基于代码修改的多目标有监督缺陷预测建模方法

陈 翔[1,2,3] 王秋萍[1]

(南通大学计算机科学与技术学院 江苏 南通 226019)[1]

(南京大学软件新技术国家重点实验室 南京 210093)[2]

(桂林电子科技大学广西可信软件重点实验室 广西 桂林 541004)[3]

摘 要 基于代码修改的缺陷预测,具有代码审查量少、缺陷定位和修复快的优点。文中首次将该问题建模为多目标优化问题,其中一个优化目标是最大化识别出的缺陷代码修改数,另一个优化目标是最小化需要审查的代码量。这两个优化目标之间存在一定的冲突,因此提出了MULTI方法,该方法可以生成一组具有非支配关系的预测模型。在实证研究中,考虑了6个大规模开源项目(累计227417个代码修改),以ACC和POPT作为评测预测性能的指标。实验结果表明,MULTI方法的预测性能均显著优于经典的有监督建模方法(EALR和Logistic)和无监督建模方法(LT和AGE)。

关键词 软件缺陷预测,多目标优化,代码修改,实证研究

中图法分类号 TP311.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.06.028

Multi-objective Supervised Defect Prediction Modeling Method Based on Code Changes

CHEN Xiang[1,2,3] WANG Qiu-ping[1]

(School of Computer Science and Technology, Nantong University, Nantong, Jiangshu 226019, China)[1]

(State Key Laboratory for Novel Software Technology at Nanjing University, Nanjing 210093, China)[2]

(Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin, Guangxi 541004, China)[3]

Abstract Defect prediction based on code changes has the advantage of smaller code inspection cost, easy fault localization and rapid fixing. This paper firstly formalized this problem as a multi-objective optimization problem. One objective is to maximize the number of identified buggy changes, and the other objective is to minimize the cost of code inspection. There exist an obvious conflict between two objectives, so this paper proposed a novel method MULTI. This method can generate a set of non-dominated prediction models. In the empirical studies, this paper chose six large-scale open source projects (with 227417 code changes in total) and considerd ACC and POPT as evaluation indicators of performance. Final results show that the proposed method can perform significantly better than the state-of-the-art supervised methods (i.e., EALR and Logistic) and unsupervised methods (i.e., LT and AGE).

Keywords Software defect prediction, Multi-objective optimization, Code changes, Empirical studies

软件缺陷预测[1]通过挖掘软件历史仓库(如版本控制系统、缺陷跟踪系统等)来构建缺陷预测模型,并用该模型来预测被测项目内的可疑程序模块,从而达到优化测试资源分配并提高被测项目质量的目的。传统的缺陷预测研究一般将程序模块粒度设置为包、类或者函数,而基于代码修改的软件缺陷预测(Defect Prediction for Software Changes, DPSC)则尝试构建模型来及时对开发人员向版本控制系统提交的代码进行缺陷预测,即将代码修改预测为有缺陷代码修改(Buggy Change)或无缺陷代码修改(Clean Change)。有缺陷代码修改会造成缺陷的引入,而无缺陷代码修改则不会引入任何缺陷。与传统缺陷预测方法相比,所提方法的优点是:1)与传统缺陷预测研究中的模块粒度相比,代码修改需要的代码审查量更少;2)对代码修改进行及时预测,可以使相关开发人员在熟悉代码内容的同时迅速完成缺陷的定位和修复。

Mockus等[2]最早在一个大型交换机系统上对DPSC问题进行了探索,他们在分析代码修改特征的同时,综合考虑了代码修改的规模、分散度以及开发人员的经验等。随后,Kamei等[3]在开源项目和商业项目中对DPSC问题进行了大规模的实证研究,并提出了EALR方法。Yang等[4]提出了一种基于深度学习的DPSC方法——Deeper,他们借助深度信

到稿日期:2017-04-24 返修日期:2017-07-19 本文受国家自然科学基金(61202006,61602267),南京大学计算机软件新技术国家重点实验室开放课题(KFKT2016B18),广西可信软件重点实验室研究课题(kx201610),江苏省高校自然科学研究项目(15KJB520030,16KJB520038),南通市科技平台项目(CP120130001)资助。

