开源软件开发者的演化研究

李其锋[1,3] 李 兵[2]
(武汉大学计算机学院软件工程国家重点实验室 武汉 430072)[1]
(武汉大学国际软件学院 武汉 430072)[2] (湖北大学计算机与信息工程学院 武汉 430062)[3]

摘 要 开源软件的开发主要依靠开发人员的自我管理和志愿贡献。在软件开发过程中,开发人员的组成、分工、能力等会发生经常性的改变,这些改变都会通过交互行为的变化反映出来。已有研究成果表明开源软件存在核心团队,他们主导项目的开发与实施,但对开发团队随时间而发生的结构变化的研究较少。通过观测软件项目生命周期中开发者总体的变化情况,研究了开发者网络与软件的协同演化机理,以GNU/Linux公共数据集为例,讨论了开源软件开发者数量、自愿度、开发者经验与项目的协同演化现象,并给出了演化的合理解释。

关键词 开源软件,开发者,协同演化
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.12.010

Evolution of Contributors in Open Source Software Development

LI Qi-feng[1,3] LI Bing[2]
(State Key Laboratory of Software Engineering, School of Computer, Wuhan University, Wuhan 430072, China)[1]
(International School of Software, Wuhan University, Wuhan 430072, China)[2]
(School of Computer Science and Information Engineering, Hubei University, Wuhan 430062, China)[3]

Abstract Open source software development is heavily based on voluntary contributions. Developers are self-selected. But as open source software projects evolve, changes in the development team affect their organization and decision structure. New members enter the group and others leave. From the related research it is known that a small group of very active developers are responsible in general for the proper evolution of a project. But it does not attend to the time axis that evolution requires. In this paper, we analyzed how software developers evolve in open source software projects. As case studies, we selected GNU/Linux for researching the behavior of contributors to work on a open source software project. Our aim is to give quantitative insight about the evolution of maintainers. We studied how many these developers remain from the beginning of the project and what happens to packages maintained by those developers who left the project.

Keywords Open source software, Developer, Co-evolution

开源软件开发与维基百科等简单的知识型生产相比,更具复杂性。由于软件开发有明确的目标、功能、性能、质量、时限等,必须在满足这些目标约束的前提下,去衡量和理解开发人员在群体开发过程中组织配置与交互合作行为,开源软件开发需要同时考虑开发者和软件产品的多种因素,包括开发者之间的交流与合作、软件产品之间的内部关系、开发者与软件产品的关系等,因此,开源软件的研究需要面对的是一个复杂的社会-技术系统,软件开发过程中开发者和软件产品两种系统之间的协同关系的研究对于指导软件开发有一定的实践意义。

1 开发者协同合作行为研究

分布在全世界的开发者通过开源社区发起或参与项目、沟通交流和分享知识,开源社区作为他们全球协作的必要途径,在推动开源软件发展的过程中起着举足轻重的作用。因此,开源社区中开发者的参与动机、组织结构、合作模式等一直是研究人员关注的焦点。Crowston等人较早分析了SourceForge上的120个项目,根据缺陷追踪系统中的通信交互研究这些项目组的构成,但没有发现典型的通用组织结构[1]。为了更好地描述和模拟大规模的开源社区结构,一些研究者开始使用社会网络分析方法处理收集到的数据,发现一些项目组是高度集中的组织方式,而其他的则不是;对于规模较大的项目,其开发组的交互结构从最初的星型逐渐向"核心成员-外围开发者"模式转变,并展现出更多的模块化特征[2]。此外,由于将开源社区的组织结构抽象为了网络模型,Xu等人也陆续发现了"无尺度"(scale free)和"小世界"(small

