数据挖掘技术在软件工程中的应用综述

毛澄映[1,2] 卢炎生[2] 胡小华[1,3]

(江西财经大学软件学院 南昌 330013)[1] (华中科技大学计算机科学与技术学院 武汉 430074)[2]

(Drexel大学信息科学与技术学院 费城 19104)[3]

摘 要 随着软件系统的规模和复杂性日益增长,软件开发已经演变成一项复杂的系统工程。软件工程中的对象、活动和过程更加难以控制和管理,因此该领域原有的经验直觉型的处理模式已经不能适应新的需求,而数据挖掘技术的引入为实现知识智能型软件工程提供了重要契机。以软件工程领域中的数据对象为主线,对在程序代码分析、故障检测、软件项目管理、开源软件开发等软件活动中所运用到的数据挖掘技术进行了系统的介绍和归纳,并在每一环节作了方法间的优劣性对比分析。最后还指出了若干值得进一步研究的方向。

关键词 数据挖掘,软件工程,预测,软件项目管理,开源软件

中图法分类号 TP311 **文献标识码** A

Data Mining Applications on the Software Engineering Tasks: A State of the Art

MAO Cheng-ying[1,2] LU Yan-sheng[2] HU Xiao-hua[1,3]

(School of Software, Jiangxi University of Finance and Economics, Nanchang 330013, China)[1]

(College of Computer Sci. and Tech., Huazhong University of Sci. and Tech., Wuhan 430074, China)[2]

(College of Information Sci. and Tech., Drexel University, Philadelphia, PA, 19104, USA)[3]

Abstract With rapid increase of the size and complexity of software system, software development has evolved into a complex and systematic engineering. The objects, activities and processes in software engineering will to be controlled even more with difficulty, so the traditional modes, such as treatment with experience and intuition, are unable to adapt properly to the new requirements. However, the introduction of data mining techniques can promote the development of knowledgeable and intelligent software engineering. In the perspective of the data to be mined in software engineering field, the paper systematically described and summarized the data mining techniques adopted in the activities such as program code analysis, fault detection, software project management, and open source software (OSS) development. The comparison analysis about these techniques is also addressed in each section. Furthermore, some on-going research issues in this direction are also discussed in the end.

Keywords Data mining, Software engineering, Prediction, Software project management, OSS

随着软件需求的日益增长和开发技术的快速发展,软件系统的规模和复杂性急剧增长,其开发活动更加难以控制。在软件工程领域中,那些传统的定性的方法和简单的统计技术难以解决数据及信息量爆炸式增长所带来的困扰。因此,从程序、文档及相关数据集中发现规律并用以指导软件工程活动就显得尤为必要。

数据挖掘技术在软件工程领域的应用可追溯到上世纪90年代初,当时主要是用以发现一些可复用的代码[1]。但随着软件系统复杂性的快速增长和数据挖掘技术的日益更新,极大地促进了该项技术在软件工程领域的广泛应用。从上世纪90年代末起,逐渐有软件工程或数据挖掘领域的学者从事这方面的研究,并通过实例证实数据挖掘技术是实现低成本、高效率完成软件工程任务的有效解决方案。直至目前,诸如ICSE, ISSTA, ASE, FSE, KDD等顶级国际会议专门开设了相关议题;从2004年起挖掘软件资源库(mining software repositories, MSR)研讨会的成功召开标志着该方向已成为软件工程的一个重要研究分支。

数据挖掘技术在软件工程中的应用主要是采用有效的分类、聚类、预测和统计分析技术从各种软件资源库中发现潜在的知识、规则等用以反馈指导软件工程活动,以达到改进软件产品质量、提高开发效率的目的。本文以软件工程领域中的数据对象为主线,系统地介绍和对比了在程序代码分析、故障检测、软件项目管理、开源软件开发等软件活动中所运用到的数据挖掘技术。

到稿日期:2008-06-03 本文受国家自然科学基金项目(60803046, 70571025),中国博士后科学基金(20070410946),湖北省自然科学基金(2005ABA266),江西省教育厅科技项目(赣教技字[2007]267号)资助。

毛澄映(1978—),男,博士后,副教授,CCF会员,主要研究方向为软件测试、数据挖掘等,E-mail: maochy@yeah.net;**卢炎生**(1949—),男,教授,博士生导师,主要研究方向为软件工程、数据挖掘等;**胡小华**(1965—),男,博士,副教授,主要研究方向为数据挖掘等。

1 面向程序代码及结构的挖掘

1.1 克隆代码检测

所谓克隆代码(clone code)就是以复用为目的进行拷贝和粘贴的代码段,有时也会作部分修改。一般在软件系统中会占到代码总量的7%至23%[2],对克隆代码进行检测将有利于防止故障(fault)的拷贝性传播,还有利于软件在演化过程中的维护。克隆代码检测是在软件工程领域最早的数据挖掘需求,现已形成了多种方法,可归纳如下:

(1)基于文本对比的方法。通过对比程序代码中的语句行进行判断,其后期改进主要表现在提高字符串的匹配效率,例如采用Hash函数技术,相应的工具有Duploc。

(2)基于标识符对比的方法。典型的方法是对分词形成的标识符序列构造前缀树,再在此基础上作对比,该类型的工具有Dup,CCFinder等。

(3)基于度量的方法。对程序代码进行结构性度量,再着重考虑度量值相近的代码段是否形成克隆,这方面的工具有CLAN。

(4)基于程序结构表示的方法。采用抽象语法树AST(或抽象语法前缀树)和程序依赖图PDG(program dependency graph)等形式表示程序代码的语法信息,再在树或图上开展频繁子树(图)的挖掘,相应的工具有CloneDR,Bauhaus,Duplix和GPLAG等。

