对某一软件模块进行修改的开发人员数量与缺陷的产生并无直接关系.

也有研究将跨版本数据驱动的方法用于分析缺陷的产生原因,并提取相应的规则用于指导版本迭代中软件开发过程的改进. 如文献[29]采用统计分析方法对 Beta 版本中有缺陷软件模块进行过程属性度量,发现版本发布后有缺陷和无缺陷软件模块的过程属性存在较大差异,例如软件模块的代码变更次数较少或者变更提交者数量较小时反而存在缺陷的可能性较大,及时的发现这些特殊规则并用于后序版本的开发,可以避免类似缺陷产生,有效提高软件产品的质量.

### 4.3　跨项目数据驱动的缺陷预测方法

项目内数据驱动的预测方法要求目标项目必须有历史积累数据,但在实践中需要进行缺陷预测的往往是新开发的软件项目,新项目一般没有或者只有少量的积累数据无法用于模型的训练. 为此有研究者提出跨项目数据驱动的缺陷预测方法,即采用相关源项目中已标注的数据构建模型实现目标项目的预测. 但由于不同项目的软件缺陷数据存在漂移问题[30],直接采用传统机器学习方法难以获得理想的效果,已有研究关注于选择迁移学习技术实现跨项目的缺陷预测.

(1)基于特征迁移的跨项目预测

基于特征迁移的方法是指对源项目和目标项目的软件属性特征进行处理,在保留各自缺陷特征的前提下使得跨项目软件缺陷数据分布最为接近,然后基于传统算法构建模型. 文献[31]提出属性补偿方法,用于实现不同程序语言源项目和目标项目之间的跨项目缺陷预测;文献[32]则基于迁移成分分析 TCA 方法实现模型迁移,在保留缺陷特征的约束条件下将源项目和目标项目的属性特征映射到潜在的特征空间实现预测模型.

也有研究提出基于特征选择的方法实现模型迁移. 如文献[33]在实验中通过迭代选择的方式获得使跨项目预测模型性能最优的属性子集,然后基于该属性子集实现预测模型的迁移.

(2)基于实例迁移的跨项目预测

软件模块可以看作是迁移学习中的实例,基于实例迁移的跨项目预测方法是指从源项目中获取适合于目标项目的软件模块实例,然后基于新的实例数据学习目标项目预测模型. 在研究中通常采用数据选择的方法获取目标模型训练数据. 如文献[34]提出基于 KNN 算法的软件模块实例数据选择方法;文献[35]提出基于项目特征的多源项目数据选择方法. 也有研究提出基于实例权值的方法获取模型数据. 如文献[36]提出的迁移贝叶斯 TNB 算法,基本思想是根据对目标模型的作用实现源项目模块的权值分配,然后基于加权源项目数据实现模型的迁移.

考虑到软件系统的复杂性,有研究提出局部模型的方法实现跨项目预测. 如文献[37]采用聚类算法形成源项目和目标项目的模块实例簇,以簇为单位选择数据并构建局部模型,实验表明其性能要优于全局模型. 但文献[38]发现多个局部模型性能平均后这种优势有可能会抵消,建议仍然采用全局模型方法用于软件缺陷预测.

本文认为在未来的研究中应该将数据特征和实例选择结合起来,在考虑项目特征的前提下实现软件模块实例的选择,可以充分利用源项目的缺陷信息,提高跨项目模型的性能. 另外,如果可以结合目标项目模块的缺陷预测结果及修复过程进一步完善模型,也可以有效提高已有模型的泛化能力.

## 5 未来研究工作展望

综合该领域研究面临的挑战、研究热点和应用前景,结合目前的研究工作,认为未来的研究可以从以下几个方面展开:

(1)相关技术的进一步探索. 关于属性度量和模型算法等关键技术已经取得了大量的成果,但是在实践中如何根据预测任务选择合适的属性度量方法,以及考虑模型成本的前提下如何选择模型算法仍然没有确定性结论. 因此对于需求适应的属性度量和模型算法选择问题需要进一步深入研究.

(2)应用研究的深入. 现有研究的目标是实现未知软件的缺陷预测. 如何从预测结果中获取可理解性的规则,进行组织或可视化后用于指导软件开发实践,对于改进软件开发过程具有重要意义. 因此关于数据驱动的软件缺陷预测在应用领域的深入有待进一步探索.

(3)软件缺陷数据共享资源库构建. 目前互联网上已有大量开源软件缺陷数据公开用于研究,而商业软件则由于隐私问题只有少量数据实现了共享[39],造成很多研究成果难以重现和继续深入. 如何在保证数据商业隐私的前提实现其有效利用,以及如何合理组织和管理数据并构建统一的软件缺陷数据共享资源库是未来值得探索的方向.