陈 翔(1980-),男,博士,副教授,CCF会员,主要研究方向为软件缺陷预测,E-mail:xchencs@ntu.edu.cn(通信作者);王秋萍(1993-),女,硕士生,主要研究方向为软件缺陷预测(通信作者)。

念网络尝试在原有特征集上构造出新的特征。Kim 等[5]借助文本挖掘方法对代码修改进行分析，但文本挖掘会生成大量特征并引发维数灾难问题，因此 SHIVAJI 等[6]借助特征选择方法来进一步提高模型的预测性能。上述研究工作均基于有监督建模方法。在最新的一项研究工作中，Yang 等[7]提出了简单的无监督建模方法，并发现了一些方法在预测性能上显著优于有监督建模方法(如 EALR[3])。但该项研究与我们的直觉相违背，因为在大部分情况下，无监督建模方法由于可以充分利用有标记数据，其构建的模型性能在大部分情况下优于无监督建模方法。通过深入分析文献[7]发现，这些无监督方法在构建模型时没有考虑需要审查的代码量，或者仅考虑了单一优化目标。基于上述分析，本文首次从多目标优化角度出发，将 DPSC 问题建模为多目标优化问题，其中第一个优化目标是最大化识别出的缺陷代码修改数，另一个优化目标是最小化需要审查的代码量。由于这两个优化目标存在一定的冲突，因此提出了一种有监督建模方法 MULTI，该方法可以生成一组具有非支配关系的预测模型。

1 MULTI 方法

MULTI 方法的提出主要受到基于搜索的软件工程思想[8]的启发，该思想最早由 Harman 提出并逐渐成为软件工程领域的一个研究热点。目前，该思想已经成功应用于软件开发生命周期的各个阶段，包括需求分析、软件设计、软件测试到软件维护。Harman[9]认为该思想也适用于软件缺陷预测研究领域。

针对 DPSC 问题，文中重点考虑两个优化目标。不难看出，这两个优化目标存在一定的冲突。若需要识别出更多的缺陷代码修改，则需要更多的代码审查量；相反，若减少代码审查量，则可能会遗漏掉一些缺陷代码修改。

本文借助 Logistic 回归来构建模型，假设代码修改用 n 个特征进行度量，则模型的系数可以用向量 $w=\{w_1,w_2,\cdots,w_n\}$ 表示。给定模型的系数向量 w 和需要预测的代码修改 m_i，其中代码修改 m_i 的第 j 个特征取值为 $v_{i,j}$，则可以借助式(1)计算该模块含有缺陷的概率：

$$y(m_i,w)=\frac{1}{1+e^{-(w_0+w_1v_{i,1}+\cdots+w_nv_{i,n})}} \tag{1}$$

本文将 DPSC 问题建模为二元分类问题并将分类阈值设置为 0.5，即若预测的概率值大于 0.5，则将该代码修改预测为有缺陷代码修改，否则预测为无缺陷代码修改。其计算式如下：

$$Y(m_i,w)=\begin{cases}1, & \text{if } y(m_i,w)>0.5\\ 0, & \text{if } y(m_i,w)\leqslant 0.5\end{cases} \tag{2}$$

随后给出两个优化目标的计算式，假设需要预测的代码修改集合为 M，候选解为 w，则第一个优化目标的计算式为：

$$benefit(w)=\sum_{m_i\in M}Y(m_i,w)\times buggy(m_i) \tag{3}$$

其中，$buggy(m_i)$表示代码修改 m_i 是否含有缺陷，若含有缺陷，则其取值为 1，否则其取值为 0。

第二个优化目标的计算式为：

$$cost(w)=\sum_{m_i\in M}Y(m_i,w)\times LOC(m_i) \tag{4}$$

其中，$LOC(m_i)$表示代码修改涉及的代码行数，其假设是代码修改涉及的行数越多，需要的代码审查量越大。

基于上述分析，可以将 DPSC 问题建模为双目标优化问题，并提出 MULTI 方法，为方便随后的描述，首先给出与多目标优化相关的定义。

定义 1(Pareto 支配关系) 假设用于构建 DPSC 模型的候选解为 w_i 和 w_j，则 w_i Pareto 支配 w_j，当且仅当：

($benefit(w_i)>benefit(w_j)$ and $cost(w_i)\leqslant cost(w_j)$) or ($benefit(w_i)\geqslant benefit(w_j)$ and $cost(w_i)<cost(w_j)$)