此外,还有一些运用频繁项集、潜在语义索引LSI的方法[3]和工具(如CP-Miner[4])。虽然该问题提出的时间早,但目前较成熟的方法仅考虑了代码中的语法信息而忽略了潜在的语义,因此仍是知名国际会议上的重要议题之一。当前研究趋势有三:结合语义的查准率(precision)和查全率(recall)改进,面向Web应用的克隆检测,以及下节将介绍的Aspect挖掘。

1.2 Aspect挖掘

面向方面软件开发(aspect-oriented software development,AOSD)是为解决程序关注点之间的散列和缠结问题而提出的一种新的程序开发范型[5]。将遗留系统改造为面向方面软件要解决的两个关键性问题是横切关注点的挖掘(也称作Aspect挖掘)和重构。由于某一关注点在程序中常常表现为相同或相近的代码段,因此传统的克隆代码检测技术仍是Aspect挖掘的一种重要解决途径。此外,还有基于执行模式挖掘[14,15]、形式概念分析[9,10,16]、度量分析[7,17,19]以及调用关系分析[14,18]等针对性较强的方法。我们对上述方法列表总结如表1所列,其中部分方法的定量评估可参见文献[20]。

表1 Aspect挖掘方法的总结与对比

类别	挖掘方法	代表工具	特点
代码文本分析	基于PDG的挖掘	Ophir[6]	可发现before型advice;存在假性aspect
	基于克隆代码度量的方法[7]	结合1.1节工具进行	取决于启发式度量规则;存在假性aspect
	基于文本和类型的分析	AMT[8]	文本分析通过简单模式匹配实现;仅考虑类对象而忽略基本类型;当前版本未考虑继承层次
	形式概念分析法[9,10]	DelfSTof[9]	在计算结果的基础上需要人工检查
	基于自然语言处理的方法[11]		根据命名与编码习惯,结合语义信息;对结果需人工检查
	基于聚类分析的方法[12,13]	AMAV[12]	根据方法名或方法调用关系进行聚类
执行模式挖掘	基于方法执行关系的挖掘	DynAMiT[14]	发现的aspect受执行场景限制;结合静态信息的混合式方法[15]精度更高
	形式概念分析法	Dynamo[16]	计算结果受限于程序的执行场景
调用关系分析	扇入分析	FINT[17]	准确性可达到52%左右
	软件探查(exploration)技术	SEXTANT[18]	基于试探的方法,即设置种子再进行导航查询
	基于耦合与PageRank度量的方法[19]	PAM[19]	能用于大型系统;查准与查全能力较强;需要领域知识的协助

对于一个实际应用的系统而言,运用上述算法得到的横切代码候选集一般较为庞大。因此,如何在候选集的指导下开展面向方面的重构也是一个难题。其实数据挖掘技术也可在这方面发挥重要作用,如Anbalagan等人通过对横切代码候选集进行聚类,并结合字符串匹配算法指导连接点(join point)的自动生成[21]。

1.3 代码(构件)检索

复用一直是软件开发方法学中所倡导的一个重要原则。如何从(本地的或面向Web的)代码(构件)库中检索出适用于当前项目的代码或构件是软件复用领域近20年来的重要议题。其中代表性的方法有如下几类:(1)分类存储并检索的方法:根据构件的刻面[22]或关键词[23]进行分类或索引,再在此基础上检索;(2)基于输入-输出映射的检索:通过记录构件的输入与输出关系的检索方式[24];(3)基于构件形式化描述的方法[25];(4)基于代码相似性的方法:如根据文本[26]或代码结构[27]相似;(5)基于语义相似性的检索[27];(6)基于构件交互关系图的方法[28]等。所涉及到的技术主要是分类、聚类、高效索引和语义分析等几种。

随着Web应用范围的日益拓展,存在于Web中的不仅仅是文档,还包含大量可供参考的代码,因此Google Code、Koders和Mica等代码搜索引擎应运而生。此外,文献[29,30]还介绍了Agora,IntelliZap,Strathcona,MARACATU,Prospector和PARSEWeb等代码搜索原型系统。这里我们对这些工具列表对比,如表2所列。

表2 代码检索工具的归纳分析

工具名称	形式	技术特点
Google Code	Web portal	针对代码的抽象语法树进行相似性的度量,返回较多的无关结果。
Koders	Web portal	针对开源软件的搜索引擎,工作机制与Agora及MARACATU类似。
Mica	Web portal	使用Google Web APIs发现相关的网页,再通过分析页面上的内容过滤出高度相关的部分。
Agora	SDK工具箱	根据构件模型进行分类,能自动生成软件产品资源库,主要用于构件搜索。
IntelliZap	客户端应用模式	结合查询词所在上下文进行检索,涉及语义关键词提取、聚类等技术。
Strathcona	Eclipse plugin	根据实例代码(sample code)运用启发式规则进行搜索,度量方式为代码结构相似性,无关结果较多。
MARACATU	Eclipse plugin	基于Lucene引擎进行,主要采用文本挖掘和刻面技术,从CVS服务器中挖掘Java代码或构件。
Prospector	Eclipse plugin	以类类型对的形式作为输入开展查询,通过遍历API函数的声明进行匹配,同样会返回较多无关结果。
PARSEWeb	Eclipse plugin	基于Google Code的二次开发,以对象类型规则式的形式作为输入开展查询,相较Google Code,Strathcona,Prospector精度有较大提高。

2 面向程序执行记录的挖掘

除了从程序代码及其结构(如控制流图 CFG、程序依赖图 PDG)中可以发现一些用于改进软件质量的代码片断、规则外,程序执行过程中的外在表现和跟踪信息同样可以用于发现一些潜在的模式和规则。一般而言,从程序的执行记录中可以挖掘程序的规格说明(specification,也称规约)、交互模式等,并可用于指导故障的快速定位。

