软件学报ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.000000]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

静态软件缺陷预测方法*

陈　翔[1,2], 顾　庆[2], 刘望舒, 刘树龙, 倪超

[1](南通大学计算机科学与技术学院,江苏南通 226019)
[2](计算机软件新技术国家重点实验室(南京大学),江苏南京 210023)

通讯作者: 陈翔, E-mail: ****@ntu.edu.cn

摘　要: 静态软件缺陷预测是软件工程数据挖掘领域中的一个研究热点.通过分析软件代码或开发过程,设计出与软件缺陷相关的度量元;随后,通过挖掘软件历史仓库来创建缺陷预测数据集,旨在构建出缺陷预测模型,以预测出被测项目内的潜在缺陷程序模块,最终达到优化测试资源分配和提高软件产品质量的目的.对近些年来国内外学者在该研究领域取得的成果进行了系统总结.首先,给出了研究框架并识别出了影响缺陷预测性能的3个重要影响因素:度量元的设定、缺陷预测模型的构建方法和缺陷预测数据集的相关问题;接着,依次总结了这3个影响因素的已有研究成果;随后,总结了一类特殊的软件缺陷预测问题(即,基于代码修改的缺陷预测)的已有研究工作;最后,对未来研究可能面临的挑战进行了展望.

关键词: 软件质量保障;软件缺陷预测;软件度量元;机器学习;数据集预处理

软件缺陷(software defect)产生于开发人员的编码过程,需求理解不正确、软件开发过程不合理或开发人员的经验不足,均有可能产生软件缺陷.而含有缺陷的软件在运行时可能会产生意料之外的结果或行为,严重的时候会给企业造成巨大的经济损失,甚至会威胁到人们的生命安全.在软件项目的开发生命周期中,检测出内在缺陷的时间越晚,修复该缺陷的代价也越高.尤其在软件发布后,检测和修复缺陷的代价将大幅度增加.因此,项目主管借助软件测试或代码审查等软件质量保障手段,希望能够在软件部署前尽可能多地检测出内在缺陷.但是若关注所有的程序模块会消耗大量的人力物力,因此,项目主管希望能够预先识别出可能含有缺陷的程序模块,并对其分配足够的测试资源. 软件缺陷预测[1,2]是其中一种可行方法,根据软件历史开发数据以及已发现的缺陷,借助机器学习等方法来预测软件项目中的缺陷数目和类型等.目前,已有的软件缺陷方法可以简单分为静态缺陷预测方法和动态缺陷预测方法[1],其中:静态预测方法基于缺陷相关的度量数据,对程序模块的缺陷倾向性、缺陷密度或缺陷数进行预测;而动态缺陷预测方法则是基于缺陷或失效产生的时间对系统缺陷随时间的分布进行预测,以发现软件缺陷随其生命周期或其中某些阶段的时间关系的分布规律. 本文重点对静态软件缺陷预测的已有研究工作进行综述.具体来说,该类方法通过分析软件代码或开发过程设计出与软件缺陷相关的度量元,随后,通过挖掘软件历史仓库(software historical repositories)来创建缺陷预测数据集.目前,可以挖掘与分析的软件历史仓库包括项目所处的版本控制系统(例如CVS,SVN或Git等)、缺陷跟踪系统(例如Bugzilla,Mantis,Jira或Trac等)或相关开发人员的电子邮件等.最后,基于上述搜集的缺陷预测数据集,构建缺陷预测模型,并用于预测出项目内的潜在缺陷程序模块. 静态软件缺陷预测属于当前软件工程数据挖掘领域[3]中的一个重要研究问题.随着新的数据挖掘技术的不断涌现以及研究人员对软件历史仓库挖掘的日益深入,静态软件缺陷预测研究在近些年来取得了大量的研究成果,其缺陷预测结果也逐渐成为判断一个系统是否可以交付使用的重要依据.因此,针对该问题的深入研究对提高和保障软件产品的质量具有重要的研究意义. 为了对该研究问题进行系统的分析、总结和比较,我们首先在IEEE,ACM,Springer,Elsevier和CNKI等论文数据库中进行检索,检索时采用的主要英文关键词包括“defect prediction”,“software defect prediction”,“fault prediction”和“software fault prediction”等;然后,对检索出的论文,通过人工审查方式移除掉与研究问题无关的论文,并通过查阅相关论文的参考文献和相关研究人员发表的论文列表来进一步识别出遗漏的论文;最终,我们选择出与该研究问题直接相关的高质量论文共113篇(截止到2015年7月).从选择出的论文所发表的会议和期刊来看,绝大部分论文发表在软件工程领域的权威会议或期刊上,例如ICSE会议(21篇)、ESEC/FSE(或FSE)会