定义 2(Pareto 最优集) 一个候选解 w 是 Pareto 最优解，当且仅当不存在其他候选解 w^*，使得该解可以 Pareto 支配 w。

定义 3(Pareto 最优解集) 所有 Pareto 最优解构成 Pareto 最优解集。

研究人员提出不同类型的多目标优化算法，这类算法主要借助演化算法来构造 Pareto 最优解集。MULTI 方法主要基于其中一种经典的多目标优化算法 NSGA-Ⅱ[10]，其伪代码如算法 1 所示。

算法 1 MULTI 方法

输入：种群规模 N，最大迭代次数 T

输出：Pareto 最优解集

```
1. i←0
2. P_i←initPop(N)
3. while i<T do
4.    C_i←makeNewPop(P_i)
5.    B_i←P_i∪C_i
6.    F←fastNondominatedSort(B_i)
7.    P_(i+1)←∅
8.    j←1
9.    while |P_(i+1)|+|F_j|≤N do
10.      crowingDistanceAssign(F_j)
11.      P_(i+1)←P_(i+1)∪F_j
12.      j←j+1
13.   end while
14.   sort(F_j) //according to crowding distance
15.   P_(i+1)←P_(i+1)∪F_j[1:(N-|P_(i+1)|)]
16.   i←i+1
17. end while
18. return P_i 内的所有 Pareto 最优解
```

种群中的染色体可编码为模型的系数向量，在训练集上可以基于染色体对应的系数向量来构建预测模型，并借助 *benefit* 函数和 *cost* 函数来分别计算该模型的两个优化目标值。MULTI 方法首先初始化种群，染色体对象的向量元素随机赋值。随后，借助变异算子和演化算子生成新的染色体，其中交叉算子会根据交叉概率随机选择两条染色体来进行交叉从而生成两条新的染色体，变异算子会根据变异概率随机选择一条染色体来进行变异从而生成一条新的染色体。

随后，MULTI 方法借助选择算子从已有染色体中选择更高质量(基于 Pareto 支配关系分析)的染色体到下一轮种群中。首先，将原有的染色体与借助变异算子和交叉算子生成的新染色体合并到集合 B_i 中；然后，使用 *fastNondominatedSort* 函数来计算 B_i 中的每条染色体的 *NDR*(Non-

Dominated Rank)值。具体来说，首先从 B_i 中识别出所有不被 Pareto 支配的染色体，将它们的 NDR 值设置为 1，并将它们从集合 B_i 移到集合 F_1。随后，继续从 B_i 中识别出所有不被 Pareto 支配的染色体，将它们的 NDR 值设置为 2，并将它们从集合 B_i 移到集合 F_2。重复上述过程，直至集合 B_i 为空。基于染色体的 NDR 值，MULTI 方法会尽力选择 NDR 值更小的染色体，并将其移到下一轮种群中。步骤 14 使用到了拥挤距离(Crowding Distance)[10]的概念，通过拥挤距离可以避免选择出具有高相似度的染色体，从而保证种群的多样性。

当达到指定的最大迭代次数后，MULTI 方法满足终止条件并返回当前种群中的所有 Pareto 最优解。

需要注意的是，MULTI 方法是基于训练集来构造 Pareto 最优解集的，然后基于每个最优解(即系数向量)构造缺陷预测模型，并在测试集上计算该模型的预测性能。

2　实证研究

2.1　评测对象

本文采用了 6 个大规模开源项目作为 MULTI 方法的评测对象，这些对象也是当前 DPSC 问题研究中经常使用的数据集[3,7]，因此具有一定的代表性。这 6 个大规模项目覆盖了不同类型的应用，例如：Bugzilla 是基于 Web 的缺陷跟踪系统；Eclipse JDT 是 Java 编程语言开发工具；Mozilla 是 Web 浏览器；PostgreSQL 是一个数据库系统。表 1 列出了这些数据集的项目名(简称)、数据集搜集的时间段、提取的代码修改数量以及有缺陷代码修改所占的比例。

