的属性.也就是说,CFS 属性选择算法既考虑了属性间的相关性,又考虑了属性与缺陷之间的相关性,因此本文中没有再做相关分析;

- BestFirst 搜索方法是回溯的贪婪搜索.在 CFS 属性选择方法中,对于每个数据,选择的度量元被赋值为1,否则为 0.本文采用两种跨版本验证,共 22 组实验,属性选择结果见表 6.

Table 6 Results of metrics selection

表 6 属性选择结果

项目	训练集	测试集	属性选择后的度量元
ANT	V1	V2	*ce,moa,cloc,cmfa*
	V2	V3	*rfc,ce,lcom3,cbm,cdam,cmoa,awmc,amax_cc*
	V1,V2	V3	*rfc,ce,moa,ic,cdam,awmc,anoc,arfc,aloc,acam,amax_cc*
	V3	V4	*wmc,rfc,loc,moa,cam,clcom,cloc,cavg_cc,age*
	V1,V2,V3	V4	*wmc,rfc,ce,lcom3loc,moa,crfc,clcom,clcom3,cdam,amax_cc,age*
XALAN	V1	V2	*noc,rfc,lcom,loc,dam,amc,crfc,cmfa,anpm,alcom3,duration*
	V2	V3	*noc,rfc,lcom3,loc,cam,amc,crfc,ccam,duration,pcd*
	V1,V2	V3	*rfc,lcom3,loc,amc,crfc,duration*
CAMEL	V1	V2	*noc,cbo,dam,cbm,amc,max_cc,avg_cc,cwmc,cnoc,clcom,cca, cnpm,camc,ace,anpm,adam,duration,pcd*
	V2	V3	*npm,clcom3,duration,pcd*
	V1,V2	V3	*noc,npm,max_cc,cdit,cca,cloc,anpm,duration,pcd*
JEDIT	V1	V2	*dit,rfc,crfc,cce,cmoa,cmax_cc,awmc,anpm,aamc,duration,pcd*
	V2	V3	*rfc,npm,ccbo,cnpm,adam,duration,pcd*
	V1,V2	V3	*rfc,ce,ccbo,cmax_cc,arfc,amoa,duration,pcd*
	V3	V4	*ce,npm,lcom3,max_cc,clcom,cmoa,cmax_cc,arfc,duration,pcd*
	V1,V2,V3	V4	*rfc,npm,lcom3,moa,*max_cc,*cnpm,cmax_cc,arfc,duration,pcd*
POI	V1	V2	*dit,lcom,avg_cc,cnpm,alcom3,aloc,amoa,amax_cc*
	V2	V3	*dit,lcom3,dam,cbm,ccam,duration*
	V1,V2	V3	*lcom3,pcd*
XERCES	V1	V2	*cam,cnoc,cmoa,cmfa,ccbm,duration,pcd*
	V2	V3	*ca,ce,lcom3,dam,moa,ic,cbm,avg_cc,ccbo,ccam,camc,age*
	V1,V2	V3	*ca,ce,lcom3,ic,cbm,avg_cc,cwmc,cnoc,cdam,cmoa,acam,duration,pcd*

根据度量元在 22 组实验中出现的次数进行排名,前 10 个最重要的度量元为 *Duration*(score=14),*PCD*(score=12),*rfc*(score=11),*lcom*3(score=10),*ce*(score=8),*moa*(score=6),*npm*(score=5),*loc*(score=5),*cbm*(score=5),*Crfc*(score=5)和 *Cmoa*(score=5).实验结果可以看出:36%的所选度量元属于两类软件演化度量元——2 个属于软件模式相关度量元,2 个属于代码变更度量元,其中,2 个演化模式相关度量元排在前 2.除此外,与规模、耦合性、内聚性和继承相关的度量元也在前 10 中出现.

针对选出来的前 10 个最重要的度量元,我们做了度量元与缺陷间的 Spearman 相关性分析,结果显示,该 10 个度量元与一个类是否有缺陷在置信度(双测)为 0.01 时相关性是显著的.其中,*Duration*,*lcom*3 和一个类是否有缺陷是负相关的,其他都是正相关的.