到稿日期:2015-02-22 返修日期:2015-05-29 本文受国家重点基础研究发展规划(973)(2014CB340401),国家自然科学基金(61273216,61272111,61202048,61202032),湖北省重大科技创新计划(2013AAA020)资助。
李其锋(1981-),男,博士生,主要研究领域为软件工程、复杂网络;李 兵(1969-),男,教授,博士生导师,CCF高级会员,主要研究领域为软件工程、复杂网络、云计算、人工智能,E-mail:bingli@whu.edu.cn。

world)的复杂网络特性[3,4]，类似组织结构是否是真实开源软件项目更高效、更易发展的原因，还有待深入研究。

标签作为一种软件制品(如代码、文档等)分类和索引的工具，能够支持开源社区中的开发者更好地进行合作，包括导航、协调和获取项目组的相关知识。作为一种新的技术尝试，Treude 等人针对拥有 175 名开发者的大型项目进行实证研究[5]，发现社会化标注已被采用和完全接受，并在改善小组协作开发实践中发挥了重要作用；进一步的研究[6]表明，该技术在支持任务发现、工作衔接、信息交换方面非常有效，有助于提高开源软件开发小组的工作效率。考虑到软件开发的社会性和技术性因素，Storey 等人将标签赋予更丰富的语义信息，通过开发的工具[7]辅助支持开发者查找和处理自己或是他人编写的代码评论，合作完成分配的任务。

另一方面，在开源软件版本不断更新的过程中，有大量的缺陷被发现和提交，需要将它们分派给合适的开发者(或众包给可能的开发者社群)去处理。一种思路是将开发者看作是他以往解决过的缺陷的关键术语集合(基于模糊集理论)[8]，通过与新的缺陷报告中的术语进行相似度计算[9]，来确定分派任务的概率；另一种思路是利用机器学习方法来构造分类器，Anvik 等人对 5 个开源软件的实验[10]表明正确率可达到 70%～98%；由于可能受到噪音数据的影响，Zou 等人提出了使用特征提取来约简训练集的方法[11]，有助于提高机器学习方法的正确率。此外，一些研究者也开始尝试用图模型来描述缺陷分派。Jeong 等人提出了一种基于马尔可夫链的图模型[12]来捕获缺陷再分配的历史记录，针对 44.5 万个缺陷报告的实验显示，该模型减少了 28%的再分配事件，并与传统的分派方法相比，其正确率提高了 23%。

已有工作以开源社区中的开发者为研究对象，侧重于开源社区组织结构分析和成员协作开发新技术(如编码、缺陷分派等)的研究，却忽视了开发者之间在开源社区的交互和反馈。

本文将重点关注开发团队在不同时间和版本的演化分析研究，并选用 GNU/Linux 这个开源项目作为研究对象，相关研究数据可以从项目网站下载，通过观测软件项目生命周期中开发者总体的变化情况，分析开发者数量变化、开发者自愿度、开发者经验、开发者流失与包丢失情况等，给出了相应的分析结果。

2 开发者数量演化分析

图 1 是项目开发者承担程序包维护工作的趋势图，图中展示了最近 5 次稳定版本发行过程中，其开发者数量的变化情况。正如我们所假设的，项目的开发者数量不断增长，2.0 版本(1998 年 7 月)发布时，项目有 216 名开发者，而 3.0 版本(2002 年 7 月)发布时，人员数达到 859 名，到 3.1 版本(2004 年 12 月)时，人员数却将近 1237 名。从时间统计来看，该项目开发者数量年均增长率约为 35%。

从图 1 还可以看出，开发者维护的平均软件包的数量(Average)稍有增长，其值在 5 和 6 之间。随着开发人员对项目熟悉程度的增加和项目新开发维护工具的引入，开发者会变得更高效，投入相同的时间，能维护数量更多的软件包。在几个版本中，中值未发生明显变化，除 3.0 版本为 4 外，其他版本的中值均为 3，这表明，50%的开发者所负责的软件包数量最多不超过 4。从众数的统计来看，每个版本的众数均为 1，对应的开发者数量分别为 52、76、122、208、348，约占对应版本开发者总数的 25%，这表明有大约 1/4 的开发者只是维护项目其中的一个包。从图中基尼系数的走势和值的情况来看，软件开发工作更倾向于不均等的分配方式，有小部分人员负责大部分的软件包，而大部分人员所负责的软件包数量比较少。