2.1 程序规约挖掘

所谓程序规约挖掘就是通过分析程序的执行跟踪,发现程序代码所展示的协议[31]。其本质是根据执行跟踪信息进行逆向建模,将有助于实施程序理解、验证和维护。该类挖掘的一般过程如下:对被分析系统进行初步的插装,记录软件对API(应用编程接口)或基本模块的调用和系统状态变量的值(我们称这些为跟踪信息),再对跟踪信息进行必要的过滤、聚类和约简,形成能表征系统功能的规约模型,基本挖掘过程如图 1 所示。根据最终形成规约的形式来看,目前比较典型的有基于自动机和基于规则的两种规约挖掘方法。

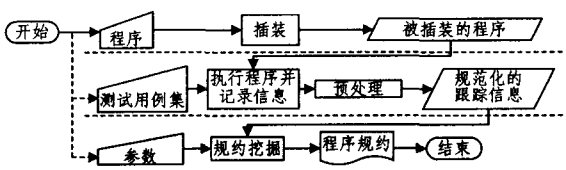

图 1 程序规约挖掘的基本过程

Ammons 等人首先提出了规约挖掘的概念[31],并提出了一种实用的执行跟踪概率学习方法,用于挖掘程序与 API 或抽象数据类型 ADT 交互时所应遵循的时态或数据依赖关系。他们构建的挖掘系统主要包含跟踪记录、流依赖注解、场景提取和自动机学习这 4 个主要部件,最终形成的规约运用概率有限状态自动机 PFSA 表示。类似地,Lo 和 Khoo 根据 API 交互的跟踪记录提取用自动机表示的时态规约,并给出了名为 SMArTIC 的挖掘框架[32]。Lo 等人的方法的显著特点是增加了排除错误跟踪信息的过滤过程和对强相关的执行跟踪进行聚类的过程。Shoham 等人则提出了一种基于自动机抽象表示的 API 规约挖掘方法[33],通过静态分析客户端程序对 API 的使用情况而实施挖掘。

在基于规则的挖掘方面,主要是根据程序的执行行为发现形如"前提事件→结果事件"的规则,一般用时态逻辑表达式来表示。通常,API 使用模式(特别是顺序调用模式)的挖掘将给系统的安全、健壮操作提供指导。Engler 等人[34]和 Weimer 等人[35]首先考虑了两事件规则的挖掘,其后 Lo 和 Khoo 给出了多事件规则的挖掘算法[32,36],虽然挖掘该类规则是一个 NP 难问题,但他们采用了基于 Apirio 的剪枝和冗余项检测等手段保证产生正确和完全的规则。此外,在 API 使用序挖掘方面,还有基于模型检验器、基于偏序关系发现和迭代模式挖掘等方法。

2.2 故障定位

为了保证测试覆盖的充分性,在实际测试活动中一般会用大批量的测试用例进行测试,从而导致引发程序失效的用例数目不可小视。那么,在后续的程序诊断过程中如何快速而全面地找出故障就困难起来,仅靠调试人员进行人工检查难以奏效,而结合数据挖掘技术往往能得到比较理想的效果。

在软件维护实践中能发现这样一条规律:往往程序的多次失效仅由于同一个故障引起。如此看来,就没有必要对每次失效都进行调试,对它们进行聚类就能避免这类重复。Dickinson 和 Podgurski 首先提出了对程序失效进行聚类以降低调试代价的思想[37],毛澄映等人通过将失效与非失效用例分离、相异性度量改进等手段提高了聚类精度[38]。此后,还有研究者通过分析程序失效与函数返回值、函数调用对等之间的关联信息进行故障定位[39,40];类似地,Li 和 Zhou[41]运用频繁项集挖掘技术对程序元素(函数、变量和数据类型等)进行使用模式挖掘,再运用挖掘结果指导故障定位。

程序切片(program slicing)是最传统的故障定位技术,但维护人员在理解大型复杂系统的切片(特别是静态切片)结果时仍存在困难,从而难以准确定位故障。基于上述事实,可以在切片结果或程序插装记录上作进一步地统计分析(Bayes 分析[42]、Bayes 信念网[43]和假设检验[44]等),大大缩小了故障查找的范围。另一种相近思想的处理方法是:对用例的每次执行跟踪均作记录,再对这些跟踪作谱分析并可视化地显示与失效联系紧密的程序语句集[45]。刘彦斌等人同样采用程序谱来抽象表达程序执行轨迹,根据成功的运行和含有故障的运行之间的差异来进行故障定位[46]。

此外,Renieris 和 Reiss 同样在分析程序运行剖面的距离和差异的基础上,给出了基于最近邻查询的故障定位方法,并提出了故障定位的通用框架[47]。吴际等人[48]进行软件输入级的故障定位,定义了软件输入响应对象及其构成的测试序列,以及测试用例对应的失效观察序列,建立了故障定位的统计模型。Ren 等人[49]则从程序演化的角度看待故障定位问题,运用分类技术和启发式规则发现可能引起程序失效的变更。

2.3 故障(失效)预测

根据软件的度量值或初步测试结果对其中的故障进行预测将为软件系统的后期维护提供有意义的指导。目前这方面的研究可分为两大类[52]:(1)基于量化分析技术,对软件系统中潜伏故障的数目进行预测;(2)基于分类技术,关注预测系统中的哪些模块具有故障倾向。就研究现状而言,数据挖掘领域的绝大多数分类和预测技术在软件故障或失效的预测应用方面均有体现。