4.2 不同度量元的预测性能分析

针对已有类的演化特征,本文提出了两类软件演化度量元.为验证演化度量元的性能,本文在 6 个开源软件产品上建立缺陷预测模型——ANT,CAMEL,XALAN,JEDIT,XERCES 和 POI,并且使用召回率、精度、*F*-值和 AUC 来评价预测性能.本文把使用代码度量元的缺陷预测模型作为对比的基准.表 7 给出了仅使用代码度量元(*CM*)、演化模式相关度量元(*EP*)和代码变更度量元(*delta*,*churn*)的缺陷预测模型的性能.从表 7 中可以观察到:

(1) 不管是从性能平均值来看还是中位数来看,仅使用演化模式相关度量元可以有效地提高预测的召回率和 *F*-值;

(2) 演化模式相关度量元在精确度 *P* 和 AUC 上与代码度量元差别不大;在精确度 *P* 上,演化模式相关度量元有着更小的标准差(*Stdev*),也就意味着,演化度量元比代码度量元在精确度 *P* 上更稳定.在 AUC 上,虽然演化度量元的标准差不是最小的,但是与代码变更度量元的标准差的差距不大,并且比代码度量元要小,因此,演化模式相关度量元在 AUC 上较稳定;

(3) 使用代码变更度量元的预测性能在精确度 P 上相差不大,但在召回率 R、F 值和 AUC 上要低于代码度量元和演化模式相关度量元;除此之外,在统计学上,代码变更度量元的预测性能在召回率和 F 值上显著低于演化模式相关度量元.

综上所述,与代码和代码变更度量元相比,演化模式度量元有着相对较好的预测性能.

Table 7 Performance of models built by *CM*, *EP*, *Delta* and *Churn*

表 7 仅使用 *CM*,*EP* 和{*Delta*,*Churn*}的缺陷预测模型的性能

	方法	*CM*				*EP*				{*Delta*,*Churn*}			
		P	*R*	*F*	AUC	*P*	*R*	*F*	AUC	*P*	*R*	*F*	AUC
平均值	NB	0.53	0.55	0.46	**0.69**	0.50	**0.64**	**0.53**	0.65	**0.54**	0.40	0.33	0.64
	BLR	0.48	0.54	0.43	**0.65**	**0.50**	**0.67**	**0.53**	**0.65**	0.49	0.49	0.35	0.63
	J48	**0.53**	0.53	0.46	**0.65**	0.48	**0.65**	**0.51**	0.64	0.52	0.46	0.35	0.61
	RF	0.49	0.54	0.45	**0.65**	**0.50**	0.58	**0.50**	**0.65**	**0.50**	0.48	0.35	0.63
标准差	NB	0.31	0.24	0.21	0.22	**0.25**	**0.20**	0.21	**0.14**	0.27	0.29	**0.19**	**0.14**
	BLR	0.30	0.22	**0.19**	0.18	**0.25**	**0.21**	0.21	0.15	0.26	0.30	**0.19**	**0.11**
	J48	0.29	0.23	**0.20**	0.19	**0.26**	**0.21**	0.22	0.14	0.30	0.29	**0.20**	**0.09**
	RF	0.29	**0.21**	**0.19**	0.19	**0.25**	0.23	0.22	0.15	0.27	0.31	**0.19**	**0.11**
中位数	NB	0.48	0.56	0.46	**0.78**	0.51	**0.63**	**0.54**	0.68	**0.54**	0.37	0.35	0.64
	BLR	0.41	0.54	0.41	**0.70**	0.51	**0.68**	**0.57**	0.68	**0.52**	0.49	0.37	0.61
	J48	0.48	0.45	0.48	**0.69**	0.46	**0.64**	**0.49**	0.65	**0.54**	0.50	0.34	0.61
	RF	0.48	**0.55**	0.44	**0.74**	**0.55**	0.54	**0.50**	0.66	0.53	0.48	0.36	0.61

为了验证组合度量元的预测性能是否优于单独的度量元,本文给出了代码(*CM*)、演化模式(*EP*)、代码变更度量元(*Delta*,*Churn*)及其组合度量元,共 7 组,即,3 种度量元的排列组合 $(C_3^1+C_3^2+C_3^3)$.这些度量元的预测性能如图 3 所示.7 组度量元及其组合分别如下:

(1) *M*1: *CM*, *M*2: *EP*, *M*3: {*Delta*,*Churn*};

(2) *M*4: *CM*+*EP*, *M*5: *CM*+{*Delta*,*Churn*}, *M*6: *EP*+{*Delta*,*Churn*};

(3) *M*7: *CM*+*EP*+{*Delta*,*Churn*}.

