软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(5):1422−1450 [doi: 10.13328/j.cnki.jos.005520]

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

对软件工程中经验研究的调查*

张 莉[1,2], 蒲梦媛[1], 刘奕君[1], 田家豪[2], 岳 涛[3], 蒋 竞[2]

[1](北京航空航天大学 软件学院,北京 100191)

[2](北京航空航天大学 计算机学院,北京 100191)

[3](Simula Research Laboratory, University of Oslo, Norway)

通讯作者: 蒋竞, E-mail: jiangjing@buaa.edu.cn

摘 要: 为了描述、理解、评估、预测、控制、管理或者改善与软件相关的内容,研究者常常使用经验研究的方法.经验研究在软件工程领域已经得到广泛的应用并备受关注.为了了解近年来软件工程中经验研究的特点,并希望经验研究方法为更多研究者所了解,通过系统映射的方法,对软件工程中经验研究的典型期刊《Empirical Software Engineering》(ESE)近 5 年的论文做了调研,搜集了 2013 年 1 月~2017 年 6 月发表在该期刊的 250 篇论文.通过定性和定量的分析,给出了软件工程领域采用经验研究的主要目的、常用的经验研究方法以及这些方法在软件工程各个领域中的使用情况和呈现的一些新特征.之后,分析了经验研究的主要数据来源、采集手段、常用的数理统计方法以及开源项目在经验研究中的使用情况等,给出了研究者对有效性和可重现性问题的关心程度.最后进行了有效性分析,并进一步探讨了经验研究的发展方向和大数据时代下经验研究面临的机遇和一些开放性问题.

关键词: 经验方法;文献调研;经验软件工程

中图法分类号: TP311

中文引用格式: 张莉,蒲梦媛,刘奕君,田家豪,岳涛,蒋竞.对软件工程中经验研究的调查.软件学报,2018,29(5):1422−1450. http://www.jos.org.cn/1000-9825/5520.htm

英文引用格式: Zhang L, Pu MY, Liu YJ, Tian JH, Yue T, Jiang J. Investigation of empirical researches in software engineering. Ruan Jian Xue Bao/Journal of Software, 2018,29(5):1422−1450 (in Chinese). http://www.jos.org.cn/1000-9825/5520.htm

Investigation of Empirical Researches in Software Engineering

ZHANG Li[1,2], PU Meng-Yuan[1], LIU Yi-Jun[1], TIAN Jia-Hao[2], YUE Tao[3], JIANG Jing[2]

[1](School of Software, BeiHang University, Beijing 100191, China)

[2](School of Computer Science, BeiHang University, Beijing 100191, China)

[3](Simula Research Laboratory, University of Oslo, Norway)

Abstract: To depict, understand, evaluate, predict, control, manage or enhance software-related artifacts, researchers and practitioners often rely on empirical methods. Empirical methods have been widely used in software engineering, and they are attracting increasing attention over the years. By conducting a systematic mapping, this paper aims to provide a literature survey of 250 papers published in a typical journal—Empirical Software Engineering, from January 2013 to June 2017. With qualitative and quantitative analysis, this survey reveals the commonly used empirical research methods, research purposes, and the application of the methods in subfields of software engineering, including the solved problems and some new features. The findings also cover the use of open source projects, data source, data collection methods and commonly used mathematical statistics methods. Finally, this paper illustrates validity threats and discusses the future work, opportunity and some open issues of empirical research in the era of big data.

* 基金项目: 国家自然科学基金(61672078)
Foundation item: National Natural Science Foundation of China (61672078)
收稿时间: 2017-09-28; 修改时间: 2017-11-07; 采用时间: 2017-12-05; jos 在线出版时间: 2018-01-09
CNKI 网络优先出版: 2018-01-11 17:25:06, http://kns.cnki.net/kcms/detail/11.2560.TP.20180111.1724.018.html

Key words: empirical method; literature survey; empirical software engineering

经验研究方法(empirical research method,也译为实证研究方法)在物理学、医学、心理学以及社会科学等其他领域已经得到了广泛应用[1].1986年,Basili等人[2]率先将实验方法引入软件工程,开启了软件工程经验研究的先河.从此,经验软件工程(empirical software engineering)作为软件工程的子领域开始受到关注.10年后,1996年,Basili等人[3]在一篇综述文献中探讨了软件工程学科作为"实验学科"的本质.2005年,Kitchenham等人[4]提出了基于证据的软件工程,尤其是给出了系统文献综述的方法.近年来,几乎所有ICSE,ESEC/FSE和EMSE等期刊都有涉及经验研究的论文[5].2016年,在美国奥斯丁举办的ICSE(Int'l Conf. of Software Engineering)上,程序委员会宣布:在被录用的论文中,有关经验研究的论文共32篇,高居榜首.

随着经验研究方法的广泛应用,经验研究受到软件工程领域的广泛关注,越来越多的高校开始开设经验软件工程课程,中国计算机学会(CCF)软件工程专委会成立了经验软件工程学组,并于2016年完成了关于经验软件工程的第一本译著[6].但与此同时,经验研究也受到一些质疑:首先,中英文翻译面临的问题,中文中"经验"多少带有一些主观的色彩;而维基百科中对Empirical Reseach的解释,更强调其客观性,指出"它是一种直接或间接地通过观察或者实验获得知识的方法".物理学、医学等学科往往采用经验研究方法去探索未知和验证假设.那么,在软件工程领域,研究者们往往采用经验研究方法解决什么问题呢?采用了哪些常用的经验研究方法?这些方法在软件工程各个子领域的使用是否存在领域显著性?实验数据真的大部分都是学生做的"玩具"系统吗?随着开源社区的流行,开源数据在经验研究中所占比例确实越来越大吗?有多少研究来自工业界的真实案例?经验研究的有效性和可重现性作为经验研究的重要特性,是否得到了研究者的充分重视?这一系列问题,促使我们对软件工程中经验研究的现状进行一个较为系统的调研,从而为大家理解和使用软件工程经验研究方法提供帮助.

该调研面临的第1个问题是文章的收集和挑选问题.考虑到目前存在大量的经验研究论文分散在不同方向的期刊,难以一一搜集分析;同时,目前还不存在一个标准可以客观地界定一篇论文是否是经验研究论文(我们曾对ICSE 2016的论文进行标注,发现不同人员标注的结果并不一致).Basili等人在1996年创办的经验软件工程学术期刊《Empirical Software Engineering》(ESE)作为一种针对经验研究的软件工程期刊,比较有代表性.该期刊20多年间共发表了802篇论文(数据来源:ESE期刊官网 http://link.springer.com/journal/10664),如图1所示.从图中可以看出,2013年之后,论文数量呈快速上升趋势.2016年,ESE进入科学期刊JCR Quartile 1(Q1)分区,其影响因子(journal impact factor)达到3.275,在Computer Science,Software Engineering类别中的排名为7/106.为此,我们收集了ESE在2013年1月~2017年6月共250篇论文,作为经验软件工程代表性论文,对其进行分析整理,以期得到一些定性和定量的分析结论,为软件工程的研究者和实践者更好地了解和学习经验软件工程提供帮助.

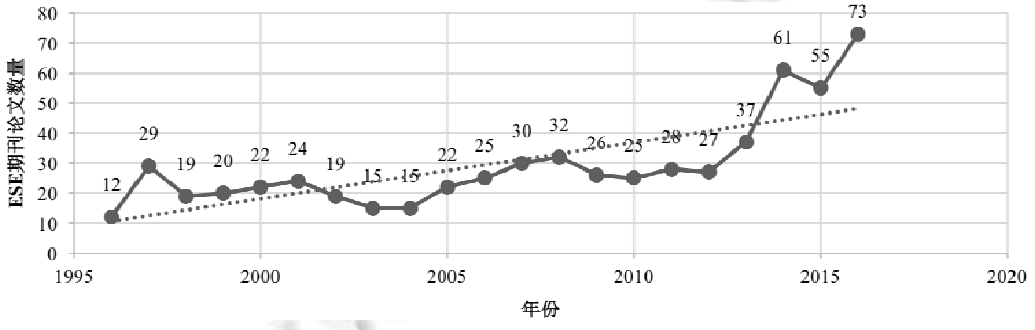


Fig.1 From 1996 to 2016 the number of ESE journal articles per year

图1 1996年~2016年的ESE期刊每年的文章数量

本文第1节提出研究问题.第2节阐述研究方法.第3节根据研究问题,对经验调查的结果进行分析.第4节

进行有效性分析和进一步讨论.第 5 节对本文进行总结.

1 研究问题

为了解经验软件工程的研究现状和更好地进行经验研究,本文提出以下研究问题(research question).

- 研究问题 1:经验研究论文的整体情况、关注的主要话题以及主要涉及了软件工程的哪些研究方向/子领域?

经验软件工程是研究如何将经验研究方法应用于软件工程领域的一种方法,即使用经验研究方法来科学地分析定量和定性的数据,理解并改进软件产品、软件开发过程和软件管理.我们希望通过对 ESE 论文的分析,来了解经验软件工程近几年关注的热点话题、涉及的软件工程子领域以及各子领域的研究话题,分析是否存在典型应用场景.

- 研究问题 2:常用的经验研究方法和经验研究的研究目的有哪些及其在各个子领域的使用情况如何?

软件工程领域常用的经验研究方法有哪些?在软件工程各个领域是否都采用了这些常用的经验方法?软件工程的经验研究用于何种研究目的?使用情况如何?是否存在领域显著性?是否存在典型应用场景?

- 研究问题 3:软件工程经验研究中数据来源、数据收集手段、数据分析方法与数据分析工具呈现什么特点?

文献[7]曾根据不同的经验策略给出了不同的数据收集方式和数据处理策略.随着互联网的广泛应用、数据挖掘等技术的快速发展、Github 等开源项目托管平台及大量开源社区的流行,软件工程中的经验研究在数据的获取和处理方面是否呈现新的形态?

为此,我们期望回答在项目数据的来源中开源项目和工业项目所占比例如何?最常用的开源项目有哪些?主要用于软件工程哪些方面的研究?研究使用的项目/案例数量的情况如何?经验研究常用的数据采集方式有哪些?经验研究最常用的数理统计方法与数据分析工具有哪些?

- 研究问题 4:经验研究人员对经验研究的有效性和可重现性问题的关注情况如何?

这个问题关注目前软件工程经验研究中研究者对经验研究有效性和可重现性的认知程度.首先是研究者对经验研究自身的有效性(validity)是否有充分的认识,是否讨论了研究结论的有效性和适用条件?哪些类别的有效性被关注得最多?

同时,经验研究本身是否是可信的?可否可通过第三方重现?研究者是否考虑了可重现性问题(possibility of replication)?有多少研究提供了可重现的资源或线索?

2 研究方法

由于目前在软件工程领域存在大量的经验研究论文,数量庞大,难以一一搜集分析,因而难以采用系统文献综述(systematic literature review,简称 SLR)方法[8];同时,本文调研的研究问题较为宽泛,更适合采用映射研究(systematic mapping)和概览研究(scoping)的方法[9].映射研究常用于为某一类研究搜索确定范围的领域,以得到某主题的当前技术发展水平或实践水平的概况.因此,我们决定借鉴系统的映射研究和概览研究的方法,即通过指定范围进行某一主题的文献调研来进行本文的调研.为此,我们不再通过关键字去搜索相关论文,而是聚焦 ESE 期刊文章(从 2013 年 1 月~2017 年 6 月).选择理由如下.

- ESE 作为经验软件工程重要国际期刊,专注于经验方法在软件工程领域的应用,其研究内容涵盖了软件工程各个领域,具有较高的研究价值和代表性,能够从一定程度上反映经验软件工程的特性.
- ESE 期刊出版 20 年了,每年有 6 期,大部分文章篇幅在 30 页~40 页左右.ESE 在软件工程领域中影响因子较高,最近 5 年一直处于期刊引证报告(journal citation report)的 Q1 区或者 Q2 区.
- 根据图 1 的统计数据,2013 年之后,文章数据增长较快.我们选取了 2013 年 1 月~2017 年 6 月的共 250 篇论文,可代表最近 5 年经验软件工程的研究特点.

本文的研究方法和过程如图 2 所示.

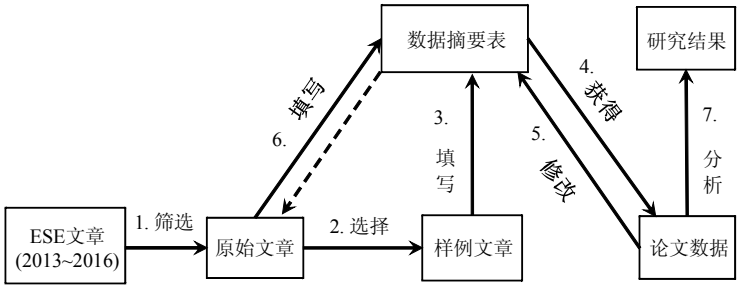


Fig.2 Process of using the research method and forming the data extraction table

图 2 文献调研方法和数据摘要表形成过程

- 第 1 步,我们对每篇文章进行了筛选,排除了期刊编辑写的文章、导论、感谢等,确保文章是进行经验研究的文章,经统计一共 250 篇.针对前面的问题设计了初步的数据摘要表后,迭代完善数据摘要表.
- 第 2 步是在原始文章中随机选取样例文章.
- 第 3 步是填写数据摘要表.
- 第 4 步得到样例文章的数据后进行分析.
- 第 5 步对数据摘要表进行修改.

然后重复第 2 步~第 5 步,经过反复多次的迭代后,形成最终的数据摘要表,见表 1.

Table 1 Data extraction table

表 1 数据摘要表

数据项	具体内容	备注
文章来源和时间	填写具体期刊或会议名称,以及出版日期	
文章名称	文章的标题	
作者	作者的姓名	
研究领域	□软件需求 □软件设计 □软件构造 □软件测试 □软件维护 □软件配置管理 □软件工程管理 □软件工程过程 □软件质量 □软件工程模型和方法 □软件工程职业实践 □软件工程经济学 □其他	多选
研究话题	参照 SWEBOK 的子话题,在标题和摘要中摘取研究话题	
使用的经验方法	□受控实验(controlled experiment) □准实验(quasi experiment) □案例研究(case study) □重现研究(replication research) □仿真实验(simulation) □调查研究(survey) □文献综述(literature review) □系统映射研究(systematic mapping) □试点研究(pilot study) □系统文献综述(systematic literature review) □其他	多选
文章研究目的	□解释性研究 □探索性研究 □技术验证 □ 其他	多选
数据来源	□工业项目 □开源项目 □开放测试集/数据集 □工业基准 □实验室项目 □其他	多选
数据采集手段	□问卷信息 □访谈信息 □观察信息 □归档数据 □预处理 □其他	多选
数据处理和分析方法	在文章中所有处理和分析数据的方法	
使用项目名称和工具	在文章中所有使用项目名称、软件和数学工具	
分析结果使用的图表	在文章结果分析中所有使用的图表	
有效性分析	○没有提到有效性威胁 ○有提到有效性威胁(包括 weakness,disadvantage,limitation 等词) ○进行了有效性分析,并包括: □结构有效 □内部有效 □外部有效 □结论有效	○为单选 □为多选
对可重现性的考虑	○提供重现包,并将实验或案例研究的重现资源发布至网上 ○提供访谈/问卷的问题列表,或综述的原始文章列表 ○提供数据来源的网址 ○在之前的研究中可以找到重现资源 ○可以向文章作者索要重现资源 ○有明确的理由,不能获取数据 ○没有提及数据的可获取性 ○其他	单选

- 然后进行第 6 步,填写全部文章的数据摘要表后获得论文数据.
- 第 7 步是通过统计等方法进行分析,形成研究结果.

