ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.7, July 2008, pp.1565−1580
DOI: 10.3724/SP.J.1001.2008.01565

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

软件缺陷预测技术*

王 青[1+], 伍书剑[1,3], 李明树[1,2]

[1](中国科学院 软件研究所 互联网软件技术实验室,北京 100190)

[2](中国科学院 软件研究所 计算机科学国家重点实验室,北京 100190)

[3](中国科学院 研究生院,北京 100049)

Software Defect Prediction

WANG Qing[1+], WU Shu-Jian[1,3], LI Ming-Shu[1,2]

[1](Laboratory for Internet Software Technologies, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](State Key Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[3](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: E-mail: wq@itechs.iscas.ac.cn

Wang Q, Wu SJ, Li MS. Software defect prediction. *Journal of Software*, 2008,19(7):1565−1580.
http://www.jos.org.cn/1000-9825/19/1565.htm

Abstract: Software defect prediction has been one of the active parts of software engineering since it was developed in 1970's. It plays a very important role in the analysis of software quality and balance of software cost. This paper investigates and discusses the motivation, evolvement, solutions and challenges of software defect prediction technologies, and it also categorizes, analyzes and compares the representatives of these prediction technologies. Some case studies for software defect distribution models are given to help understanding.
Key words: software defect; metric; defect prediction; defect model; classification technology

摘 要: 软件缺陷预测技术从 20 世纪 70 年代发展至今,一直是软件工程领域最活跃的内容之一,在分析软件质量、平衡软件成本方面起着重要的作用.研究和讨论了软件缺陷预测技术的起源、发展和当前所面临的挑战,对主流的缺陷预测技术进行了分类讨论和比较,并对典型的软件缺陷的分布模型给出了案例研究.
关键词: 软件缺陷;度量;缺陷预测;缺陷模型;分类技术
中图法分类号: TP311 **文献标识码**: A

所谓缺陷(defect),目前为止,学术界、产业界有很多相关的术语和定义,比如故障、缺陷、bug、错误、失误、失效、失败等.根据 ISO 9000 对缺陷的定义"未满足与预期或者规定用途有关的要求",缺陷是软件中已经存在的一个部分,可以通过修改软件而消除.另一个重要的概念是失效(failure).当系统或者软件运行时,出现不正确的输出,则称为失效.严格地说,失效可能由软件缺陷引起,也可能由其他诸如人为因素、硬件故障等引起.如果我

* Supported by the National Natural Science Foundation of China under Grant Nos.60573082, 90718042 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2007AA010303 (国家高技术研究发展计划(863)); the National Basic Research Program of China under Grant No.2007CB310802 (国家重点基础研究发展计划(973))

Received 2007-06-27; Accepted 2008-03-12

们每观察到一次失效,就可以发现一个或者多个软件缺陷,那么纠正这些缺陷就可以避免类似失效的重复出现.

软件已经成为影响国民经济、军事、政治乃至社会生活的重要因素.高可靠和复杂的软件系统非常依赖于其采用的软件的可靠性.软件的缺陷是导致相关系统出错、失效、崩溃甚至机毁人亡的潜在根源.例如,1996 年 6 月,欧洲"阿丽亚娜"号航天飞机因导航系统的计算机软件出现故障,致使航天飞机坠毁,造成了数亿美元的巨大损失;2005 年 3 月 31 日,欧空局的 SMART-1 月球探测器和 NASA 的"雨燕"太空望远镜的使用状态均因软件故障而受到了很大影响;2005 年 4 月,软件失灵、继而导航失误,导致耗资 1.1 亿美元的 NASA 自主交会任务 DART 实验失败.

然而软件技术发展至今,任何检验、验证手段都不可能发现并排除所有的缺陷,软件作为一种无形的产物,虽然不会磨损用坏,却随时可能因为我们不易查知的原因出现故障甚至失效.

CeBASE(Center for Empirically-Based Software Engineering)[1]是美国国家科学基金支持的经验软件工程中心,拥有世界范围著名的专家和学术机构的合作组织.2002 年,CeBASE组织了几次著名的网上研讨,并在加拿大渥太华的第 8 届度量大会(METRICS 2002)上组办了名为"What we have learned about fighting defects"的研讨会.会后根据大家研讨的结果,整理出了软件缺陷所带来的影响及缺陷分布和检测问题:软件发布后发现和修复缺陷的成本大幅增加(例如,对于一个严重问题,其成本较之需求和设计阶段增加了 100 倍);软件项目返工工作量比例居高不下,但随着过程成熟度的提高而减少;软件工程中返工工作量与缺陷、软件失效与缺陷,以及缺陷在软件模块中的分布都是符合 2-8 原则的;同行评审、评审准备工作等有助于发现更多的缺陷;有经验的开发人员可以显著减少缺陷的引入率.

上述讨论结果表明,缺陷对软件质量甚至对软件经济有重要影响;同时也说明缺陷分布问题的复杂性和差异性,以及现有的缺陷预测技术在解决实际问题上的不足等.事实上,从第一个软件诞生,就伴随出现软件缺陷的检测和预测技术.检测技术在于发现缺陷,而预测技术则在于预测还未发现的缺陷.

20 世纪 70 年代,出现了利用统计学习技术,根据历史数据以及已经发现的缺陷等软件度量数据预测软件系统的缺陷数目及类型.缺陷预测技术的目的在于统计计算软件系统的缺陷数、没有发现但还可能存在的缺陷数,以决定系统是否可以交付使用.缺陷预测技术为软件质量的提高和保证起着非常重要的作用,同时,也促进了软件工程技术向前大大地发展了一步.

纵观软件缺陷预测技术的发展,从 20 世纪 70 年代起,软件缺陷预测技术大体上分为静态和动态两种缺陷预测技术,如图 1 所示.静态预测技术,主要是指基于缺陷相关的度量数据,对缺陷的数量或者分布进行预测的技术;而动态技术则是基于缺陷或者失效产生的时间,对系统缺陷随时间的分布进行预测的技术.

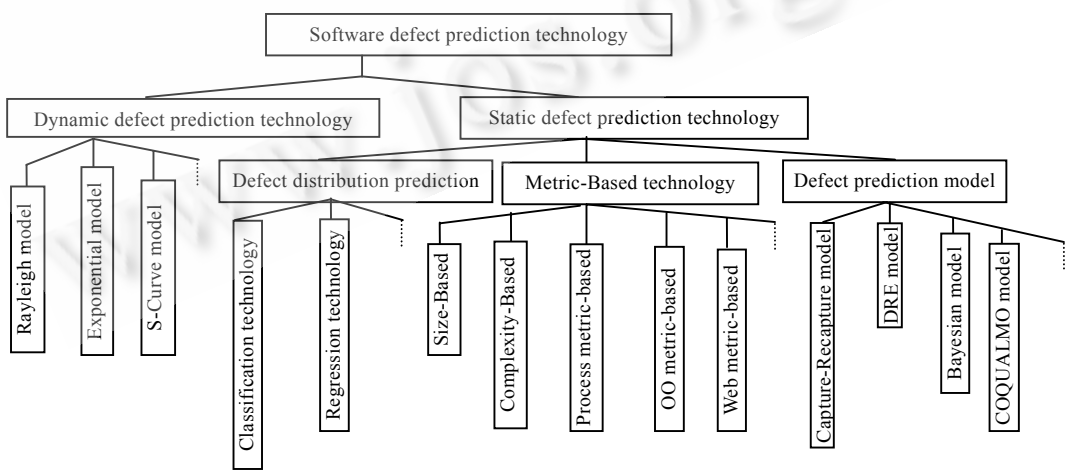

Fig.1 Classification of software defect prediction technologies

图 1 软件缺陷预测技术分类

1 静态的缺陷预测技术

静态缺陷预测技术起源较早,早期集中在基于软件规模等度量元的缺陷预测方面,亦即研究缺陷和软件规模、复杂度等基本属性之间的关系,以此预测软件可能存在的缺陷数量.20 世纪 90 年代初,人们发现缺陷并非在软件中平均或者完全随机地分布,出现了针对缺陷分布的预测技术.此外,缺陷在软件生命周期不同阶段的引入和移除对遗留缺陷也有严重的影响,一些相应的软件缺陷预测模型也取得了非常好的成果和应用.

下面按照时间顺序给出基于度量元的缺陷预测技术、缺陷分布预测技术以及缺陷预测模型技术.