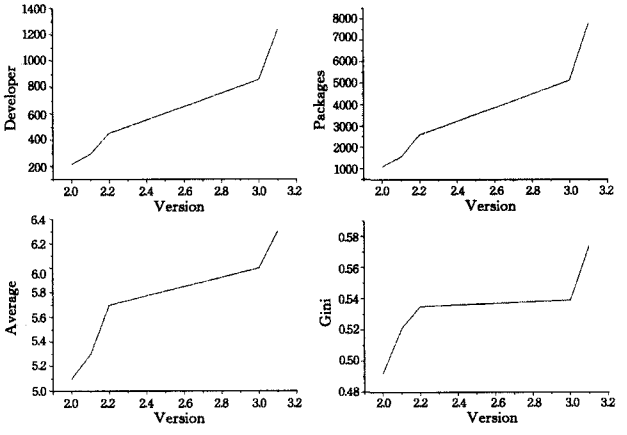

图 1 各版本中开发者承担程序包的情况

3 开发者自愿度分析

为了研究开发者的自愿度，我们以研究的起点 2.0 版本的数据为研究基础，跟踪分析 2.0 版本发行时项目团队成员的变化情况。2.0 版本发行时共有 216 名开发者。在计划实施过程中，参与其中的开发者数量是动态变化的。

图 2 展示了每个版本发行时，保留在原有团队的开发者人数以及他们所负责的软件包数量。开发者人数在不断下降，2004 年 12 月，2.0 时期的 216 名开发者中，只有 121 名还在继续参与该计划，仅占 55.8 %。在随后的人员演化过程中，历经六年半的时间，原来的 2.0 版本时期的开发团队人员在不断地减少，根据这种趋势计算，该项目团队的半衰期是 7 年半。同样，如果我们不以 2.0 作为初始版本，而是以其他发行版本作为研究的起始点，可以发现类似的现象，可以说这是一个开源项目的普遍现象。造成该现象的原因在于一些参与者随着项目的开发进展，会逐渐产生疲劳感，直至离开该项目；此外，项目管理者在把握项目进度和控制项目质量上也会有所考虑，会对项目人力资源进行适当调整。

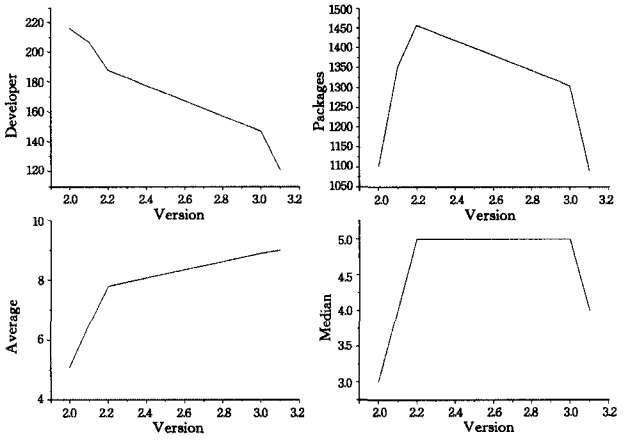

图 2 版本 2.0 中开发者的跟踪分析

从开发者负责的软件包数量上看，2.0 版本的 216 名开发者共负责 1102 个软件包，到 2.1 版本发行时（约 9 个月后），保留在原团队中的开发者所负责的软件包数量增加到 1351 个，而 2.2 版本发行时，又增加到 1457 个，在 3.0 版本发行时，这一数目为 1305，在最后的版本中，相对于最初版本，开发人员减少了一半，维护的软件包的数量却没有减少。

图 2 中均值数据变化表明，那些从一开始就没有离开项目的开发者所负责的软件包数量在持续增长。1998 年 7 月，每个开发者负责 5 个软件包；2004 年 12 月，每个开发者则负责 9 个软件包，增幅虽然减慢，但总体上均值在不断增长；我们还可以看出，开发者效率的显著提升，维护包的数量接近翻番。