本文共选择 90 多篇 ESE 样例文章进行试读.为了保证数据收集的客观性,每篇文章由第 2 作者、第 3 作者及部分自愿者分别阅读和填写数据摘要表,然后对摘取结果进行比对.如果摘取的内容不同,则由全体作者一起讨论确定.这个方法也是在社会科学领域广泛应用[10].完成所有的数据摘取和核对后,再对摘取结果进行统计和分析.每一个研究问题的数据收集、统计和分析都有明确的过程,因而确保了可重现性.

如表 1 所示,在软件工程研究领域分类方面,参考了 IEEE 计算机协会最新发布的软件工程知识体系第 3 版(guide to the software engineering body of knowledge version 3.0,简称 SWEBOK)[11],将软件工程研究划分为 12 个子领域,分别是软件需求(software requirement)、软件设计(software design)、软件构造(software construction)、软件测试(software testing)、软件维护(software maintenance)、软件配置管理(software configuration management)、软件工程管理(software engineering management)、软件工程过程(software engineering process)、软件工程模型和方法(software engineering models and methods)、软件质量(software quality)、软件工程职业实践(software engineering professional practice)、软件工程经济学(software engineering economics).如有难以判断研究子领域的文章,归为其他类,之后再通过全体作者讨论确定.当然,按照这个分类,软件工程研究也可能同时覆盖多个子领域.例如,文献[12]研究作为需求规范的主要质量属性之一的完整性,通过实验对比了两种完整性验证方法的结果.根据 SWEBOK,研究内容涉及需求规范(requirement specification)划分到软件需求,同时研究内容涉及软件质量需求(software quality requirement)划分到软件质量.这种交叉在任何一种分类中都是难以避免的.参考了 SWEBOK 中对知识领域和知识单元的定义,通过阅读论文标题和摘要,获取了论文的研究话题.

对于文章研究目的,通过试读,我们发现有解释性研究、探索性研究和技术验证.用数据解释某一现象或结果,称为解释性研究.探索性地发现某一现象、问题或规律,称为探索性研究.用于评估新提出的技术(包括策略、算法、模型、方法、工具等)称为技术验证.如有其他研究目标,归为其他类,并给出说明.

对于文章使用的经验方法,我们主要根据论文作者声称的研究方法进行摘取和判断.通过对样例文章进行试读,我们发现作者使用到的经验研究方法有受控实验、准实验、案例研究法、调查研究、仿真实验、重现研究、系统文献综述、系统映射研究、试点研究、文献综述.如果有使用未列举的经验方法,则归为其他类,并给出说明.

为了了解经验研究的数据来源,通过试读发现,进行经验研究的数据来源有工业项目、开源项目、开放测试集/数据集、工业基准、实验室项目.如有其他数据来源,我们归为其他.研究者数据采集手段有问卷信息、访谈信息、观察信息、归档数据、预处理,如有其他数据采集手段,我们归为其他.为了了解经验研究在数据的处理和分析方面,我们摘取了每篇文章的数据处理和分析方法或工具以及在分析结果时使用的统计图标,例如直方图、饼图.

在经验研究中,另一个备受关注的问题是有效性[5].有效性评价是由研究者自行衡量研究结果的一种方式,有效性可以细分为结构有效性、内部有效性、外部有效性、结论有效性(可靠性)[13].

可重现性是经验研究内在蕴含的重要性质,经验社区的研究者逐渐意识到可重现性的重要性[14].一篇经验研究若考虑了可重现性,就增强了经验研究的可信性.同时,本文研究的 250 篇 ESE 期刊原始文章列表和数据见网站(http://t.beihangsoft.cn/lily/ch/index.html)的实验数据,以供研究人员参考.

3 结果与讨论

在这部分,通过阅读和摘取最近 5 年的 ESE 期刊的文章,我们统计、分析并呈现了文献调研的结果.这些结果是根据之前第 1 节提出的研究问题而组织的.

3.1 研究问题1:经验研究论文的整体情况、关注的主要话题以及主要涉及了软件工程的哪些研究方向/子领域?

3.1.1 2013 年~2017 年期间 ESE 期刊论文的整体情况

首先,我们关注经验研究的整体情况.从图 1 我们可以看出,ESE 期刊发表的论文从 2013 年开始,数量有了大幅度的增加,每期论文数量如图 3 所示(注:图中*全为专刊,**为包含专刊的正刊),在 2016 年 6 月达到了 19 篇(其中包括《Special Section on Software Reverse Engineering》中的 6 篇),所以整体稳定在 10 篇左右.其中,包括

15 次专刊(special issue),部分专刊包含在正刊中,见表 2.整体上看,专刊中文章数量在 5 篇左右.

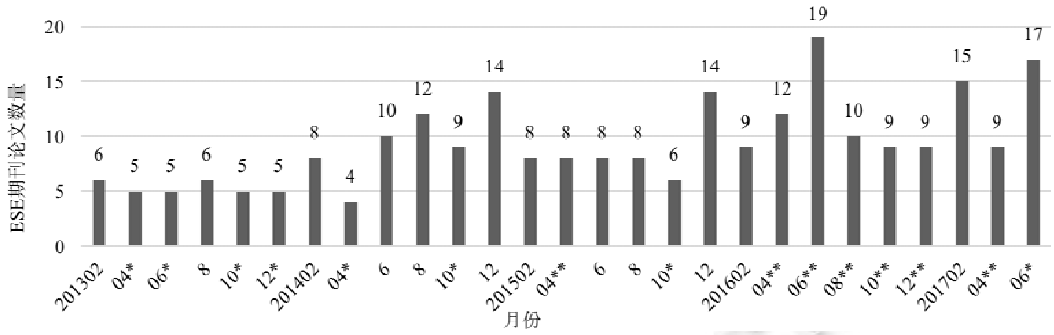


Fig.3 From January 2013 to June 2017, the number of articles per issue in the ESE journal

图 3 2013 年 1 月~2017 年 6 月,ESE 期刊每期的文章数量

Table 2 Number of articles on the special issues of ESE journal from January 2013 to June 2017

表 2 2013 年 1 月~2017 年 6 月,ESE 期刊的专刊的文章数量

时间	主题	专刊文章数量/总量
201304	Program comprehension	5/5
201306	Predictive models, search-based software engineering	5/5
201310	Reverse engineering	5/5
201312	Mining software repositories	5/5
201404	Replication	4/4
201410	Global software engineering, program comprehension, search-based software engineering	9/9
201504	Software maintenance and evolution	5/8
201510	Software maintenance and evolution research	6/6
201604	Mining software repositories, empirical evidence on software product line engineering	5/12
201606	Software reverse engineering	6/19
201608	Empirical evidence on software product line engineering	6/10
201610	Mining software repositories	5/9
201612	Search-based software engineering	3/9
201704	Research in search-based software engineering	3/9
201706	Software and systems traceability, mining software repositories, program comprehension	17/17

从表 2 的专刊列表中我们可以看出,近几年,软件工程经验研究关注的热点方向有程序理解、逆向工程、基于搜索的软件工程、软件仓库挖掘、软件维护和演化、软件产品线工程等,其中,15 期专刊有 4 次涉及基于搜索的软件工程、软件仓库挖掘,3 次涉及程序理解,2 次涉及软件维护与演化、逆向工程、产品线等.

3.1.2 经验方法在软件工程各子领域的使用情况

如前所述,关于软件工程的子领域有不同的划分方法,本文参考 SWEBOK[11],将软件工程分为 12 个子领域.如果一篇文章研究的软件工程问题不在这 12 个子领域中,则划分为其他子领域.在对 ESE 文章进行子领域分类时,一篇文章可能涉及 1 个或多个子领域.例如,文献[15]研究如何优化敏捷方法以适应软件维护的实践.根据 SWEBOK,软件维护实践属于软件维护子领域,敏捷方法属于软件工程过程,则把其同时划分到软件维护和软件工程过程子领域中.得到图 4(按数量排序).

从图 4 可以看出,经验研究的论文覆盖了软件工程所有子领域.

- 软件维护是经验研究应用最多的子领域,有 80 篇(占 32%).这与 ESE 期刊有 5 期专刊涉及程序理解、软件维护和逆向工程有关.
- 其次是软件质量和软件测试,各有 41 篇(占 16%).
- 文章数量较少的子领域分别是软件工程过程(4%)、软件设计(4%)、软件工程经济学(4%)、软件配置管理(3%).
- 涉及其他子领域的文章有 4 篇,分别是:文献[16]对软件工程的重现研究进行了一个映射研究;文献[17]

提出了一种 Alitheia Core 平台工具,使用真实的工业项目评估该工具;文献[18]提出了一个辅助筛选原始文章的半自动化工具;文献[19]讨论了经验方法的改进.

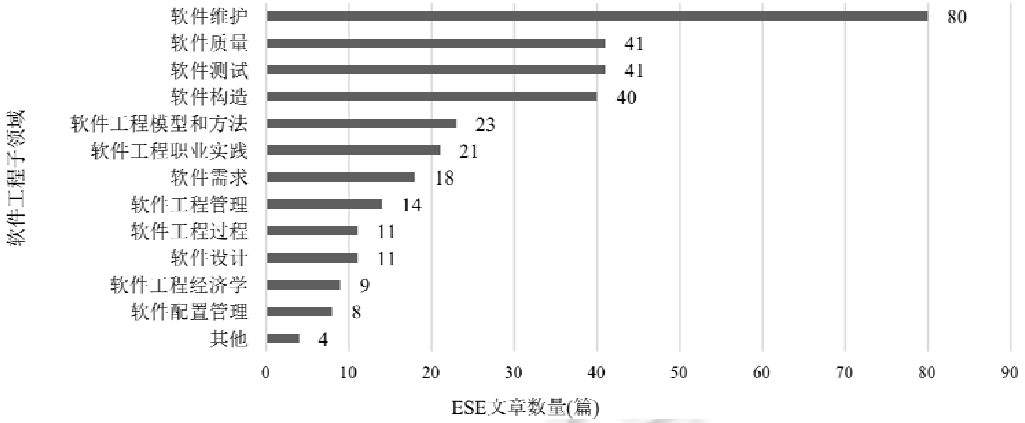


Fig.4 Number of articles in subfields of software engineering

图 4 软件工程子领域的论文数量

为了进一步了解采用经验研究的研究方向,我们参考了 SWEBOK 中对于知识领域和知识单元的定义,摘取了论文的研究子话题,并得到表 3.

Table 3 Research subfields and research topics of ESE journal

表 3 ESE 期刊文章的研究子领域及研究主题

研究子领域	研究子话题	相关文献	文章数/百分比
软件需求	需求过程和管理、需求捕获、需求分析、需求规约、需求验证、需求追踪等	[20–26]	18/7%
软件设计	软件体系结构设计、UI 设计、软件设计质量分析与评估、软件设计策略和重用(面向对象的类重用、设计模式)等	[27–30]	11/4%
软件构造	代码分析与度量、代码重构、构造技术(API 设计和使用、编码方法、编程语言等)、软件构造工具、编程实践等	[31–41]	40/16%
软件测试	测试技术和方法、测试技术和方法的评估与验证、测试需求分析、测试用例选择、缺陷分析、缺陷分析和评估,测试过程、测试手段与工具等	[42–52]	41/16%
软件维护	代码变更与软件演化、软件质量预测、重构、缺陷管理和预测、程序理解、逆向工程等	[53–63]	80/32%
软件配置管理	软件发布管理、软件配置管理工具、软件配置管理过程管理、软件配置控制等	[64–69]	8/3%
软件工程管理	可行性分析、软件项目计划(工作量评估、项目风险管理)、开源社区管理、全球软件工程中的项目管理等	[70–76]	14/6%
软件工程过程	软件过程管理、软件过程实践(如敏捷实践)、软件过程改善、软件过程度量等	[77–84]	11/4%
软件工程模型和方法	建模方法和建模质量分析、基于搜索的软件工程、模型驱动的软件工程、软件产品线模型等	[85–92]	23/9%
软件质量	软件安全、软件质量管理过程(软件质量保证、验收和验证)、软件质量实践(软件质量管理技术、缺陷特征)、软件质量工具等	[93–101]	41/16%
软件工程职业实践	开发者行为分析、开发者观点、软件团队构建、提高编程技能、博客与开发过程、软件开发的绩效、社交活动对开源项目成功的影响、人的因素在软件开发中的影响等	[102–110]	21/8%
软件工程经济学	软件工程经济学基础(效率、生产力)、生命周期经济学、风险和不确定性、经济学分析方法(软件外包、软件经济成本估计、制定准确性和可靠性进行评估)等	[111–117]	9/4%
其他	重现研究、挖掘软件版本库平台工具、半自动化的文献综述工具	[16–18]	4/1%

综上所述,ESE 期刊在 2013 年 1 月~2017 年 6 月,每年论文总量呈增长趋势,研究领域覆盖了软件工程全部

子领域(参考 SEWBOK 分类)及软件生命周期全过程在软件维护、软件质量、软件测试、软件构造子领域的经验研究论文数量较多.15 期专刊有 4 次涉及基于搜索的软件工程、软件仓库挖掘,3 次涉及程序理解,2 次涉及软件维护与演化、逆向工程、产品线等,也说明这些子领域有可能较多地采用了经验研究的方法.

3.2 研究问题2:常用的经验研究方法和经验研究的研究目的有哪些及其在各个子领域的使用情况如何?

3.2.1 软件工程中的经验研究方法

在 250 篇文章中,有 9 种经验研究方法被使用,见表 4.其中,

- 实验方法在 145 篇文章中被使用,占比 58%.有 126 篇论文采用了受控实验方法;其他为准实验方法,其主要用于无法随机分配主体的实验[118].
- 案例研究方法的使用总量次之,在 86 篇文章中被使用,占比 35%.软件工程中的案例研究(case study)是针对真实环境中当前存在的某种现象进行调查研究的一种经验方法[119].关于案例研究和实验,在一些论文中的区分并不清晰,我们主要采用了源论文自己的判断.在存在二义性时,我们的原则是:若存在人为的控制变量,则归为实验研究;否则归为案例研究.例如,文献[120]提出一种检测和分类剥离二进制文件中使用的高级数据结构的工具,在 10 个现实世界应用程序中评估该工具的精度.文章声称其使用案例研究方法,即通过 10 个案例进行研究.但我们认为,该文章是使用实验方法进行技术验证.
- 再其次是文献综述法,有 21 篇文章使用了文献综述法,占比 8%.文献综述法是基于出版物的经验研究,是一种二次研究方法(secondary study)[121].其中,采用了系统文献综述方法的有 11 篇,使用系统映射研究[122]方法的有 6 篇文章.
- 有 20 篇文章使用调查研究法,占比 8%.调查研究法是一种通过收集来自于人或者与人有关的信息来描述、比较或者解释人们的知识、态度和行动的方法[123].调查法常见的形式有问卷调查、访谈等[7].调查研究作为一种独立完整的方法,应包括调查研究的目的、设计、实施、数据分析、形成结论、有效性分析等过程.如果在其他方法中仅仅采用了调查问卷等形式采集数据,不形成结论,则不被认为是调查研究方法,而仅仅作为数据收集的一种手段,将在下一节讨论.
- 重现研究[124]在 8 篇文章中,包括在重现研究专刊中的 5 篇.有 5 篇文章使用了试点研究,试点研究是指在小范围真实环境中进行的研究[125].例如,文献[126]在进行大规模的调查研究之前先进行一个小规模的试点研究,以确保大规模调查的有效性和可行性.文献[18]提出了一个自动选择部分综述文章的策略(SCAS),随后,应用一个小型的试点研究来评估该策略的精度和误差.有 3 篇文章使用仿真实验方法,常用于难以实际进行实验的情况.

