GitHub 开源软件治理文件数据集

Yibo Yan Seth Frey Amy Zhang Vladimir Filkov Likang Yin 加州大学戴维斯分校 加州大学戴维斯分校 UWSeattle 加州大学戴维斯分校 加州大学戴维斯分校 ybyan@ucdavis.edu sethfrey@ucdavis.edu axz@cs.uw.edu vfilkov@ucdavis.edu lkyin@ucdavis.edu

arXiv:2304.00460v1 [cs.SE] 2 Apr 2023

摘要——**过去几十年来，开源软件（OSS）已成为工业界和学术界的重要资源。尽管开源软件项目及其社区构建了创新的结构来支持项目，但它们的需求复杂，也面临着诸如项目被弃置等风险。为了管理内部社会动态和社区演变，开源软件开发者社区开始依赖书面治理文件，为不同的社区参与者分配角色和职责。**

为了便于研究正式治理文件对开源软件项目及社区的影响和成效，我们构建了一个纵向数据集，涵盖 710 个托管在 GitHub 上且包含 GOVERNANCE.MD 治理文件的开源软件项目。该数据集包括代码仓库中的所有提交、在 GitHub 上创建的所有议题和评论，以及对治理文件所做的所有修订。我们希望该数据集的开放能够激发更多研究兴趣，推动人们研究开源软件社区如何治理其项目，以及治理文件对社区的影响。

I. 引言

作为全球规模最大的在线开放代码托管平台之一，GitHub 为众多开源软件（OSS）项目和社区提供了广泛可用、易于使用的平台。由于 OSS 数字痕迹可免费获取，大规模研究 OSS 已成为可能，关于 OSS 代码仓库的学术研究也日益增多。

对 OSS 社区的研究涵盖代码质量 [1], [2], 可持续性 [3], [4], [5], 以及近期的治理 [6], [7]等主题。GitHub 治理不同于其他 OSS 托管组织中的治理。例如，Apache 软件基金会（ASF）是一家知名的非营利组织，为 OSS 项目提供孵化器。ASF 孵化器（ASFI）中的项目可获得指导，其工作由 ASFI 委员会引导和治理。GitHub 上的大多数 OSS 项目和社区都没有这样的委员会来支持 OSS 活动。为此，许多 OSS 社区会起草自己的治理文件，并利用 GitHub 的“issue”功能来治理和协调 OSS 活动。基于使用基于文本的文档来治理集体活动的既有惯例（例如 Contributor Covenant），开源社区开始采用托管名为 GOVERNANCE.MD 的基于文本文件这一做法，努力营造更健康的社区，并更清晰地传达分工和参与预期。随着项目发展，治理文件通常会定期修订，社区成员会据此引入、修改或删除治理规则。

研究治理文件有助于了解开源软件项目如何协调工作，并可能揭示其社会技术动态。了解 GitHub 上的开源软件项目如何治理协作工作，以及项目中的社会技术动态如何变化，可以进一步推动对开源软件项目可持续性的研究，从而找出开源软件项目和社区成功治理的关键因素 [8], [9]。

我们介绍 GitHub 开源软件治理文档数据集。该数据集包含托管在 GitHub 上的 710 个开源软件项目的治理文件、提交历史和议题。为便于开展纵向研究，我们提供了独立的表格，记录 GOVERNANCE.MD 文件的提交历史，并以行级粒度详细记录每次提交对 GOVERNANCE.MD 文件所做的更改。

据我们所知，这是首次向实证软件工程领域介绍此类以治理文档为中心的 GitHub 托管开源软件项目数据集。接下来，我们将介绍数据集的详情、数据抓取方法和存储方式，随后展示两个可从这些数据中受益的初步研究示例。数据集及其抓取脚本已发布在 Zenodo：https://doi.org/10.5281/zenodo.7530768。

II. 相关工作

从社会技术视角分析开源软件项目的社区组织、结构和治理，有助于理解开源软件项目质量随时间的动态变化 [10]，以及开源软件项目中的协调成效 [11], [12] [13], [14]。除 GitHub 外，许多大型组织和基金会，如 OSGeo [15]、Apache 软件基金会 [16], [17], 和 Linux 基金会 [18] ，也支持众多开源软件项目。已有多种工具用于挖掘这些项目的数据，例如，Perceval 提供统一入口，用于从各种后端收集软件仓库的数据 [19]；GHTorrent 提供了从 GitHub 收集镜像数据的简化方法 [20]；CHAOSS 项目提供指标、模型和软件，以了解开源软件社区健康度 [21]。我们的治理文档数据集与这些数据源互为补充，结合其他来源后，可支持对开源软件项目演变开展整体纵向分析。