1.1 基于度量元的缺陷预测技术

1.1.1 基于软件规模的缺陷预测技术

最早的软件缺陷预测技术可以追溯到20世纪70年代.早期的缺陷预测技术认为缺陷的多少取决于软件规模的大小,典型的代表有:

1) 早期的软件缺陷估算主要根据经验来估计,如人们普遍认为一个软件模块中缺陷数量约为每 60 行代码 1 个缺陷(或乐观地估计为 100 行代码 1 个缺陷),并据此来估计项目验证和测试阶段所需要的人力和计算机资源;最早的量化关系式由Akiyama[2]提出,他明确给出了软件缺陷与代码行的关系式:D=4.86+0.018L;同时,他还探讨了复杂度度量元(如the number of decisions,subroutine calls等)对于缺陷的影响;在Akiyama的关系式中,估算的软件缺陷是指在程序验证开始阶段之前,程序中所可能有的软件缺陷数量;

2) Halstead[3]提出的缺陷与软件体积度量元(volume metric)的关系:$D=V/3000$,其中软件体积V与开发语言相关,$V=N(\log_2(n)), N=N_1+N_2, n=n_1+n_2$,$n_1$代表不同操作符数量,$n_2$代表不同操作数数量,$N_1$代表总的操作符数量,$N_2$代表总的操作数数量;在Halstead的关系式中,估算的软件缺陷是指在程序验证阶段发现的软件缺陷数量;Ottenstein[4,5]使用上述关系式来估算在测试和集成阶段开始前系统中仍然存在的缺陷数,以及在验证阶段发现和修复缺陷所耗费的时间等;

3) Lipow[6]在Halstead的基础上进行了改进,提出了缺陷与可执行代码行L的关系:$D/L=A_0+A_1\ln L+A_2\ln^2 L$,其中系数$A_i$与语言相关,不同的语言有着较大的差异(例如,对高级语言来说,系数较大,如Fortran语言:A_0=0.0047,A_1=0.0023,A_2= 0.000043;而汇编语言:A_0=0.0012,A_1=0.0001,A_2=0.000002);

4) Takahashi[7]将代码与具体的文档数量对应起来,给出了缺陷密度的估计式:$\hat{B}=67.98+0.46f_1-9.69f_2-0.08f_3$,其中$\hat{B}$为缺陷率(单位:The number of errors/KLOC,KLOC为千行代码),f_1为程序规约变更的频率(单位:The number of pages/KLOC),f_2为程序员技能(单位:Years),f_3为程序设计文档(单位:The number of pages/KLOC);

5) Malaiya等人[8]在假设模块规模符合指数分布的情况下,给出了缺陷密度估算公式:缺陷密度$D(s)=a/s+b+cs$,其中s为规模,a,b,c为经验值,它们取决于程序员能力、过程成熟度以及测试程度等.

1.1.2 基于软件复杂度的缺陷预测技术

随着软件规模的增加,软件的复杂度也在逐步攀升,人们开始意识到软件的缺陷不仅与规模有关,还与软件的复杂度有关.从 20 世纪 70 年代后期,出现了许多关于软件规模和复杂度的度量,其中最著名的是McCabe Cyclomatic Complexity复杂度度量元[9],简称CC.CC用两种方法来度量软件模块中控制流的复杂度,一种是通过图理论中的节点和边的数量来计算($CC=e-n+2$,e代表边数,n代表节点数);另一种是通过决策路径数来计算($CC=bd+1$,bd代表控制图中 2 项决策数;如果控制图中有n条决策路径,该控制图就有$n-1$个 2 项决策数);两种计算方式得到的CC数据是一致的.Aivosto公司的经验研究[10]给出了CC与误修复概率(bad fix probability)的关系:CC(1~10),5%;CC(20~30),20%;CC(>50),40%;CC(接近 100),60%;该研究表明,当CC在 1~10 时,修复缺陷可能引入新缺陷的概率是 5%,随着复杂度的提高,这个概率也越高,当CC接近 100 时,修复缺陷引入新缺陷的概率高达 60%.

1.1.3 基于多维软件度量元的缺陷预测技术

20 世纪 90 年代,随着软件规模和复杂度的不断增长,出现了面向对象的软件开发技术,从而诞生了许多面

向对象的度量元来衡量软件规模和复杂度等.对于软件模块复杂度的度量也从原来的模块内部度量发展到模块外部度量,如内聚性、耦合度、继承深度等.缺陷预测开始关注更多的度量元,如OO度量元、Web类度量元,或者是产品、过程、技术等度量元,称为基于度量元(metrics-based)的缺陷预测技术[11−13].特别是随着软件过程技术的发展,人们逐渐认识到软件过程对于质量的影响,基于过程的缺陷度量成为人们关注的焦点,如:CMM等级与遗留缺陷密度关系的经验结果[14],采用净室(cleanroom)方法得到的遗留缺陷密度参考数据[15]等.

这一类的技术往往强调预测结果的准确性与实际情况的符合.尽管缺陷预测技术发展了 30 多年,但是预测结果与实际情况相比远没有达到人们期望的那样令人满意.早期的模型虽然简单、易用,但是由于不同的组织特点和过程成熟度、人员经验以及技术类型,建立的缺陷与规模、复杂度的关系式都不一样,无法将另一组织的预测模型及结果拿来简单地对照使用.所以,从 20 世纪 90 年代开始,对缺陷的预测也从仅仅关注缺陷数量的估计,发展到关注缺陷的分布、移除、遗留等问题,以求对软件质量的理解更加准确.

1.2 缺陷分布预测技术

Kitchenha,Basili,Khoshgoftaar等人通过采集缺陷与软件产品度量(software product metrics)、软件技术(OO,Web)、软件过程和执行程序等关联的度量数据,采用分类或回归技术研究缺陷的分布问题.研究结果表明,缺陷在软件模块中的分布符合 2-8 原则,要识别或者预测这些包含大多数缺陷的少量模块,就需要分类或者回归技术来进行判定.分类技术和回归技术都属于学习问题[16],在学习问题上常常将分类技术划分到解决模式识别问题上,即用于估计指示函数集合(指示函数即只有 0 或 1 两种取值的函数);而回归技术则用于解决回归函数估计问题,即用于估计实函数.具体到缺陷预测方面,分类技术通常用于识别模块是不是高缺陷率或低缺陷率的,而回归技术更多地用于估算缺陷数量.所以说两类技术都可以用来指导测试和其他质量保证活动的计划及执行,节省昂贵的测试及评审的时间和成本.

常见的分类技术(也常被称为质量分类技术)有:线性判别分析(linear discriminant analysis,简称 LDA),布尔判别函数(Boolean discriminant function,简称 BDF),分类回归树(classification and regression tree,简称 CART),优化集精简方法(optimized set reduce,简称 OSR),聚类分析(clustering analysis,简称 CA),支持向量机(support vector machine,简称 SVM)等,简单描述如下:

LDA也称作费舍尔(Fisher)判别方法.LDA的优点是易于实现,计算复杂度低,它的学习过程是根据训练数据估计高斯分布参数的过程,其假设是预测对象的每个分类都服从具有相同协方差矩阵的多元高斯分布;在缺陷预测领域中,LDA主要用于高缺陷风险的程序识别[17].

BDF类似于LDA,不同之处是,它针对每一个度量元设置了一个关键值(阈值),如果有任意一个度量元超过这个值,则将这个模块判定为高缺陷率模块,否则判定为低缺陷率模块;在训练过程中,BDF可以人为地调整I类错误和II类错误的比例;BDF的缺点在于判定低风险模块的条件过于苛刻,虽然很少产生II类错误,但I类错误率非常高,造成项目有限的资源浪费在大量错判为高风险的模块上,因此,Khoshgoftaar提出改进的BDF方法,在一定程度上放宽了对于低风险模块的判别条件[18].

CART是一个统计工具,它能够挖掘数据中的模式和关系,并自动进行分类;使用该工具时,需要进行训练,根据训练数据集构造决策二叉树;该树的所有非叶子节点均有两个子节点和一个判定准则,所有叶子节点均包含一个预测输出值;训练完成后,决策树的结构不再改变,当输入新的度量数据时,就能够按照事先确定的非叶子节点的判定准则决定转向哪一个子节点,直到叶子节点,也就是输出的预测值;由于分类树的思想容易被人理解,而且其决策判定准则是给定的门限值,因此可以用于变量间的非线性和非单调关系建模,在缺陷分布预测方面得到一定的应用,如Khoshgoftaar等人将其用在通信软件中预测使用时缺陷,以及分类树中I类错误和II类错误率的优化分析等[19].

OSR与CART方法类似,也是使用分类回归树作为预测模型,其主要思想是通过对训练集进行适当的裁剪来获得更好的分类回归树结构;与CART使用所有的训练集数据相比,OSR主要集中于训练集的裁剪[20].

CA的目标是将输入的数据按照其自身的特点分为几个类别.在缺陷预测领域中使用的聚类分析算法主要有两种,*K*-均值聚类和Neural-Gas聚类.首先利用聚类分析将软件模块按照预先设定的分类数进行自动分类,然

后由度量人员手工将这些类标定为高缺陷率或低缺陷率[21].

SVM是一种通用的前馈神经网络,由Vapnik在 1992 年首先提出,可以用于模式分类和非线性回归[16];SVM的学习算法来自于统计学理论,其核心是经验风险最小化原则;在 2005 年北京师范大学郭平的一篇论文[22]中,给出了基于SVM的软件质量预测,主要依据模块复杂度进行软件缺陷分类.