Table 4 Empirical methods and usage quantity in ESE journal

表 4 在 ESE 期刊中的经验方法和使用数量

编号	名称	文章数量	研究性质	备注
1	实验方法	145	原始研究	126 篇为受控实验,其他为准实验
2	案例研究	86	原始研究	–
3	文献综述法	21	二次研究	11 篇使用系统文献综述方法,6 篇文章使用系统映射研究方法
4	调查研究法	20	原始研究	–
5	重现研究	8	二次研究	–
6	试点研究	5	原始研究	–
7	仿真实验	3	原始研究	–
8	其他	1	–	行动研究(action research)

我们发现,除了数据摘要表中列出的这几种经验方法外,其他经验方法还有 1 篇,即行动研究,研究人员作为实际项目的参与者进行的研究[15].

文献综述法自 2005 年 Kitchenham 将其引入软件工程领域,得到了广泛的关注.除了这里的 21 篇论文以外,还根据其关注的领域广泛分布于不同的期刊和会议(见第 4 节进一步讨论),文献综述法已成为软件工程领域的常用的经验方法之一.

由表 4 可以看出,在原始研究中,实验研究最多,案例研究次之,调查研究再次之,三者占了文章总数的 95%.虽然调查研究法相对较少,但问卷和访谈作为调查研究中常用的数据采集手段,累计在 100 多篇文章中出现(见第 3.3.2 节数据采集手段).我们认为这 3 种方法都非常重要,是经验研究者必须掌握的方法.这 3 种方法在 2013 年~2017 年的 250 篇文章中占比如图 5 所示,无特定趋势,总体上趋于均衡,说明这 3 种方法一直是经验研究很常用的方法,也是学习经验研究最应该掌握的方法.

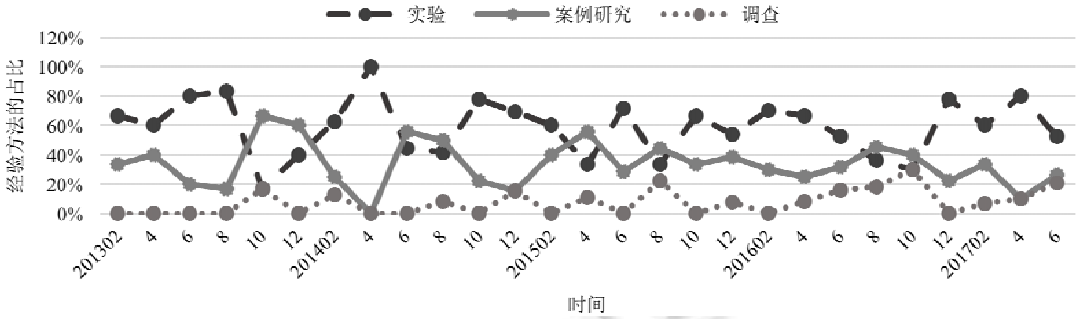


Fig.5 Percentage trends of experiment, case study and survey

图 5 实验、案例研究、调查的百分比趋势图

3.2.2 不同子领域在采用经验研究方法上的特点

我们分析统计了不同的经验方法在各个子领域的使用情况,结果见表 5.所有的软件工程子领域都使用了实验和案例研究方法.

Table 5 Different empirical methods and the usage in software engineering subfields

表 5 不同经验方法在软件工程子领域的使用情况

经验方法	软件需求	软件设计	软件构造	软件测试	软件维护	软件配置管理	软件工程管理	软件工程过程	软件工程模型和方法	软件质量	软件工程职业实践	软件工程经济学	其他
实验	15	8	23	24	51	4	9	3	11	25	6	3	2
案例研究	2	3	18	18	24	4	6	7	4	16	12	5	1
调查	2	0	7	4	7	0	1	1	2	0	5	1	0
综述	1	2	2	1	4	0	1	2	10	2	1	0	1
重现研究	1	0	0	3	1	0	0	0	0	2	1	0	0
试点研究	1	0	1	3	2	0	0	0	0	0	0	0	0
仿真实验	1	1	0	0	2	0	0	0	0	1	0	0	0
其他	0	0	0	0	0	0	0	0	0	0	1	0	0

采用实验方法的文章较多地应用在在软件维护、软件质量、软件测试、软件构造、软件需求、软件工程模型和方法等子领域中.在这类子领域中,可变因素相对于易于控制、易于获取的历史数据(如软件版本库、缺陷报告等)来做实验,以评估新方法和技术的有效程度.例如,文献[52]使用开源浏览器 Firefox 的数据做实验,评估错误预测方法能否预测脆弱性(vulnerability).文献[53]将一种文本挖掘技术应用到开源软件的缺陷报告中,通过实验对 5 个开源软件的缺陷报告进行挖掘,以研究统计重复提交缺陷的数量,并和人工统计做对比.软件需求子领域中,实验方法使用得最多.例如,文献[127]通过实验方法对两种新的自动化需求文档的满意度评估技术进行对比.在软件需求子领域中,方法的评估主观性比较大,大部分实验无法依靠纯技术的方法进行,需要辅助问卷或访谈的形式来获得实验结果.又如,文献[128]研究个人因素对需求审查(requirements inspection)的影响,实验人员向参与者提供实验材料,并让参加者找到需求文档的错误并将其记录在错误列表中.

在软件工程过程、软件工程管理、软件工程职业实践和软件工程经济学子领域中,采用案例研究方法的文章数量较多.这些子领域的研究具有由于受到多因素的影响,难以独立控制多种因素、周期长、难以仿真实验、

成本高等特点,因此与实验相比,案例研究方法放宽了对多种因素的控制,更易被采用.例如,文献[79]通过调查问卷和访谈等方式采集了横跨欧洲、亚洲和美洲的 66 个真实的跨地区全球性软件开发项目的数据,以对比采用敏捷流程(scrum,XP 等)和结构化流程(RUP、瀑布)在分布式全球化软件项目的差异.没有采用实验方法,是因为在真实的项目中,实验方法难以控制除软件工程流程外的其他因素.虽然在软件工程过程中实验方法比较少,但是也有一些应用场景.文献[80]首先提出了一种使用机器学习技术的新型半自动化软件过程评估方法,为了验证该方法的有效性,采用了对比实验的方法,即将新方法用于评估九个实际的工业软件项目中缺陷管理流程,并将结果与现有常规方法进行比较.

调查研究法的文章数量少于实验和案例研究方法,较多地使用在软件构造和软件维护子领域.调查法的结果也受到很多因素的影响(如参与者的选择),调查法常用于探索某一话题的定性研究,能够快速地了解参与者对该话题的态度和看法.例如,文献[61]研究对重构的实践和态度的行业调查.调查研究结果显示,参与者对重构看法不一,并提出目前重构存在的问题,以期待今后的研究进行解决.

仿真实验和试点研究并无明显的领域显著性.软件需求、软件设计、软件维护、软件质量子领域都使用了仿真实验.试点研究使用在软件需求、软件构造、软件测试、软件维护子领域中.例如,文献[129]研究测试用例的选择,评估不同信息检索方法的性能实验,实验前先进行一个试点实验来探索信息检索方法是否可以用于测试用例的选择.

3.2.3 经验研究的研究目的及其常用方法

经过对 250 篇 ESE 期刊文章的分析发现,有 49%(122 篇)的文章的研究目的属于探索性(exploratory),比如,通过观察现象,或基于证据或数据,发现问题或可能存在的规律;有 43%(108 篇)的文章是为了进行技术验证(evaluation),即用于评估新提出的策略、算法、模型、方法、工具等;有 11 篇文章是解释性研究(explanation),比如软件工程存在的因果关系.此外,有 7 篇文章先进行探索性研究,再进行技术验证.

如图 6 所示,在探索性研究中,案例研究方法使用的最多,如文献[71]从 Linux 开源案例中探索开发者的工作量和集中开发时间的关系.在技术验证中,实验方法使用得最多,常用于对新提出的方法进行技术验证,对已有方法进行再次验证或比较,如文献[42,45].解释性研究用于解释某一现象、规律的因果关系,如文献[128,130];有 7 篇文章先使用探索性研究,用于发现某一问题,提出一种解决方法后,再进行技术验证,如文献[131,132].

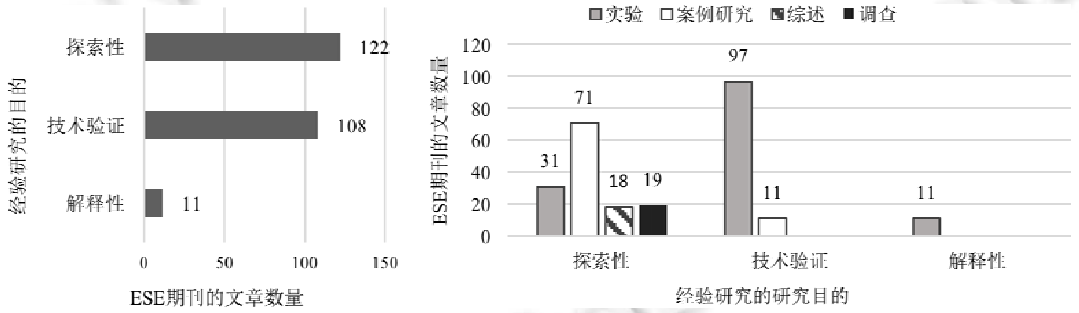


Fig.6 Purpose of empirical research

图 6 经验研究的目的

在进行经验研究时,研究者有时会组合使用多种经验方法以达到不同的研究目的,例如,

- 有 5 篇文章组合使用调查法和案例研究.例如,文献[133]首先进行调查研究法以了解开发者对开源平台上的自动推荐工具等的看法,然后,通过案例研究采集 Github 上的数据以验证开发者的实际行为和看法的一致性.
- 有 4 篇文章组合使用综述和调查法,例如,文献[103]首先使用系统文献综述探究全球软件工程中新术语遇到的问题,如新术语的概念与分类不清楚等,再通过对全球软件专家进行访谈等调查研究方法来解决综述发现的问题.

- 有 1 篇经验研究组合使用案例研究和实验方法.文献[34]对一个大型 Smalltalk 代码库进行案例研究,发现动态特征分类中存在的问题;然后提出一种自动动态分类方法,并使用实验方法对新方法进行了技术验证.
- 有 1 篇文章使用综述、实验和案例研究方法.文献[134]先使用综述探索了检测克隆网页技术和工具存在的局限,通过案例研究采集了 Google、Yahoo、Twitter 的数据以分析和了解它们在克隆网页中存在的问题;然后提出一个检测克隆网页的工具,使用实验方法来评估该工具的有效性,发现比其他研究者提出的工具的精度高.

3.2.4 不同子领域使用经验方法的研究目的

为了进一步了解软件工程采用经验研究的目的,我们对各子领域的情况进行了统计,结果见表 6.从表中可以看出,所有子领域都采用了经验研究方向,进行了探索性和技术验证研究.虽然图 6 显示经验研究目的最多的是探索性研究,但在软件需求、软件设计、软件测试和软件维护等子领域中,使用经验方法进行技术验证的文章数量要多一些.而在软件构造、软件工程管理、软件工程过程、软件工程职业实践、软件工程经济学等子领域中,探索性研究的文章数量偏多.总体上可以看出,与管理相关的子领域,探索性论文偏多一些;与技术相关的领域,技术验证文章多一些.解释某一现状的因果关文章数量最多的子领域是软件质量.

Table 6 Different research purposes in the software engineering subfields

表 6 软件工程子领域的不同研究目的

经验研究目的	软件需求	软件设计	软件构造	软件测试	软件维护	软件配置管理	软件工程管理	软件工程过程	软件工程模型和方法	软件质量	软件工程职业实践	软件工程经济学	其他
探索性	4	3	24	17	32	5	9	8	15	18	18	6	2
技术验证	12	8	14	23	43	2	5	4	8	17	2	2	1
解释性	2	0	0	1	3	0	1	0	0	5	1	1	1

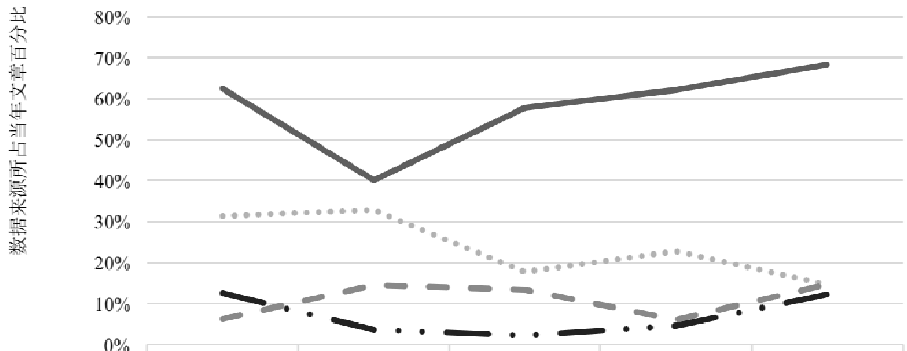
综上所述,软件工程研究中常采用经验方法,主要有实验研究、案例研究、调查研究法、综述研究等,其中使用最多的是实验方法,案例研究次之.所有的软件工程子领域都使用了实验和案例研究方法,但各个子领域使用这两种经验方法存在一定的领域差异.在因素易于控制的子领域中,实验方法较为适用;在具有受到多种因素影响且难于独立控制多种因素特点的软件工程过程等子领域,案例研究方法比实验研究更为适用.调查研究、试点研究、仿真实验无明显的领域显著性.

软件工程的经验研究主要用于探索性、技术验证、解释性等,其中,探索性研究最多,技术验证次之.总体上可以看出,与管理相关的子领域,探索性论文偏多一些;与技术相关的领域,技术验证文章多一些.大部分技术验证论文采用了实验的方法,而大部分探索性论文采用了案例研究的方法.解释性论文在软件质量领域数量最多.

3.3 研究问题3:软件工程经验研究中数据来源、数据收集手段、数据分析方法与数据分析工具呈现什么特点?

3.3.1 经验研究的数据来源和使用情况

文献综述的数据来源为公开发表的文献,本文重点关注去掉 11 篇仅使用文献综述一种研究方法的文章后余下 239 篇文章的数据来源.经调查发现,239 篇文章中的数据主要来自于开源项目、企业界的真实项目(简称“工业项目”)、科研实验室项目以及标准数据集/测试集/基准;而且发现,调查研究的数据多来自于访谈和问卷,这将在第 3.3.2 节中详细讨论.本文按年份统计了 239 篇文章的数据来源的占比情况,如图 7 所示.一篇文章可能有多种数据来源,例如,文献[135]既使用了开源项目的数据,又使用了工业项目的数据.经统计,在这 239 篇文章中,采用开源项目进行经验研究的文章占 57.3%,包括使用开源软件的代码、过程数据等,以及对开源社区开展的经验研究.在 2014 年,使用开源项目的文章数量的比例为 40%;在 2017 年上半年,使用开源项目的文章数量占 68.3%.使用开源项目的比例呈明显上升趋势.采用工业项目进行研究的文章占比 23.8%,总体上有下降趋势.有 10.9%文章采用了实验室项目,还有少量研究采用了标准数据集或者公开的测试集/基准.这说明在软件工程领域,可供研究人员进行研究的标准数据集和测试集还是非常缺乏的.