III. 开源软件治理文档数据集

该数据集以不同粒度记录了每个收集项目的 GOVERNANCE.MD 文件的完整修订历史。除开发者进行的每次提交的详细信息外

我们还提取了治理文件中每次提交所涉及的行变更信息。

1) 数据访问：GitHub 提供一系列接口，供开发者获取与代码仓库相关的数据。GitHub 支持用于数据检索的 REST [22] 和 GraphQL [23] API，我们利用这两种 API 抓取数据。由于 GitHub 提供一手数据访问，通过 GitHub 官方 API 收集的数据至少与其他社交编程网站的数据一样可靠且及时。虽然 GHTorrent 可用于挖掘 GitHub 数据，但它可能不适用于某些特定的研究方向，因为它通过全面镜像 GitHub 数据生成了过量数据，却遗漏了治理文件逐行修订的详细信息。

2) 代码仓库：根据 GitHub 的搜索结果，包含 GOVERNANCE.MD 文件的项目超过 160 万个。然而，并非所有项目都能提供有意义的信息，因为这些项目通常并未为自身项目提供 GOVERNANCE.MD 文件，而是包含了其他带有 GOVERNANCE.MD 文件的软件包或依赖项。[1] 因此，为了减少数据集中的噪声，我们使用 REST API 的代码搜索端点 [2]，仅获取根目录中包含 GOVERNANCE.MD（不区分大小写且非精确匹配）文件的代码仓库。[3] 初始数据集包含 1,899 个根目录中有 GOVERNANCE.MD 文件的唯一项目。为确保项目具有意义，我们只保留至少有 1 次提交/问题的项目。最终数据集包含 710 个项目。我们收集了每个代码仓库的基本元数据，例如代码仓库名称和治理文件的文件名。

由于 GitHub 的搜索 API 具有非确定性，我们反复调用搜索 API，直到最近搜索的 100 个代码仓库中已有 99 个被遇到过。我们使用 *gql*[4] 获取每个代码仓库的提交、议题以及每个议题中的评论。此外，我们使用 *PyDriller* [24] 在本地定位所有涉及治理文件变更的提交。借助 *PyDriller*，我们通过处理解析后的差异输出，按行详细获取了每个治理文件提交的信息，从而了解每次提交中具体哪些行及其编辑内容被添加或删除。

3）提交、议题和评论：我们使用 GraphQL 端点 https://docs.github.com/en/graphql/guides/forming-calls-with-graphql#the-graphql-endpoint，收集提交、议题以及每个议题中的评论。我们通过相应地调整游标，依次发送 GraphQL 查询，以遍历整个搜索空间。

对收集的数据进行了初步的项目级汇总，为每个代码仓库生成基本指标随后将这些指标添加到仓库列表中。因此，仓库列表包含星标数、分叉数、提交数、提交者人数、议题数和评论数；议题和评论的提交人数；以及治理文件的提交数和提交者人数。

4) 治理文件的提交历史：由于 GitHub 不支持检索特定文件的提交历史，我们使用 PyDriller [24]检查了涉及治理文件变更（即添加、删除或修改）的提交。我们提取了基本元数据，例如作者、提交者、文件行数，以及提交前后的内容和差异输出。我们还提取了治理文件中每一行在各次提交中的修改情况（即添加或删除）。我们对收集到的治理文件相关提交列表运行脚本，获得：a) 治理文件各章节对应的提交列表；[5] 以及 b) 通过提取治理文件最后一次已收集提交之后的内容，获取每个代码仓库最新的治理文件。所有数据——提交、议题、评论——的收集时间范围为相应项目在 GitHub 上的创建日期至 2022年6月，脚本最后一次更新的月份。

5）数据库：大多数数据直接存储在 MongoDB 数据库（一种 NoSQL 数据库）中，之后导出为不同的数据格式，即 CSV、SQL 和 MongoDB 归档转储。少部分数据（如仓库列表）最初由*pandas* 在内存中处理，之后由脚本写入 MongoDB 数据库。最后，所有数据都重新导出了一次。我们使用 PyMongo[6] 通过 Python 脚本与 MongoDB 数据库交互。相关数据模式如图 1 所示。模式中的每个表都对应数据集中的一个 CSV 表、SQL 表或 MongoDB 集合。

表***repo-list*** 包含托管在 GitHub 上的项目元数据列表。每行代表一个项目。各表之间没有显式外键；相反，*“repo name”* 和*“filename”*是两个主要键，用于定位相应代码仓库的提交、议题和评论。