一些回归技术也用来预测缺陷和度量元之间的关系.由于不同度量元之间存在数据相关性等问题,通常使用主成分分析技术压缩和降低这些度量元的维度,选取正交的度量元来估算软件缺陷.常见的回归技术有:人工神经网络(artificial neural network,简称 ANN)、逻辑回归(logistic regression,简称 LR)、多元线性回归(multiple linear regression,简称 MLR)、基于用例推理(case-based reasoning,简称 CBR)等.以下是对这些回归技术的介绍:

ANN起源于对生物神经系统的研究和模仿,目前,在软件缺陷预测领域使用的神经网络主要是基于误差反向传播算法(error back propagation,简称BP)或者该算法的某个分支/变种[23];BP算法的基本思想是将训练集的输入数据放入ANN进行运算,将运算结果与期望的输出进行比较,根据两者的误差反向调整ANN的权值等参数.

LR是对线性回归的进一步改善,针对线性回归难以解决分类时出现的 0~1 之外的非法概率值,采用对数变换,其结果值将不再局限于 0~1 的区间,而是负无穷大和正无穷大之间的任意值;与线性回归一样,需要找出能与训练数据匹配得较好的权值,这里使用模型的对数似然最大化的权值来测量匹配的良好程度,这在软件缺陷分类技术中应用得非常广泛[24].

MLR是一种统计学方法,用于估计或者预测自变量为一系列已知变量的函数;其训练过程也就是确定模型参数的过程,为了提高回归分析的适应能力,通常会对自变量集合进行优化调整,首先决定选用或者不选用某些自变量,然后按照最小平方的原则估计参数;Khoshgoftaar在 1992 年时提出用相对最小平方和平均相对误差作为误差函数进行线性回归建立缺陷预测模型[25],并针对两个已完成的实际项目数据进行验证,其结果表明,应用这种改进的回归方法在预测性能上和数据拟合方面都比原来有所提高,但却未能验证该模型在新项目中的预测结果.

CBR是通过获取过去相似的事例来解决预测问题;其预测模型是由训练数据和一些相关的参数如相似度函数组成,欧几里德距离、绝对差都可以作为相似度函数;使用CBR进行缺陷预测,其优点主要是使用的度量元可多可少,依据数据收集情况而定;而且,该方法是依据人们的经验和直觉来判断,易于理解,应用比较方便;在CBR方法中,还可以集成半数投票(majority voting)和数据聚类(data clustering)的规则来更好地预测缺陷,Khoshgoftaar在 2003 年就缺陷估算给出了Analogy-Based 实际分类规则的CBR方法[26].

对于上述技术,我们通常用I类和II类错误率来进行评价,相关的概念和计算方法见表 1[27].I类错误就是指将原本没有缺陷的模块划分到高缺陷率的模块当中;II类错误就是指将原本有缺陷的模块划分到低缺陷率的模块当中.I类和II类错误率所带来的成本是不一样的,就I类错误来说,耗费了测试和评审资源却对没有缺陷的模块进行大量的重点测试和评审;而II类错误缺乏对高缺陷率的模块进行必要的测试和评审,从而造成维护和返工工作量的增加.

Table 1 Type I and Type II misclassification

表 1 I 类和 II 类错误率

		Module actually has defects	
		No	Yes
Classifier predicts no defects	No	a	b
Classifier predicts some defects	Yes	c	d
		Accuracy=$(a+d)/(a+b+c+d)$	
		Probability of detection=$d/(b+d)$	
		Probability of false alarm=$c/(a+c)$	
		Type I error of misclassification=$c/(a+c)$	
		Type II error of misclassification=$b/(b+d)$	

目前,对于各类预测技术的评价和比较分析还很少,NASA[28]专门提供了一些缺陷数据以方便人们使用不

同的模型和方法进行训练和预测,用于各类模型和技术在缺陷分类技术上的比较.如Zhong等人[21]就引用了NASA 上提供的数据以及近 20 种预测方法的预测性能.Khoshgoftaar在 2004 年的两篇经验研究文章中[29,30]对7类分类技术和6类回归技术进行了比较和评价:由于不同的分类方法产生不同的I类和II类错误率,单纯凭借某类错误率来评价不同的技术是难以得到大家认同的,需要在两类错误类之间取得平衡;如果按照I类和II类错误率的总计来评价这几类预测技术也存在一定的问题,因为I类和II类错误的成本是不同的,所以Khoshgoftaar按照经验值给出了I类和II类错误的成本对比,使用ECM(expected cost of misclassification)的概念将成本因素考虑进去来对这 7 类技术进行评价;对于回归技术,Khoshgoftaar给出了这些技术在相同测试数据上其AAE(average absolute errors)和ARE(average relative errors)的对比结果,从而得到了这些回归技术相对优劣比较分析的经验结果.

在实际使用过程中,还需要考虑更多的因素,如不同分类和回归技术的适用性、它们对于训练数据的要求程度、预测精度要求、系统的可靠性和安全要求(要求高的系统尽量选择 II 类错误低的技术)、资源消耗、算法的效率以及各类技术其验证是否简单、可理解性方面、稳定性方面等等.表 2 给出了上面介绍的分类及回归技术在缺陷预测方面的比较.

Table 2 Comparison of the classification and regression methods
表 2 常用分类及回归方法比较

Methods	Accuracy	Type I vs. II error rate	Required resources	Others
LDA	Low	Unchangeable	Low	Only be applicable to the classification
BDF	Low	Changeable	Low	High of Type I error rate; Only be applicable to the classification
CART	High	Changeable	Medium	Need the PCA to decrease dimensionality; Easy to over fitting
OSR	High	Changeable	Medium	Similar with which of the CART
CA	High	Unchangeable	High	Need the personnel for the analysis; Low of the repeatability; High cost of prediction and easy to make mistaken
SVM	High	Unchangeable	High	Good model at learning and generalizing; Lack of effective methods to determine the best parameters
ANN	High	Unchangeable	Medium	Low speed of the convergence during the modeling
MLR	Low	Unchangeable	Low	Need the PCA to decrease dimensionality and model selection (stepwise selection, forward elimination, etc.)
LR	High	Changeable	High	Could be combined with other methods and improved in the classification
CBR	High	Unchangeable	High	Can take advantage of availability of new or revised information

1.3 缺陷预测模型技术

前面介绍的两类技术都是基于一些选取的度量元,通过经验、分类回归等方法找到缺陷和主度量元之间的关系,以通过采集这些度量元的数据来预测软件的缺陷和缺陷的分布.但事实上,引起软件缺陷的因素非常多,任何单一或者回归后简化的预测算法都不可避免地会理想化约束缺陷产生的环境,从而导致缺陷估计上的不准确.此外,随着人们开发软件工作量的增大,人为原因造成缺陷的增加或者减少,比如需求、编码等工作会引入更多的缺陷,而评审、测试工作会减少缺陷数量等,但总的来说,假定过程、人的能力技术等因素是稳定的,则缺陷与软件规模是一个正比变化关系.一些模型技术开始用来建立较为复杂和综合的软件缺陷预测模型,旨在通过比较全面的缺陷影响因素,基于历史和当前的缺陷数据预测软件的缺陷.这些模型主要考察影响缺陷引入和排除的因素,建立缺陷随这些因素变化而受影响的关系.一方面指导人们采取措施减少缺陷的引入,另一方面增加缺陷排除的能力.

1.3.1 COQUALMO

美国南加州大学的Boehm等人[31−33]认为,缺陷是随着软件产品的开发而引入进来,又伴随着此过程逐渐被移除出去的.他们认为缺陷和软件规模有着直接的关系,此外,还有一些其他因素对软件缺陷产生影响.

1997 年,Boehm 在其经典的软件成本估算模型(constructive cost model,简称 COCOMO)的基础上提出软件质量估算模型 COQUALMO(constructive quality model),如图 2 所示.

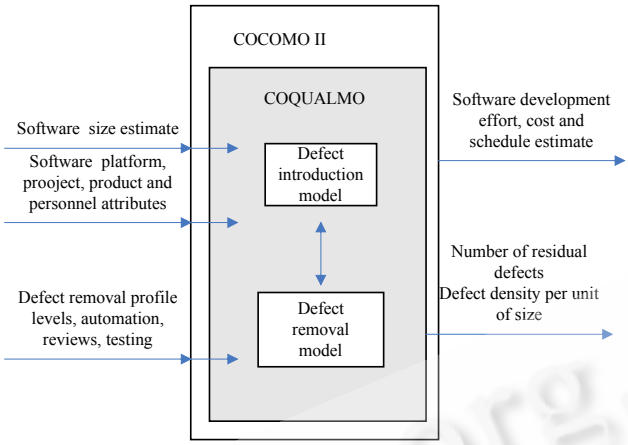

Fig.2 COQUALMO model

图 2 COQUALMO 模型

COQUALMO 包括两个子模型,分别为缺陷引入模型(defect introduced model,简称 DI)和缺陷移除模型(defect removal model,简称 DR).DI 模型有 23 个可裁剪的成本驱动参数,以软件规模为输入,估算软件在需求、设计、编码、文档等过程引入的缺陷.DR 模型则考虑软件开发过程中采用的自动分析技术、评审技术、测试技术等参数,估算可能排除的缺陷,最后得到交付时的缺陷率.