议(13篇)、ISSRE会议(5篇)、TSE期刊(16篇)、IST期刊(7篇)和JSS期刊(4篇)等. 本文第1节对静态软件缺陷预测方法的研究框架进行总结,并识别出其中3个重要的影响因素(即,度量元的设定、缺陷预测模型的构建方法和缺陷预测数据集的相关问题).第2节对已有的度量元设计进行总结.第3节对已有的缺陷预测模型构建方法进行总结.第4节对缺陷预测数据集相关问题的产生根源和解决方法进行总结.第5节对一类特殊的软件缺陷预测问题(即,基于代码修改的缺陷预测)的已有研究工作进行总结.传统的缺陷预测问题重点预测的是程序模块内部是否含有缺陷,而该问题的特殊之处在于需要预测出提交的代码修改是否会产生缺陷.最后总结全文,并对未来值得关注的研究方向进行初步探讨.

1 研究框架

静态软件缺陷预测可以将程序模块的缺陷倾向性、缺陷密度或缺陷数设置为预测目标.以预测模块的缺陷倾向性为例,其典型研究框架如图1所示. 图1上半部分总结的是软件缺陷预测过程,该过程主要包括两个阶段:模型构建阶段和模型应用阶段.具体来说,模型构建阶段包括3个步骤.

(1). 挖掘软件历史仓库,从中抽取出程序模块.程序模块的粒度根据实际应用的场景,可设置为文件、包、类或函数等.随后,将该程序模块标记为有缺陷模块或无缺陷模块,在图1中,我们将有缺陷模块用红色进行标记,无缺陷模块用绿色进行标记;

(2). 通过分析软件代码或开发过程设计出与软件缺陷存在相关性的度量元,借助这些度量元对程序模块进行软件度量,并构建出缺陷预测数据集;

(3). 对缺陷预测数据集进行必要的数据预处理(例如噪音移除、特征子集选择、数据归一化等)后,借助特定的建模方法构建出缺陷预测模型.大部分建模方法都基于机器学习方法,其常用的模型性能评测指标包括准确率(accuracy)、查准率(precision)、查全率(recall)、F-measure或AUC(area under the ROC curve)取值等.

而在模型应用阶段,当面对新程序模块时,在完成对该模块的软件度量后,基于前一阶段构造出的缺陷预测模型和具体度量元取值,可以完成对该模块的分类,即将该模块预测为有缺陷倾向性(defect-proneness,简称FP)模块或无缺陷倾向性(non defect-proneness,简称NFP)模块

若将预测目标设置为缺陷密度或缺陷数时,其预测流程与图1基本相同,主要的不同点是模型构建阶段中的模块标记(即,需要标记出已有模块内的缺陷密度或缺陷数)和模型预测阶段中的新模块的类型输出(即,预测输出的是新模块内的缺陷密度或缺陷数). 通过分析上述软件缺陷预测过程,我们识别出影响缺陷预测性能的3个重要影响因素(如图1的下半部分所示).

(1). 度量元的设计(见第2节).
挖掘软件历史仓库、设置新颖的与软件缺陷存在强相关性的度量元,是构建高质量缺陷预测模型的关键.本文将已有的度量元分为两类,其中:第一类重点关注的是程序模块的代码规模和内在复杂度;而第二类则重点分析软件开发过程,从分析代码修改特征、开发人员经验、模块间的依赖性以及项目团队组织构架等角度出发来设计度量元.

(2). 缺陷预测模型的构建方法(见第3节).
本文将已有的构建方法分为两类,其中:基于机器学习的方法是目前主流的建模方法,根据预测目标的不同,可以进一步细分为分类方法和回归分析方法;而基于缓存的方法则借助缺陷的局部性原理来尝试识别出缺陷模块.