图 3 中给出了不同度量元组合在不同方法下的平均性能情况.

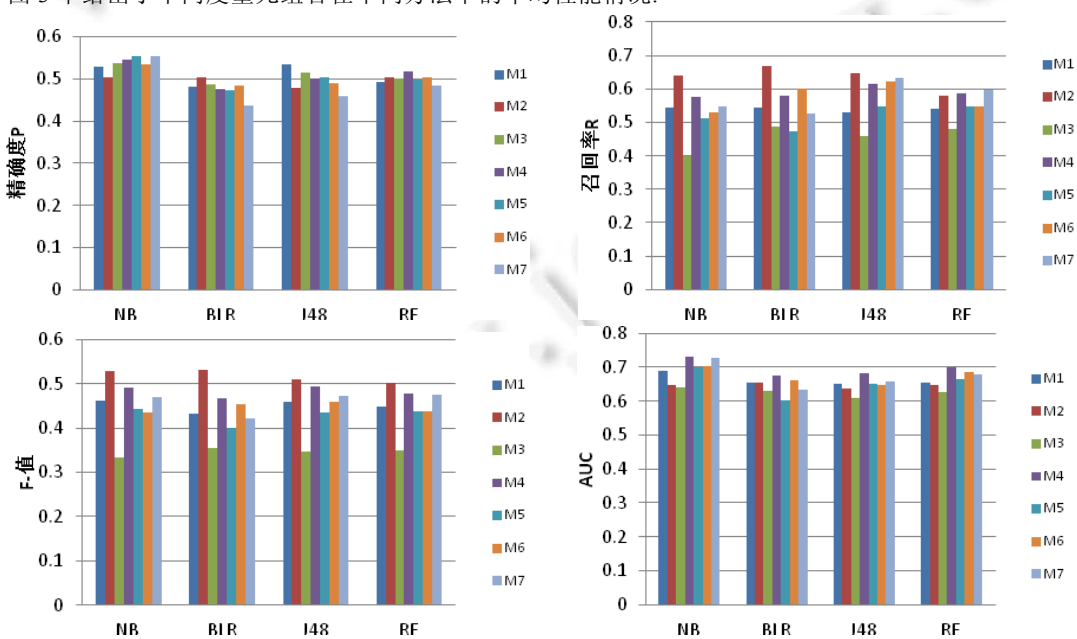

Fig.3 Performance of models built by various metrics combinations

图 3 使用不同度量元组合的模型性能

从中可以看出:

(1) 在精确度 *P* 上,7 组度量元组合在不同的方法下,性能各有不同,性能间的差距不显著:当使用朴素贝叶斯(NB)时,*M*7(*CM*+*EP*+{*Delta*,*Churn*})有着最好的预测性能;当使用二元逻辑回归时,*M*2(*EP*)有着最好的预测性能;当使用 J48 决策树时,*M*1(*CM*)有着最好的预测性能;当使用随机森林(RF)时,*M*4(*CM*+*EP*)有着最好的预测性能;

(2) 在召回率 *R* 上,在朴素贝叶斯、二元逻辑回归和 J48 决策树算法上,*M*2(*EP*)都有着最好的预测性能;在随机森林算法上,虽然 *M*7(*CM*+*EP*+{*Delta*,*Churn*})有着最好的性能,但 *M*2 与 *M*7 的差距也不大.即演化模式相关的度量元在不同的建模方法下都有着较好的召回率;

(3) 在 *F*-值上,不管在什么方法上,*M*2(*EP*)都有着相对最好的预测性能.即演化模式相关的度量元在不同的建模方法下都有着最好的 *F*-值;

(4) 在 AUC 上,不管在什么方法上,*M*4(*CM*+*EP*)都有着较好的预测性能.即代码度量元和演化模式度量元的组合有着较高的召回率且较低的假正率;

(5) 除此之外,代码变更度量元(*M*3)在召回率 *R*、*F* 值和 AUC 上的预测性能比其他度量元相对较差.