1.3.2 DRE 模型

缺陷移除矩阵(defect removal matrix)是业界流行的一种缺陷模型方法,最早出现在IBM公司的内部技术报告中[34];该方法统计软件生命周期各个阶段引入和移除的缺陷,以计算整个系统的缺陷移除率DRE(defect removal efficiency),见表 3 中一组数据.

Table 3 DRE model used in practice

表 3 DRE 模型实例数据

Defect removal step	Defect injection phase				Total
	Requirements	High-Level design	Low-Level design	Coding	
Requirement analysis and review	13				13
High-Level design inspections	2	8			10
Low-Level design inspections	2	3	7		12
Code inspections and testing	2	2	1	18	23
Customer detected	1	1	1	2	5
Total	20	4	9	20	63

整个系统的缺陷移除效率=(13+10+12+23)/63=92%.

各个阶段的缺陷移除效率:需求阶段 DRE=13/(13+2+2+2+1)=0.65;概要设计阶段 DRE=10/((2+2+2+1)+(8+3+2+1))=0.33;详细设计阶段 DRE=12/((2+2+1)+(3+2+1)+(7+1+1))=0.6;代码评审与测试阶段 DRE=23/((2+1)+(2+1)+(1+1) +(18+2))=0.82;用户发现 DRE=5/(1+1+1+2)=1.00.

从以上的缺陷移除效率可以看出,对这组数据样本而言,设计移除缺陷的效率最低,测试在缺陷移除过程中最有效.这样就可以有针对性地改进过程,提高软件开发过程中缺陷的移除效率.该方法对于软件过程的控制和改进非常有效,业界通常称为DRE模型.最早的缺陷移除效率这一概念出自IBM公司,后来Jones作了进一步的改进[35],Remus和Zilles在 1979 年ICSE上详细给出了DRE模型的计算方法、过程和应用[36].

DRE模型常与正交缺陷分类(orthogonal defect classification,简称ODC[37])方法结合使用.ODC方法提供了一个从缺陷中提取关键信息的测量范例,用于评价软件开发过程,提出过程改进方案.ODC分类方法主要包括缺陷的 8 个属性特征:发现缺陷的活动、缺陷影响、缺陷引发事件、缺陷载体、缺陷年龄、缺陷来源、缺陷类型和缺陷限定词.该方法覆盖的语义信息包括缺陷的位置、缺陷产生的征兆、最终影响以及缺陷的形成机制等.湖南大学也在软件正交缺陷分类方面做了一些研究[38].

1.3.3 Bayesian 模型

Fenton等人于 2004 年利用贝叶斯网络(Bayesian network)技术,初步提出了遗留缺陷预测模型,并于 2007 年提出了在不同的生命周期模型中适用的软件缺陷预测模型[39].该模型根据缺陷引入和移除过程建立对遗留缺陷的估算,该模型在缺陷引入和排除因子中根据因果关系建立了它们与缺陷之间的Bayesian网络,利用Bayesian技术的前向和后向推理,一方面进行遗留缺陷的估算,另一方面根据估算结果与实际值的比较不断地校准模型,以逐步提高估算的准确度.该模型的核心结构如图 3 所示.

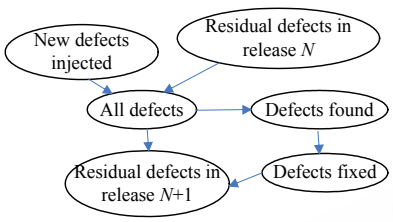

Fig.3 Bayesian model

图 3 Bayesian 模型

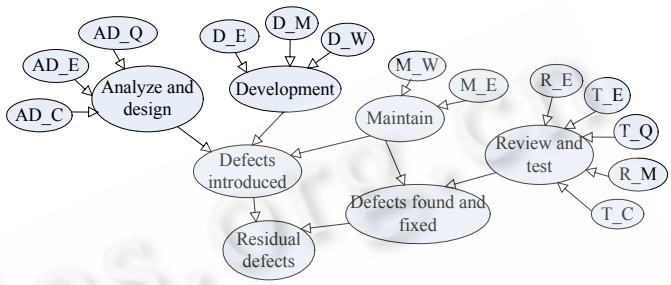

Fig.4 Bayesian model for predicting residual defects

图 4 可预测遗留缺陷的 Bayesian 模型

与其他模型相比,Bayesian 技术对于解决不确定性和数据不完整方面存在一定的优势.

Bayesian模型刚提出来的时候,主要是基于Fenton所提出的一个最基本的因果关系模型:发现的缺陷(defects detected)取决于测试有效性(test effectiveness)和当前存在的缺陷(defects present).随着Fenton等人对Bayesian模型的不断改进,目前已结合项目数据和定性因子给出了Bayesian模型中缺陷引入、发现、修复及遗留过程所受到一系列因子的影响程度,并且给出了Bayesian模型在实际项目中预测效果的验证[40].

Bayesian 模型在缺陷预测方面具有很好的应用前景,但还存在着不足.比如,在如何选取与缺陷相关的一些因子以及这些因子对于缺陷的影响程度,都是一个经验性的判断,需要结合专家经验来建立因子对于缺陷的影响关系.另外,在因子过多的情况下,模型的结构比较复杂,容易出现状态空间爆炸的问题,虽然可以通过增加中间结点来解决,但是计算效率和推理过程也将变得非常缓慢而不易被人们所接受.

针对以上问题,在实际使用中应该结合项目的相似历史数据、经验数据和知识(如项目经理、质量成员、专家等在缺陷方面的经验知识)对影响缺陷引入和排除的驱动因子进行裁剪;目前的裁剪方法,可以采纳COQUALMO模型中的驱动因子,结合项目实际信息进行裁剪工作;也可以采纳Fenton等人[40]所提出的 5 类定性因子设计的 27 个定性描述和问答进行判断,并结合Bayesian模型中缺陷因果关系,给出裁剪后的Bayesian缺陷预测模型.如图 4 所示,中国科学院软件研究所(ISCAS)在Fenton等人研究的基础上提出了基于 13 个驱动因子的Bayesian模型,主要用于分析缺陷引入、缺陷发现和排除以及遗留缺陷过程,用于过程质量控制和软件发布时机的决策;并依靠专家经验、组织数据以及定性准则得到Bayesian模型的因子值[41].国内关于Bayesian技术研究软件缺陷的还有其他一些成果[42].

1.3.4 Capture-Recapture 模型

捕捉模型(capture-recapture model)来自于生物学家对于自然界中生物种群总数的估算.由于自然界中某类生物的总数只能通过估算得出,生物学家通过对该类生物的捕捉、标记、释放、再捕捉等过程,根据再捕捉中有标记生物与无标记生物的比例,估算出该类生物的总数.

在软件工程领域最早使用捕捉模型的是Mills[43],它根据测试过程中发现的缺陷数来估计总的缺陷.随着该模型在生物界的广泛应用和改进,在软件工程领域也发展了系列模型,根据Briand等人[44]的研究,目前有 4 类捕捉模型(Model M0,Mh,Mt,Mth),其分类主要是依据模型的假设.

在生物界中应用该模型的 3 个基本假设条件是:其一,捕捉的次数只允许 1 次;其二,封闭性,即在研究过程中没有生物离开或进入该种群;其三,捕捉能力,所有的生物在每次捕捉过程中被捉的几率均等.在软件工程界使

用该模型时,也应该满足上述3个基本假设条件.条件1和条件2容易满足,例如,通过加权平均的方式,可以对多次捕捉过程满足一次过程的要求,而对于封闭性的要求通过只对当前的捕捉过程时的总数进行估算即可满足,但是,对于条件 3,显然无法满足,因为有些缺陷确实比另外一些缺陷难以发现,这也是大家的普遍共识.

所以,4类捕捉模型虽然可以通过简单的转换满足两个基本条件,却难于满足第3个条件.依据对于第3个条件满足的不同情形,目前的研究可以归类到模型M0,Mh,Mt,Mth这4类,其假设条件分别是:模型M0假设所有的缺陷被发现的几率相等,而且所有的评审人员在发现缺陷的能力或效率方面均等;模型Mh比起M0,增加了多样性(heterogeneity)变量,即假设每个缺陷i具有自己的发现概率P_i,但是对于不同的评审人员来说,它们被发现的概率不变,都是P_i;模型Mt增加了对评审人员在缺陷发现能力方面的考虑,即评审人员j发现缺陷的概率为P_j,但是对于不同的评审人员来说,同一缺陷被发现的概率相等;模型Mth则认为每个缺陷都有自己被发现的概率P_i,而且每个评审人员也有自己发现缺陷的概率P_j,这样,评审人员j发现缺陷i的概率为$PiPj$.于是我们可以看出,模型M0是其他3个模型的特例,模型Mh和模型Mt是模型Mth的特例.