表 1　数据集的统计特征

Table 1　Statistical characteristics of datasets

项目名	时间段	#代码修改	缺陷代码/%
Bugzilla(BUG)	1998/8－2006/12	4620	37
Columba(COL)	2002/11－2006/7	4455	31
Eclipse JDT(JDT)	2011/5－2007/12	35386	14
Eclipse Platform(PLA)	2011/5－2007/12	64250	15
Mozilla(MOZ)	2000/1－2006/12	98275	5
PostgreSQL(POS)	1996/7－2010/5	20431	25

这些数据集针对代码修改共考虑了 14 种特征，这些特征可以简单分为 5 组。第一组主要考虑代码修改的分散度，其包含的特征有：NS(修改的子系统数)、ND(修改的目录数)、NF(修改的文件数)和 $Entropy$(基于熵计算出的代码修改的分散度)。第二组主要考虑代码修改的规模，其包含的特征有：LA(新增的代码行数)、LD(删除的代码行数)和 LT(修改前的相应代码行数)。第三组考虑代码修改的目的，其特征是 FIX(是否用于修复缺陷)。第四组考虑代码修改的历史，其包含的特征有：$NDEV$(代码修改涉及的开发人员数)、AGE(代码修改间的平均时间间隔)、NUC(与修改文件相关的代码修改数)。第五组考虑开发人员的经验，其包含的特征有：EXP(开发人员的经验)、REXP(最近开发人员的经验)和 SEXP(某个子系统上的开发人员经验)。特征的具体含义可参见文献[3]。

2.2　模型性能的评测指标

与 Kamei 等[3]的研究一样，本文重点考虑 ACC 指标和 POPT 指标，因为这两个指标都考虑了代码审查开销。本文将代码修改的规模(即 $LA+LD$)作为代码审查的开销。其中，ACC 指标与可以使用的测试资源(用代码修改规模表示)有关，其计算的是当使用了指定比例的测试资源后缺陷代码修改的查全率。而 POPT 指标是基于代码审查开销的评测指标的归一化版本，其具体含义参见文献[3]。上述两个指标的取值越大，其预测性能越好。

2.3　模型性能评测的方法

本文在模型性能评测时考虑了代码修改的时序因素[11]，即代码修改按照其提交的时间进行排序。假设将代码修改划分到 n 个区间(即将在同一个月内提交的代码修改划分到同一个区间内)，首先基于区间 i 和 $i+1$ 来构建模型，并用该模型预测区间 $i+4$ 和 $i+5$ 内的代码修改。这样一方面可以确保训练集中的代码修改和测试集上的代码修改有两个月的时间差，另一方面也可以确保测试集和训练集内有足够的代码修改。

2.4　结果分析

为了评估 MULTI 方法的有效性，本文考虑了 DPSC 研究领域内的一些经典的有监督建模方法和无监督建模方法。

有监督建模方法包括 EALR 和 Logistic 回归(简称 Logistic)。EALR 是 Kamei 等[3]提出的经典方法，其使用 $buggy(m_i)/LOC(m_i)$ 作为因变量，借助线性回归来构建模型。Logistic 回归是目前软件缺陷预测研究中经常使用的建模方法，其基于训练集来直接确定模型中的系数取值，与 MULTI 方法中借助演化算法来确定 Logisitic 回归模型中的系数不同。

无监督建模方法重点考虑 Yang 等[7]提出的方法，在该方法中仅基于测试集上指定的特征取值，按从小到大的顺序对代码修改进行排序。其假设是特征取值越小，其含有缺陷的概率越大。在实证研究中，他们发现存在一些无监督建模方法，其预测性能优于有监督建模方法。本文主要考虑他们提出的 LT 方法和 AGE 方法，这两种方法分别根据 LT 特征和 AGE 特征对代码修改进行排序，并且在本文中是预测性能最好的两种无监督建模方法。

在构建模型之前，我们进行了一系列数据预处理：1)移除具有高相关性的特征；2)对所有数值型特征取值进行 Log 处理；3)在训练集上使用随机欠采样方法来缓解数据集内的类不平衡问题，即在训练集上随机移除无缺陷代码修改，直至有缺陷代码修改的数量与无缺陷代码修改的数量一致。其预处理方式与文献[3,7]一致。除此之外，为了方法的公平性，针对无监督建模方法，我们仅使用了测试集中的代码修改来构建模型。

MULTI 方法的参数和取值设置如下：种群规模为 200，系数向量成员的有效取值区间为[－10000,10000]，在种群初始化阶段，系数向量成员的有效取值区间限定为[－10,10]，最大迭代次数为 400，交叉概率为 0.5，变异概率为 0.1。