4.3 演化历史中数据集选择对预测性能的影响分析

本文选择历史版本数大于 1 的产品来验证使用前一个版本数据(PRE)的预测效果是否比使用所有版本数据(ALL)的预测效果要好,并且选择演化模式相关度量元和代码与演化相关的组合度量元来分析.表 8 给出了其平均性能的对比.

Table 8 Comparison of performance of models built on ALL/PREVIOUS data

表 8 使用所有和前一个版本数据的缺陷预测模型的性能对比

模型	度量元	方法	ALL				PRE			
			P	*R*	*F*	AUC	*P*	*R*	*F*	AUC
*M*1	*CM*	NB	0.617	**0.598**	0.505	0.787	**0.625**	0.593	**0.515**	**0.790**
		BLR	**0.558**	**0.603**	**0.473**	0.738	0.541	0.518	0.422	0.668
		J48	**0.589**	0.583	**0.500**	0.731	0.564	**0.633**	0.480	**0.736**
		RF	0.559	0.581	**0.476**	**0.724**	**0.565**	**0.591**	0.460	0.722
*M*2	*EP*	NB	**0.542**	0.602	0.515	0.678	0.510	**0.733**	**0.557**	**0.684**
		BLR	**0.537**	0.654	0.521	0.687	0.502	**0.798**	**0.568**	**0.693**
		J48	**0.498**	0.620	0.486	0.657	0.477	**0.812**	**0.543**	0.651
		RF	**0.538**	0.502	0.471	0.673	0.512	**0.718**	**0.556**	**0.684**
*M*3	{*Delta*,*Churn*}	NB	0.510	0.448	0.333	0.649	**0.546**	**0.497**	**0.400**	**0.681**
		BLR	0.519	**0.508**	0.379	**0.664**	**0.520**	0.472	**0.392**	0.599
		J48	0.523	**0.484**	0.382	**0.638**	**0.529**	0.469	**0.391**	0.604
		RF	0.513	**0.488**	**0.369**	**0.646**	**0.537**	0.410	0.360	0.641
*M*4	*CM*+*EP*	NB	0.616	0.601	0.516	0.783	**0.635**	**0.616**	**0.543**	**0.813**
		BLR	0.546	0.600	**0.495**	**0.736**	**0.562**	**0.625**	0.483	0.714
		J48	**0.545**	**0.666**	**0.520**	0.717	0.517	0.660	0.471	**0.744**
		RF	0.577	0.616	0.499	**0.755**	**0.582**	**0.645**	**0.511**	0.750
*M*5	*CM*+{*Delta*,*Churn*}	NB	**0.606**	0.560	0.470	0.779	0.601	**0.576**	**0.487**	**0.789**
		BLR	**0.541**	0.515	0.418	**0.662**	0.515	**0.527**	**0.418**	0.646
		J48	**0.559**	0.533	0.443	**0.706**	0.555	**0.553**	**0.466**	0.706
		RF	**0.569**	0.587	0.462	**0.730**	0.557	**0.608**	**0.468**	0.713
*M*6	*EP*+{*Delta*,*Churn*}	NB	0.534	0.549	0.453	0.703	**0.557**	**0.623**	**0.514**	**0.757**
		BLR	**0.514**	0.648	0.501	0.661	0.507	**0.667**	**0.522**	**0.676**
		J48	**0.517**	0.674	**0.523**	0.659	0.464	**0.778**	0.518	**0.664**
		RF	0.514	0.581	0.464	0.687	**0.527**	**0.609**	**0.510**	**0.707**
*M*7	*CM*+*EP*+{*Delta*,*Churn*}	NB	0.611	0.572	0.493	0.777	**0.626**	**0.597**	**0.524**	**0.815**
		BLR	0.497	0.545	0.437	0.687	**0.532**	**0.623**	**0.486**	**0.705**
		J48	0.527	0.634	**0.494**	0.693	**0.546**	**0.640**	0.483	**0.746**
		RF	0.552	0.632	**0.507**	0.735	**0.578**	**0.650**	**0.507**	**0.767**