(4)工具软件的开发与应用. 该领域大部分研究仍处于理论实验阶段,如何将已有成果应用于实践是学术界和工业界都比较关注的问题. 目前谷歌[40]、微软[41]等公司已经进行了尝试,基于部分成果开发了相应的原型系统,但在实际中相关的工具软件还很缺乏. 因此对于数据驱动缺陷预测相关工具软件的研究和开发很有必要.

## 6 结语

数据驱动的软件缺陷预测作为保证软件可靠性和提高软件测试效率的重要途径,近年内其研究受到了学术界和工业界的广泛关注,并积累了大量的研究成果. 本文从属性度量和模型算法两个视角,分类介绍了软件工程中不同场景下缺陷预测的实现方法,并列举了部分具有代表性的研究文献,为该问题的进一步的研究和应用提供了参考. 从研究现状来看,该领域在相关技术、应用和支撑工具等方面仍需要深入研究.

### 参考文献


[1] Lyu M R. Software reliability engineering:a roadmap[A]. 2007 Future of Software Engineering[C]. Washington:IEEE Computer Society,2007. 153 - 170.

[2] Caglayan B,Tosun Misirli A,Bener A B,et al. Predicting defective modules in different test phases[J]. Software Quality Journal,2015,23(2):205 - 227.

[3] Shepperd M,Bowes D,Hall T. Researcher bias:the use of machine learning in software defect prediction[J]. IEEE Transactions on Software Engineering,2014,40(6):603 - 616.

[4] Hall T,Beecham S,Bowes D,et al. A systematic literature review on fault prediction performance in software engineering[J]. IEEE Transactions on Software Engineering,2012,38(6):1276 - 1304.

[5] Halstead M H. Elements of Software Science (Operating and Programming Systems Series)[M]. New York:Elsevier Science Inc,1977. 1 - 128.

[6] Mccabe T J. A complexity measure[J]. IEEE Transactions on Software Engineering,1976,SE - 2(SE - 2):308 - 320.

[7] Menzies T,Greenwald J,Frank A. Data mining static code attributes to learn defect predictors[J]. IEEE Transactions on Software Engineering,2007,33(1):2 - 13.

[8] Radjenovi? D,Heri? ko M,Torkar R,et al. Software fault prediction metrics:a systematic literature review[J]. Information and Software Technology,2013,55(8):1397 - 1418.

[9] Basili V R,Briand L C,Melo W L. A validation of object-oriented design metrics as quality indicators[J]. IEEE Transactions on Software Engineering,1996,22(10):751 - 761.

[10] Elish M O,Al-Yafei A H,Al-Mulhem M. Empirical comparison of three metrics suites for fault prediction in packages of object-oriented systems:a case study of eclipse[J]. Advances in Engineering Software,2011,42(10):852 - 859.

[11] Olague H M,Etzkorn L H,Gholston S,et al. Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes[J]. IEEE Transactions on Software Engineering,2007,33(6):402 - 419.

[12] Bell R M,Ostrand T J,Weyuker E J. Does measuring code change improve fault prediction?[A]. Proceedings of the 7th International Conference on Predictive Models in Software Engineering[C]. New York:ACM,2011. 1 - 8.

[13] Munson J C,Elbaum S G. Code churn:a measure for estimating the impact of code change[A]. Proceedings of the International Conference on Software Maintenance[C]. Washington:IEEE Computer Society,1998. 24 - 31.

[14] Kpodjedo S,Ricca F,Galinier P,et al. Design evolution

metrics for defect prediction in object oriented systems [J]. Empirical Software Engineering, 2011, 16(1): 141 – 175.

[15] Nagappan N, Zeller A, Zimmermann T, et al. Change bursts as defect predictors[A]. Proceedings of the 21st IEEE International Symposium on Software Reliability Engineering[C]. Washington: IEEE Computer Society, 2010. 309 – 318.

[16] Khoshgoftaar T M, Allen E B, Halstead R, et al. Using process history to predict software quality[J]. Computer, 1998, 31(4): 66 – 72.

[17] Illes-Seifert T, Paech B. Exploring the relationship of a file's history and its fault-proneness: an empirical method and its application to open source programs[J]. Information and Software Technology, 2010, 52(5): 539 – 558.

[18] Okutan A, Yildiz O T. Software defect prediction using Bayesian networks[J]. Empirical Software Engineering, 2014, 19(1): 154 – 181.

[19] Ostrand T J, Weyuker E J, Bell R M. Programmer-based fault prediction[A]. Proceedings of the 6th International Conference on Predictive Models in Software Engineering [C]. New York: ACM, 2010. 1 – 10.

[20] Malhotra R. A systematic review of machine learning techniques for software fault prediction[J]. Applied Soft Computing, 2015, 27(0): 504 – 518.