由于 MULTI 方法的内部存在随机因素，因此在每个数据集上独立运行该方法 10 次，每次选择不同的随机数。每次在训练集上获得一组 Pareto 最优解集，因此独立运行 10 次后会获得 10 组 Pareto 最优解集。借助 MULTI-B，从 10 组最优解集中选出可以在测试集上取得预测性能最好的解。借助

MULTI-B,从 10 组最优解集中选出测试集上可以使预测性能达到中位值的解。其运行过程如图 1 所示。

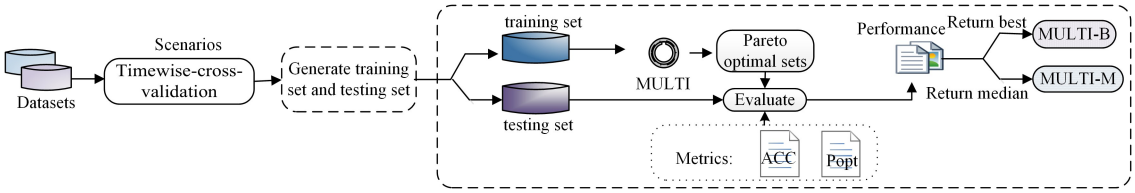


图 1 MULTI 方法的运行过程

Fig. 1 Running process of MULTI method

本文首先基于 POPT 指标将 MULTI 方法与基准方法进行比较,其在每个项目中的预测性能如表 2 所列。由于每个数据集上可能存在多组预测,因此表 2 罗列了在每个项目上的中位值,最后一行统计的是每个方法在不同项目上的均值。同时,图 2 给出了不同方法在所有预测上的取值分布。从表 2 中不难看出,基于均值,MULTI-B 方法相比 EALR,Logistic,LT 和 AGE 在 POPT 指标上可以提高 49.5%,128.3%,22.2%和 23.9%。MULTI-M 方法相比 EALR,Logistic,LT 和 AGE 在 POPT 指标上可以提高 43.8%,119.7%,17.6%和 5.7%。随后通过计算 Benjamini-Hochberg 修正后的 p 值(其显著水平设置为 0.05)[12]发现,MULTI 方法的预测性能显著优于 4 种基准方法(即对应的 p 值均远小于 0.05)。

表 2 基于 POPT 指标的结果比较

Table 2 Comparison of results based on POPT

项目名	MULTI-B	MULTI-M	EALR	Logistic	LT	AGE
BUG	0.904	0.832	0.600	0.460	0.721	0.661
COL	0.919	0.885	0.619	0.325	0.732	0.786
JDT	0.847	0.824	0.588	0.401	0.709	0.685
PLA	0.871	0.861	0.583	0.379	0.717	0.709
MOZ	0.814	0.786	0.498	0.408	0.651	0.638
POS	0.867	0.837	0.600	0.316	0.742	0.731
均值	0.870	0.837	0.582	0.381	0.712	0.702

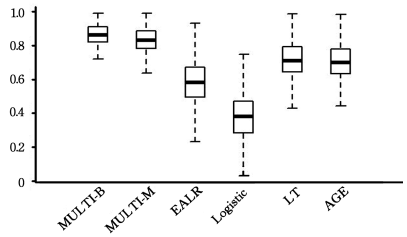


图 2 不同方法基于 POPT 指标的盒图

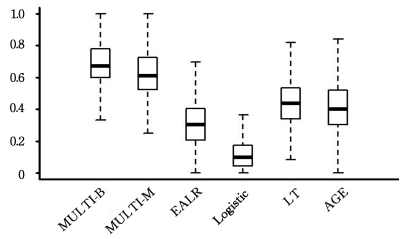
Fig. 2 Box plot of different methods based on POPT