从表 7 中可以观察到:

(1) 对于代码度量元(*CM*)来说,时间邻近的演化历史对预测性能的影响不明显;

(2) 对于演化模式相关度量元,时间邻近的演化历史降低了 0.025 左右的精确度,但是提高了召回率、*F*-

值和 AUC 上的预测性能.其中,召回率最高提高了 0.216,*F*-值最高提高了 0.085,AUC 的提高不大;

(3) 对于代码变更度量元,时间邻近的演化历史提高了预测的精确度和 *F*-值,但是减低了召回率和 AUC;

(4) 对于代码和演化相关度量元的组合,时间邻近的演化历史对预测性能有改进作用,主要针对精确度和召回率;

(5) 对于代码和代码变更度量元的组合,时间邻近的演化历史可以提高预测的召回率和 *F*-值,并且在精确度和 AUC 上差别不大;

(6) 对于演化模式相关的度量元和代码变更度量元的组合,时间邻近的演化历史对除精确度外的预测性能有改进作用,且最大改进了 0.104 的召回率、0.061 的 *F*-值和 0.054 的 AUC;

(7) 对于 3 种度量元的组合,时间邻近的演化历史对精确度、召回率、*F*-值和 AUC 都有改进作用,且最高改进了 0.035 的精确度、0.078 的召回率、0.049 的 *F*-值和 0.053 的 AUC.

综上所述,案例研究表明:除代码度量元外,最近版本数据建立的模型比全部历史数据建立的模型具有更好的预测性能.这说明在软件进化历程中,最近的演化轨迹对产品的影响更大.这也符合自然界进化的规律.而代码类度量元表征产品的本质特征,随时间的影响不大.

5 结束语

本文提出了一种基于软件演化历史的缺陷预测方法,该方法的目的是预测一个 Java 类是否存在缺陷.本文把软件的演化过程看成一个物种的进化过程,通过改进的软件演化矩阵来模拟 Java 类的演化过程.根据一个 Java 类是否存在演化历史,把 Java 类分为新增类和已有类,其中,已有类有自身演化数据而新增类没有.本文主要是针对已有类进行缺陷预测.已有类是指在历史版本中曾存在过的 Java 类,它们有着自身的演化历史数据.针对该类的特点,本文提出两类代码演化度量元来度量已有类的演化,它们分别从粗粒度和细粒度上分析软件演化过程中版本级别和代码级别上软件属性的变化.案例研究选择了 6 个著名开源软件项目和 4 种常用的软件缺陷预测建模方法建立缺陷预测模型,实验结果验证了软件演化度量元的有效性.

在未来的研究中,将进一步开展以下几个方面的工作.

(1) 扩展数据源.本文选用的数据集都是 PROMISE 数据库中版本数大于 4 的 Java 项目,具有一定的局限性.在更多不同类型的连续项目上验证演化度量元的有效性,是下一步工作的重点;

(2) 完善演化度量元集.本文中使用两类与演化相关的度量元:一类是在版本级别上类的演化模式,另一类是软件代码变更相关的度量元.未来计划通过深入分析软件演化的特征来提出更多度量元,从而完善现有演化度量元集;

(3) 提取与新增类相关的度量元.本文中仅分析了软件演化对已有类的影响,下一步将深入分析由哪些因素影响着新增类的演化,从而提高新增类缺陷预测的性能;

(4) 应用在实际项目中.本文已经验证了演化度量元的有效性,下一步计划将实验结果应用于实际的项目中.用演化模式相关度量元来预测易出缺陷模块,帮助软件人员更好的检测和移除软件缺陷.

References:

[1] Halstead M. Elements of Software Science. New York: Elsevier North-Holland. 1977.