当然,使用捕捉模型在软件工程界与生物界还是有很大差别,最主要的就是在生物界估算的种群数往往很大,不需要精确的估算结果,而在软件缺陷估算过程中,总缺陷数的数量级远没有生物界的高,因此,估算结果的准确性相对来说不高,还需要进一步的研究和改进,如北京航空航天大学提出的改进的软件缺陷预测方法[45].

2 动态基于时间关系的软件缺陷预测技术

另外一类重要的缺陷预测技术是基于时间关系的.很多动态的可靠性模型都是基于这类缺陷预测技术.

这类研究主要是基于经验研究和统计技术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,最著名的有 Rayleigh 分布模型、指数分布模型和 S 曲线分布模型(exponential and s-curves arrival distribution models).

2.1 Rayleigh分布模型

Rayleigh模型是一个可以对软件开发全生命周期的缺陷分布进行预测的模型,是一种最常用的可靠性模型[46].Rayleigh模型基于Weibull[47]统计分布,Weibull分布是一个广泛用于不同领域可靠性分析的统计分布族,有大量的统计数据支持.Weibull分布的一个重要特征是其概率密度函数的尾部逐渐逼近 0,但永远达不到 0.1982 年,Trachtenberg[48]观测一组软件项目每个月的缺陷数据,发现这些项目的综合缺陷模式符合Reyleigh曲线.1984年,IBM联邦系统部(Federal Systems Division)的Gaffney[49]报告,在IBM使用的6个公共缺陷检测阶段(高级设计检查、低级设计检查、代码审查、单元测试、集成测试、系统测试)所发现的缺陷随这些阶段在软件生命周期的时间分布也符合Rayleigh曲线.

Rayleigh模型的概率分布密度函数 $f(t)=2K(t/c^2)\mathrm{e}^{-(t/c)^2}$, 累积分布函数 $F(t)=K(1-\mathrm{e}^{-(t/c)^2})$,其中$t$是时间,$c$是一个常量($c=\sqrt{2}t_m$,t_m是$f(t)$达到峰值对应的时间),K是曲线与坐标形成的面积(总缺陷数).可以得到缺陷在t_m时间的比率($F(t_m)/K$)约等于 0.4,即在$f(t)$达到最大值时,已出现的缺陷数大约占总缺陷的 40%,因此,按照这个推导,在某一时间就可以估算出总的缺陷数以及具体的Rayleigh分布参数,从而将缺陷的计算过程简化.

例如,有表4所示的一组缺陷数据.

Table 4 Defects distribution in schedule

表 4 缺陷数据随时间分布

Weeks	1	2	3	4	5	6	7	8
Defects	69	333	316	62	25	10	23	7

从以上数据我们可以得出在第 2 周发现的缺陷数最大,则按照上面的简化计算过程,截止到第 2 周出现的缺陷总数大约占全部缺陷总数的 40%,则总缺陷数K=(前 2 周的缺陷总数)/0.4=(69+333)/0.4=1005,则t_m=2,由此可以得到分布函数 $f(t)=2K(t/c^2)\mathrm{e}^{-(t/c)^2}=1005(t/4)\mathrm{e}^{-t^2/8}=251.25t\mathrm{e}^{-t^2/8}$, $F(t)=1005(1-\mathrm{e}^{-t^2/8})$,如图 5 所示.

使用 Rayleigh 模型还可以对组织过程的质量性能目标进行控制,按照 Putnam 和 Myers 在文献[50]中的公

式: $f(t)=(6K/(T_d^2))te^{-3(t/T_d)^2}$ (由$F(T_d)$/=0.95 计算出具体的常量c,再把c代入$f(t)$中就能得到所期望的时间间隔t与所发现的缺陷关系;Putnam和Myers由此预测到了在里程碑 4-系统集成测试到里程碑 7-软件发布所发现的缺陷约占总缺陷的 17%),其中T_d是项目的整个周期;该周期内大概 95%的总缺陷被发现,而K是整个寿命周期中期望的总缺陷数.对于成熟的组织,当项目周期、软件规模和缺陷密度确定时,可以得到确定的缺陷分布曲线,并可以据此对项目过程的缺陷率加以控制.

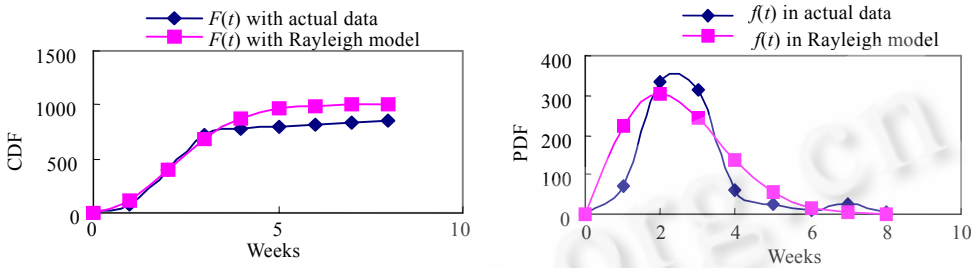

Fig.5 Rayleigh model-CDF (left) and PDF (right)

图 5 Rayleigh 模型的 CDF(左)和 PDF(右)

例如,已知组织开发的历史项目中的缺陷密度为每千行代码 10.53 个缺陷,预测的新项目的总代码行数为100KLOC,要求新项目与历史项目相比期望达到约减少 5%的缺陷,那么,该项目期望的总缺陷数为10.53×100×(1−5%)=1000,假如项目估算的周期T_d为 26 周,那么将数据代入到上面的Rayleigh模型的$f(t)$公式就可以得到每周应发现缺陷数的时间分布,如图 6 所示,当项目开发过程中实际发现的缺陷和分布曲线预计的缺陷出现重大偏差时,说明该过程出现异常,需要采取相应的纠正措施.

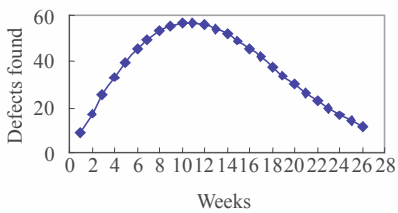

Fig.6 Defects distribution prediction

图 6 用于缺陷控制的分布预测

2.2 指数分布模型

指数模型是针对测试阶段,尤其是验收类测试阶段的缺陷分布.其基本原理是在这个阶段出现的缺陷或者失效模式,是整个产品可靠性的良好指征.指数模型也称为可靠性增长模型,分为故障/失效计数模型(fault/failure count model)和失效间隔时间模型(time between failures model).指数模型的缺陷概率分布密度函数(probability density function,简称 PDF)为 $f(t)=K(\lambda e^{-\lambda t})$,累积缺陷分布函数(cumulative distribution function,简称 CDF)为 $F(t)=K(1-e^{-\lambda t})$.其中 t 为时间,K 为总缺陷数,λ为缺陷探测率或者失效出现率,也称为尺度参数或形状参数.指数模型是可靠性研究中最简单也是最重要的一种模型,它是许多其他可靠性增长模型的基础.

Jeinski-Moranda(J-M)模型是最早的失效间隔时间模型之一[51],它假设在测试阶段发现的每个故障导致失效的影响是均等的.此外,它还假设故障的修复时间可以忽略不计,并且每个故障都可以完美地被修复.因此,每次故障的修复都可以均等地改进软件产品的失效率,失效率(λ)函数为$Z(t_i)=\phi[N-(i-1)]$,其中N是测试开始时的缺陷数,ϕ是比例常量(proportionality).所以在两次相邻失效之间,失效率函数是一个常量,但ϕ随着故障的排除而递减,两次失效的间隔会越来越长.

类似的模型还有Littlewood(LW)模型[52],其假设不同的故障对失效的影响不同,早期大量的故障被排除,平均出错规模(error size)会越来越低.Littlewood后续还提出了一些其他模型,如非齐次泊松过程模型(littlewood nonhomogeneous Poisson process model,简称LNHPP)[53],LNHPP类似于LW,但他假设失效率连续变化,而不是

LW模型的离散下降.

1979 年,Goel和Okumoto[54]提出了一个在测试区间的失效次数模型(GO model),他们假设在时间t_i,累计观测到的失效次数为$N(t_i)$是一个非齐次泊松过程(nonhomogeneous Poisson process model,简称NHPP),有时间依赖的失效率,并满足指数分布.