表***commit-list*** 包含***repo-list***中所有项目的提交；每行代表一次提交。

表 ***governance-change-commit*** 包含每个项目治理文件的提交记录。每一行代表一次提交。由于此列表是通过在本地分析 Git 代码仓库收集的，因此未展示任何与 GitHub 相关的信息。

表 ***governance-change-content*** 包含所有已收集项目中治理文件每次提交的行变更信息列表。每一行代表某次提交中的一条行变更信息。要定位某次特定提交中的所有行变更，应使用对象标识符（*oid*）、*repository.nameWithOwner*和文件名 的组合作为键。

表格 ***governance-change-commit-by-section***包含的信息与 ***governance-change-commit***相同。对于 ***governance-change-commit*** 表中的每一行，*‘sourceFile’*

[1]这是因为 npm 会在项目目录中安装软件包/依赖项，而项目开发者没有将依赖文件夹加入“gitignore”。[2] https://docs.github.com/en/rest/search?apiVersion=2022-11-28#search-code[3]因此，结果中会出现“Governance-Committee Charter.md”、“IPEP-29:- Project-Governance.md”和“GovernancePolicy.md”等文件。[4] https://github.com/graphql-python/gql

[5]本节指 Markdown 文件中两个标题元素之间的内容。标题元素由“#”“##”等符号后接标题内容构成。[6] https://github.com/mongodb/mongo-python-driver

