*论文题目：开源代码贡献支撑工具效果的实证研究（原文标题：Effectiveness of code contribution: from patch-based to pull-request-based tools）

*作者：朱家鑫(1)，周明辉(1)，AudrisMockus(2)

*单位：1.北京大学，2.田纳西大学

联系方式：zhujiaxin@pku.edu.cn

*文章发表信息：In Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2016). ACM, New York, NY, USA, 871-882. DOI: https://doi.org/10.1145/2950290.2950364

*原文链接地址：http://dl.acm.org/citation.cfm?id=2950364

*正文：

背景与问题：

开源世界拥有着一种美妙的开发生态：贡献者们自发地通过一次次的代码提交不断地为项目注入创新的活力与发展的动力，于此同时，他们参与开源开发的能力与在社区中的声望逐渐提高，软件产品也在向着他们所期待的样子持续演化。

这种生态催生了眼下的“开源盛世”，但相关研究提醒我们，其中仍然存在诸多问题。有研究表明，大量的代码提交（约 70%）以失败告终[1]，也就是说人们为这种“盛世”付出了高昂的代价。从古至今，很多成功的工具极大地促进了人类生产效率的提高。那么代码贡献支撑工具能帮助我们降低贡献活动的成本吗？当下备受青睐的 pull-request 系统是否比其他工具更具优势？开源社区中的参与者们应该如何去做？基于前人的研究成果，我们对 12 个典型开源项目进行量化分析来揭晓答案。

方法：

在对开源项目及相关文献的调查中，我们发现人们所使用的代码贡献支撑工具可以分为基于 patch 和基于 pull-request 的两类。基于 patch 的工具以邮件列表和问题追踪系统为代表（如图 1-a 所示），使用这种工具时，贡献者将修改后的代码以补丁的形式封装在附件中提交至邮件列表或者问题追踪系统中，开发者们对其进行评审、讨论、修订，最终完成代码的集成。基于 pull-request 的工具（如图 1-b 所示）诞生于 GitHub 等社会化协作平台，相比于基于 patch 的工具，它们具有一系列旨在便捷化代码贡献过程的新特性，如代码的提交与合并通过对版本库分支的 pull-request 完成，操作过程只需用户点击几个按钮。

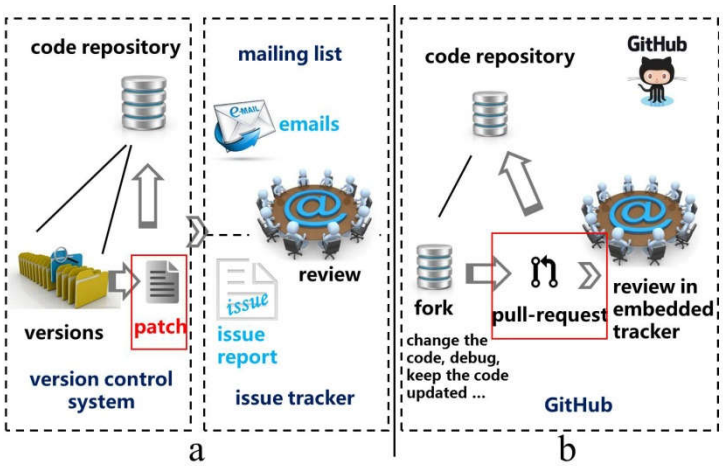

图 1 基于 patch 的(a)与基于 pull-request 的(b)代码贡献支撑工具

基于上述认识，我们选取使用两类工具的代表性开源项目，对他们的代码贡献活动进行“数字考古”，分析、对比不同工具下的代码贡献活动效率。

什么是低成本、高效的代码贡献？我们希望给出一种方法，帮助大家通过数据回答这个问题。第一步，我们要找出代码贡献活动的关键方面来评估其效率。我们对最近 10 年发表于软件工程顶级期刊和会议的文献进行调研，综合和扩展已有研究成果后，得出了三个值得关注的方面：参与者的产出、时间的开销以及他们参与的活跃度。在产出方面，我们选取了之前被广泛使用的代码贡献的成功率，另外，考虑到贡献成功率低的原因可能是有很多贡献被直接忽略，我们又加入了贡献的忽略率作为补充；在时间开销方面，我们同样复用了以往研究中所使用的贡献处理周期和响应时间；在活跃度方面，我们引入了归一化的每月代码贡献提交量，用该量度来刻画贡献者对工具效果所表现出的态度。

影响代码贡献效率的因素众多，如何把工具的作用剥离出来呢？我们设计了项目内与项目间两类对比来处理这个问题。在项目内的对比中我们使用了 Rails 项目，该项目在 2010 年之前使用基于 patch 的工具而在之后迁移到了 GitHub 上使用 pull-request 系统。在项目间的对比中，我们使用了两组项目，一组是使用邮件列表的项目，包括 Linux、Apache 等，他们的代码贡献数据及部分度量结果来自于前人的研究[1][2]，另一组是使用 GitHub pull-request 系统的项目，包括 Rails、jQuery 等。