2.3 S曲线分布模型

1983 年,Yamada等人[55]提出测试过程不仅仅是发现缺陷,还包括缺陷的隔离.当发现失效的时候,我们需要时间去分析失效原因,所以从第 1 次观测到失效到报告这个失效有着不可忽略的延迟时间,随时间观测到的延迟缺陷累计数符合S曲线分布,这种情况也称为延迟(delay) S模型,也是一种可靠性增长模型.S曲线模型满足非齐次泊松过程,其 CDF: $F(t)=K(1-(1+\lambda t)\mathrm{e}^{-\lambda t})$,其中:$t$ 为时间,K 为总缺陷数或者累计缺陷率,λ为缺陷探测率;其 PDF: $f(t)=K\lambda^2 t\mathrm{e}^{-\lambda t}$.

1984 年,Ohba[56]提出另一种S分布模型,称为变形(inflection)S模型.该模型认为发现缺陷之间是相互依赖的,发现的失效越多,就有越多还没有发现的失效变得可探测.这个模型讨论了缺陷之间的相互关系,更贴近于实际情况,是早期众多模型的重大改进;其 CDF: $F(t)=K(1-\mathrm{e}^{-\lambda t})/(1+\phi\mathrm{e}^{-\lambda t})$,其中:$t$ 为时间,K 为总缺陷数或者累计缺陷率, λ为缺陷探测率;其 PDF: $f(t)=K\lambda\mathrm{e}^{-\lambda t}(1+\phi)/(1+\phi\mathrm{e}^{-\lambda t})^2$.

图 7 是指数模型、延迟 S 曲线及变形 S 曲线模型的 CDF 和 PDF 比较图.

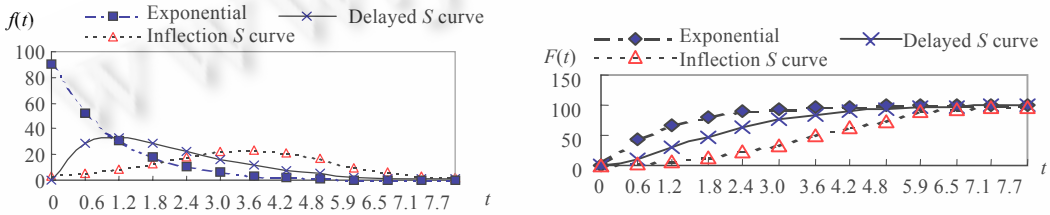

Fig.7 Exponential, delayed S, and inflection S models-CDF (left) and PDF (right)

图 7 指数模型,延迟 S 曲线和变形 S 曲线模型的 CDF(左)和 PDF(右)

软件可靠性原理上也是一种缺陷预测技术,包括动态和静态的预测技术,主要研究软件在约定环境下运行不失效的概率.国内在软件可靠性方面的研究成果颇多,如哈尔滨工业大学在软件可靠性增长模型方面的研究[57,58],上海交通大学在软件缺陷及软件可靠性技术方面的探讨以及缺陷管理和预防过程及方法[59,60],北京交通大学有关软件缺陷的因果分析[61]等,中国科学院软件研究所关于缺陷度量及过程管理方面的研究[62,63]都很好地推动了中国在这一领域的研究.在过去的 20 多年里,软件可靠性模型是软件工程领域最活跃的方向之一,大约有 100 多种模型发表在各种专业刊物和学术会议上,每种模型也都有一定的假设前提、应用范围和限制.但是,大多数模型并没有得到实际数据的验证,原因是多方面的,主要包括 1) 收集数据可能很昂贵;2) 模型本身可能难以理解;3) 一些模型虽然验证简单但又可能不能很好地工作等等.所以,至今也只有少数模型得到应用.

3 评价及经验研究

缺陷预测技术经历了长期的发展并持续保持活跃.各种预测技术有着不同的适应面,解决不同的问题,也有不同的局限.具体的客户需求、环境、产品特征、组织的历史数据等情况不同,适用的缺陷预测技术也不同.表 5 总结了一些典型技术的优缺点和使用条件.

软件缺陷预测技术的各种方法之间并不是孤立的,如在静态缺陷预测技术中,Metric-Based 方法是最基本的,它来自于对软件度量的基本信息,缺陷分布以及缺陷模型都要基于这些基本缺陷度量.动态预测技术中上百种预测模型也是彼此有关联和继承的.

此外,随着软件开发技术的发展,现有的预测技术也面临新的挑战,以Web类应用软件为例,与传统的软件类型相比,此类软件的度量还没有统一的标准[64].另外,与缺陷预测密切相关的软件可信性研究也在国内迅速开展

起来[65,66],这也要求软件缺陷预测技术能够很好地为可信性研究提供参考.

Table 5 Comparisons of software defect prediction technologies

表 5 软件缺陷预测技术比较

Defect prediction technology			Usage scene	Strengths	Weaknesses
Dynamic distribution technology	Rayleigh model		As early as possible in the life cycle when the defect data can be collected; Need historical defect density data in addition to the estimated and actual software size, and consistently tracked defect counts.	The defect density by time period can be predicted, which is enabling the estimation of defects to be found and validated during the testing.	No insight or adjustment mechanism for changes in the product, personnel, platform, or project, which may affect the prediction of defects.
	Exponential model		The model is based on the data from the formal testing phases, so it is applicable mainly to these phases.	One of the simplest and most important models, easy to use and quick to implement.	Testing effort is homogeneous throughout the testing phase
	S-Curve model		Testing process consists of both the detection and isolation; Compared with other models, it considers the delay between the observation and reporting.	Apply to the larger software projects with a fairly large number of defects; More effective when testing effort increases or decreases throughout the test period	More complicated than the above two models; it is difficult to use across the projects
Static technology	Classification	LDA, BDF, etc	Fault-Prone module classification and prediction; apply to make planning and testing.	Focus on the fault-prone modules (classes, files etc) and save the testing and review resources.	Difficult to balance between the Type I and Type II misclassification
	Regression	MLR, CBR, etc			
	Metric-Based technology	Size-Based	Make testing plan; Need the data of the size, complexity, coupling, historical defect data, etc	Efficient and effective focus of defect detection activities	Defect density by modules may not predict operational failure density; effort may be misdirected; models and assumptions need calibrated to be hold across the different systems.
		Complexity-Based			
		Process metric-based			
		OO metric-based			
		Web metric-based			
	Model-Based	COQUALMO	Used in the planning phase; required size of the product and ratings for 21 quality adjustment factors.	Quantifies the effect of different discovery techniques on the detection and removal of defects. Considers the effects of attributes belong to the product, personnel, project, and platform.	Covers a small number of phases in the life of cycle; Without consider the factors used for the prediction of the test or post deployment defect.
		DRE	Suitable to the stable life of cycle and the process model.	Easy to understand and use; modeling the process of the defect injection and removal in addition to the effectiveness of the phases.	Need reliable defect classification and matrix approach mapping to the defect origin and found; Delay of the time between origin and found affect the effectiveness of the model.
		Bayesian	Be lack in the defect data and need risk analysis and decision-making support.	Defect probability distribution results; Causal analysis and reasoning, the trade-offs.	Need expert experience and judgment.
		Capture-Recapture	Determining whether a software product should undergo re-inspection; Required detailed defect descriptions from each inspector	More robust when simply used to predict whether a criterion for re-inspection has been exceeded	Need the stringent assumptions; Relaxing assumptions requires more complicated estimation.

4 结 论

缺陷预测技术的主要目的是预测软件还存留的缺陷,它根据软件的基本属性(规模、复杂度、开发方法、过程等等)、软件已经发现的缺陷来预测软件可能还遗留的、尚未发现的缺陷.缺陷预测技术可以帮助我们清楚并客观地了解软件的质量状态,软件是否可以交付使用,甚至软件使用过程中的失效模式.

软件缺陷预测技术从 20 世纪 70 年代开始,半个世纪以来,伴随并见证了软件工程发展的历史,也是软件工程学科中一个重要的组成部分.已有的软件缺陷预测技术还未解决的问题主要有:

1) 简单度量元技术是否逊于复杂度量元技术.传统的简单度量元技术基于规模度量(如Size,CC等)来估算和预测软件缺陷,随着软件开发技术从传统的结构化设计技术过渡到面向对象技术(OO等)、过程技术(XP,Cleanroom等),新的技术所带来的度量元的变化(如产品度量元、执行度量元、过程度量元等体系的建立)是否能够在原来的预测技术上得到改进,这还没有确凿的证据,甚至有研究表明,使用简单度量元技术(Size或CC)甚至获得了比使用复杂度量元技术还要好的结果[55].

2) 经验研究中的结果是否能够推广到其他项目及使用环境中.目前,关于缺陷预测方面的研究大都是经验性的,预测技术受限于所使用的数据以及项目的具体情况.除了 COQUALMO 模型和 Bayesian 方法考虑到了不同项目及环境因素对缺陷的影响以外,其他方法和技术都受限于组织中的数据,难以跨平台推广和使用.