(1)软件故障数目预测

故障数目预测可用静态预测和动态预测两种方法解决。所谓静态方法就是利用程序的一些静态结构信息(如程序复杂性度量值)进行预测;动态方法则是根据历史数据进行学习后开展预测。通常,静态预测方法是根据实践统计数据形成一些经验公式,有 Akiyama 模型、Halstead 模型和 Linpow 模型等。最为典型的动态预测方法是根据已有的部分测试记录构造一些预测误差尽可能低的回归模型[50]。而基于神经网络的预测方法相较传统的线性回归模型预测精度更高,且降低了数据预处理的要求[51];回归树(regression tree)是由抽象树模型表示的决策规则的集合,一些独立变量(称为预测器)用于预测诸如故障数目等响应变量。该模型具有很好的解释性,相较神经网络方法在速度上有优势,且容易转化为 SQL 语句。目前,Bayes 信念网在软件故障预测中也广为采用[52,53],其特点是充分运用了过程度量值和软件产品度量

值，还可以从软件开发全生命周期来考察故障。

(2)故障倾向模块预测

除了预知故障数目为后期维护提供科学决策外，维护人员往往更关心哪些模块最有可能包含故障，称这样的模块为故障倾向(fault-proness)模块。由于故障倾向模块的预测在维护活动中更具应用价值(例如可为模块测试及快速诊断提供指导)，因此这方面的研究非常活跃，文献[54]也对一些常用预测方法作了对比分析。从数据挖掘的角度看，故障数目与故障倾向模块的预测之间并没有本质区别，因而所采用的方法均以分类技术居多。再者，两种预测结果往往可以互相转化：根据预测的故障数目结合故障分布的分析可以估计出故障倾向模块；反之，根据故障倾向模块的发现也可以初步估算出整个系统潜伏的故障数目。

(3)关键度量因素选取

在进行软件故障预测时，关于软件的度量信息是至关重要的输入，通常可分为软件结构度量、软件过程度量和软件执行度量3大类。一般而言，众多的度量指标之间存在关联，并且如果预测模型的输入量过多将降低可操作性。为此，从度量指标集中选取具有代表性的度量子集也是一项重要任务，目前使用的技术也主要是一些统计分析技术，如区分分析、主成分分析等。机器学习方法应用到其中也取得较好的效果：Menzies等人[55]运用附加对数转换的朴素Bayes分析选择出来的度量子集可达到71%的查全率，比单个度量阈值设置和决策树算法J48的效果要好；而利用模糊Rough集推理技术选取关键度量属性的方法[56]既能提高关联精度又能降低关联集合的大小，表现出较好的实用性。

3 面向软件项目管理数据集的挖掘

软件开发发展到今天已经成为集开发技术、管理学、经济学、组织行为学等众多知识于一身的综合性学科。因此，当代软件开发企业除了注重软件开发技术的革新外，更注重规范、科学的管理。在软件项目管理过程中形成的文档主要有如下几类：(1)软件演化跟踪信息，如版本控制信息、缺陷记录等；(2)项目人员管理信息；(3)项目经费及时间进度信息；(4)软件开发过程记录信息等。在上述信息上开展挖掘将有利于发现软件演化规律、项目资源(人员、经费及时间)的优化配置、以及部分软件过程的复用。本文针对软件项目管理的不同类型的数据集，阐述在其之上的数据挖掘应用，并指出所带来的益处。

3.1 版本控制信息的挖掘

版本控制系统(如CVS，SVN，Visual SourceSafe等)有利于程序修改的有序性、可回溯性以及全局统一性等，目前软件开发企业或开源组织基本上均会采用它来组织多参与者或分布式的软件开发活动。当前在版本控制信息上挖掘应用大多是挖掘软件变更历史的，并用以发现不同程序模块(子系统)间的逻辑依赖[57,58]，预测程序故障的引入方式[59-62]，或是预测今后的变更可能[62,63]。通过上述挖掘活动，将有利于降低系统后期维护代价，便于系统逆向分析和前瞻性地预防一些因变更而引入的故障。

基于程序变更信息的挖掘为程序内部逻辑依赖分析提供了一种新途径：CAESAR方法[57]从分析系统的多个发布记录入手，找出软件模块之间的常规变更模式；Rysselberghe则对变更历史数据进行可视化表示并发现一些有价值或有问题的部分来方便逆向工程活动[58]，如发现内聚程序单元。

故障引入方式的挖掘会为软件系统的后期维护提供警示作用。Graves等人将软件演化过程看作是“年龄”增长过程，运用广义线性模型等统计分析技术发现故障引入分布特征[59]，他们的实验表明：基于变更历史的故障率预测比基于常规代码度量的预测更有用，并发现软件“年龄”增长一年以后其故障数将大致降至原来的三分之一。文献[60]则通过扩展JUnit形成一个变更分类工具JUnit/CIA，该工具分别用不同颜色表征不同故障引起软件失效的可能性，可以方便开发人员快速诊断高危险性故障。挖掘工具DynaMine[61]则在软件修复历史记录中挖掘一些常规的错误模式，其挖掘结果有助于软件开发者避免这些常见错误的再次引入。

后续变更的预测为项目的管理及相关决策提供重要的参考依据，较为典型的做法是在CVS版本控制文档上采用关联规则挖掘算法找出当前的程序变更将会引起的后续变更[62]，Zimmermann等人在此基础上开发出了用于版本控制信息挖掘的Eclipse插件系统ROSE[63]，该系统具有较强的适时挖掘能力，在粒度上除文件级别外还支持类、函数等级别间的关联分析能力。

此外，挖掘版本库所形成的知识还有利于不同版本程序的对比，找出相匹配或近似代码。对于大型软件项目的开发来讲，所形成的版本库也是十分庞大的，聚类、搜索以及可视化等手段也常常应用到其中以方便程序变更规律的发现。