然后,本文基于 ACC 指标将 MULTI 方法与基准方法进行了比较,这里假设可以使用的测试资源比例为 20%。其在每个项目中的预测性能如表 3 所列,其中罗列了每个方法在每个项目中的中位值,最后一行统计了每个方法在不同项目中的均值。同时,图 3 给出了不同方法在所有预测上的取值分布。从表 3 中不难看出,基于均值,MULTI-B 方法相比 EALR,Logistic,LT 和 AGE 在 ACC 指标上可以提高119.3%,525.7%,59.3%和 64.7%。MULTI-M 方法相比 EALR,Logistic,LT 和 AGE 在 ACC 指标上可以提高98.1%,465.1%,43.9%和 48.8%。随后,通过计算 Benjamini-Hochberg 修正后的 p 值(其显著水平设置为 0.05)发现,MULTI 方法的预测性能显著优于 4 种基准方法(即对应的 p 值均远小于 0.05)。

表 3 基于 ACC 指标的结果比较

Table 3 Comparison of results based on ACC

项目名	MULTI-B	MULTI-M	EALR	Logistic	LT	AGE
BUG	0.717	0.574	0.306	0.188	0.449	0.375
COL	0.733	0.690	0.400	0.100	0.440	0.568
JDT	0.664	0.616	0.330	0.118	0.452	0.408
PLA	0.719	0.688	0.305	0.076	0.432	0.429
MOZ	0.606	0.543	0.180	0.099	0.363	0.280
POS	0.652	0.583	0.345	0.071	0.432	0.426
均值	0.682	0.616	0.311	0.109	0.428	0.414



注:假设可以使用的测试资源比例为 20%

图 3 不同方法基于 ACC 指标的盒图

Fig. 3 Box plot of different methods based on ACC

最后,进一步分析基于 ACC 指标时可以使用的测试资源比例对结果的影响。本文主要考虑的比例是 5%,10%,15%和 20%,最终结果如图 4 所示,其中横坐标表示可供使用的测试资源比例,纵坐标表示各个方法在所有项目上的 ACC 均值。从图 4 中不难看出,基于不同的阈值,本文所提 MULTI 方法与其他基准方法相比,均具有更高的 ACC 值。

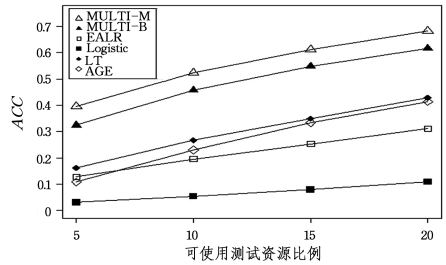


图 4 基于 ACC 指标时可使用的测试资源比例对结果的影响

Fig. 4 Impact on performance when considering different ratio of available test resources based on ACC

结束语 虽然在最近的研究工作中,Yang 等[7]发现存在一些无监督建模方法,其预测性能显著优于无监督建模方法(如 EALR[3]),但上述研究的结论与已有直觉相违背。本文首次将 DPSC 问题建模为多目标优化问题并提出了 MULTI 方法,结果表明该方法的预测性能显著优于基于无监督建模方法的 LT 方法和 AGE 方法。该结果表明,针对 DPSC 问

题,有监督学习方法仍然具有很高的研究价值。论文还有很多后续的研究工作值得关注,例如:需要考虑更多的商业项目来验证 MULTI 方法的有效性,可以考虑其他类型的多目标优化算法(例如 SPEA[13])来完善 MULTI 方法,借助更加有效的特征选择方法[14-15]来移除数据集中的冗余特征和无关特征。

致　谢　感谢赵英全、袁智丹和张亦晶在本文所提方法的实现和实证研究中做出的贡献。

参 考 文 献

[1] CHEN X,GU Q,LIU W S,et al. Survey of static software defect prediction[J]. Journal of Software,2016,27(1):1-25. (in Chinese)
陈翔,顾庆,刘望舒,等. 静态软件缺陷预测方法研究[J]. 软件学报,2016,27(1):1-25.

[2] MOCKUS A,WEISS D M. Predicting risk of software changes [J]. Bell Labs Technical Journal,2000,5(2):169-180.

[3] KAMEI Y,SHIHAB E,ADAMS B,et al. A large-scale empirical study of just-in-time quality assurance[J]. IEEE Transactions on Software Engineering,2013,39(6):757-773.

[4] YANG X,LO D,XIA X,et al. Deep learning for just-in-time defect prediction[C]// International Conference on Software Quality,Reliability,and Security. 2015:17-26.

[5] KIM S,JR E J W,ZHANG Y. Classifying software changes: clean or buggy?[J]. IEEE Transactions on Software Engineering,2008,34(2):181-196.