3) 分类和回归技术受限于数据自身的问题.分类和回归技术都需要历史数据来构建模型和校验,各技术间的优劣取决于数据自身以及它们在简单性、可理解性、稳定性方面的比较和分析.目前,这些技术在缺陷预测方面的研究还只是给出经验结果,主要表现在:构造缺陷预测的自变量的选择还是经验性的,有选择产品度量元的,有选择开发过程的,也有部署和使用阶段的度量元等等.对各类技术优劣性的比较研究还很少,多在分类技术的比较,而回归技术的比较研究较少,分类和回归技术用于部署和使用阶段的缺陷预测方面的研究更少由于这些技术还有待改进和发展,因此,也存在着发展的广阔前景.

4) 模型技术在考虑产品、项目、人员及平台方面是否足够或过于考虑这 4 个方面.已有的模型技术有些考虑到了产品、项目、人员及平台方面对缺陷预测的影响,如 COQUALMO 模型,该模型通过对上述 4 个方面的考虑,能够很好地将软件缺陷引入、排除以及缺陷修复成本等诸多因素考虑在一个大的框架中,并综合寿命周期的不同阶段来得出包括缺陷密度、修复成本在内的一系列估算结果;但是,这一框架是否完备还有待进一步研究,所考虑的因素是否过多还难以界定,尽管这一模型在业界已经有了一定的应用,其他模型更是缺乏广泛的应用和验证.

随着软件给整个社会带来的巨大冲击,软件工程学科中的主要技术,包括缺陷预测技术不得不时刻面临新的挑战,以不断的创新去适应不断变化的需求.21 世纪,缺陷预测技术面临的挑战主要有:

1) 适应全球 24 小时不间断软件服务模式的需求.2005 年,Friedman 在其“The World is Flat”一书中说到:“低成本的光纤通信,使得全球 24 小时不间断的服务提供成为可能,全球变成了平坦的,同时也使得所有发达国家的经济都严重依赖软件.”这种软件产品模式的变化,要求软件的运行可靠性、友好性、可维护性更好,可信性更高.互联网环境下软件产品形态的变化,要求软件缺陷预测必须关注软件的部署环境、运行模式等对缺陷的影响,以及 24 小时不间断模式下软件的失效模型等等.正如开发北京 2008 年奥运会票务系统的服务商始料未及的场景:中国这一庞大用户群在特定环境下对票务系统的访问,直接导致该网络系统大面积拥堵并停机.

2) 适应快速市场进入的需求.在全球激烈的竞争下,快速和有预见性的市场进入已经成为竞争成败的关键.缺陷预测也成为评估和控制风险的重要依据.如何更准确地预测遗留缺陷、缺陷的种类、严重程度、分布以及基于市场时间的失效模型等等都有非常重要的意义.

3) 适应复杂软件系统的需求.越来越多的系统由软件来控制,软件日渐成为主要的成本和风险源,复杂软件系统对软件工程的各个分支都提出了新的挑战.其特征和对缺陷预测技术的挑战如下:

复杂性:大量的组织、接口、供应商、协调组等,以及软件开发过程、方法、支持工具的适应性、可伸缩性等要求,必然带来新的缺陷影响因素.

动态性:快速的变化影响分析、变更合成、协商、开发、确认、实现等技术,必然对软件质量产生新的影

响,一些新的度量参数,如变化数/月、过程的平均变化时间是缺陷预测必须关注的新问题.

突发性:很多需求无法预先说明,必然导致新的质量问题,不确定环境下的软件缺陷预测技术也将是一个新的挑战.

4) 适应软件价值经济的需要.传统的软件工程技术认为每一个需求、用例、对象、测试用例以及缺陷都是等价的,孤立地认为软件工程的任务就是把需求转换成验证的代码.但事实上,大部分事情都遵从经济上著名的 2-8 原则,亦即 20%的工作,创造 80%的效益,因而Boehm提出了价值软件工程的思想[67].李明树也提出了Triso-Model三维的软件工程环境,其中一个维度就是关注软件工程的价值经济[68].缺陷预测是估算和计划成本、确定软件交付、制定维护计划的重要依据,如何关注其间的价值因素,是软件缺陷预测技术必须应对的另一项挑战.

References:

[1] http://www.CeBASE.org

[2] Akiyama F. An example of software system debugging. In: Proc. of the Int'l Federation of Information Proc. Societies Congress. New York: Springer Science and Business Media, 1971. 353−359.

[3] Halstead MH. Elements of Software Science. New York: Elsevier, North-Holland, 1977.

[4] Ottenstein L. Predicting numbers of errors using software science. ACM SIGMETRICS Performance Evaluation Review, 1981, 10(1):157−167.

[5] Ottenstein L. Quantitative estimates of debugging requirements. IEEE Trans. on Software Engineering, 1979,SE-5(5):504−514.

[6] Lipow M. Number of faults per line of code. IEEE Trans. on Software Engineering, 1982,8(4):437−439.

[7] Takahashi M, Kamayachi Y. An empirical study of a model for program error prediction. IEEE Trans. on Software Engineering, 1989,15(1):82−86.

[8] Malayia Y, Denton J. Module size distribution and defect density. In: Proc. of the 11th Int'l Symp. on Software Reliability Engineering. New York: IEEE Computer Society Press, 2000. 62−71. http://portal.acm.org/citation.cfm?id=851024.856229

[9] McCabe TJ. A complexity measure. IEEE Trans. on Software Engineering, 1976,SE-2(4):347−354.

[10] http://www.aivosto.com/project/help/pm-complexity.html. 2007.

[11] Khoshgoftaar TM, Herzberg A, Seliya N. Resource oriented selection of rule-based classification models: An empirical case study. Software Quality Control, 2006,14(4):309−338.

[12] Li PL, Herbsleb J, Shaw M. Forecasting field defect rates using a combined time-based and metrics-based approach: A case study of OpenBSD. In: Proc. of the 16th Int'l Symp. on Software Reliability Engineering. 2005. 193−202. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1544734

[13] Basili V, Briand L, Walcelio L. A validation of object oriented design metrics as quality indicators. IEEE Trans. on Software Engineering, 1996,22(10):751−761.

[14] Diaz M, Sligo J. How software process improvement helped Motorola. IEEE Software, 1997,14(5):75−81.

[15] Dyer M. The Cleanroom Approach To Quality Software Development. New York: John Wiley & Sons, Inc., 1992.

[16] Vapnik V, Wrote; Zhang XG, Trans. The Nature of Statistical Learning Theory. 2nd ed., New York: Springer-Verlag, 1999 (in Chinese).

[17] Munson JC, Khoshgoftaar TM. The detection of fault-prone programs. IEEE Trans. on Software Engineering, 1992,18(5):423−433.

[18] Khoshgoftaar TM, Seliya N. Improving usefulness of software quality classification models based on boolean discriminant functions. In: Proc. of the 13th Int'l Symp. on Software Reliability Engineering. IEEE Computer Society Press, 2002. 221−230. http://portal.acm.org/citation.cfm?id=851033.856311

[19] Khoshgoftaar TM, Yuan X, Allen EB. Balancing misclassification rates in classification-tree models of software quality. Empirical Software Engineering, 2000,5(4):313−330.

[20] Briand LC, Basili V, Hetmanski CJ. Developing interpretable models with optimized set reduction for identifying high-risk software components. IEEE Trans. on Software Engineering, 1993,19(11):1028−1044.

[21] Zhong S, Khoshgoftaar TM, Seliya N. Analyzing software measurement data with clustering techniques. IEEE Intelligent Systems, 2004,19(2):20−27.

[22] Xing F, Guo P, Lyu MR. A novel method for early software quality prediction based on support vector machine. In: Proc. of the 16th IEEE Int'l Symp. on Software Reliability Engineering (ISSRE 2005). Chicago, 2005. 213−222. http://portal.acm.org/citation.cfm?id=1104997.1105250

[23] Khoshgoftaar TM, Lanning DL. A neural network approach for early detection of program modules having high risk in the maintenance phase. Journal of Systems and Software, 1995,29(1):85−91.

[24] Briand LC, Melo WL, Wüst J. Assessing the applicability of fault-proneness models across object-oriented software projects. IEEE Trans. on Software Engineering, 2002,28(7):706−720.

[25] Khoshgoftaar TM, Munson JC, Bhattacharya B, Richardson D. Predictive modeling techniques of software quality from software measures. IEEE Trans. on Software Engineering, 1992,18(11):979−987.

[26] Khoshgoftaar TM, Seliya N. Fault prediction modeling for software quality estimation: Comparing commonly used techniques. Empirical Software Engineering, 2003,8(3):255−283.

[27] http://mdp.ivv.nasa.gov/mdp_glossary.html

[28] http://mdp.ivv.nasa.gov

[29] Khoshgoftaar TM, Seliya N. Comparative assessment of software quality classification techniques: An empirical case study. Empirical Software Engineering, 2004,9(3):229−257.

[30] Khoshgoftaar TM, Seliya N. Fault prediction modeling for software quality estimation: Comparing commonly used techniques. Empirical Software Engineering, 2004,8(3):255−283.

[31] Chulani S. Bayesian analysis software cost and quality models [Ph.D. Thesis]. Los Angeles: University of Southern California, 1999.