从各版本的中值变化趋势可以看出，开发者所负责的软件包数量越来越多。该数据也表明相比总的开发者人数，仅负责一个软件包的开发者在以更快的速度下降（前者从 216 名下降到 121 名，降幅为 56%，而后者则从 52 名下降到 38 名，降幅为 38%）。造成这一现象的原因有两个：其一，这些开发者可能已经退出了这个项目；其二，他们可能负责了更多的软件包，任务更加繁重。换句话说，他们要么更多地参与这个项目，要么退出。

4 开发者经验分析

我们通过对 2.0 版本的开发人员进行追踪分析，得到了他们自愿参与项目的演化情况。在此我们将从相反的角度入手，以最近发行版本为研究对象进行逆向分析，研究 3.1 版本现有开发者加入该项目的时间及维护软件包的数量变化。

图 3 展示了在 3.1 版本中开发者初次参与到该项目的时间及承担任务的情况。从图中可以看到，有 121 人从 2.0 版本开始就持续参与该项目，55 人于 2.1 发布前加入，114 人于 2.2 发布前加入。在最近的 3.0 和 3.1 两次版本发布中，分别有 393 人和 554 人加入。

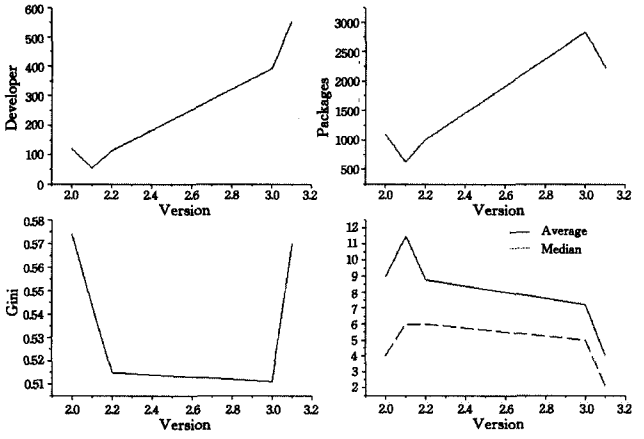

图 3 版本 3.1 中开发者的加入时间及其承担的开发任务分析

图 3 同时展示了每个开发者负责的软件包数量的演化情况，从均值的变化情况看，对于在前 3 个版本发布时加入的开发者，其人均处理软件包的数量在 8.8 到 11.5 之间变化；而对于在后两个版本发布时新加入的开发者，该值远低于前者，总体趋势就是：经验更丰富的开发者负责维护更多的软件包。对于 2.1 版本，每个开发者负责维护的包的数量达到最大，明显高于其他版本。对于中位数，早期参与项目的开发者的中位值要高，这表明该时期加入的大部分开发者在维护多个软件包。3.1 版本的中位数为 2，也再次说明开源软件开发过程中，开发者维护一个软件包只是一种暂时的状态。

5 开发者流失与程序包维护情况分析

当开发者离开项目后，他们所负责的软件包便处于无人维护的状态（软件包被丢弃）。这些软件包一般有两种处理方式，1）交由其他开发人员负责（软件包被采纳），2）被新的版本淘汰掉，不会出现在未来的稳定版本中。我们统计了两两不同版本中被丢弃的和被采纳的软件包统计情况，从统计结果发现，被采纳的软件包几乎在每个版本中都超过了 60%，另一个值得关注的结果是，每出一个新的版本，采纳软件包/丢弃软件包的比例就会下降，这说明了被丢弃的软件包的增长的速度比被采纳的软件包增长速度要快。在每个新的版本中，有些软件包会丢失，如果一个无人维护的软件包未出现在下一个版本中，那么它将极有可能不会在出现在未来的版本中。