3.2 人员组织关系的挖掘

在组织大规模软件开发过程中，如何协调、调度并分配人力资源也是软件工程领域的一个严峻问题。人们在解决复杂问题时往往会采用“分而治之”的思想，例如软件模块化设计就是一例典型应用。在构建一个大型复杂软件系统时，一般是几百人甚至上千人参加的一个系统工程(例如Windows操作系统的研发)，而这些开发人员之间不可避免地通过讨论会、文档传递、电子邮件等途径产生交互，深入挖掘组织人员之间的关系将有利于小组划分、任务指派等管理活动。软件组织内部员工(以及软件客户)之间的关系网络是一种典型的社会网络，随着近年来复杂网络研究的兴起，挖掘该网络内部关系促进软件项目管理的研究已初现端倪。

早在1968年，M. E. Conway就在其论文“How Do Committees Invent?”中给出这样一个假设(后来Fred Brooks在《人月神话》中称之为Conway定律)：软件系统的结构是其开发组织结构的直接反映，如图2所示。随后，Bowman等人分别对Linux，Mozilla和一个商用软件系统开展实证发现：软件概念结构、真实结构和开发人员间的关系结构三者之间表现出十分相近的结构特征[64]。

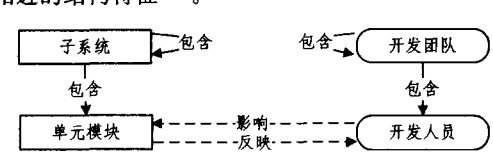

图2 开发组织结构与软件结构的潜在关系映射

近年来，Amrit[65]在Conway定律的指导下对由研究生组成的4个分布式研究小组进行问卷调查，进而形成开发成员之间、成员与任务之间的交互图，再结合复杂网络的度量指标以发现更优的任务分派策略。此外，他们还给出了组织过

程结构的矩阵表达及计算方法。文献[66]以一个大型的信息系统开发为例,通过收集25个小组在分析与设计阶段的实际数据发现:人员组织结构(如内聚性)是影响软件系统性能的至关重要的指标。

基于上述事实可知:对开发人员的科学组织和任务的合理分派将会给软件项目的代价、进度以及成功性带来重要影响。在这方面一个比较典型的研究方法就是对软件过程实施模拟建模[67]。

3.3 经济效益及开发过程的挖掘

(1)软件经济效益挖掘

软件项目的成功与否一般受资金、进度和软件质量3方面的因素制约。关于软件质量和进度的挖掘、度量和预测等方面的研究已经较为深入,而数据挖掘技术在软件项目经济效益(代价)方面的应用则相对薄弱。早期,Barry Boehm在对大量软件项目数据集进行统计(回归)分析的基础上提出了著名的软件工程经济学模型COCOMO及其改进COCOMOII。最近,英国学者Bahsoon和Emmerich提出了经济驱动的软件挖掘(economics-driven software mining,EDSM)的概念[68]。他们认为从软件项目相关的数据仓库中提取的信息不仅可以用于技术层面,也可为软件系统的开发维护和演化提供投资决策。他们还进一步指出了EDSM的3个应用方向:(1)补充现有一般挖掘技术以实现特定的经济相关查询;(2)基于证据的发现,寻找表面上不相干的概念之间的潜在联系;(3)在挖掘过程中融合经济模型,即将挖掘结果作为模型的输入,其输出是软件开发、管理活动中要寻求的目标。

软件开发往往在软件发布的时机上难以抉择:如果发布了没有充分排除缺陷的软件将会失去客户,而过于彻底的测试又会导致得不偿失。因此,在软件项目中如何找到最佳的发布时机以及发现缺陷修复代价的规律等问题一直备受关注。Ling等人[69]提供了一个将软件发布相关的最大获益问题转化为代价敏感学习(cost-sensitive learning)问题的通用框架,并通过大量实验发现代价敏感决策树方法是较好的解决途径。在软件故障修复代价的发现方面,Morisaki等人发现了一系列故障特征和修复代价方面的关联规则[70],这些规则将会为后期维护活动提供量化指导。

(2)软件开发过程挖掘

软件复用发展到今天已经超出了简单的代码、测试用例复用的范畴,软件开发过程已经成为一种新的复用对象。主要从历史项目中挖掘软件项目成功的子过程、控制策略等,或是从失败的历史项目中找到反面警示。目前,国内外在软件过程的挖掘与复用方面尚处于起步阶段。已出现的成果均是在历史项目控制数据集中逆向建立可供后续参考的过程模型:例如,中科院软件所Internet软件技术实验室提出了从历史项目数据中自动构建过程代理(process-agent)的技术[71];荷兰的Rubin等学者给出了从软件配置管理系统中挖掘形成软件过程模型的框架[72]。

4 面向开源软件项目的挖掘

随着网络的快速发展与普及,以及全局软件开发和开源意识的增强,Web环境下开源软件(open source software,OSS)的发展非常之快。仅以sourceforge为例,目前已经超过15.7万个开源项目,近168万个注册用户。由于OSS的开发模式跟传统的商用软件存在很大不同,开发环境是开放式的,项目参与者也是动态变化的,正由于这些原因使得OSS项目的控制和管理变得异常困难。而数据挖掘技术在开源项目中的应用将有助于规律的发现,进而提高OSS的质量。

由于OSS是Web上可供利用代码的主要对象,怎样准确、高效地检索开源系统中的代码是代码检索领域中的研究重点。一般而言,1.3节中所介绍的检索方法均适用于开源软件代码的挖掘。OSS之间共用代码是非常普遍的情况,有实验研究表明有50%以上的文件应用到两个以上的开源项目中[73]。因此,在克隆代码检测方面除了检测单个OSS内的重复代码外,发现OSS之间的代码克隆也十分必要。显著的有,日本大阪大学的Livieri等人在CCFinder的基础上设计了一个分布式克隆代码挖掘系统D-CCFinder[74],既可实现大规模系统的挖掘,还可以实现多个开源项目的全局挖掘。