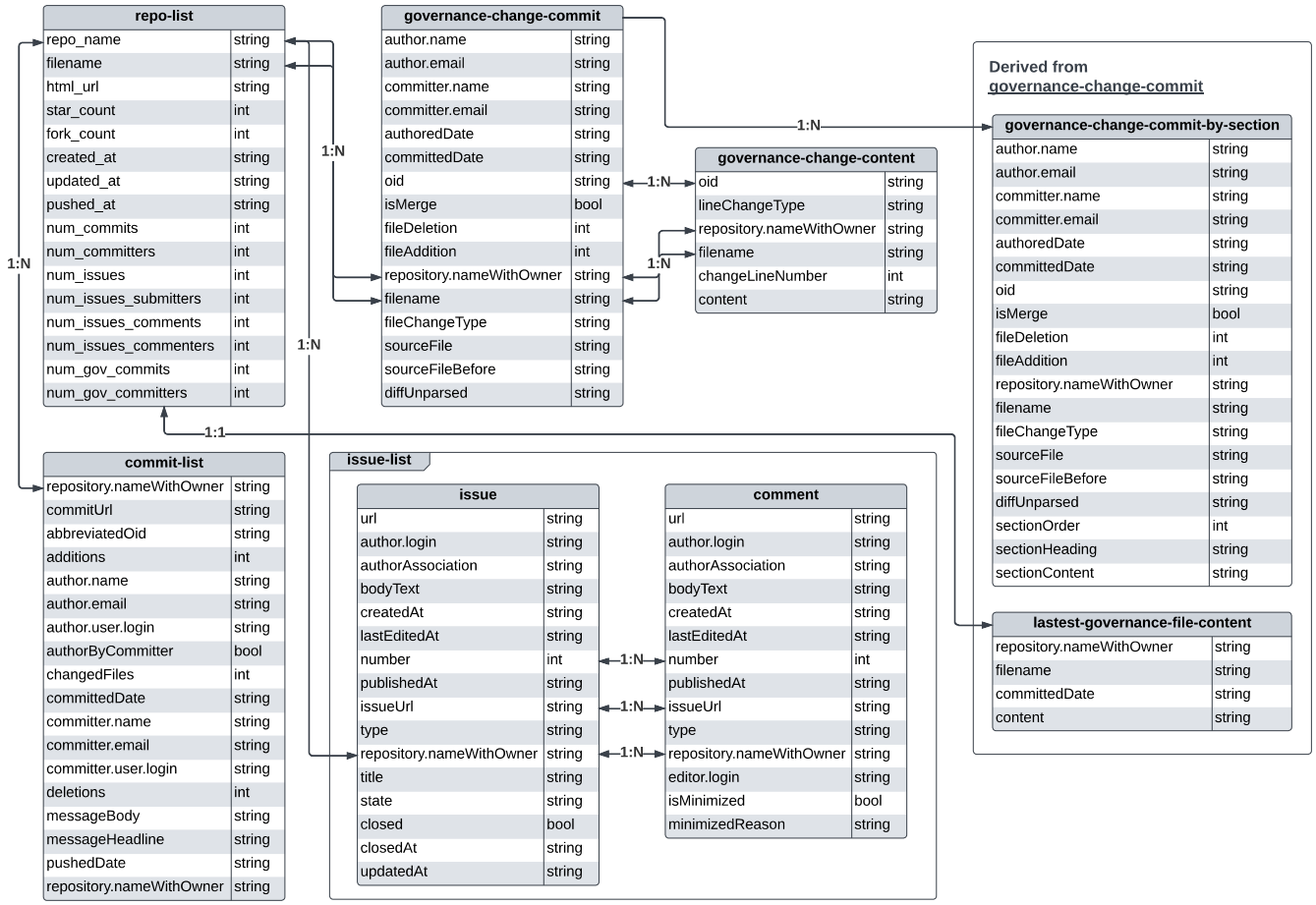


图 1. GitHub 开源软件治理文档数据集的数据模式。注：根据存档数据来源的不同，实际数据类型可能有所差异。例如，对于 MongoDB 转储文件，所有日期字段（如 'createdAt'）在 MongoDB 环境中都是原生的 日期 类型。

字段用于提取章节信息。每条章节信息都作为一行存储在 ***governance-change-commit-by-section*** 表中。大多数字段与***governance-change-commit*** 表中的字段相同。

表 最新治理文件内容 包含每个收集到的项目中治理文件的最新版本。

6) 指标： 我们根据开源软件治理文档数据集，计算了一组用于衡量开源软件项目活动的标准软件工程指标，包括：星标数（星标数）、分叉数（复刻数）、提交次数（提交次数）、提交者人数（提交者人数）、议题数（议题数）、议题提交者人数（议题提交者人数）和议题评论数（议题评论数）；对议题发表评论的人数（议题评论者人数）；对治理文件的提交次数（治理提交次数）；以及提交过该文件的人数（治理提交者人数）。表 I 列出了这 710 个托管于 GitHub 的项目在这些指标上的描述性统计数据。

IV. 潜在用例

本节介绍了基于所提供数据集可研究的两个潜在用例。

表 I 含有治理 Markdown 文件的 710 个托管于 GitHub 的项目的项目级统计数据

统计指标	Mean	中位数	标准差	Min	Max
星标数	1500.8	42.5	5902.7	0	85775
复刻数	308.3	23	1305.3	0	22861
提交次数	2,865.3	350.5	6,896.2	0	60,612
提交者人数	164.5	19	495.7	0	4,033
议题数	419.2	43	1,453.8	1	23,609
议题提交者人数	149.34	12	629.8	1	10,069
议题评论数	1,344	71	5,724.3	0	83,183
议题评论者数	168.9	13	842.1	0	14,285
治理提交数	5.5	2	9.1	1	79
治理提交者数	2.4	1	2.8	1	26

A. 案例一：研究数字公地中的治理

在线上语境下研究自组织自治的学者长期以来一直借鉴埃莉诺·奥斯特罗姆等公共池资源研究者的理论 [25]，[26]。该领域尤其重要的一项贡献是《可持续公共池资源设计原则》

管理 [27]。这些原则最初用于研究管理渔业、森林和水系统等自然资源的制度，但后来已扩展应用于开源软件和其他数字资源 [28]。然而，当前仍迫切需要检验这些设计原则在开源软件语境中的普适性，而这需要一个涵盖正式开源软件治理记录的综合比较数据集。例如，借助 GitHub 上大量正式治理文件的集合，学者可以通过提取对 GitHub 平台角色构念（如用户、提交者和贡献者）的引用，来检验“原则一：明确界定边界”。对于“原则五：对不遵守社区规则的资源使用者实施分级制裁”，学者可以搜索 GOVERNANCE.MD 文件，查看其中的规则是否包含多种制裁措施（既包括警告，也包括封禁，而不是二者之一）。借助该数据集，我们可以将这些原则应用于托管在 GitHub 上的任何开源项目，观察其实际运作方式，以及最佳实践是否与这些原则一致。我们还可以研究协作机制、社区中制定的规则与规范，以及维护者、协作者和贡献者等制度在项目治理中所发挥的作用。