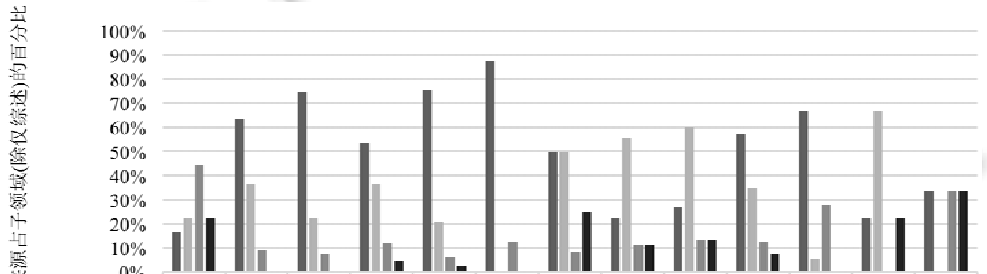


	2013	2014	2015	2016	2017（上半年）
开源项目	62.5%	40.0%	57.8%	62.1%	68.3%
工业项目	31.3%	32.7%	17.8%	22.7%	14.6%
实验室项目	6.3%	14.5%	13.3%	6.1%	14.6%
标准数据集/测试集/基准	12.5%	3.6%	2.2%	4.5%	12.2%

Fig.7 Trend of the proportion of data sources in empirical study

图 7 经验研究的数据来源比例的趋势

本文还对研究子领域和项目来源进行了交叉分析,图 8 是在去掉仅文献综述的文章后,统计的各领域不同数据来源占该领域文章数量的百分比.



	软件需求	软件设计	软件构造	软件测试	软件维护	软件配置管理	软件工程管理	软件工程过程	软件工程模型和方法	软件质量	软件工程职业实践	软件工程经济学	其他
开源项目	16.7%	63.6%	75.0%	53.7%	75.6%	87.5%	50.0%	22.2%	26.7%	57.5%	66.7%	22.2%	33.3%
工业项目	22.2%	36.4%	22.5%	36.6%	20.5%	0.0%	50.0%	55.6%	60.0%	35.0%	5.6%	66.7%	0.0%
实验室项目	44.4%	9.1%	7.5%	12.2%	6.4%	12.5%	8.3%	11.1%	13.3%	12.5%	27.8%	0.0%	33.3%
标准数据集/测试集/基准	22.2%	0.0%	0.0%	4.9%	2.6%	0.0%	25.0%	11.1%	13.3%	7.5%	0.0%	22.2%	33.3%

Fig.8 Research subfields and data sources

图 8 研究子领域与数据来源

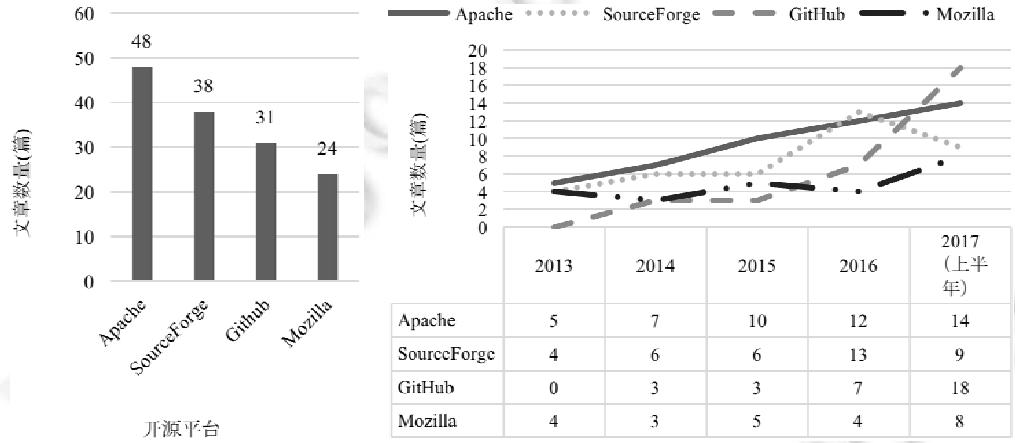
从图 8 中可以看出,采用开源项目进行研究的文章数量占比最多的子领域有软件设计、软件构造、软件测试、软件维护、软件配置管理、软件质量和软件工程职业实践;较多使用工业项目作为数据来源的子领域有软件工程过程、软件工程模型和方法以及软件工程经济学;而在软件工程管理子领域,这二者几乎持平.由此可见,关注成本、过程以及新的开发模型和方法的研究偏向于使用工业项目作为数据来源;而软件配置管理、软件测试、维护以及构造等有关的数据更容易从开源项目中获取.这是由于开源软件自身的使命和特点决定的,因为

开源软件是由地理和时间分散的、不同背景的贡献者共同开发的,因而软件构造、维护和配置管理是目前开源项目最为关注的子领域,同时也最易获得大量的软件工程实践数据.在两类项目中,均可能获得测试相关数据.选择实验室项目作为数据来源的经验研究较少,一是难以模拟复杂的软件工程问题,二来也难以被其他研究者重现.我们还发现,采用标准数据集/测试集/基准作为数据来源的文章也较少,说明软件工程领域目前可用于研究者进行研究的标准数据集不多.大部分领域的研究者都倾向于使用开源项目进行研究:在需求领域,实验室项目偏多,标准数据集/测试集也相对丰富;软件配置管理领域没有采用工业界数据的经验研究,说明很难获得相关数据.

(1) 经验研究中采用较多的开源软件

从前面的分析可以看到,几乎所有子领域的经验研究都不同程度地采用了开源软件.这是由于随着开源软件的快速发展和开源社区的逐渐成熟,一些著名的开源软件,如 Linux 操作系统、Apache Web 服务器等,其软件质量甚至超过了一些商业竞争软件[136].同时,由于开源软件的易获取性以及便于进行重现研究,因而为越来越的研究人员所采用.为了便于大家在从事经验研究中使用开源软件,我们进一步分析了这些文章对开源软件的使用情况.

首先,我们对开源托管平台进行了统计,排在前 4 名的分别是 Apache、SourceForge、GitHub 和 Mozilla,如图 9 所示.



	2013	2014	2015	2016	2017 (上半年)
Apache	5	7	10	12	14
SourceForge	4	6	6	13	9
GitHub	0	3	3	7	18
Mozilla	4	3	5	4	8

Fig.9 Most commonly used open source platform

图 9 最常使用的开源平台

统计结果表明,使用 GitHub 的文章呈明显上升趋势,而且使用 Apache 和 Mozilla 的文章也有所增加.说明有越来越多的研究者选择使用开源平台的项目或数据进行研究.有 48 篇文章使用的数据来自于 Apache 项目.Apache 项目属于 Apache Software Foundation(ASF),ASF 是一个非营利组织并将 Apache 项目开源.Apache 项目中,常用于经验研究的有 HTTP Server、Ibatis、Tomcat、Wicket、Maven2、Ant、Derby、Xerces、HTTPD、Hadoop、Log4j、Struts、Lucene、Axis2_c、Pluto、Solr、Joda-Time、Hibernate、Cocoon、Jmeter、Cpptasks 等.有 38 篇文章中的开源项目来自 SourceForge. SourceForge 是一个开源的软件开发平台和仓库.经验软件工程的研究者从 SourceForge 中选取相应的开源软件进行经验研究.例如,文献[53]对 SourceForge 上的 FileZilla、jEdit、phpMyAdmin、Pidgin、Slash 开源软件的缺陷报告进行文本聚类分析以预测缺陷解决时间;文献[86]从 SourceForge 中随机抽取 100 个 Java 项目研究搜索算法的参数值设置问题.来自开源托管平台 GitHub 的 Decentralized Systems、MDG、JbidWatcher、Dnsjava、Closure Library、SproutCore 等开源项目被 31 篇文章使用.Mozilla 非盈利基金会的项目被 24 篇文章使用,使用的开源项目有 Firefox,Firefox 的扩展 Firebug、Rhino 等.

之后,我们进一步统计了开源项目的使用数量,得到图 10:使用最多的 10 个开源项目,分别是 Eclipse、Linux kernel、Jedit、Firefox、Lucene、JhotDraw、ArgoUML、PostgreSQL、Tomcat、HTTPD 和 Rhino.有 29 篇文章使用 Eclipse 项目的数据,如 Eclipse 源代码、版本信息、bug 报告和 Eclipse 上的开源插件进行经验研究.Linux Kernel 是一种用 C 语言写成的计算机操作系统内核,Linux 内核方法、内核源码、版本信息可作为经验研究的数据来源.JEdit 是一个用 Java 语言开发的文本编辑器,例如,文献[55]利用 JEdit 评估提出的一种用于特征定位的数据融合模型.Firefox 是一个适用于 Windows,Linux 和 MacOS X 平台的浏览器,Lucene 是一个全文检索引擎工具包,JHotDraw 是一个二维的 GUI 框架,如,文献[137]使用 JHotDraw 的数据来评估一种自动化的类重构方法.ArgoUML 是一个 UML 模型工具,PostgreSQL 是一个关系型数据库管理系统,Tomcat 是一个 Web 应用服务器,HTTPD 是 Apache 超文本传输协议(HTTP)服务器的主程序.Rhino 是一个 3D 建模软件.这些常被使用的开源项目的共同特点是文档齐全、持续更新版本、覆盖用户广、代码质量高.

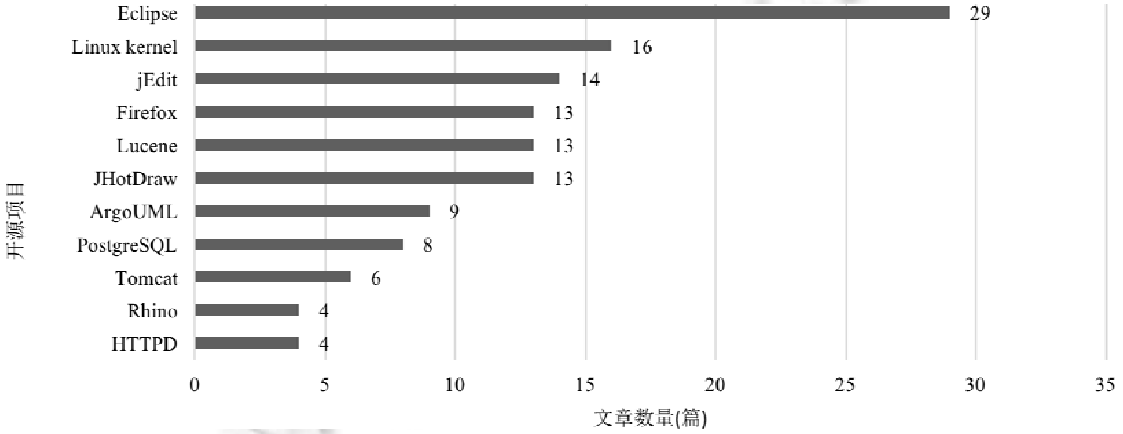


Fig.10 Most commonly used ten open source projects

图 10 最常用的 10 个开源项目

(2) 经验研究使用的项目/案例数量

无论是研究者还是实践者,都非常关心选择多少项目进行经验研究最为合适,这也是目前最具争议的话题之一.本文首先按照研究目的的不同,对研究者使用项目数量(项目数量指开源项目和工业项目之和)进行了统计,统计结果见表 7;然后再按照研究子领域对使用的项目数量进行统计,统计结果见表 8.

Table 7 Purpose of the study and the number of projects used

表 7 研究目的与使用项目的数量

项目数量	探索性	技术验证	解释性	探索性+技术验证	整体
最大值	1547	3286	21	19	3 286
最小值	1	1	1	1	1
平均数	62.13	84.74	5.9	7.43	68.07
中位数	3	3	3	6	3

Table 8 Subfileds of the study and the number of projects used

表 8 研究子领域与使用项目的数量

项目数量	软件需求	软件设计	软件构造	软件测试	软件维护	软件配置管理	软件工程管理	软件工程过程	软件工程模型和方法	软件质量	软件工程职业实践	软件工程经济学	其他
最大值	6	25	600	1 547	2 117	15	1 385	1 547	3 286	1 385	1 316	1 316	5
最小值	1	1	1	1	1	1	1	1	1	1	1	1	1
平均数	3.09	7.1	35.05	66.47	67.16	3	124.30	239.5	220.6	43.95	160	177.66	3
中位数	2	2	3	3	4	1	7	4	1	3	2	4	3

在统计研究者使用项目数量时我们发现:

- 探索性研究中,文献[67]使用的项目最多,达到了 10 713 个项目.研究者通过对 GooglePlay 中 10 713 个移动应用的更新频率的统计发现了一定的规律,从而给开发者提出了改进建议.
- 在技术验证的研究中,文献[138]为了通过大量应用程序来解决恶意软件检测问题,选取了来自 Google Play 的 52 000 个 Android 应用对其提出的检测方法进行验证.

这两篇文章选择的项目数都过大,极大地影响了平均值.为了使统计结果更贴合普通情况,统计时去掉了使用项目数量过万的文章.

下面我们从研究目的和研究子领域两个角度来了解目前经验研究所使用的项目数量.

- 研究目的与使用的项目/案例数量

从表 7 中我们可以发现,技术验证和探索性研究所使用的项目的平均数较多.为了验证新方法,或者从现象中发现结果,研究者一般会选取较多的项目进行研究.而进行解释性研究的研究者是为了验证对某一现象和规律的因果关系解释,故选择的项目数会较少.但是整体上看中位数都不大,在 3 个~6 个之间.探索性和技术验证性的方差更大一些.

进一步分析发现:239 篇文章中,使用项目数超过 1 000 的仅有 10 篇文章,占比 4%.其中,

- 6 篇文章的研究目的是技术验证,例如,文献[49]是用 SourceForge 和 GoogleCode 托管的 1 385 个开源项目对作者提出的通用缺陷预测模型进行验证.
- 4 篇文章是进行探索性研究,例如,文献[107]是通过 1 316 个 Gnome 项目对生态系统贡献者的工作量和参与程度如何在项目和活动类型之间变化进行探索性研究.

使用一个项目进行研究的有 58 篇文章,占总体的 23.2%.在这 58 篇文章之中,有 43%的文章是进行技术验证,51%的文章是进行探索性研究.

- 研究子领域与使用的项目/案例数量

通过表 8 发现:有的领域的最大值与最小值相差很大,而且平均值也受到最大值的影响,数据普遍偏大.统计分析发现:软件测试、软件维护、软件工程管理、软件工程过程、软件工程模型和方法、软件质量、软件工程职业实践和软件工程经济学这 8 个领域使用的项目数的最大值都超过了 1 000;然而软件需求、软件配置管理和软件设计的最大值相对于其他领域比较小,可能是因为这 3 个领域的研究者不易获取其所需的数据.

中位数趋于数据的中间位置,受最大值和最小值得影响较小,代表性较好,我们对各领域研究数目的中位数进行分析.

- 首先,软件工程管理的的中位数最大为 7.
- 其次是软件构造、软件测试、软件维护、软件工程过程、软件质量、软件工程经济学,这些领域的中位数在 3~5 之间,结合图 8 可以发现,这些领域中有的经常使用开源项目作为数据来源,开源项目较易获取,项目数量的中位数就稍大.
- 软件配置管理、软件工程模型和方法中位数最少.这两个领域都有过半数的文章选择使用一个项目,说明这两个领域对一个项目进行研究有时就能基本满足研究者的需要并得到同行评审的认可,或者说明这两个子领域难以获得需要的项目数据.