[2] McCabe T. A complexity measure. IEEE Trans. on Software Engineering, 1976,2(4):308−320. [doi: 10.1109/TSE. 1976.233837]

[3] Chidamber S, Kemerer C. A metrics suite for object oriented design. IEEE Trans. on Software Engineering. 1994,20(6):476−493. [doi: 10.1109/32.295895]

[4] Moser R, Pedrycz W, Succi G. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In: Proc. of the 30th Int'l Conf. on Software Engineering. 2008. 181−190. [doi: 10.1145/1368088.1368114]

[5] Kpodjedo S, Galinier P, Antoniol G. A google-inspired error correcting graph matching algorithm. Technical Report, EPM-RT-2008-06, Ecole Polytechnique de Montreal, 2008.

[6] Kpodjedo S, Ricca F, Galinier P, Antoniol G. Not all classes are created equal: Toward a recommendation system for focusing testing. In: Proc. of the Int'l Workshop on Recommendation Systems for Software Engineering. New York, 2008. 6−10. [doi: 10.1145/1454247.1454250]

[7] Kpodjedo S, Ricca F. Evolution and search based metrics to improve defects prediction. In: Proc. of the 1st Int'l Symp. on Search Based Software Engineering. 2009. 23−32. [doi: 10.1109/SSBSE.2009.24]

[8] D'Ambros M. On the evolution of source code and software defects [Ph.D. Thesis]. Universita Della Svizzera Italiana of Washington, 2010.

[9] Wang D. Research on the analysis and prediction techniques of defect and its removal [Ph.D. Thesis]. Graduate University of Chinese Academy of Sciences, 2014 (in Chinese).

[10] Gall H, Jazayeri M, Klosch R, Trausmuth G. Software evolution observations based on product release history. In: Proc. of the Int'l Conf. on Software Engineering. 1997. 160−166.

[11] Gall H, Jazayeri M, Riva C. Visualizing software release histories: The use of color and third dimension. In: Proc. of the Int'l Conf. on Software Maintenance. IEEE Press, 1999. 99−108. [doi: 10.1109/ICSM.1999.792584]

[12] Lanza M. The evolution matrix: Recovering software evolution using software visualization techniques. In: Proc. of the 1st Workshop on Principles of Software Evolution. New York: ACM Press, 2001. 37−42. [doi: 10.1145/602461.602467]

[13] Wu J, Holt R, Hassan A. Exploring software evolution using spectrographs. In: Proc. of the 11th Working Conf. on Reverse Engineering. IEEE Press, 2004. 80−89. [doi: 10.1109/WCRE.2004.20]

[14] Wu JW, Spitzer CW, Hassan AE, Holt RC . Evolution spectrographs: Visualizing punctuated change in software evolution. In: Proc. of the 7th Int'l Workshop on Principles of Software Evolution. ACM Press, 2004. 57−66.

[15] Grba T, Ducasse S. Modeling history to analyze software ecolution. Journal on Software Maintenance and Evolution: Research and Practice, 2006,18(3):207−236. [doi: 10.1002/smr.325]

[16] Robbes R, Lanza M. A change-based approach to software evolution. Electronic Notes in Theoretical Computer Science, 2007,166: 93−109. [doi: 10.1016/j.entcs.2006.06.015]

[17] Graves TL, Karr AF, Marron JS, Siy H. Predicting fault incidence using software change history. IEEE Trans. on Software Engineering, 2000,26(7):653−661. [doi: 10.1109/32.859533]

[18] Nagappan N, Ball T. Use of relative code churn measures to predict system defect density. In: Proc. of the 27th Int'l Conf. on Software Engineering. 2005. 284−292. [doi: 10.1109/ICSE.2005.1553571]

[19] Sliwerski J, Zimmermann T, Zeller A. When do changes induce fixes? ACM Sigsoft Software Engineering Notes, 2005,30(4):1−5. [doi: 10.1145/1082983.1083147]

[20] Bell RM, Ostrand TJ, Weyuker EJ. Looking for bugs in all the right places. In: Proc. of the Int'l Symp. on Software Testing and Analysis. New York: ACM Press, 2006. 61−72. [doi: 10.1145/1146238.1146246]

[21] Zimmermann T, Premraj R, Zeller A. Predicting defects for eclipse. In: Proc. of the 3rd Int'l Workshop on Predictor Models in Software Engineering. 2007. 9. [doi: 10.1109/PROMISE.2007.10]