[32] Chulani S. Results of Delphi for the defect introduction model, sub-model of the cost/quality model extension to COCOMO II. Technical Report, USC-CSE-97-504, 1997.

[33] Chulani S, Boehm B. Modeling software defect introduction and removal: COQUALMO (COnstructive QUALity MOdel). Technical Report, USC-CSE-99-510, 1999.

[34] Fagan ME. Design and code inspections to reduce errors in program development. IBM System Journal, 1976,15(3):182−211.

[35] Jones TC. Measuring programming quality and productivity. IBM System Journal, 1978,17(1):39−63.

[36] Remus H, Zilles S. Prediction and management of program quality. In: Proc. of the 4th Int'l Conf. on Software Engineering. Munich: IEEE Computer Society, 1979. 341−350. http://portal.acm.org/citation.cfm?id=800091.802957

[37] Chillarege R, Bhandari I, Jarik K, Michael J, Diane S, Bonnie K, Man-Yuen W. Orthogonal defect classification-a concept for in-process measurements. IEEE Trans. on Software Engineering, 1992,18(11):943−956.

[38] Chen L, Liu HH, Sheng C, Chen W. Combination of reliability growth model and orthogonal defect classification and application to qualitative analysis of process. Science Technology and Engineering, 2005,5(14):963−966 (in Chinese with English abstract).

[39] Fenton NE, Martain N, William M, Peter H, Lukrad R, Paul K. Predicting software defects in varying development lifecycles using bayesian nets. Information and Software Technology, 2007,49(1):32−43.

[40] Fenton NE, Martain N, William M, Peter H, Lukrad R, Paul K. Project data incorporating qualitative facts for improved software defect prediction. In: Proc. of the 3rd Int'l Workshop on Predictor Models in Software Engineering. Washington: IEEE Computer Society, 2007. 11−20. http://ieeexplore.ieee.org/xpls/abs_all.jsp?isnumber=4273254&arnumber=4273258&count=14&index=3

[41] Yang D, Wan YX, Tang ZN, Li MS. COCOMO-U: An extension of COCOMO II for cost estimation with uncertainty. In: Wang W, *et al*., eds. Software Process Change, SPW/ProSim 2006. LNCS 3966, Berlin, Heidelberg: Springer-Verlag, 2006. 132−141.

[42] Hu YP, Chen ZP, Lin YP, Li JY. Defect analysis model of Bayesian and its application in software testing. Computer Applications, 2005,25(4):808−810 (in Chinese with English abstract).

[43] Mills H. On the statistical validation of computer programs. Technical Report, FSC-72-6015, IBM Federal Systems Division, 1972.

[44] Briand L, Freimut, BG. A comprehensive evaluation of capture-recapture models for estimating software defect content. IEEE Trans. on Software Engineering, 2000,26(6):518−540.

[45] Zhu YC, Xu H. Empirical-Based software defect content estimation improvement. Journal of Beijing University of Aeronautics and Astronautics, 2003,29(10):947−950 (in Chinese with English abstract).

[46] Xie M. Software Reliability Modelling. Singapore: Word Scientific Publishing Co. Pte. Ltd., 1991.

[47] Lyu M. Handbook of Software Reliability Engineering. Singapore: McGraw-Hill, 1996.

[48] Trachtenberg M. Discovering how to ensure software reliability. RCA Engineer, 1982,27(1):53−57.

[49] Jr Gaffney JE. On predicting software related performance of large-scale systems. In: Proc. of the Int'l Conf. of the Computer Measurement Group, CMG XV. San Francisco, 1984. http://www.cmg.org/proceedings/1984.html

[50] Putnam LH, Myers W. Familiar metric management-reliability.1995. http://www.qsm.com

[51] Jelinski Z, Moranda P. Software reliability research. In: Freiberger W, ed. Statistical Computer Performance Evaluation. New York: Academic Press, 1972. 465−484.

[52] Littlewood B. Stochastic reliability growth: A model for fault removal in computer programs and hardware designs. IEEE Trans. on Reliability, 1981,R-30:313−320.

[53] Abdel-Ghaly AA, Chan PY, Littlewood B. Evaluation of competing software reliability predictions. IEEE Trans. on Software Engineering, 1986,12(9):950−967.

[54] Goel AL, Okumoto K. A time-dependent error-detection rate model for software reliability and other performance measures. IEEE Trans. on Reliability, 1979,R-28(3):206−211.

[55] Yamada S, OhbaM, Osaki S. S-Shaped reliability growth modeling for software error detection. IEEE Trans. on Reliability, 1983, R-32(5):475−478.
[56] Ohba M. Software reliability analysis models. IBM Journal of Research and Development, 1984,28(4):428−443.
[57] Liu HW, Yang XZ, Qu F, Zhao JH. Software reliability growth models of nonhomogenous Poisson process. Journal of Tongji University (Natural Science), 2004,32(8):1071−1074 (in Chinese with English abstract).
[58] Liu HW, Yang XZ, Yue XG, Qu F. A framework for NHPP software reliability growth models. Computer Engineering and Science, 2005,27(4):1−3 (in Chinese with English abstract).
[59] Jiang LT, Xu GZ. Software fault and software reliability. Computer Simulation, 2004,21(2):141−144 (in Chinese with English abstract).
[60] Zhang W, Cao J. Process and method of software defect prevention. Computer Engineering, 2004,30(Suppl.):23−25 (in Chinese with English abstract).
[61] Wang XY, Jiang MM, Huang T. The discussion of software defect causal analysis. Modern Computer, 2003,157(2):19−21 (in Chinese with English abstract).
[62] Wang Q, Jiang N, Gou L, Liu X, Li MS, Wang YJ. BSR: A statistic-based approach for establishing and refining software process performance baseline. In: Proc. of the 28th Int'l Conf. on Software Engineering. China, 2006. 585−594. http://portal.acm.org/citation.cfm?id=1134285.1134368#
[63] Wang Q, Li MS. Measuring and improving software process in China. In: Proc. of the 4th Int'l Symp. on Empirical Software Engineering. 2005. 183−192. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1541827
[64] Mendes N, Mosley N, Counsell S. Early Web size measures and effort prediction for Web costimation. In: Proc. of the 9th Int'l Software Metrics Symp. (METRICS 2003). 2003. 18−29. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1232452
[65] Chen HW, Wang J, Dong W. High confidence software engineering technologies. Acta Electronica Sinica, 2003,31(12A): 1933−1938 (in Chinese with English abstract).
[66] Guo SH, Lan YG, Jin MZ. Some issues about trusted components research. Computer Science, 2007,34(5):243−246 (in Chinese with English abstract).
[67] Boehm B. Value-Based software engineering-overview and agenda. Technical Report, USC-CSE 2005-504, 2005.
[68] Li M. Assessing 3-D integrated software development processes: a new benchmark. In: Wang W, *et al*., eds. Software Process Change, SPW/ProSim 2006. LNCS 3966, Berlin, Heidelberg: Springer-Verlag, 2006. 15−38.

附中文参考文献:

[16] Vapnik V,著;张学工,译.统计学习理论的本质.北京:清华大学出版社,2000.12−17.
[38] 陈莉,刘海红,盛昌,陈威.可靠性增长模型和正交缺陷分类的结合及在过程定性分析中的应用.科学技术与工程,2005,5(14):963−966.
[42] 胡玉鹏,陈治平,林亚平,李军义.贝叶斯缺陷分析模型及其在软件测试中的应用.计算机应用,2005,25(4):808−810.
[45] 朱永春,徐红.一种基于历史数据的软件缺陷预测方法改进.北京航空航天大学学报,2003,29(10):947−950.
[57] 刘宏伟,杨孝宗,曲峰,赵金华.非齐次泊松过程类软件可靠性增长模型.同济大学学报(自然科学版),2004,32(8):1071−1074.
[58] 刘宏伟,杨孝宗,岳晓光,曲峰.一个 NHPP 类软件可靠性增长模型框架.计算机工程与科学,2005,27(4):1−3.
[59] 蒋乐天,徐国治.软件缺陷及软件可靠性技术.计算机仿真,2004,21(2):141−144.
[60] 张文浩,曹健.软件缺陷预防过程与方法.计算机工程,2004,30(增刊):23−25.
[61] 王晓艳,蒋明清,黄涛.软件缺陷的因果分析.现代计算机,2003,157(2):19−21.
[65] 陈火旺,王戟,董威.高可信软件工程技术.电子学报,2003,31(12A):1933−1938.
[66] 郭树行,兰雨晴,金茂忠.基于目标的软件可信性需求规约方法研究.计算机工程,2007,33(11):37−41.

王青(1964－),女,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为软件过程技术,需求工程,软件质量与管理.

李明树(1966－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为软件工程方法学,软件过程技术,需求工程,软件工程经济学.

伍书剑(1974－),男,博士生,主要研究领域为软件估算,软件质量技术.