3.3.2 经验研究的数据采集手段

虽然调查研究的数据主要来自于问卷和访谈,但是问卷和访谈的方法也常常在实验和案例研究方法中被采用.在此,我们对所有原始研究中显示地说明了的数据采集手段进行统计(不包含作者直接使用项目代码等情况),得到表 9.为了了解各数据采集手段在不同经验方法中的使用情况,我们进行了分类统计,也见表 9;为了了解各数据采集手段和子领域的相关性,我们按照研究子领域对其进行统计,得到表 10.

Table 9 Empirical research methods and data collection methods

表 9 经验研究方法与数据采集手段

文章数量	问卷	访谈	归档数据	爬虫	观察
总计	66	35	53	10	1
实验	44	8	33	3	0
案例研究	16	21	16	7	1
调查研究	15	11	5	1	1
试点研究	2	0	2	0	0
仿真实验	1	0	1	0	0
其他	2	0	1	0	1

Table 10 Research subfileds and data collection methods

表 10 研究子领域与数据采集手段

文章数量	软件需求	软件设计	软件构造	软件测试	软件维护	软件配置管理	软件工程管理	软件工程过程	软件工程模型和方法	软件质量	软件工程职业实践	软件工程经济学	其他
问卷	9	1	15	9	19	3	2	6	5	5	6	3	0
访谈	4	0	3	5	8	0	3	9	3	2	2	4	0
归档数据	4	3	11	6	16	3	3	2	4	8	4	2	1
爬虫	0	0	3	2	3	2	0	0	1	0	1	1	0
观察	0	0	0	0	0	0	0	0	0	0	1	0	0

从表 9 中可以看出,原始研究中采用问卷方式最多(66 篇);归档数据排名第二(53 篇);有 35 篇文章使用访谈采集数据,位居第三,其中有 10 篇同时采用了问卷调查和访谈的方式获取数据,有 36.4%的文章的数据采集对象包括人.所有的研究方法以及所有的子领域都采用了问卷调查的方式来获取数据,问卷采集方式的特点是形式灵活,易于结构化[123].常用的在线问卷调查有基于邮件和问卷调查网站(online).而访谈常被案例研究和调查研究用来获取数据.访谈的主要设计形式有结构化访谈、半结构化访谈、无结构化访谈[1].结构化访谈与问卷调查类似,都有明确的问题.但访谈更为灵活,易于深入,但是成本较高.

实验、案例研究、调查研究等方法都用到了归档数据来获取数据,而且归档数据也是实验方法较为常用的数据获取方式.归档数据通常是指不同开发阶段的文档、故障数据、组织结构图、财务记录以及其他一些存档的历史数据[7].

还有 10 篇文章使用了爬虫来获取数据.至于观察,有一篇文章[139]通过观察获取参与者的个性和项目偏好数据,从而对参与者的行为进行研究.

如表 10 所示,通过研究领域与数据采集手段的交叉分析我们发现:问卷调查最常用于软件维护领域,其次是软件构造;软件工程过程和软件维护则较多地采用了访谈的手段获取数据.问卷调查和访谈都需要人员参与,所以涉及到人参与的领域有时需要对参与者进行调查获取相应的信息.归档数据最常用于软件质量、软件维护领域等领域,往往是对开发日志、故障数据等进行研究.

3.3.3 数据处理、分析方法和工具

(1) 统计图表的使用情况

通过通读样例文章,我们发现几乎所有的经验研究都使用了描述性统计学方法.作为分析的第 1 步,描述性统计将收集的数据可视化,包括结合图表进行分析.我们进一步统计了文章使用的具体图表,结果如图 11 所示,几乎所有的文章都使用了表格进行分析,其次使用较多的是折线图、箱形图、条形图、直方图、散点图.为此,我们认为这些图表都是经验研究者应该掌握的分析工具.

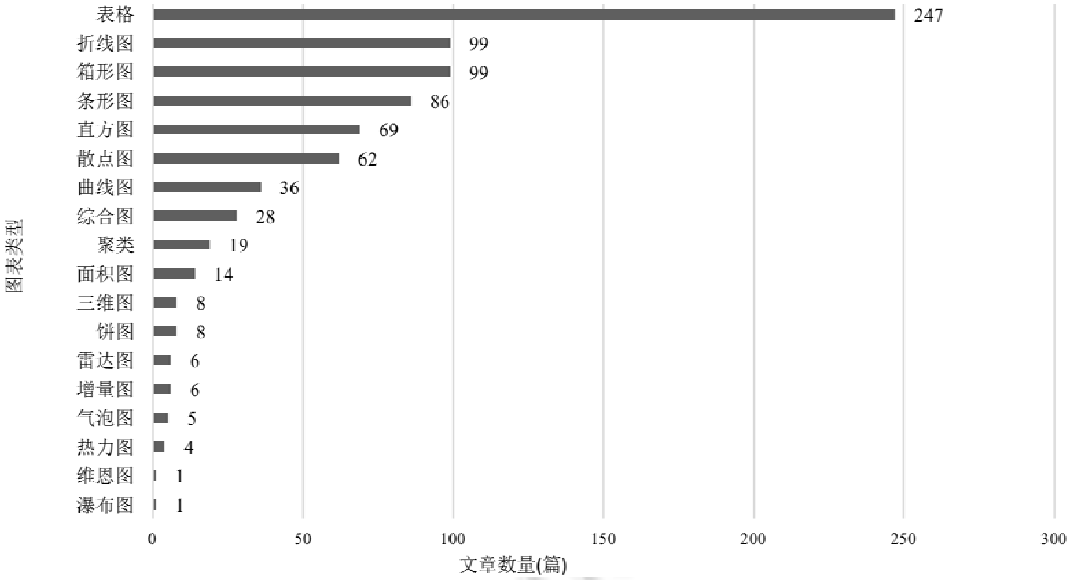


Fig.11 Figures used in the empirical study

图 11 经验研究中使用的图表

(2) 数理统计方法的使用情况

本文对常用数理统计方法进行了统计,见表 11.Wilcoxon 检验、t 检验、Mann-Whitney 检验、方差分析、Kruskal-Wallis 检验是较常用的数理统计方法.研究者应当掌握这些方法,以便在研究过程中选择合适的数理统计方法进行统计分析.

Table 11 Mathematical statistics methods used in empirical studies

表 11 经验研究中使用的数理统计方法

样本数量	样本规模	数理统计方法	使用文章数量	说明
单个	大(服从正态分布)	卡方检验	10	单个正态总体方差的检验
单个或两个	大(服从正态分布)	t 检验	54	单个正态总体方差未知时总体均值的检验,两个独立正态总体方差未知但相等时对均值差的检验
两个	大(服从正态分布)	F 检验	8	两个独立正态总体方差相等的检验
		配对 t 检验	5	两个配对正态总体方差未知时对总体均值的检验
	小(不服从正态分布)	Wilcoxon 检验	79	两个不服从正态分布的配对样本的非参数检验
		Mann-Whitney 检验	49	两个不服从正态分布的独立样本的非参数检验
	没有规定	符号检验	2	两个配对样本的均值一致性检验(对样本是否来自正态总体没有严格规定)
两个及两个以上	大(服从正态分布)	方差分析	43	两个及两个以上正态总体的均数差别的显著性检验
	小(不服从正态分布)	Kruskal-Wallis 检验	19	两个及两个以上不服从正态分布的样本的均数差别的显著性检验

- 单个或两个样本

检验单个或两个样本,且样本规模较大时,t 检验是最常用的参数检验,常用于单个正态总体方差未知时总体均值的检验或者两个独立正态总体方差未知但相等时对均值差的检验.

- 两个样本

Wilcoxon 检验和 Mann-Whitney 检验都适用于样本较小的非正态分布的两组数据检验.两者的区别在于:Wilcoxon 检验是对两配对样本的非参数检验,是相同受试的配对检验;Mann-Whitney 检验是对两独立样本的非参数检验,是不同受试的组间检验.例如,程序员甲按同一需求开发的两个不同程序之间的正确率的对比要用

Wilcoxon 检验;程序员甲和程序员乙按同一需求开发的程序运行正确率的对比就用 Mann-Whitney 检验.

- 两个或两个以上样本

方差分析和 Kruskal-Wallis 检验都是用于分析两个或两个以上样本的均数差别的显著性检验.区别在于:方差分析用于分析规模较大的两个或两个以上样本的均数差别显著性,常被用来分析、推断出哪些因素对研究事物有显著影响;当多个样本的规模较小时,常选择 Kruskal-Wallis 检验进行分析.Kruskal-Wallis 检验是一种基于序的方差分析方法,用于检验各个样本的总体是否相同.

(3) 数据分析工具使用情况

数据分析是经验研究中的重要环节,而统计工具的使用让研究者快速地获得有效的结论,所有统计分析工具中最常用的工具是 Microsoft Excel、SPSS、R 统计分析工具和 Matlab.最常用的办公软件 Microsoft Excel 还是研究者使用最多的统计工具,有 34 篇文章提到使用其进行统计分析.有 18 篇文章使用 SPSS 进行统计分析,因其操作简单、功能强大的特点,SPSS 一直是经常用于统计学分析运算、数据挖掘、预测分析和决策支持任务的软件.有 14 篇文章使用 R 统计分析工具进行统计分析,R 作为统计计算和统计制图的优秀工具,一直比较受研究者的青睐.有 11 篇文章使用兼具算法开发、数据可视化、数据分析等功能一体的 Matlab 进行数据分析.

以上 4 种工具主要是对定量数据进行分析的工具,最常用的定性分析工具是 Nvivo9,有 4 篇文章使用其对定型数据进行分析.NVivo 作为一款支持定性研究方法和混合研究方法的软件,可以收集、整理和分析访谈、焦点小组讨论、问卷调查、音频等内容,方便研究者对定性数据进行处理与分析.

综上所述,经验研究的主要数据来源有开源项目、工业项目和实验室项目等,有 57.3%的文章采用开源项目作为项目来源,23.8%的文章有选择工业项目为项目来源,而实验室项目只占 10.9%.在经验研究中,采用开源项目的文章数量呈现上升趋势.开源项目更容易获得软件构造、软件维护、软件配置管理等有关数据,而且常被采用的开源项目具有文档齐全、版本持续更新、覆盖用户广、代码质量高等特点,如 Eclipse,JEdit 等.研究者常用的开源项目托管平台有 SourceForge、Github 等.在进行经验研究时,有 4%的文章采用项目数量超过 1 000 个,有 23.2%的文章采用 1 个项目的数据进行经验研究,整体上,项目数量选择的中位数为 3.问卷、访谈和归档数据是使用最多的数据采集手段.最常用的数据处理方法是 Wilcoxon 检验、Mann-Whitney 检验、t 检验、方差分析等,分别针对小样本和大样本数据进行分析处理.常用的数据可视化工具有表格、折线图、箱形图、条形图、直方图和散点图等.建议拟采用经验研究的研究者应较熟练地掌握这些方法.

3.4 研究问题4:经验研究人员对于关于经验研究的有效性和可重现性问题的关注情况如何?

3.4.1 经验研究对有效性的关注

有效性是衡量一个经验研究结果有效程度的重要指标.任何一个经验研究都不可能避免有效性威胁,研究者在进行经验研究时应给予有效性充分考虑,以减轻一些有效性威胁.Yin 等人提出了一个有效性讨论的指南[140],将有效性分为结构有效性、内部有效性、外部有效性、结论有效性.

通过统计数据摘取表中的有效性分析,我们发现 ESE 论文的作者对有效性有较好的认识,结果如图 12 所示.在 250 篇 ESE 文章中,97%的文章提到了有效性威胁(validity threat),或缺点(weakness),或局限性(limitation),或其他意思相近的词语,大多数研究者意识到了经验研究的有效性,只有 7 篇(3%)文章没有提及有效性威胁.有 70%的文章参考结构有效性、内部有效性、外部有效性、结论有效性(可靠性)的分类方法对有效性进行了论述.

外部有效性是被讨论得最多的有效性,外部有效性评估被认为是度量研究结论应用到实践中的范畴[53].有 92%的文章讨论了外部有效性,有 89%的文章讨论了内部有效性,有 73%的文章讨论了结构有效性,有 45%的文章讨论了结论有效性(可靠性).

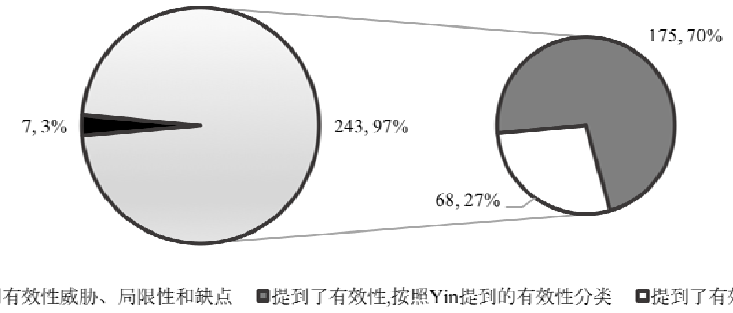


Fig.12 Discussions on the effectiveness of threats in the 250 articles

图 12 在 250 篇文章中对有效性威胁的讨论

3.4.2 经验研究对可重现性的关注

可重现性(possibility of replication)是经验研究中关注的一个重要问题.近年来,由于实验不可重现的问题时有发生,甚至被撤稿,有越来越多的研究者开始呼吁提高对可重现性的重视[5,16],因而我们希望了解软件工程领域经验研究对可重现性的重视情况.为此,我们在数据摘要表中设置了可重现性这一项,并根据实际情况将研究者对可重现性的考虑归纳分为 7 类,结果见表 12.我们认为,第 1 类~第 6 类,研究者都考虑了可重现性;在第 7 类中,研究者没有考虑可重现性.结果表明,大多数研究者(95%的文章)考虑了可重现性.

- 第 1 类是作者明确提到重现,并且将实验或案例研究的重现资源打包发布在网上,在 250 篇 ESE 期刊文章中,27%的文章对可重现性的考虑属于第 1 类.
- 第 2 类是明确提到了问卷或访谈的问题列表,或原始文章列表,独立研究者可根据这些数据进行重现研究,这类文章占 5%.
- 第 3 类是提供数据或项目的网址,有 52%文章是第 3 类,这类文章数量最多.
- 第 4 类是数据来自于之前的研究,可从之前的研究中获取重现资源.例如,文献[141]先明确提及实验数据来自之前的两篇文章提供的网址,然后直接使用这些数据进行实验.这类文章占 4%.
- 第 5 类需要向文章作者索要重现资源.例如,文献[142]明确提出源代码和数据是可获取的,可以发邮件向第 1 作者索要.这类文章数量仅 1%.
- 第 6 类是明确提数据不能获取,原因有闭源系统、保密性问题等,这类文章占 5%.例如,提到案例来源于匿名的工业公司,数据属于不公开的内部项目,又如明确指出系统是闭源的,或研究数据来自于保密系统等.
- 第 7 类文章中没有提及数据的可获取性及途径,这类文章占 6%.