[6] SHIVAJI S,WHITEHEAD E J,AKELLA R,et al. reducing features to improve code change-based bug prediction[J]. IEEE Transactions on Software Engineering,2013,39(4):552-569.

[7] YANG Y,ZHOU Y,LIU J,et al. Effort-aware just-in-time defect prediction:simple unsupervised models could be better than supervised models[C]// Proceedings of the International Symposium on Foundations of Software Engineering. 2016,157-168.

[8] HARMAN M,MANSOURI S A,ZHANG Y. Search-based software engineering:trends,techniques and applications[J]. ACM Computing Survey,2012,45(1):1-61.

[9] HARMAN M. The relationship between search based software engineering and predictive modeling[C]// International Conference on Predictive Models in Software Engineering. 2010:1-13.

[10] DEB K,PRATAP A,AGARWAL S,et al. A fast and elitist multi-objective genetic algorithm:NSGA-II[J]. IEEE Transactions on Evolutionary Computation,2002,6(2):182-197.

[11] TAN M,TAN L,DARA S,et al. online defect prediction for imbalance data[C]// International Conference on Software Engineering. 2015:99-108.

[12] BENJAMINI Y,HOCHBERG Y. controlling the false discovery rate:a practical and powerful approach to multiple testing[J]. Journal of the Royal Statistical Society,Series B (Methodological),1995,57(1):289-300.

[13] ZITZLER E,THIELE L. Multiobjective evolutionary algorithms:a comparative case study and the strength pareto approach [J]. IEEE Transactions on Evolutionary Computation,1999,3(4):257-271.

[14] LIU W S,CHEN X,GU Q,et al. A cluster-analysis-based feature-selection method for software defect prediction[J]. SCIENCE CHINA:Information Sciences,2016,46(9):1298-1320. (in Chinese)
刘望舒,陈翔,顾庆,等. 软件缺陷预测中基于聚类分析的特征选择方法[J]. 中国科学:信息科学,2016,46(9):1298-1320.

[15] LIU W S,CHEN X,GU Q,et al. A Noise Tolerable Feature Selection Framework for Software Defect Prediction[J]. Chinese Journal of Computers,2018,41(3):506-520. (in Chinese)
刘望舒,陈翔,顾庆,等. 一种面向软件缺陷预测的可容忍噪声的特征选择框架[J]. 计算机学报,2018,41(3):506-520.

(上接第 160 页)

[8] ROZINAT A,VAN DER AALST W M P. Conformance checking of processes based on monitoring real behavior [J]. Information Systems,2008,33(1):64-95.

[9] TINI S,LARSEN K G,GEBLER D. Compositional bisimulation metric reasoning with probabilistic process calculi[J]. Logical Methods in Computer Science,2017,12(4):1-38.

[10] POLYVYANYY A,WEIDLICH M,CONFORTI R,et al. The 4C spectrum of fundamental behavioral relations for concurrent systems[C]// International Conference on Applications and Theory of Petri Nets and Concurrency. Springer,Cham,2014:210-232.

[11] ARMAS-CERVANTES A,BALDAN P,DUMAS M,et al. Diagnosing behavioral differences between business process models:An approach based on event structures [J]. Information Systems,2016,56(C):304-325.

[12] WEIDLICH M,MENDLING J. Perceived consistency between process models [J]. Information Systems,2012,37(2):80-98.

[13] WEIDLICH M,MENDLING J,WESKE M. Efficient consistency measurement based on behavioral profiles of process models [J]. IEEE Transactions on Software Engineering,2011,37(3):410-429.

[14] ARMAS-CERVANTES A,DUMAS M,GARCíA-BAÑUELOS L,et al. On the suitability of generalized behavioral profiles for process model comparison [M]// Web Services,Formal Methods,and Behavioral Types. Springer,Cham,2014:13-28.

[15] POLYVYANYY A,ARMAS-CERVANTES A,DUMAS M,et al. On the expressive power of behavioral profiles[J]. Formal Aspects of Computing,2016,28(4):597-613.

[16] ENGELFRIET J. Branching processes of Petri nets [J]. Acta Informatica,1991,28(6):575-591.

[17] COUVREUR J M,POITRENAUD D,WEIL P. Branching processes of general Petri nets[J]. Fundamenta Informaticae,2013,122(1/2):31-58.