开源项目的开发环境是全局、动态和开放的,因此对该类软件的开发过程管理将有别于传统的软件。一般而言,成熟的开源项目对参与到其中的开发者的活动、错误报告以及软件的使用等均有比较完整的记录。参与其中的开发者形成的是一个典型的社会网络,由于其开放性将导致参与的人员不断发生变化,对其动态演变性等网络特征的挖掘将会促进开源项目的高效管理。例如:英国牛津大学开发的Simal系统能对开源项目的开发者及使用者进行系统化地跟踪管理[75]。此外,由于开源软件的开发模式相较传统软件发生了很大变化,关于OSS项目的代价估计问题也是一个重要的研究方向[76]。开源项目的一个重要特征是其故障资源库、版本变更信息以及代码库等均是公开的,因此目前大多数研究人员采用它们作为实验数据集。

结束语 随着软件系统日趋复杂、涉众(stakeholders)增多以及技术不断更新,软件开发已经演变成一项复杂的系统工程。相应地,软件工程也变成一个"计算型"的工程学科。在该项活动中原有的靠单个人员的决策行为已经不复存在,更多的则是对软件项目数据进行挖掘、分析和智能处理的基础上进行"量化"管理。本文以软件开发活动中形成的数据集为对象,对其上实施的挖掘技术及其主要用途进行了系统的归纳与总结,旨在给国内的同行以适当的借鉴。

不难看出,数据挖掘领域的主要技术均能在软件工程实践活动中找到应用之处,并产生了一定的获益。但由于软件工程中数据形式、管理方式和多重影响因素等原因,数据挖掘技术在该领域的应用研究尚处于初步发展期,还有很多工作值得进一步深入探究。例如,(1)结合自然语言处理及语义分析的软件项目文档挖掘;(2)软件特征准确定位;(3)软件安全脆弱点的挖掘等;这些就是今后研究方向的范例。

参考文献

[1] Allen K, Krutz W, Olivier D. Software Reuse: Mining, Refining, and Designing, 1990: 222-226

[2] Bellon S, Koschke R, Antoniol G, et al. Comparison and Evaluation of Clone Detection Tools[J]. IEEE Trans. on Soft. Eng., 2007, 33(9): 577-591

[3] Marcus A, Maletic J. Identification of High-Level Concept Clones in Source Code[C]//Proc. of ASE'01. 2001: 107-114

[4] Li Z, Lu S, Myagmar S, et al. CP-Miner: A Tool for Finding Copy-paste and Related Bugs in Operating System Code[M].