结束语 本文探讨了 GNU/Linux 计划开发者维护产品程序包的行为，定量分析了开发者的演化及其与程序包的维护关系，包括开发者数量演化分析、开发者自愿度分析、开发者经验对产品维护影响分析、开发者的流失与对应程序包丢弃行为分析。研究对象为项目的 5 个版本，时间跨度为 7 年。经研究发现，项目开发者及开发者平均维护的程序包持续增长，项目开发者团队的半衰期为 90 个月，超过一半的开发者仍然在当前版本中继续贡献，选择留下的开发者随着开发者经验的不断积累，他们负责维护的包的数量越来越多，任务越来越重，开发者的离开对项目造成一定影响，表现为维护程序包的丢弃。在实际研究中，我们发现在开源社区中，程序包并未因开发者的离开而被大量丢弃，相反，大多数程序包被其他开发者所接管。

参 考 文 献

[1] Crowston K, Howison J. The social structure of Free and Open Source software development [J]. First Monday, 2005, 10(2): 2

[2] Long Y, Siau K. Social Network Structures in Open Source Software Development Teams [J]. Journal of Database Management, 2007, 18(2): 25-40

[3] Xu J, Gao Y, Christley S, et al. A topological analysis of the open source software development community [C]// Proc. of the 38th Hawaii International Conference on System Sciences (HICSS 2005). USA, 2005: 198a

[4] Singh P. The small-world effect: The influence of macro-level properties of developer collaboration networks on open-source project success [J]. ACM Transactions on Software Engineering and Methodology, 2010, 20(2): 6

[5] Treude C, Storey M-A. How tagging helps bridge the gap between social and technical aspects in software development [C]// Proc. of the 31st International Conference on Software Engineering (ICSE 2009). Canada, 2009: 12-22

[6] Treude C, Storey M-A. Work Item Tagging: Communicating Concerns in Collaborative Software Development [J]. IEEE Transactions on Software Engineering, 2012, 38(1): 19-34

[7] StoreyM-A, Ryall J, Singer J, et al. How Software Developers Use Tagging to Support Reminding and Refinding [J]. IEEE Transactions on Software Engineering, 2009, 35(4): 470-483

[8] Tamrawi A, Nguyen T T, Al-Kofahi J M, et al. Fuzzy set-based automatic bug triaging [C]// Proc. of the 33rd International Conference on Software Engineering (ICSE 2011). USA, 2011: 884-887

[9] Chen L, Wang X, Liu C. An Approach to Improving Bug Assignment with Bug Tossing Graphs and Bug Similarities [J]. Journal of Software, 2011, 6(3): 421-427

[10] Anvik J, Murphy G C. Reducing the effort of bug report triage: Recommenders for development-oriented decisions [J]. ACM Transactions on Software Engineering and Methodology, 2011, 20(3): 10

[11] Zou W, Hu Y, Xuan J, et al. Towards Training Set Reduction for Bug Triage [C]// Proc. of the 35th IEEE Annual Computer Software and Applications Conference (COMPSAC 2011). Germany, 2011: 576-581

[12] Jeong G, Kim S, Zimmermann T. Improving bug triage with bug tossing graphs [C]// Proc. of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2009). Netherlands, 2009: 111-120

(上接第 42 页)