[22] Aversano L, Cerulo L, Grosso CD. Learning from bug-introducing changes to prevent fault prone code. In: Proc. of the Int'l Workshop on Principles of Software Evolution. New York: ACM Press, 2007. 19−26. [doi: 10.1145/1294948.1294954]

[23] D'Ambros M. Supporting software evolution analysis with historical dependencies and defect information. In: Proc. of the Int'l Conf. on Software Maintenance. 2008. 412−415. [doi: 10.1109/ICSM.2008.4658092]

[24] Kim S, Whitehead EJ, Zhang Y. Classifying software changes: Clean or buggy? IEEE Trans. on Software Engineering, 2008,34(2): 181−196. [doi: 10.1109/TSE.2007.70773]

[25] Abreu R, Premraj R. How developer communication frequency relates to bug introducing changes. In: Proc. of the Joint Int'l Workshop on Principles of Software Evolution. New York: ACM Press, 2009. 153−158. [doi: 10.1145/1595808.1595835]

[26] Eyolfson J, Tan L, Lam P. Do time of day and developer experience affect commit bugginess? In: Proc. of the 8th IEEE Working Conf. on Mining Software Repositories. New York: ACM Press, 2011. 153−162. [doi: 10.1145/1985441.1985464]

[27] Rahman F, Devanbu PT. Ownership, experience and defects: A fine-grained study of authorship. In: Proc. of the 33rd Int'l Conf.on Software Engineering. New York: ACM Press, 2011. 491−500. [doi: 10.1145/1985793.1985860]

[28] Shivaji S, Whitehead EJ, Akella R, Kim S. Reducing features to improve code change-based bug prediction. IEEE Trans. on Software Engineering, 2013,39(4):552−569. [doi: 10.1109/TSE.2012.43]

[29] Yuan Z, Yu LL, Liu C. Bug prediction method for fine-grained source code changes. Ruan Jian Xue Bao/Journal of Software, 2014, 25(11):2499−2517 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4559.htm [doi: 10.13328/j.cnki.jos.004559]

[30] Jureczko M, Madeyski L. Towards identifying software project clusters with regard to defect prediction. In: Proc. of the 6th Int'l Conf. on Predictive Models in Software Engineering. New York: ACM Press, 2010. 9−19. [doi: 10.1145/1868328.1868342]

[31] He Z, Peters F, Menzies T, Yang Y. Learning from open-source projects: An empirical study on defect prediction. In: Proc. of the ACM/IEEE Int'l Symp. on Empirical Software Engineering and Measurement. 2013. 45−54. [doi: 10.1109/ESEM.2013.20]

[32] Hall GA, Munson JC. Software evolution: Code delta and code churn. The Journal of Systems and Software, 2000,54:111−118. [doi: 10.1016/S0164-1212(00)00031-5]

[33] Menzies T, Caglayan B, He Z, Kocaguneli E, Krall J, Peters F, Turhan B. The PROMISE repository of empirical software engineering data. West Virginia University, Department of Computer Science, 2012. http://promisedata.googlecode.com

[34] Menzies T, Turhan B, Bener A, Gay G, Cukic B, Jiang Y. Implications of ceiling effects in defect predictors. In: Proc. of the 4th Int'l Workshop on Predictor Models in Software Engineering. New York: ACM Press, 2008. 47−54. [doi: 10.1145/1370788.1370801]

[35] He Z. Research on the data shift problem of software defect prediction and cost estimation [Ph.D. Thesis]. Graduate university of Chinese Academy of Sciences, 2014 (in Chinese with English abstract).

[36] Han J, Kamber M Wrote; Fan M, Meng XF, Trans. Data Mining: Concepts and Techniques. 2nd ed., Beijing: China Machine Press, 2007 (in Chinese).

[37] Quinlan JR. Introduction of decision trees. Machine Learning, 1986,1:81−106.