Table 12 Considerations about possibility of replication

表 12 可重现性考虑情况

类型编号	是否考虑了重现性	类型说明	文章数量/百分比
1	是	提供重现包,并将实验或案例研究的重现资源发布至网上	67/27%
2	是	提供访谈/问卷的问题列表,或综述的原始文章列表	13/5%
3	是	提供数据来源的网址	129/52%
4	是	在之前的研究中可以找到重现资源	9/4%
5	是	可以向文章作者索要重现资源	3/1%
6	是	有明确的理由,不能获取数据	13/5%
7	否	没有提及数据的可获取性	16/6%

综上所述,本节分析了 ESE 论文作者对经验研究的有效性和可重现性的关注情况.结果表明:在软件工程领域,97%的研究者考虑了有效性威胁,并对经验研究的有效性进行了阐述,仅 3%的文章没有提及有效性威胁;95%的研究者考虑了可重现性,并提供了获得重现资源的途径.仍有 5%的文章没有提及数据的可获取性.

4 有效性分析及进一步讨论

4.1 有效性分析

参考 Yin[140]提出的方法,本文将从内部有效性、外部有效性、结构有效性和结论有效性这 4 个方面讨论本文调研的有效性.

(1) 内部有效性

内部有效性关注研究的因果关系[5].无论是系统文献综述,还是系统映射研究,文献调研方法的常见内部有效性威胁是选择偏见和主观性.

本文选择 ESE 期刊论文来分析近几年软件工程领域经验研究的情况,首先就是为了保证分析结果的内部有效性.由于 ESE 期刊明确的定位以及严格的审稿过程,我们可以认为 ESE 期刊刊出的论文既属于软件工程领域,同时又是进行经验研究的论文.如果广泛收集论文,而目前业界还不存在一个公认的标准来判断一篇论文是否是关于经验研究的论文,比如是否采用了问卷调查就是经验研究呢?是否做了一个实验就是经验研究呢?是否分析了经验数据、历史证据就是经验研究论文呢?非常难以界定.同时,计算机科学与软件工程的交叉融合,也很难界定一篇文章是否归属于软件工程.这些都势必会影响本文分析的内部有效性.在确保内部有效性的情况下,我们最后确定选择 ESE 期刊近 5 年的全部论文进行分析,以确保被分析论文的合理性.

主观性问题反映在设计和填写数据摘要表时的主观判断会影响研究结果的内部有效性.为了提高本文的有效性,采用了图 2 的方法,对数据摘要表进行了多次试填;对于有二义性的选项,给出了判断的标准,为了减少人为的误判,采用多个作者同时进行判定的方法,如果判定结果不一致,则通过集体讨论决定;对于领域划分,参考了 SWEBOK 3.0 中个知识域的描述,以最大程度地减少主观判断的有效性威胁.但由于数据表的填写主要依据论文的摘要、各节的标题、关键词、总结性描述,并未能充分阅读和理解 250 篇论文,难免存在误判情况.

(2) 外部有效性

外部有效性关注研究结果的适用及推广范围[5].仅选择 ESE 期刊论文是否可以代表整个软件工程领域的经验研究情况呢?答案显然是不能完全代表.根据前面内部有效性分析,在不能做到完全正确描述整个软件工程领域的经验研究的情况下,我们期望能够较大程度地反映近几年软件工程领域的经验研究情况.ESE 期刊论文无疑是一个最佳选择.ESE 期刊 20 多年来一直专注于经验方法在软件工程领域的应用,其研究内容涵盖了软件工程各个领域,每年有 6 期,每期论文数量稳定,每篇论文篇幅在 30 页~40 页左右.在软件工程领域影响因子较高,最近 5 年一直处于期刊引证报告的 Q1 区或者 Q2 区,认可度高,因而具有较高代表性.本文无选择地全部分析了近 5 年半的 250 篇论文,没有引入选择偏见,总体上应能较好地反映软件工程领域经验研究的特征.

在分析过程中,我们也意识到一个问题,即大量的系统文献综述、映射论文可能发表在面向领域的期刊或者综述性期刊上,因而本文的调研结果大部分反映的是原始经验研究的情况(参见表 4).关于系统文献综述方法的使用情况,将在第 4.2 节做简要的补充.

(3) 结构有效性

在设计数据摘要表时,存在结构有效性威胁.因此,本文采取了迭代设计方法,随机选取文章进行试填写,然后根据结果调整数据摘要表,重复几次这个过程后,得到最终的数据摘要表.迭代的设计方法有效减轻了结构有效性威胁.

(4) 结论有效性

结论有效性,反映数据统计和分析结果的有效程度[7].本文对研究方法和过程进行了详细的阐述,以确保研究过程可重现,也希望有独立研究者进行重现研究.本文中的结论有效性威胁存在于数据统计和分析过程.在进行数据统计和分析时,本文使用统计分析工具来辅助统计和分析.

4.2 关于系统文献综述(SLR)应用情况的补充说明

本文尝试分析近年来软件工程领域内系统文献综述文章的情况,以“literature+review”为关键词,2010~2017 年为年限,在 DBLP(DataBase Systems and Logic Programming)库中搜索文献,然后在结果中筛选软件工程领域

内期刊/会议的文章,最终得到256篇SLR文章.选择DBLP作为文献数据库,因为DBLP是一个整合了计算机领域内研究的成果的英文文献的集成数据库系统.该数据库按年代列出了作者的科研成果,包括国际期刊和会议等公开发表的论文,其收录的期刊和会议论文质量较高,DBLP 的文献更新速度很快,很好地反映了国外学术研究的前沿方向.

按年份对这256篇综述论文进行统计,得到图13.由图13可见:近8年来,论文数量总体呈现明显的增长趋势;且从2015年开始到2017年来,每年的论文数量相对于以前,有较高的增幅.2010年~2014年每年SLR论文数量都在25篇以下,平均为21篇;而后,2015年~2017年,每年的论文数量都超过了40篇,平均为49篇.可见,近3年来,学者们开始更加频繁地使用SLR方法来进行研究.图14给出了这256篇综述文章按照各个期刊或会议的分布情况.

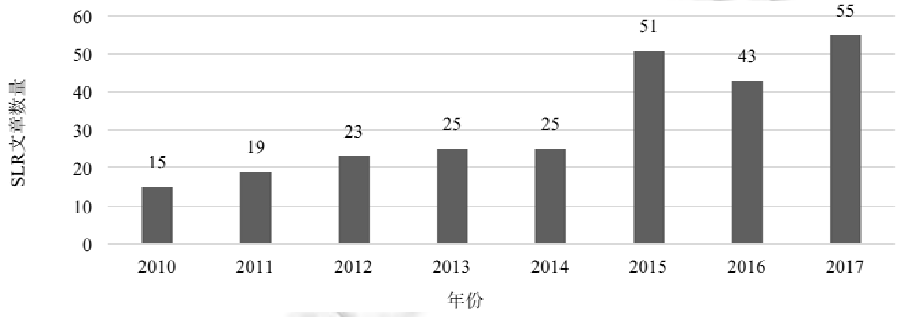


Fig.13 Number of SLR articles per year in software engineering from 2010 to 2017

图13 软件工程领域2010年~2017年每年的SLR文章数量

Fig.14 From 2010 to 2017, the number of SLR articles in the journals and conferences of software engineering

图14 软件工程领域2010年~2017年各期刊/会议各期刊/会议SLR文章数量

由图 14 可见,各期刊/会议中 SLR 文章数量的分布差别较为明显,其中,Information & Software Technology 期刊或会议中 SLR 的数量最多,达到了 61 篇,远远高于其他期刊中 SLR 文章的数量;其他期刊或会议的 SLR 数量都在 25 篇以下.这些期刊/会议中,有 12 个,其 SLR 文章数量在 1 篇~3 篇;有 13 个,其文章数量介于 5 篇~15 篇;有 3 个,其文章数在 20 篇~25 篇;还有 1 个,其 SLR 数量为 61 篇.这与期刊发表的文章总量以及所接收的文章类型有很大关系,如,《Information & Software Technology》文章基数较大,《ACM Transactions on Computer Systems》和《Computer Science Review》主要接收综述或调研类的文章.

关于 SLR 的详细分析不在本文中详细阐述,可参见文献[143−145].

4.3 进一步讨论:收获、问题和趋势

通过文本的调研可以发现,经验研究已经覆盖了软件工程领域的所有方面,包括软件全生命周期及所有子领域.实验研究法、案例研究法、调查研究法和系统文献综述法被广泛使用.不同子领域采用经验研究方法时有一定的领域侧重性.

软件工程的经验研究目的有探索性、技术验证、解释性等.探索性研究较多地使用了案例研究方法;在技术验证研究中,实验方法使用最多.在软件工程各个子领域都存在探索性研究和技术验证研究.

经验研究的数据来源以开源项目和工业项目为主,将近 80%.实验室项目占比较少.采用开源项目作为数据来源的文章数量逐年增多,有部分替代工业界项目的趋势.开源软件为软件工程经验研究提供了很好的平台,也是对目前软件工程领域缺乏足够标准数据集/测试集/基准集的一个很好的补充.

目前最常用的开源托管平台是 Apache、SourceForge、GitHub 和 Mozilla,且对 GitHub 的使用呈上升趋势.我们给出了最常用的 10 个开源项目,供研究者参考.同时,通过对数据采集方法的分析,给出了建议经验研究者应掌握的数据分析方法和工具.

在本文调研的过程中,我们也遇见和发现了一些问题,如前所述,首先是如何区分一篇论文是否是经验研究论文.我们发现,经验研究方法已深入到软件工程和计算机研究的方方面面.但是如何判断一篇文章是否为经验研究的论文呢?在分析 ESE 期刊论文时,我们也遇到了类似的问题.我们发现有两类论文:一类以经验研究为主,通过经验研究方法解决问题;另一类是针对问题提出解决方案/方法,然后采用经验研究方法去验证所提方案/方法的有效性,称为以经验研究为辅.但是实际区分时,却很难界定主次,这个边界难以确定.此外,在术语的使用方面也缺乏规范,最典型的是提出一种方法,进行验证时,不少论文采用了“案例研究/分析”作为标题,而实际上却是作者做的一个受控实验,甚至就是一个典型受控对比实验.

我们非常高兴地发现:ESE 期刊论文从经验研究的角度来看,大多非常规范和完整,而且研究者对经验研究的有效性和可重现性有非常好的认识,95%以上的作者都考虑了有效性威胁和可重现性问题,但是其他文章没有这么乐观,我们通过抽样分析发现:部分采用了经验方法的论文,写作不规范,过程不完整,比如采用调查问卷方法,缺设计的依据,人员选择合理性分析等;实验设计不完整;有效性分析不足等.因而,建议应加强经验软件工程的系统学习和教育.

未来的趋势:根据论文分析,可以发现开源数据的使用在经验研究中呈现上升趋势.我们还发现:在本文收集的 250 篇文章中,有 46 篇论文采用了对大数据进行分析、挖掘和学习的方法.根据维基百科对“经验研究”的定义,即经验研究是使用经验证据的研究,是一种通过直接或者间接观察与实验的方式获取知识的一种方法.在数字化时代,尤其是大数据环境下的数字化时代,更容易获得更多的经验数据(真实项目的各种数据),因而我们认为:引入机器学习、数据挖掘等新技术,是大数据时代经验研究的一种新形态.

5 结束语

本文通过分析 ESE 期刊最近 5 年刊出的全部 250 篇论文,总结分析了软件工程领域经验研究近 5 年的整体情况和研究特点、经验研究在软件工程各个子领域的应用情况和特点、各类经验方法在软件工程的使用情况及经验研究的目的.本文还分析了经验研究的数据来源,包括常用的开源平台、开源项目以及常用的数据采集、数据分析处理方法和工具,以供学习者了解学习和使用.给出了经验研究者对有效性和可重现性问题的关

注程度,以提高研究者对有效性和可重现性的重视.在分析了本文的有效性后,对经验研究的调研进行了总结,并针对一些问题做了进一步的补充和讨论.

在以后的工作中,将考虑扩展经验研究文章的范围,通过有针对性地选择期刊或会议论文,以及随机抽样经验研究论文的方式,对更多、更广泛的经验研究论文进行分析,以得到关于软件工程经验研究更全面的情况.同时,考虑利用自然语言处理方法开发自动或半自动的工具辅助,完成数据摘取和分析过程.

致谢 在此,我们向对本文的工作给予支持和建议的老师和同学表示由衷的感谢.

References:

[1] ShulL F, Singer J, Sjøberg DIK. Guide to Advanced Empirical Software Engineering. London: Springer-Verlag, 2008. [doi: 10.1007/978-1-84800-044-5]

[2] Basili VR, Selby RW, Hutchens DH. Experimentation in software engineering. IEEE Trans. on Software Engineering, 1986,12(7): 733−743. [doi: 10.1109/TSE.1986.6312975]

[3] Basili VR. The role of experimentation in software engineering: Past, current, and future. In: Proc. of the 18th Int'l Conf. on Software Engineering (ICSE). IEEE Press, 1996. 442−449. [doi: 10.1109/ICSE.1996.493439]

[4] Dybä T, Kitchenham BA, Jorgensen M. Evidence-Based software engineering for practitioners. IEEE Software, 2005,22(1):58−65. [doi: 10.1109/MS.2005.6]

[5] Siegmund J, Siegmund N, Apel S. Views on internal and external validity in empirical software engineering. In: Proc. of the Int'l Conf. on Software Engineering. IEEE Press, 2015. 9−19. [doi: 10.1109/ICSE.2015.24]

[6] Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wsslen A, Wrote; Zhang L, Wang Q, Peng R, Xuan Q, Trans. Experimentation in Software Engineering. Beijing: China Machine Press, 2015 (in Chinese).

[7] Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wsslen A. Experimentation in Software Engineering: An Introduction. Springer-Verlag, 2012. [doi: 10.1007/978-3-642-29044-2]

[8] Kitchenham B, Brereton OP, Budgen D, Turner M, Bailey J, Stephen L. Systematic literature reviews in software engineering: A systematic literature review. Information and Software Technology, 2009,51(1):7−15. [doi: 10.1016/j.infsof.2008.09.009]

[9] Petersen K, Feldt R, Mujtaba S, Mattsson M. Systematic mapping studies in software engineering. In: Proc. of the 12th Int'l Conf. on Evaluation and Assessment in Software Engineering (EASE). 2008. 68−77.

[10] Petticrew M, Roberts H. Systematic Reviews in the Social Sciences: A Practical Guide. John Wiley & Sons, 2008.

[11] Bourque P, Fairley RE. Guide to the Software Engineering Body of Knowledge (SWEBOK): Version 3.0. IEEE, 2014.

[12] Jurkiewicz J, Nawrocki J, Ochodek M, Głowacki T. HAZOP-Based identification of events in use cases. Empirical Software Engineering, 2015,20(1):82−109. [doi: 10.1007/s10664-013-9277-5]

[13] Cook TD, Campbell DT. Quasi-Experimentation: Design & Analysis Issues for Field Settings. Rand McNally College Publishing Company, 1979.