表 1 异构的代码贡献活动数据和统一度量的方法

<table>
<tr><td colspan="2">支撑工具</td><td>pull-request 系统</td><td>问题追踪系统</td><td>邮件列表</td></tr>
<tr><td colspan="2">数据源</td><td>Rails、jQuery、PPSSPP、Rust 项目在 GitHub 中的代码贡献活动数据</td><td>Rails 项目在 Lighthouse 中的代码贡献活动数据</td><td>文献[1][2]的数据包含的项目：Apache、SVN、Linux、FreeBSD、KDE、Gnome、Python、Postgress</td></tr>
<tr><td rowspan="6">度量的统一</td><td>成功的代码贡献的识别</td><td>状态为 merged 的，或者状态为 closed 且 commit 出现在核心仓库中的 pull-request</td><td>状态为 committed 或 resolved 的问题报告中的 patch</td><td>能够与核心库中某两个相邻版本间 diff 匹配的 patch</td></tr>
<tr><td>被忽略的代码贡献的识别</td><td>没有 comment 且不是在一小时内被贡献者自己关闭的 pull-request</td><td>没有 comment 且没有状态改变的问题报告的 patch</td><td>没有收到回复邮件的 patch</td></tr>
<tr><td>代码贡献的开始时间</td><td>pull-request 的提交时间</td><td>问题报告的提交时间</td><td>邮件的发送时间</td></tr>
<tr><td rowspan="2">代码贡献的首次响应时间</td><td>pull-request 的第一个 comment 或者 event 的时间</td><td>问题报告的第一个 comment 或者状态改变的时间</td><td>第一封回复邮件的时间</td></tr>
<tr><td colspan="3">不考虑被忽略的代码贡献</td></tr>
<tr><td rowspan="2">代码贡献的结束时间</td><td>最后一个 comment 的时间（和邮件列表对比）、最后一个状态改变的时间(和问题追踪系统对比)</td><td>最后一个状态的改变时间</td><td>最后一封回复邮件的时间</td></tr>
<tr><td colspan="3">不考虑被忽略的代码贡献</td></tr>
</table>

有了度量的设计和要研究的开源项目，第二步，我们就要实现具体的量度了。由于这项研究涉及了不同支撑工具，我们要解决数据异构的问题，实现度量的统一。表 1 展示了这些异构数据和统一度量的方法。

在进行对比时，针对不同方面的效率量度我们采用不同的统计工具来量化二者间的差异，对于是二元变量的，我们使用 Fisher's exact test，而对于是连续型变量的，我们则使用 Wilcoxon rank-sum test。此外，我们还重现了以往研究中的效率分析模型[1][3]，在已发现的效率影响因素中加入工具因素，进一步检验工具对效率的影响，例如下面对贡献处理周期进行分析的模型。

$$log(resolve\_time) \sim log(Churn + 1) + log(\#Reviewers + 1) + log(CExperience + 1) + log(RExperience + 1) + log(CExpertice + 1) + log(RExpertice + 1) + is\_PR$$

注：$Churn$：增加和删除的代码行数，

$\#Reviewers$：代码审查者的数量，

$C(R)Experience$：代码贡献者(审查者)参与代码贡献活动的时长，

$C(R)Expertice$：代码贡献者(审查者)做代码贡献(审查)的次数，

is_PR：是否使用 pull-request。

结果：

对比、分析的结果如何呢？首先，关于贡献的成功与否、被忽略与否，在两类对比中，工具因素表现出了不同的显著性及效应量，说明贡献的成功可能主要取决于其他的因素；其次，关于贡献的处理周期及响应时间，所有的对比都显示在基于 pull-request 的工具中贡献的处理周期更短，模型重现的结果也显示了工具因素的显著性，值得注意的是，模型拟合的某些结果与之前的研究结果[1]存在明显的差异，这有待进一步探索；最后，关于贡献的活跃度，所有对比也都显示项目使用基于 pull-request 的工具时，人们参与的活跃度更高，我们观察到的高产出、低时耗可能使参与者有了更好的体验，因而更加积极地参与贡献。基于上述结果，我们提出以下建议供开源参与者们思考并在实践中进行探索：1，有效地利用基于 pull-request 工具的特性，如分布式版本控制系统、贡献过程追踪机制、代码审查辅助机制、社会化编程平台；2，通过高级的实践方法弥补传统工具的不足，例如 Linux Kernel 项目对邮件标题进行标注增强对贡献过程的追踪；3，寻找通过工具提高代码贡献成功率的方法。

总结：

综合与扩展前人的工作，本文提出一套面向异构数据的代码贡献活动效率度量方法来量化不同支撑工具的效果，通过效率的度量、对比、建模，我们希望帮助实践者评估他们的代码贡献活动，了解工具给他们带来了怎样的影响，让他们能更有效地改进工具和实践方法。特别值得一提的，鉴于软件开发的复杂性和项目上下文的差异，同一套度量方法在不同的项目中很难能有一致的精准的解释。例如，在此工作中我们得到一些与以前工作不一致的结果。如何解决这类问题？我们建议：一方面进行多侧面的验证（例如本文采用的项目内和项目间的比较方法），另一方面通过更加深入和广阔的分析以获得更多的洞察。

参考文献：

[1]P. C. Rigby, D. M. Germ´an, L. Cowen, and M. D.Storey. Peer review on open-source software projects:Parameters, statistical models, and theory. ACMTrans. Softw. Eng. Methodol., 23(4):35, 2014.

[2]C. Bird, A. Gourley, and P. T. Devanbu.Detectingpatch submission and acceptance in OSS projects. InFourth International Workshop on Mining SoftwareRepositories, MSR 2007 (ICSE Workshop),Minneapolis, MN, USA, May 19-20, 2007,page 26, 2007.

[3]G. Gousios, M. Pinzger, and A. van Deursen.Anexploratory study of the pull-based softwaredevelopment model. In 36th International Conferenceon Software Engineering, ICSE ’14,

Hyderabad, India- May 31 - June 07, 2014, pages 345–355, 2014.

作者简介：

文发作者有北京大学的博士生朱家鑫、周明辉副教授以及田纳西大学的 AudrisMockus 教授。

说明：*为必填项。