Operating System Design Implementation, 2004: 289-302
[5] Kiczales G, Lamping J, Menhdhekar A, et al. Aspect-Oriented Programming[C]// Proc. of ECOOP. 1997: 220-242
[6] Shepherd D, Gibson E, Pollock L. Design and Evaluation of an Automated Aspect Mining Tool[C]// Proc. of SERP'04. 2004: 601-607
[7] Bruntink M, Deursen A, Engelen R, et al. On the Use of Clone Detection for Identifying Crosscutting Concern Code[J]. IEEE Trans. on Soft. Eng., 2005, 31(10): 804-818
[8] Hannemann J, Kiczales G. Overcoming the Prevalent Decomposition of Legacy Code[C]// Proc. of Workshop on Advanced Separation of Concerns at the ICSE'01. 2001
[9] Tourwe T, Mens K. Mining Aspectual Views using Formal Concept Analysis[C]// Proc. of Workshop on Source Code Analysis and manipulation (SCAM'04). 2004: 97-106
[10] 金龙飞, 刘磊. 一种基于形式概念分析的语句级自动化方面挖掘方法[J]. 小型微型计算机系统, 2006, 27(4): 677-680
[11] Shepherd D, Pollock L, Tourwe T. Using Language Clues to Discover Cross-Cutting Concerns[C]// Proc. of Workshop on the Modeling and Analysis of Concerns. 2005: 1-6
[12] Shepherd D, Pollock L. Interfaces, Aspects, and. Views[C]// Proc. of Workshop on Linking Aspect Technology (LATE). 2005
[13] 何丽莉, 白洪涛. 用聚类分析方法挖掘 Aspect[J]. 计算机集成制造系统, 2006, 12(1): 149-153
[14] Breu S, Krinke J. Aspect Mining Using Event Traces[C]// Proc. of ASE'04: 310-315
[15] Breu S. Extending Dynamic Aspect Mining with Static Information[C]// Proc. of SCAM'05. 2005: 57-65
[16] Tonella P, Ceccato M. Aspect Mining through the Formal Concept Analysis of Execution Traces[C]// Proc. of 11th Working Conference on Reverse Engineering. 2004: 112-121
[17] Marin M, Deursen A, Moonen L. Identifying Aspects using Fan-in Analysis[C]// Proc. of WCRE'04. 2004: 132-141
[18] Schafer T, Eichberg M, Mezini M. Towards Exploring Cross-Cutting Concerns[C]// Proc. of LATE'05. 2005
[19] Zhang C, Jacobsen H A. Efficiently Mining Crosscutting Concerns through Random Walks[C]// Proc. of AOSD'07. 2007: 226-238
[20] Ceccato M, Marin M, Mens K, et al. Applying and Combining Three Different Aspect Mining Techniques[J]. Software Quality Journal, 2006, 14: 209-231
[21] Anbalagan P, Xie T. Clamp: Automated Joinpoint Clustering and Pointcut Mining in Aspect-Oriented Refactoring[C]// Proc. of FSE'06 (Poster Session). 2006: 1-2
[22] Prieto-Diaz R. Implementing Facted Classification for Software Reuse[J]. Communication of the ACM, 1991, 34(5): 88-97
[23] Maarek Y S, Berry D M, Kaiser G E. An Information Retrieval Approach for Automatically Constructing Software Libraries [J]. IEEE Trans. on Soft. Eng., 1991, 17(8): 800-813
[24] Podgurski A, Pierce L. Retrieving Reusable Software by Sampling Behavior[J]. ACM Trans. on Soft. Eng. and Meth., 1993, 2(3): 286-303
[25] Garcia V C, Lucredio D, Durao F A, et al. From Specification to Experimentation: A Software Component Search Engine Architecture[C]// Proc. of CBSE'06. 2006: 82-97
[26] Holmes R, Murphy G C. Using Structural Context to Recommend Source Code Examples[C]// Proc. of ICSE'05. 2005: 117-125
[27] Sugumaran V, Storey V C. A Semantic-based Approach to Component Retrieval[J]. ACM SIGMIS Database, 2003, 34(3): 8-24
[28] Inoue K, Yokomori R, Yamamoto T, et al. Ranking Significance of Software Components Based on Use Relations[J]. IEEE Trans. on Soft. Eng., 2005, 31(3): 213-225
[29] Garcia V C, Almeida E S, Lucredio D, et al. Towards a Code Search Engine Based on the State-of-Art and Practice[C]// Proc. of APSEC'06. 2006: 61-70
[30] Thummalapenta S, Xie T. PARSEWeb: A Programmer Assistant for Reusing Open Source Code on the Web[C]// Proc. of ASE'07. 2007: 204-213
[31] Ammons G, Bodik R, Larus J R. Mining Specifications[J]. ACM SIGPLAN Notices, 2002, 37(1): 4-16
[32] Lo D, Khoo S C. SMArTIC: Towards Building an Accurate, Robust and Scalable Specification Miner[C]// Proc. of SIGSOFT'06/FSE14. 2006: 265-275
[33] Shoham S, Yahav E, Fink S, et al. Static Specification Mining Using Automata-Based Abstractions[C]// Proc. of ISSTA'07. 2007: 174-184
[34] Engler D, Chen D Y, Hallem S, et al. Bugs as Deviant Behavior: A General Approach to Inferring Errors in Systems Code[C]// Proc. of ACM Symposium on Operating Systems Principles (SOSP). 2001: 57-72
[35] Weimer W, Necula G C. Mining Temporal Specifications for Error Detection[C]// Proc. of TACAS'05. 2005: 461-476
[36] Lo D, Khoo S C. A Sound and Complete Specification Miner[C]// PLDI'07 Student Research Competition (awarded 2. nd. position)-www. acm. org/src/winners. html, 2007
[37] Dickinson W, Leon D, Podgurski A. Finding Failures by Cluster Analysis of Execution Profiles[C]// Proc. of ICSE'01. 2001: 339-348
[38] Mao Chengying, Lu Yansheng. Extracting the Representative Failure Executions via Clustering Analysis Based on Markov Profile Model[C]// Proc. of ADMA'05. 2005: 217-224
[39] Liblit B, Aiken A, Zheng A X, et al. Bug Isolation via Remote Program Sampling[J]. ACM SIGPLAN Notices, 2003, 38(5): 141-154
[40] Livshits B, Zimmermann T. DynaMine: Finding Common Error Patterns by Mining Software Revision Histories[C]// Proc. of FSE'05. 2005: 296-305
[41] Li Z, Zhou Y. PRMiner: Automatically Extracting Implicit Programming Rules and Detecting Violations in Large Software Code[C]// Proc. of ESEC/FSE'05. 2005: 306-315
[42] Liu C, Lian Z, Han J. How Bayesians Debug[C]// Proc. of ICSM'06. 2006: 382-393
[43] Burnell L, Horvitz E. Structure and Chance: Melding Logic and Probability for Software Debugging[J]. Communications of the ACM, 1995, 38(3): 31-41
[44] Liu C, Fei L, Yan X, et al. Statistical Debugging: A Hypothesis Testing-Based Approach[J]. IEEE Trans. on Soft. Eng., 2006, 32(10): 831-848

(下转第 26 页)

Sound Mechanized Proofs for Basic and Public-key Kerberos[C]// ASIACCS'08. Tokyo, Japan, March 2008

[19] Datta A, Mitchell J, Warinschi B. Computationally Sound Compositional Logic for Key Exchanged Protocols[C]// Proc. of CSFW'06. July 2006

[20] 李梦君,李舟军,陈火旺. SPVT:一个有效的安全协议验证工具[J]. 软件学报,2006,17(4):898-906

[21] 李梦君,李舟军,陈火旺. 安全协议的扩展 Horn 逻辑模型及其验证方法[J]. 计算机学报,2006,29(9):1666-1678

[22] 周倜,李梦君,李舟军,等. 基于 Horn 逻辑扩展模型的安全协议反例的自动构造[J]. 计算机研究与发展,2007,44(9):1518-1531

[23] Li Mengjun, Zhou Ti, Li Zhoujun, et al. An Abstraction and Refinement Framework for Verifying Security Protocols Based on Logic Programming[C]// ASIAN 2007, LNCS 4846. 2007:166-180

[24] Li Mengjun, Li Zhoujun, Chen Huowang, et al. A Novel Derivation Framework For Define Logic Program[J]. Electronic Notes in Theoretical Computer Science, 2008, 212:71-85

[25] Zhou Ti, Li Mengjun, Li Zhoujun, et al. Modeling and Verifying Time Sensitive Security Protocols with Constraints[J]. Electronic Notes in Theoretical Computer Science, 2008, 212:103-118