[38] Breiman L. Random forests. Machine Learning, 2001,45(1):5−32. [doi: 10.1023/A:1010933404324]

[39] Lessmann S, Baesens B, Mues C, Pietsch S. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. IEEE Trans. on Software Engineering, 2008,34(4):485−496. [doi: 10.1109/TSE.2008.35]

[40] Menzies T, Greenwald J, Frank A. Data mining static code attributes to learn defect predictors. IEEE Trans. on Software Engineering, 2007,33(1):2−13. [doi: 10.1109/TSE.2007.256941]

[41] Turhan B, Bener A, Menzies T. Regularities in learning defect predictors. In: Proc. of the 11th Int'l Conf. on Product Focused Software Development and Process Improvement. 2010. 116−130. [doi: 10.1007/978-3-642-13792-1_11]

[42] Rahman F, Posnett D, Devanbu P. Recalling the "imprecision" of cross-project defect prediction. In: Proc. of the ACM SIGSOFT Symp. on Foundations of Software Engineering. 2012. 61. [doi: 10.1145/2393596.2393669]

[43] Rahman F, Devanbu P. How, and why, process metrics are better. In: Proc. of the 35th Int'l Conf. on Software Engineering. 2013. 432−441. [doi: 10.1109/ICSE.2013.6606589]

[44] Zimmermann T, Nagappan N, Gall H. Cross-Project defect prediction: A large scale experiment on data vs. domain vs. process. In: Proc. of the 7th Joint Meeting of the European Software Engineering Conf. and the ACM SIGSOFT Symp. on the Foundations of Software Engineering. 2009. 91−100. [doi: 10.1145/1595696.1595713]

[45] Ma Y, Luo GC, Zeng X, Chen A. Transfer learning for cross-company software defect prediction. Information and Software Technology, 2012,54(3):248−256. [doi: 10.1016/j.infsof.2011.09.007]

[46] Seliya N, Khoshgoftaar TM, Hulse JV. Predicting faults in high assurance software. In: Proc. of the 12th Symp. on High-Assurance Systems Engineering. 2010. 26−34. [doi: 10.1109/HASE.2010.29]

[47] Rodriguez D, Ruiz R, Cuadrado-Gallego J, Aguilar-Ruiz J. Detecting fault modules applying feature selection to classifiers. In: Proc. of the IEEE Conf. on Information Reuse and Integration. 2007. 667−672. [doi: 10.1109/IRI.2007.4296696]

[48] Ekanayake J, Tappolet J, Gall H, Bernstein A. Tracking concept drift of software projects using defect prediction quality. In: Proc. of the 6th IEEE Working Conf. on Mining Software Repositories. Washington: IEEE Computer Society, 2009. 51−60. [doi: 10.1109/MSR.2009.5069480]

[49] Jiang Y, Cukic B, Ma Y. Techniques for evaluating fault prediction models. Journal of Empirical Software Engineering, 2008,13(5): 561−595. [doi: 10.1007/s10664-008-9079-3]

[50] Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH. The WEKA: A data mining software: An update. SIGKDD Explorations, 2009,11(1):10−18. [doi: 10.1145/1656274.1656278]

[51] Rodriguez D, Ruiz R, Cuadrado-Gallego J, Aguilarruiz J, Garre M. Attribute selection in software engineering datasets for detecting fault modules. In: Proc. of the 33rd EUROMICRO Conf. on Software Engineering and Advanced Applications. 2007. 418−423. [doi: 10.1109/EUROMICRO.2007.20]

[52] Hall M. A correlation-based feature selection for machine learning [Ph.D. Thesis]. Hamilton: The University of Waikato, 1999.

附中文参考文献:

[9] 王丹丹,软件缺陷产生和移除的因素分析及预测技术研究[博士学位论文].北京:中国科学院研究生院,2014.

[29] 原子,于莉莉,刘超.面向细粒度源代码变更的缺陷预测方法.软件学报,2012,25(11):2499−2517. http://www.jos.org.cn/1000-9825/4559.htm [doi: 10.13328/j.cnki.jos.004559]

[35] 何治民.软件缺陷预测和成本估算中数据漂移问题的研究[博士学位论文].北京:中国科学院研究生院,2014.

[36] Han J, Kamber M,著;范明,孟小峰,译.数据挖掘:概念与技术.第 2 版,北京:机械工业出版社,2007.

王丹丹(1985－),女,山东枣庄人,博士,助理研究员,主要研究领域为缺陷数据分析与预测.

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方法与技术,经验软件工程.