[14] Silva FQ, Suassuna M, França ACC, Grubb AM, Gouveia TB, Monteiro CVF, Santos IE. Replication of empirical studies in software engineering research: A systematic mapping study. Empirical Software Engineering, 2014,19(3):501−557. [doi: 10.1007/s10664-012-9227-7]

[15] Heeager LT, Rose J. Optimising agile development practices for the maintenance operation: Nine heuristics. Empirical Software Engineering, 2015,20(6):1762−1784. [doi: 10.1007/s10664-014-9335-7]

[16] Bezerra RMM, Silva FQBD, Santana AM, Magalhães CVC, Santos RES. Replication of empirical studies in software engineering: An update of a systematic mapping study. Empirical Software Engineering, 2014,19(3):501−557. [doi: 10.1109/ESEM.2015.7321213]

[17] Gousios G, Spinellis D. Conducting quantitative software engineering studies with Alitheia Core. Empirical Software Engineering, 2014,19(4):885−925. [doi: 10.1007/s10664-013-9242-3]

[18] Octaviano FR, Felizardo KR, Maldonado JC, Fabbri SCPF. Semi-Automatic selection of primary studies in systematic literature reviews: Is it reasonable? Empirical Software Engineering, 2015,20(6):1898−1917. [doi: 10.1007/s10664-014-9342-8]

[19] Kitchenham B, Madeyski L, Budgen D, Keung J, Brereton P, Charters SM, Gibbs S, Pohthong A. Robust statistical methods for empirical software engineering. Empirical Software Engineering, 2017,22(2): 579−630. [doi: 10.1007/s10664-016-9437-5]

[20] Mäder P, Egyed A. Do developers benefit from requirements traceability when evolving and maintaining a software system? Empirical Software Engineering, 2015,20(2):1−29. [doi: 10.1007/s10664-014-9314-z]

[21] Ali N, Sharafi Z, Guéhéneuc YG, Antoniol G. An empirical study on the importance of source code entities for requirements traceability. Empirical Software Engineering, 2015,20(2):442−478. [doi: 10.1007/s10664-014-9315-y]

[22] Tu YC, Tempero E, Thomborson C. An experiment on the impact of transparency on the effectiveness of requirements documents. Empirical Software Engineering, 2016,21(3):1035−1066. [doi: 10.1007/s10664-015-9374-8]

[23] Sagrado JD, Águila IMD, Orellana FJ. Multi-Objective ant colony optimization for requirements selection. Empirical Software Engineering, 2015,20(3):577−610. [doi: 10.1007/s10664-013-9287-3]

[24] McZara J, Sarkani S, Holzer T, Eveleigh T. Software requirements prioritization and selection using linguistic tools and constraint solvers. Empirical Software Engineering, 2015,20(6):1721−1761. [doi: 10.1007/s10664-014-9334-8]

[25] Bjarnason E, Runeson P, Borg M, Unterkalmsteiner M, Engström E, Regnell B, Sabaliauskaite G, Loconsole A, Gorschek T, Feldt R. Challenges and practices in aligning requirements with verification and validation: A case study of six companies. Empirical Software Engineering, 2014,19(6):1809−1855. [doi: 10.1007/s10664-013-9263-y]

[26] Hindle A, Bird C, Zimmermann T, Nagappan N. Do topics make sense to managers and developers? Empirical Software Engineering, 2015,20(2):479−515. [doi: 10.1007/s10664-014-9312-1]

[27] Robbes R, Röthlisberger D, Tanter E. Object-Oriented software extensions in practice. Empirical Software Engineering, 2015, 20(3):745−782. [doi: 10.1007/s10664-013-9298-0]

[28] Lizaranzu MJM, Rojo FC. A framework and architecture for rapid software development: A success story. Empirical Software Engineering, 2015,20(6):1456−1485. [doi: 10.1007/s10664-014-9320-1]

[29] Jaafar F, Gu, Guéhéneuc YG, Hamel S, Khomh F, Zulkernine M. Evaluating the impact of design pattern and anti-pattern dependencies on changes and faults. Empirical Software Engineering, 2016,21(3):896−931. [doi: 10.1007/s10664-015-9361-0]

[30] Parnin C, Bird C, Murphy-Hill E. Adoption and use of Java generics. Empirical Software Engineering, 2013,18(6):1047−1089. [doi: 10.1007/s10664-012-9236-6]

[31] Feigenspan J, Kästner C, Apel S, Liebig J, Schulze M, Dachselt R, Papendieck M, Leich T, Saake G. Do colors improve program comprehension in the# ifdef hell? Empirical Software Engineering, 2013,18(4):699−745. [doi: 10.1007/s10664-012-9208-x]

[32] Bavota G, Lucia AD, Marcus A, Oliveto R. Using structural and semantic measures to improve software modularization. Empirical Software Engineering, 2013,18(5):901−932. [doi: 10.1007/s10664-012-9226-8]

[33] Parnin C, Bird C, Murphy-Hill E. Adoption and use of Java generics. Empirical Software Engineering, 2013,18(6):1047−1089. [doi: 10.1007/s10664-012-9236-6]

[34] Callaú O, Robbes R, Tanter E, Röthlisberger D. How (and why) developers use the dynamic features of programming languages: The case of smalltalk. Empirical Software Engineering, 2013,18(6):1156−1194. [doi: 10.1007/s10664-012-9203-2]

[35] Davies J, German DM, Godfrey MW, Hindle A. Software bertillonage: Determining the provenance of software development artifacts. Empirical Software Engineering, 2013,18(6):1195−1237. [doi: 10.1007/s10664-012-9199-7]

[36] Dallal JA, Morasca S. Predicting object-oriented class reuse-proneness using internal quality attributes. Empirical Software Engineering, 2014,19(4):775−821. [doi: 10.1007/s10664-012-9239-3]

[37] Jbara A, Matan A, Feitelson DG. High-MCC functions in the linux kernel. Empirical Software Engineering, 2014,19(5):1261−1298. [doi: 10.1007/s10664-013-9275-7]

[38] Siegmund J, Apel S, Hanenberg S. Measuring and modeling programming experience. Empirical Software Engineering, 2014, 19(5):1299−1334. [doi: 10.1007/s10664-013-9286-4]

[39] Barros MO. An experimental evaluation of the importance of randomness in hill climbing searches applied to software engineering problems. Empirical Software Engineering, 2014,19(5):1423−1465. [doi: 10.1007/s10664-013-9294-4]

[40] Yang J, Hotta K, Higo Y, Igaki H, Kusumoto S. Classification model for code clones based on machine learning. Empirical Software Engineering, 2015,20(4):1095−1125. [doi: 10.1007/s10664-014-9316-x]

[41] Moreno-Lizaranzu MJ, Cuesta F. A framework and architecture for rapid software development: A success story. Empirical Software Engineering, 2015,20(6):1456−1485. [doi: 10.1007/s10664-014-9320-1]

[42] Yoo S, Harman M, Ur S. GPGPU test suite minimisation: Search based software engineering performance improvement using graphics cards. Empirical Software Engineering, 2013,18(3):550−593. [doi: 10.1007/s10664-013-9247-y]

[43] Greiler M, Deursen AV. What your plug-in test suites really test: An integration perspective on test suite understanding. Empirical Software Engineering, 2013,18(5):859−900. [doi: 10.1007/s10664-012-9235-7]

[44] Offutt J, Papadimitriou V, Praphamontripong U. A case study on bypass testing of Web applications. Empirical Software Engineering, 2014,19(1):69−104. [doi: 10.1007/s10664-012-9216-x]

[45] Thomas SW, Hemmati H, Hassan AE, Blostein D. Static test case prioritization using topic models. Empirical Software Engineering, 2014,19(1):182−212. [doi: 10.1007/s10664-012-9219-7]

[46] Itkonen J, Mäntylä M. Are test cases needed? Replicated comparison between exploratory and test-case-based software testing. Empirical Software Engineering, 2014,19(2):303−342. [doi: 10.1007/s10664-013-9266-8]

[47] Apa C, Dieste O, Espinosa EGG, Fonseca CE. Effectiveness for detecting faults within and outside the scope of testing techniques: An independent replication. Empirical Software Engineering, 2014,19(2):378−417. [doi: 10.1007/s10664-013-9267-7]

[48] Offutt J, Alluri C. An industrial study of applying input space partitioning to test financial calculation engines. Empirical Software Engineerging, 2014,19(3):558−581. [doi: 10.1007/s10664-012-9229-5]

[49] Fraser G, Arcuri A. 1600 faults in 100 projects: automatically finding faults while achieving high coverage with EvoSuite. Empirical Software Engineerging, 2015,20(3):611−639. [doi: 10.1007/s10664-013-9288-2]

[50] Alégroth E, Feldt R, Ryrholm L. Visual GUI testing in practice: Challenges, problemsand limitations. Empirical Software Engineering, 2015,20(3):694−744. [doi: 10.1007/s10664-013-9293-5]

[51] Fraser H, Arcuri A. Achieving scalable mutation-based generation of whole test suites. Empirical Software Engineering, 2015, 20(3):783−812. [doi: 10.1007/s10664-013-9299-z]

[52] Shin Y, Williams L. Can traditional fault prediction models be used for vulnerability prediction? Empirical Software Engineering, 2013,18(1):25−59. [doi: 10.1007/s10664-011-9190-8]

[53] Raja U. All complaints are not created equal: Text analysis of open source software defect reports. Empirical Software Engineering, 2013,18(1):117−138. [doi: 10.1007/s10664-012-9197-9]

[54] Zaidman A, Matthijssen N, Storey MA, Deursen A. Understanding Ajax applications by connecting client and server-side execution traces. Empirical Software Engineering, 2013,18(2):181−218. [doi: 10.1007/s10664-012-9200-5]

[55] Dit B, Revelle M, Poshyvanyk D. Integrating information retrieval, execution and link analysis algorithms to improve feature location in software. Empirical Software Engineering, 2013,18(2):277−309. [doi: 10.1007/s10664-011-9194-4]

[56] Lämmel R, Pek E. Understanding privacy policies. Empirical Software Engineering, 2013,18(2):310−374. [doi: 10.1007/s10664-012-9204-1]

[57] Kagdi H, Gethers M, Poshyvanyk D. Integrating conceptual and logical couplings for change impact analysis in software. Empirical Software Engineering, 2013,18(5):933−969. [doi: 10.1007/s10664-012-9233-9]

[58] Hindle A, Ernst NA, Godfrey MW, Mylopoulos J. Automated topic naming: Supporting cross-project analysis of software maintenance activities. Empirical Software Engineering, 2013,18(6):1125−1155. [doi: 10.1007/s10664-012-9209-9]

[59] Canfora G, Cerulo L, Cimitile M, Penta MD. How changes affect software entropy: An empirical study. Empirical Software Engineering, 2014,19(1):1−38. [doi: 10.1007/s10664-012-9214-z]

[60] Ljungkrantz O, Åkesson K, Fabian M, Ebrahimi AH. An empirical study of control logic specifications for programmable logic controllers. Empirical Software Engineering, 2014,19(3):655−677. [doi: 10.1007/s10664-012-9232-x]

[61] Chen J, Xiao J, Wang Q, Osterweil LJ, Li M. Perspectives on refactoring planning and practice:an empirical study. Empirical Software Engineering, 2016,21(3):1397−1436. [doi: 10.1007/s10664-015-9390-8]

[62] Chatterji D, Carver JC, Kraft NA. Code clones and developer behavior results of two surveys of the clone research community. Empirical Software Engineering, 2016,21(4):1476−1508. [doi: 10.1007/s10664-015-9394-4]

[63] Kim S, Kim D. Automatic identifier inconsistency detection using code dictionary. Empirical Software Engineering, 2016,21(2): 565−604. [doi: 10.1007/s10664-015-9369-5]

[64] Khomh F, Adams B, Dhaliwal T, Zou Y. Understanding the impact of rapid releases on software quality: The case of firefox. Empirical Software Engineering, 2015,20(2):336−373. [doi: 10.1007/s10664-014-9308-x]

[65] Germán DM, Adams B, Hassan AE. Continuously mining distributed version control systems: An empirical study of how Linux uses Git. Empirical Software Engineering, 2016,21(1):260−299. [doi: 10.1007/s10664-014-9356-2]

[66] Abebe SL, Ali N, Hassan AE. An empirical study of software release notes. Empirical Software Engineering, 2016,21(3): 1107−1142. [doi: 10.1007/s10664-015-9377-5]

[67] Mcilroy S, Ali N, Hassan AE. Fresh apps: An empirical study of frequently-updated mobile apps in the Google play store. Empirical Software Engineering, 2016,21(3):1346−1370. [doi: 10.1007/s10664-015-9388-2]

[68] Asadi M, Soltani S, Hatala M. The effects of visualization and interaction techniques on feature model configuration. Empirical Software Engineering, 2016,21(4):1706−1743. [doi: 10.1007/s10664-014-9353-5]

[69] Passos L, Guo J, Teixeira L, Czarnecki K, Borba P. Coevolution of variability models and related software artifacts: A case study from the Linux kernel. In: Proc. of the Int'l Software Product Line Conf. 2013. 91−100. [doi: 10.1007/s10664-015-9364-x]

[70] Kocaguneli E, Menzies T, Keung JW. Kernel methods for software effort estimation. Empirical Software Engineering, 2013,18(1): 1−24. [doi: 10.1007/s10664-011-9189-1]

[71] Capiluppi A, Izquierdo-Cortázar D. Effort estimation of FLOSS projects: A study of the Linux kernel. Empirical Software Engineering, 2013,18(1):60−88. [doi: 10.1007/s10664-011-9191-7]

[72] Corazza A, Martino SD, Ferrucci F, Gravino C, Sarro F, Mendes E. Using tabu search to configure support vector regression for effort estimation. Empirical Software Engineering, 2013,18(3):506−546. [doi: 10.1007/s10664-011-9187-3]

[73] Khatibi Bardsiri V, Jawawi DN, Hashim SZ, Khatibi E. A flexible method to estimate the software development effort based on the classification of projects and localization of comparisons. Empirical Software Engineering, 2014,19(4):857−884. [doi: 10.1007/s10664-013-9241-4]

[74] Borg M, Runeson P, Ardö A. Recovering from a decade: A systematic mapping of information retrieval approaches to software traceability. Empirical Software Engineering, 2014,19(6):1565−1616. [doi: 10.1007/s10664-013-9255-y]

[75] Bettenburg N, Hassan AE, Adams B, German DM. Management of community contributions. Empirical Software Engineering, 2013,20(1):252−289. [doi: 10.1007/s10664-013-9284-6]

[76] Kocaguneli E, Menzies T, Mendes E. Transfer learning in effort estimation. Empirical Software Engineering, 2015,20(3):813−843. [doi: 10.1007/s10664-014-9300-5]

[77] Beck F, Diehl S. On the impact of software evolution on software clustering. Empirical Software Engineering, 2013,18(5):970−1004. [doi: 10.1007/s10664-012-9225-9]

[78] Lee T, Gu T, Baik J. MND-SCEMP: An empirical study of a software cost estimation modeling process in the defense domain. Empirical Software Engineering, 2014,19(1):213−240. [doi: 10.1007/s10664-012-9220-1]

[79] Estler HC, Nordio M, Furia CA, Meyer B, Schneider J. Agile vs. structured distributed software development: A case study. Empirical Software Engineering, 2014,19(5):1197−1224. [doi: 10.1007/s10664-013-9271-y]

[80] Chen N, Hoi SCH, Xiao X. Software process evaluation: A machine learning framework with application to defect management process. Empirical Software Engineering, 2014,19(6):1531−1564. [doi: 10.1007/s10664-013-9254-z]

[81] Al-Baik O, Miller J. Waste identification and elimination in information technology organizations. Empirical Software Engineering, 2014,19(6):2019−2061. [doi: 10.1007/s10664-014-9302-3]

[82] Kai P, Gencel C, Asghari N, Betz S. An elicitation instrument for operationalising GQM+Strategies (GQM+S-EI). Empirical Software Engineering, 2015,20(4):968−1005. [doi: 10.1007/s10664-014-9306-z]