(3). 缺陷预测数据集的相关问题(见第4节).
本文从两个角度对缺陷预测数据集相关问题进行分析:首先分析了数据集质量对软件缺陷预测的影响,重点对其中的噪音问题、维数灾难问题和类不平衡问题的产生原因及其相应解决方案进行了分析和总结;随后,针对需要预测的目标项目可能是一个全新项目,或这个项目已有的训练数据较少的问题,分析了利用其他项目的数据集来为目标项目构建缺陷预测模型的可行性,并将该问题称为跨项目缺陷预测问题.然后,从实例选择、实例权重设置、特征映射和度量元选择等角度对基于迁移学习的跨项目缺陷预

测方法进行了总结.

2 度量元的设计

挖掘软件历史仓库、设置新颖的与软件缺陷存在强相关性的度量元,是构建高质量缺陷预测模型的关键.因此,度量元的设计一直是软件缺陷预测研究中的一个核心问题[4].早期的研究工作主要集中于分析源代码,重点关注基于软件代码(software code)的软件度量.近些年来,更多的研究工作集中于挖掘不同的软件历史存档,重点关注基于软件开发过程(software process)的软件度量.本节将重点从这两个角度出发,对已有研究工作进行系统总结.

2.1 基于软件代码的度量

在研究早期,大部分研究工作通过分析软件代码来设计度量元.这类度量元重点关注程序模块的代码规模和内在复杂度等属性,其潜在的假设是:代码规模或复杂度越高的程序模块,其内部含有缺陷的可能性越高.

研究人员[5]最早借助代码行数(lines of code,简称LOC)进行度量,例如,Akiyama给出了缺陷数(D)与LOC(L)的关系式:D=4.86+0.018L.但该度量元过于简单,难以合理地去度量软件系统的复杂性.随后,研究人员逐渐考虑了Halstead科学度量[6]和McCabe环路复杂度(cyclomatic complexity)[7].其中:

- Halstead科学度量[6]通过统计程序内操作符和操作数的数量来度量代码的阅读难度,其假设是代码的阅读难度越高,其含有缺陷的可能性也越高,涉及到的主要度量元包括程序的长度、容量、难度和工作量等;
- 而McCabe环路复杂度[7]关注的是程序的控制流复杂度,其假设是程序的控制流复杂度越高,其含有缺陷的可能性也越高.在度量时,首先将程序建模为控制流图(control flow graph),其中,节点对应的是语句,边表示从一个语句到另一个语句的控制流.随后,通过公式v(G)=e n+2计算出控制流图G的环路复杂度,其中,e表示边的数量,n表示节点的数量.最后,可以进一步计算出程序的基本复杂度(essential complexity)和设计复杂度(design complexity).

随着面向对象开发方法的普及,其特有的封装、继承和多态等特性给传统的软件度量提出了挑战.研究人员提出了适用于面向对象程序的度量元,其中最为典型的是Chidamber和Kemerer提出的CK度量元[8].CK度量元综合考虑了面向对象程序中的继承、耦合性和内聚性等特征,给定一个类,其包含的度量元名称及相关描述见表1.

表 1: CK度量元

名称	描述
WMC	类的加权方法数
DIT	类在继承树中的深度
NOC	类在继承树中的孩子节点数
CBO	与该类存在耦合关系的其他类的数目
RFC	该类可以调用的外部方法数
LCOM	类内访问一个或多个属性的方法数

Basili等人[9]基于一些中等规模的信息管理系统,首次验证了CK度量元与程序模块内的缺陷存在相关性.随后,Subramanyam和Krishnan[10]基于8个工业界项目,进一步验证了Basili等人的发现.周毓明等人[11]也对基于面向对象程序的度量元与程序模块缺陷间的相关性进行了深入的分析,随后他们[12,13]发现:类规模度量元在分析时存在潜在的混和效应,并会对缺陷预测模型的性能产生影响.因此,他们提出了一种基于线性回归的方法来尝试移除这种混和效应.最后,他们[14,15]分别对Sarkar等人提出的package-modularization度量元[16]和基于程序切片的内聚性度量元[17]与程序模块缺陷间的相关性进行了深入分析.