[21] Wang S, Yao X. Using class imbalance learning for software defect prediction[J]. IEEE Transactions on Reliability, 2013, 62(2): 434 – 443.

[22] Catal C, Sevim U, Diri B. Metrics-driven Software Quality Prediction Without Prior Fault Data[M]. Berlin: Springer, 2010. 189 – 199.

[23] Catal C, Sevim U, Diri B. Clustering and metrics thresholds based software fault prediction of unlabeled program modules[A]. Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations [C]. Washington: IEEE Computer Society, 2009. 199 – 204.

[24] Alan O, Catal C. Thresholds based outlier detection approach for mining class outliers: an empirical case study on software measurement datasets [J]. Expert Systems with Applications, 2011, 38(4): 3440 – 3445.

[25] Li M, Zhang H, Wu R, et al. Sample-based software defect prediction with active and semi-supervised learning[J]. Automated Software Engineering, 2012, 19(2): 201 – 230.

[26] Rahman F, Devanbu P. How, and why, process metrics are better[A]. Proceedings of the 2013 International Conference on Software Engineering[C]. USA: IEEE Press, 2013. 432 – 441.

[27] Kastro Y, Ay, Bener E B. A defect prediction method for software versioning[J]. Software Quality Control, 2008, 16(4): 543 – 562.

[28] Weyuker E, Ostrand T, Bell R. Do too many cooks spoil the broth? Using the number of developers to enhance defect prediction models[J]. Empirical Software Engineering, 2008, 13(5): 539 – 559.

[29] Misirli A T, Murphy B, Zimmermann T, et al. An explanatory analysis on eclipse beta-release bugs through in-process metrics[A]. Proceedings of the 8th International Workshop on Software Quality[C]. New York: ACM, 2011. 26 – 33.

[30] Turhan B. On the dataset shift problem in software engineering prediction models[J]. Empirical Software Engineering, 2012, 17(1): 62 – 74.

[31] Watanabe S, Kaiya H, Kaijiri K. Adapting a fault prediction model to allow inter language reuse[A]. Proceedings of the 4th International Workshop on Predictor Models in Software Engineering[C]. New York: ACM, 2008. 19 – 24.

[32] Nam J, Pan S J, Kim S. Transfer defect learning[A]. Proceedings of the 2013 International Conference on Software Engineering[C]. USA: IEEE Press, 2013. 382 – 391.

[33] He P, Li B, Liu X, et al. An empirical study on software defect prediction with a simplified metric set[J]. Information and Software Technology, 2015, 59: 170 – 190.

[34] Turhan B, Menzies T, Bener A B, et al. On the relative value of cross-company and within-company data for defect prediction [J]. Empirical Software Engineering, 2009, 14(5): 540 – 578.

[35] Herbold S. Training data selection for cross-project defect prediction[A]. Proceedings of the 9th International Conference on Predictive Models in Software Engineering. Baltimore[C]. New York: ACM, 2013. 1 – 10.

[36] Ma Y, Luo G, Zeng X, et al. Transfer learning for cross-company software defect prediction[J]. Information and Software Technology, 2012, 54(3): 248 – 256.

[37] Menzies T, Butcher A, Cok D, et al. Local versus global lessons for defect prediction and effort estimation[J]. IEEE Transactions on Software Engineering, 2013, 39(6): 822 – 834.

[38] Bettenburg N, Nagappan M, Hassan A E. Think locally, act globally: Improving defect and effort prediction models [A]. Proceedings of the 9th IEEE Working Conference on Mining Software Repositories[C]. USA: IEEE Press, 2012. 60 – 69.

[39] Peters F, Menzies T, Gong L, et al. Balancing privacy and utility in cross-company defect prediction [J]. IEEE Transactions on Software Engineering, 2013, 39(8): 1054

－1068.

[40] Lewis C, Lin Z, Sadowski C, et al. Does bug prediction support human developers? findings from a google case study[A]. Proceedings of the 2013 International Conference on Software Engineering[C]. USA: IEEE Press, 2013. 372－381.

[41] Czerwonka J, Das R, Nagappan N, et al. CRANE: failure prediction, change analysis and test prioritization in practice—experiences from Windows[A]. Proceedings of the 2011 Fourth IEEE International Conference on Software Testing, Verification and Validation[C]. Washington: IEEE Computer Society, 2011. 357－366.

## 作者简介

**李 勇** 男,1983 年 12 月生于山西晋中,现为南京航空航天大学计算机科学与技术学院博士研究生. 主要研究方向为实证软件工程.
E-mail: liyong@ live. com

**黄志球(通信作者)** 男,1965 年生于江苏南京,现为南京航空航天大学教授、博士生导师,CCF 杰出会员,主要研究方向为软件工程、软件分析与验证.
E-mail: zqhuang@ nuaa. edu. cn