(上接第 6 页)

[45] Jones J, Harrold M J, Stasko J. Visualization of Test Information to Assist Fault Localization[C]// Proc. of ICSE'02. 2002: 467-477

[46] 刘彦斌,朱小冬. 基于双轨迹差异分析法的软件故障定位[J]. 计算机工程,2007,33(9):43-48

[47] Renieris M, Reiss S P. Fault Localization with Nearest Neighbor Queries[C]// Proc. of ASE'03. 2003:30-39

[48] Wu Ji, Jia Xiaoxia, Liu Chang, et al. A Statistical Model to Locate Faults at Input Level[C]// Proc. of ASE'04. 2004:274-277

[49] Ren X, Ryder B G. Heuristic Ranking of Java Program Edits for Fault Localization[C]// Proc. of ISSTA'07. 2007:239-249

[50] Ostrand T J, Weyuker E J, Bell R M. Predicting the Location and Number of Faults in Large Software Systems[J]. IEEE Trans. on Soft. Eng., 2005, 31(4):340-355

[51] Khoshgoftaar T M, Pandya A S, Lanning D L. Application of Neural Networks for Predicting Faults[J]. Annals of Software Engineering, 1995, 1:141-154

[52] 罗云峰,贲可荣. 基于 BBNs 的软件故障预测方法[J]. 电子学报,2006,34(12A):2380-2383

[53] Pai G J, Dugan J B. Empirical Analysis of Software Fault Content and Fault Proneness Using Bayesian Methods[J]. IEEE Trans. on Soft. Eng., 2007, 33(10):675-686

[54] Denaro G, Pezze M. An Empirical Evaluation of Fault-proneness Models[C]// Proc. of ICSE'02. 2002:241-251

[55] Menzies T, Greenwald J, Frank A. Data Mining Static Code Attributes to Learn Defect Predictors[J]. IEEE Trans. on Soft. Eng., 2007, 33(1):2-13

[56] Ramanna S, Bhatt R, Biernot P. Software Defect Classification: A Comparative Study with Rough Hybrid Approaches[C]// Proc. of RSEISP'07. 2007:630-638

[57] Gall H, Hajek K, Jazayeri M. Detection of Logical Coupling Based on Product Release History[C]// Proc. of ICSM'98. 1998:190-198

[58] Rysselberghe F V, Demeyer S. Studying Software Evolution Information by Visualizing the Change History[C]// Proc. of ICSM'04. 2004:328-337

[59] Graves T L, Karr A F, Marron J S, et al. Predicting Fault Incidence Using Software Change History[J]. IEEE Trans. on Soft. Eng., 2000, 26(7):653-661

[60] Stoerzer M, Ryder B G, Ren X, et al. Finding Failure-Inducing Changes in Java Programs using Change Classification[C]// Proc. of FSE'06. 2006:57-68

[61] Livshits B, Zimmermann T. DynaMine: Finding Common Error Patterns by Mining Software Revision Histories[C]// Proc. of FSE'05. 2005:296-305

[62] Ying A T, Murphy G C, Ng R, et al. Predicting Source Code Changes by Mining Change History[J]. IEEE Trans. on Soft. Eng., 2004, 30(9):574-586

[63] Zimmermann T, Weibgerber P, Diehl S, et al. Mining Version Histories to Guide Software Changes[J]. IEEE Trans. on Soft. Eng., 2005, 31(6):429-445

[64] Bowman I T, Holt R C. Reconstructing Ownership Architectures To Help Understand Software Systems[C]// Proc. of IWPC'99:28-37

[65] Amrit C. Application of Social Network Theory to Software Development: The problem of Task Allocation[C]// Proc. of Computer Supported Activity Coordination. 2005:3-17

[66] Yanga H L, Tang J H. Team Structure and Team Performance in IS Development: A Social Network Perspective[J]. Information & Management, 2004(41):335-349

[67] Setamanit S, Wakeland W W, Raffo D. Exploring the Impact of Task Allocation Strategies for Global Software Development Using Simulation[C]// Proc. of SPW/ProSim'06. 2006:274-285

[68] Bahsoon R, Emmerich W. Economics-Driven Software Mining [C]// Proc. of the First International Workshop on the Economics of Software and Computation (ESC'07). 2007:3-7

[69] Ling C X, Sheng V S, Bruckhaus T F W, et al. Maximum Profit Mining and Its Application in Software Development[C]// Proc. of KDD'06:929-934

[70] Morisaki S, Monden A, Matsumura T, et al. Defect Data Analysis Based on Extended Association Rule Mining[C]// Proc. of MSR'07:3-10

[71] Zhang L, Wang Q, Xiao J, et al. A Tool to Create Process-Agents for OEC-SPM from Historical Project Data[C]// Proc. of ICSP'07:84-95

[72] Rubin V, Gunther C W, Aalst W M P, et al. Process Mining Framework for Software Processes[C]// Proc. of ICSP'07:169-181

[73] Brown A W, Booch G. Reusing Open-Source Software and Practices: The Impact of Open-Source on Commercial Vendors[C]// Proc. of the 7th International Conference Software Reuse: Methods, Techniques, and Tools. 2002:381-428

[74] Livieri S, Higo Y, Matsushita M, et al. Very-Large Scale Code Clone Analysis and Visualization of Open Source Programs Using Distributed CCFinder: D-CCFinder[C]// Proc. of ICSE'07: 106-115

[75] Simal-A Project Support Framework[OL]. http://simal.osswatch.ac.uk/index.html, 2007

[76] Asundi J. The Need for Effort Estimation Models for Open Source Software Projects[C]// Proc. of the Fifth Workshop on Open Source Software Engineering (5-WOSSE). 2005:1-3