[83] Santos V, Goldman A, Souza CRB. Fostering effective inter-team knowledge sharing in agile software development. Empirical Software Engineering, 2015,20(4):1006−1051. [doi: 10.1007/s10664-014-9307-y]

[84] Bass JM. How product owner teams scale agile methods to large distributed enterprises. Empirical Software Engineering. 2015, 20(6):1525−1557. [doi: 10.1007/s10664-014-9322-z]

[85] Mohagheghi P, Gilani W, Stefanescu A, Fernández MA. An empirical study of the state of the practice and acceptance of model-driven engineering in four industrial cases. Empirical Software Engineering, 2013,18(1):89−116. [doi: 10.1007/s10664-012-9196-x]

[86] Arcuri A, Fraser G. Parameter tuning or default values? An empirical investigation in search-based software engineering. Empirical Software Engineering, 2013,18(3):594−623. [doi: 10.1007/s10664-013-9249-9]

[87] Koziolek H, Schlich B, Becker S, Hauck M. Performance and reliability prediction for evolving service-oriented software systems. Empirical Software Engineering, 2013,18(4):746−790. [doi: 10.1007/s10664-012-9213-0]

[88] Walkinshaw N, Lambeau B, Damas C, Bogdanov K, Dupont P. STAMINA: A competition to encourage the development and assessment of software model inference techniques. Empirical Software Engineering, 2013,18(4):791−824. [doi: 10.1007/s10664-012-9210-3]

[89] Heule MJ, Verwer S. Software model synthesis using satisfiability solvers. Empirical Software Engineering, 2013,18(4):825−856. [doi: 10.1007/s10664-012-9222-z]

[90] Mcmillan C, Poshyvanyk D, Grechanik M. On using machine learning to automatically classify software applications into domain categories. Empirical Software Engineering, 2014,19(3):582−618. [doi: 10.1007/s10664-012-9230-z]

[91] Wang S, Ali S, Gotlieb A, Liaaen M. A systematic test case selection methodology for product lines results and insights from an industrial case study. Empirical Software Engineering, 2016,21(4):1586−1622. [doi: 10.1007/s10664-014-9345-5]

[92] França BBN, Travassos GH. Experimentation with dynamic simulation models in software engineering: Planning and reporting guidelines. Empirical Software Engineering, 2016,21(3):1302−1345. [doi: 10.1007/s10664-015-9386-4]

[93] Li X, Mutha C, Smidts CS. An automated software reliability prediction system for safety critical software. Empirical Software Engineering, 2016,21(6):2413−2455. [doi: 10.1007/s10664-015-9412-6]

[94] Nguyen VH, Dashevskyi S, Massacci F. An automatic method for assessing the versions affected by a vulnerability. Empirical Software Engineering, 2016,21(6):2268−2297. [doi: 10.1007/s10664-015-9408-2]

[95] Mcintosh S, Kamei Y, Adams B, Hassan AE. An empirical study of the impact of modern code review practices on software quality. Empirical Software Engineering, 2016,21(5):2416−2189. [doi: 10.1007/s10664-015-9381-9]

[96] Zhang F, Mockus A, Keivanloo I, Zou Y. Towards building a universal defect prediction model with rank transformed predictors. Empirical Software Engineering, 2016,21(5):2107−2145. [doi: 10.1007/s10664-015-9396-2]

[97] Grigera J, Garrido A, Panach JI, Distante D, Rossi G. Assessing refactorings for usability in e-commerce applications. Empirical Software Engineering, 2015,21(3):1224−1271. [doi: 10.1007/s10664-015-9384-6]

[98] Fontana FA, Mäntylä V, Zanoni M, Marino A. Comparing and experimenting machine learning techniques for code smell detection. Empirical Software Engineering, 2016,21(3):1143−1191. [doi: 10.1007/s10664-015-9378-4]

[99] Baysal O, Kononenko O, Holmes R, Godfrey MW. Investigating technical and non-technical factors influencing modern code review. Empirical Software Engineering, 2015,21(3):932−959. [doi: 10.1007/s10664-015-9366-8]

[100] Misirli AT, Shihab E, Kamei Y. Studying high impact fix-inducing changes. Empirical Software Engineering, 2016,21(2):605−641. [doi: 10.1007/s10664-015-9370-z]

[101] Munaiah N, Camilo F, Wigham W, Meneely A, Nagappan M. Do bugs foreshadow vulnerabilities? An in-depth study of the chromium project. Empirical Software Engineering, 2017,22(3):1305−1347. [doi: 10.1007/s10664-016-9447-3]

[102] Pagano D, Maalej W. How do open source communities blog? Empirical Software Engineering, 2013,18(6):1090−1124. [doi: 10.1007/s10664-012-9211-2]

[103] Šmite D, Wohlin C, Galviņa Z, Prikladnicki R. An empirically based terminology and taxonomy for global software engineering. Empirical Software Engineering, 2014,19(1):105−153. [doi: 10.1007/s10664-012-9217-9]

[104] Gómez MN, Acuña ST. A replicated quasi-experimental study on the influence of personality and team climate in software development. Empirical Software Engineering, 2014,19(2):343−377. [doi: 10.1007/s10664-013-9265-9]

[105] Barua A, Thomas SW, Hassan AE. What are developers talking about? An analysis of topics and trends in stack overflow. Empirical Software Engineering, 2014,19(3):619−654. [doi: 10.1007/s10664-012-9231-y]

[106] Salleh N, Mendes E, Grundy J. Investigating the effects of personality traits on pair programming in a higher education setting through a family of experiments. Empirical Software Engineering, 2014,19(3):714−752. [doi: 10.1007/s10664-012-9238-4]

[107] Vasilescu B, Serebrenik A, Goeminne M, Mens T. On the variation and specialisation of workload: A case study of the GNOME ecosystem community. Empirical Software Engineering, 2014,19(4):955−1008. [doi: 10.1007/s10664-013-9244-1]

[108] Ihme T, Pikkarainen M, Teppola S, Kaariainen J, Biot O. Challenges and industry practices for managing software variability in small and medium sized enterprises. Empirical Software Engineering, 2014,19(4):1144−1168. [doi: 10.1007/s10664-013-9253-0]

[109] Scholtes I, Mavrodiev P, Schweitzer F. From Aristotle to Ringelmann: A large-scale analysis of team productivity and coordination in open source software projects. Empirical Software Engineering, 2016,21(2):642−683. [doi: 10.1007/s10664-015-9406-4]

[110] Rosen C, Shihab E. What are mobile developers asking about? A large scale study using stack overflow. Empirical Software Engineering, 2016,21(3):1192−1223. [doi: 10.1007/s10664-015-9379-3]

[111] Seo Y, Bae D. On the value of outlier elimination on software effort estimation research. Empirical Software Engineering, 2013, 18(4):659−698. [doi: 10.1007/s10664-012-9207-y]

[112] Moe NB, Smite D, Hanssen GK, Barney HT. From offshore outsourcing to insourcing and partnerships: Four failed outsourcing attempts. Empirical Software Engineering, 2014,19(5):1225−1258. [doi: 10.1007/s10664-013-9272-x]

[113] Guo Y, Spinola RO, Seaman CB. Exploring the costs of technical debt management: A case study. Empirical Software Engineering, 2016,21(1):159−182. [doi: 10.1007/s10664-014-9351-7]

[114] Duarte CHC. Productivity paradoxes revisited—Assessing the relationship between quality maturity levels and labor productivity in brazilian software companies. Empirical Software Engineering, 2017,22(2):818−847. [doi: 10.1007/s10664-016-9453-5]

[115] Bavota G, Canfora G, Penta MD, Oliveto R, Panichella S. How the Apache community upgrades dependencies: An evolutionary study. Empirical Software Engineering, 2015,20(5):1275−1317. [doi: 10.1007/s10664-014-9325-9]

[116] Phannachitta P, Keung J, Monden A, Matsumoto K. A stability assessment of solution adaptation techniques for analogy-based software effort estimation. Empirical Software Engineering, 2017,22(1):474−504. [doi: 10.1007/s10664-016-9434-8]

[117] Han W. Validating differential relationships between risk categories and project performance as perceived by managers. Empirical Software Engineering, 2014,19(6):1956−1966. [doi: 10.1007/s10664-013-9270-z]

[118] Maccoun RJ. Experimental and quasi-experimental designs for generalized causal inference. Journal of Policy Analysis and Management, 2003,22(2):330−332. [doi: 10.1002/pam.10129]

[119] Runeson P, H¨ost M. Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering, 2009,14(2):131−164. [doi: 10.1007/s10664-008-9102-8]

[120] Haller I, Slowinska A, Bos H. Scalable data structure detection and classification for C/C++ binaries. Empirical Software Engineering, 2016,21(3):778−810. [doi: 10.1007/s10664-015-9363-y]

[121] Siegmund J, Schumann J. Confounding parameters on program comprehension: A literature survey. Empirical Software Engineering, 2015,20(4):1159−1192. [doi: 10.1007/s10664-014-9318-8]

[122] Kai P, Vakkalanka S, Kuzniarz L. Guidelines for conducting systematic mapping studies in software engineering: An update. Information and Software Technology, 2015,64:1−18. [doi: 10.1016/j.infsof.2015.03.007]

[123] Molleri JS, Petersen K, Mendes E. Survey guidelines in software engineering: An annotated review. In: Proc. of the ACM/IEEE Int'l Symp. on Empirical Software Engineering and Measurement. 2016. 58:1−58:6. [doi: 10.1145/2961111.2962619]

[124] Juristo N, Vegas S. Using differences among replications of software engineering experiments to gain knowledge. In: Proc. of the ACM/IEEE Int'l Symp. on Empirical Software Engineering and Measurement. 2009. 356−366. [doi: 10.1109/MSR.2010.5463362]

[125] Monteiro CV, Silva FQ, Capretz LF. The innovative behaviour of software engineers: Findings from a pilot case study. In: Proc. of the ACM/IEEE Int'l Symp. on Empirical Software Engineering and Measurement. ACM Press, 2016. 7:1−7:10. [doi: 10.1145/2961111.2962589]

[126] Hafiz M, Fang M. Game of detections: How are security vulnerabilities discovered in the wild? Empirical Software Engineering, 2016,21(5):1920−1959. [doi: 10.1007/s10664-015-9403-7]

[127] Holbrook EA, Hayes JH, Dekhtyar A, Li W. A study of methods for textual satisfaction assessment. Empirical Software Engineering, 2013,18(1):139−176. [doi: 10.1007/s10664-012-9198-8]

[128] Albayrak Ö, Carver JC. Investigation of individual factors impacting the effectiveness of requirements inspections: A replicated experiment. Empirical Software Engineering, 2014,19(1):241−266. [doi: 10.1007/s10664-012-9221-0]

[129] Unterkalmsteiner M, Gorschek T, Feldt R, Lavesson N. Large-Scale information retrieval in software engineering: An experience report from industrial application. Empirical Software Engineering, 2016,21(6):2324−2365. [doi: 10.1007/s10664-015-9410-8]

[130] Fucci D, Turhan B. On the role of tests in test-driven development: A differentiated and partial replication. Empirical Software Engineering, 2014,19(2):277−302. [doi: 10.1007/s10664-013-9259-7]

[131] Mcburney PW, Mcmillan C. An empirical study of the textual similarity between source code and source code summaries. Empirical Software Engineering, 2016,21(1):17−42. [doi: 10.1007/s10664-014-9344-6]

[132] Mcilroy S, Ali N, Khalid H, Hassan AE. Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. Empirical Software Engineering, 2016,21(3):1067−1106. [doi: 10.1007/s10664-015-9375-7]

[133] Jiang J, Lo D, He J H, Xia X, Kochhar PS, Zhang L. Why and how developers fork what from whom in GitHub. Empirical Software Engineering, 2017,22(1):547−578. [doi: 10.1007/s10664-016-9436-6]

[134] Cheung WT, Ryu S, Kim S. Development nature matters: An empirical study of code clones in JavaScript applications. Empirical Software Engineering, 2016,21(2):517−564. [doi: 10.1007/s10664-015-9368-6]

[135] Ceccato M, Capiluppi A, Falcarin P, Boldyreff C. A large study on the effect of code obfuscation on the quality of Java code. Empirical Software Engineering, 2015,20(6):1486−1524. [doi: 10.1007/s10664-014-9321-0]

[136] Gharehyazie M, Posnett D, Vasilescu B, Filkov V. Developer initiation and social interactions in OSS. Empirical Software Engineering, 2015,20(5):1318−1353. [doi: 10.1007/s10664-014-9332-x]

[137] Bavota G, Lucia AD, Marcus A, Oliveto R. Automating extract class refactoring: An improved method and its evaluation. Empirical Software Engineering, 2014,19(6):1617−1664. [doi: 10.1007/s10664-013-9256-x]

[138] Allix K, Bissyandé TF, Jérome Q, Klein J, State R. Traon YL. Empirical assessment of machine learning-based malware detectors for Android-Measuring the gap between in-the-lab and in-the-wild validation. Empirical Software Engineering, 2016,21(1):183−211. [doi: 10.1007/s10664-014-9352-6]

[139] Kosti MV, Feldt R, Angelis L. Archetypal personalities of software engineers and their work preferences: A new perspective for empirical studies. Empirical Software Engineering, 2016,21(4):1509−1532. [doi: 10.1007/s10664-015-9395-3]

[140] Yin R. Case Study Research: Design and Methods. 5th ed., Blackwell Science Ltd, 2013.

[141] Ryu D, Choi O, Baik J. Value-Cognitive boosting with a support vector machine for cross-project defect prediction. Empirical Software Engineering, 2016,21(1):43−71. [doi: 10.1007/s10664-014-9346-4]

[142] Williams BJ, Carver JC. Examination of the software architecture change characterization scheme using three empirical studies. Empirical Software Engineering, 2014,19(3):419−464. [doi: 10.1007/s10664-012-9223-y]

[143] Garousi V, Mäntylä MV. A systematic literature review of literature reviews in software testing. Information & Software Technology, 2016,80:195−216. [doi: 10.1016/j.infsof.2016.09.002]

[144] Wohlin C, Prikladnicki P. Systematic literature reviews in software engineering. Information & Software Technology, 2013,55(6): 919−920. [doi: 10.1016/j.infsof.2013.02.002]

[145] Imtiaz S, Bano M, Ikram N, Niazi M. A tertiary study: Experiences of conducting systematic literature reviews in software engineering. In: Proc. of the Int'l Conf. on Software Evaluation and Assessment in Software Engineering. EASE, 2013. 177−182. [doi: 10.1145/2460999.2461025]

附中文参考文献:

[6] Wohlin C,Runeson P,Höst M,Ohlsson MC,Regnell B,Wsslen A,著;张莉,王青,彭蓉,宣琦,译.经验软件工程.北京:机械工业出版社,2015.

张莉(1968－),女,四川成都人,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件建模与分析,需求工程,经验研究工程,软件体系结构.

田家豪(1991－),男,博士生,CCF 学生会员,主要研究领域为经验软件工程,软件体系结构.

蒲梦媛(1993－),女,硕士生,主要研究领域为经验软件工程,软件测试.

岳涛(1974－),女,博士,首席科学家,博士生导师,主要研究领域为模型驱动工程,需求工程,产品线工程,基于搜索的软件工程,经验软件工程.

刘奕君(1994－),女,硕士生,主要研究领域为经验软件工程.

蒋竞(1985－),女,博士,讲师,CCF 专业会员,主要研究领域为经验软件工程,开源软件,基于数据的分析与推荐.