(4)学生 B 进入该提问,向学生 A 进行详尽的解答,其中包括上传图片辅助说明、关键字加粗等高级编辑功能,如图 7 所示。

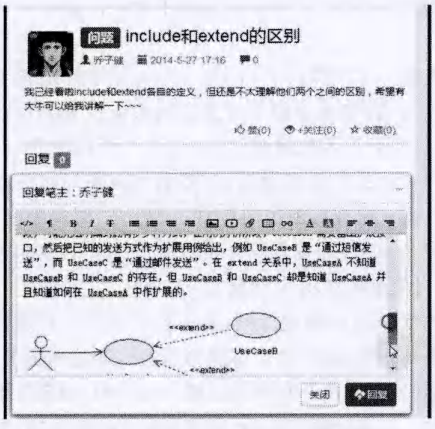

图 7 笔记回复界面

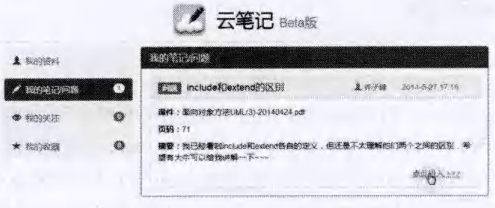

图 8 个人主页中笔记管理界面

(5)之后,学生 A 前往个人主页中笔记/问题管理界面,找到该提问并点击进入,如图 8 所示。进入后,可以看到此时讲义区中与该问题相关联的区域仍是高亮的,学生 A 经过查看学生 B 对自己问题的解答,理解了二者的关系,如图 9 所示。

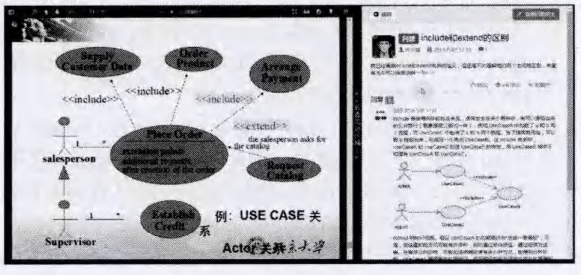

图 9 直接进入具体笔记界面

通过以上实例分析,我们可以看到课堂云笔记插件给学生提供了一个提问互助和笔记共享的平台,为学生之间创建了一个便于学习和交流的知识分享机制。实例证明,本文所提出的设计方案是可行、有效的。

结束语 本文分析了现有学习辅助平台的一些不足,例如:提问存在障碍、笔记不能针对教育资源细节、学生知识共享渠道匮乏等。与相关工作相比,本文首次提出将云笔记与讲义相结合,并将其运用到在线教学的解决方案,实现了一个面向 Chrome 浏览器的课堂云笔记插件。在该插件的辅助下,学生在遇到问题时,可以及时并有针对性地向老师、同学寻求帮助;还可以针对课程资源中的任意细节进行笔记记录,并与同学分享自己对这些知识点的理解和感悟,相互讨论;复习时,可以结合讲义及与讲义内容相关的问题和笔记,更好地进行课程学习。最后,本文进行了实例研究,证明了所提出的设计方案具有可行性和有效性。

参 考 文 献

[1] Sahami M, Guzdial M, Martin F G, et al. The revolution will be televised: perspectives on massive open online education [C]// Proceeding of the 44th ACM Technical Symposium on Computer Science Education. 2013: 457-458

[2] Notemate [EB/OL]. http://www.tiaozhanbei.net/project/18427

[3] Russell D M, Klemmer S. Will massive online open courses (moocs) change education? [C]// Extended Abstracts on Human Factors in Computing Systems. Paris, France, 2013: 2395-2398

[4] Nguyen A, Piech C, Huang J, et al. Codewebs: scalable homework search for massive open online programming courses [C]// Proceedings of the 23rd International Conference on World Wide Web (WWW'14). IW3C2, Switzerland, 2014: 491-502

[5] 魏青,杨杰,陈典铖. 基于云存储技术的云笔记产品研发[J]. 广东通信技术,2013(9):43-47
Wei Qing, Yang Jie, Chen Dian-cheng. Research and development of cloud notes products based on cloud storage technology [J]. Guangdong Communication Technology, 2013(9): 43-47

[6] 朴灵. 深入浅出 Node.js[M]. 北京:人民邮电出版社,2013
Pu Ling. Head First Nade.js[M]. Beijing: Posts and Telecom. Press, 2013

[7] 彭娜. 基于 Node.JS 博客系统的设计与实现[D]. 大连:大连理工大学,2013
Peng Na. The Design and Implementation of Blog System based on Node.js[D]. Dalian: Dalian University of Technology, 2013

[8] Herrick D R. Google this!: using Google apps for collaboration and productivity [C]// Proceedings of the 37th Annual ACM SIGUCCS Fall Conference. 2009: 55-64