B. 案例二：制度分析中的语义搜索

运用自然语言处理（NLP）方法对 GitHub 进行制度分析，可以研究在线社区的组织结构、规则和规范，以及它们与开源软件项目的治理和绩效之间的关系。借助分类后的句子，我们可以开展制度分析，理解治理机制。人们在日常话语中会以何种方式援引开源项目的规则？人们讨论的规则与实际使用的规则有何重叠？规则会如何随时间变化，变化方向又是什么？通常有多少人会参与规则制定？随着项目日趋成熟，他们所界定的不同角色通常有多少种？这些问题都有助于推动开源软件治理实践的发展，并促进同侪生产的有效开展。我们在此介绍的数据集，使在线治理研究的整体领域得以推进。

C. 案例三：治理设计模式的开发

通过考察来自不同社区的已实施治理代码仓库，治理实践者和工具开发者可以了解哪些治理组件较为常见。普遍存在的组件可以提炼为治理设计模式，进而纳入指南及其他材料，帮助正在决定采用何种治理方式的社区 [29]。这些模式也可以转化为现有软件工具包中的程序，以便通过编程方式实施治理 [30]。最后，GitHub 或第三方工具中治理相关功能的开发者可以针对常见治理组件构建定制化支持；例如，实现经常出现的特定角色、分级权限或投票机制。

V. 局限性与结论

局限性 我们注意到，该数据集在**完整性** 和潜在**偏差** 方面存在两个主要局限。**完整性** 1）由于 GitHub 的搜索 API 返回的结果既非确定，也不完整，因此该数据集并未涵盖所有包含治理文件的托管于 GitHub 的代码仓库。2）由于我们只收集了根目录中包含 GOVERNANCE.MD 文件的项目，因此有些托管于 GitHub 的项目未被纳入数据集，因为它们可能以不同方式组织和存储治理文件。例如，有些项目将治理文件直接放在自述文件中。3）我们在数据集中收集的提交历史不一定代表所有提交的完整历史。某些 Git 操作可能会丢失提交历史，例如压缩提交或变基。**偏差** 1）有些项目可能会使用相同的治理文件。在某些情况下，多个项目可能会在其项目级治理文件中放置相同的重定向链接，指向外部治理文档。这样一来，这些项目中的所有治理文件都会产生完全相同但没有实际意义的治理内容。2）有些项目最初并非托管在 GitHub 上；相反，这些项目及其相应的治理文件托管在其他平台上，并在之后的开发阶段迁移到了 GitHub。在这种情况下，治理文件的提交历史可能不完整，从而导致分析研究出现偏差。

为缓解上述完整性问题和偏差，我们可以：1）整合其他来源的数据，例如 GHTorrent 镜像的数据，以规避 GitHub API 的非确定性行为；2）利用现有的治理文件数据集训练分类器，判断某些文本文件是否包含治理内容，从而避免按特定模式搜索治理文件，并扩大搜索范围，纳入更多代码仓库。

结论 本研究介绍了一个纵向数据集的构建过程。该数据集涵盖托管在 GitHub 上的 710 个开源软件（OSS）项目，并包含有关治理文件的信息。尽管 GitHub 等托管平台广受欢迎，开源软件项目及其社区仍频频失败；治理文件有时会起草并修订，以满足社区的需求。

该数据集旨在帮助研究人员和开发者了解开源软件治理文档中的最佳实践和常见模式，并识别治理不善的项目，以促进这些项目从长远来看实现更好的可维护性和可持续性。我们介绍了数据的收集方式、用于筛选项目的具体标准，以及数据的描述和结构，并说明了数据收集过程中遇到的挑战。此外，我们还讨论了该数据集的潜在应用和价值，并鼓励研究人员利用该数据集研究不同项目和社区中的开源软件治理现状。

致谢

我们感谢匿名审稿人提出的建设性评论。本材料基于美国国家科学基金会支持的研究工作：GCR # 2020751/2020900“以循证规则和结构助力开源软件项目成功起步”，以及 DASS #2217652/2217653“推动开源软件项目转向负责任的社区治理”。

参考文献

[1] Y. Lu, X. Mao, Z. Li, Y. Zhang, T. Wang, 和 G. Yin，“角色重要吗？对 GitHub 非正式贡献者代码质量的调查”，载于*2016*年第*23*届亚太软件工程会议（*APSEC*）。IEEE，2016年，第49–56页。[2] S. St˘anciulescu、L. Yin 和 V. Filkov，“已毕业和已退役 ASFI 项目的代码、质量与过程度量指标”，载于第*30*届 *ACM* 欧洲联合软件工程会议暨软件工程基础研讨会论文集，2022年，第495–506页。[3] J. Gamalielsson 和 B. Lundell，“分叉之后开源软件社区的可持续性：LibreOffice 项目如何以及为何演变？”，《系统与软件杂志》，第89卷，第128–145页，2014年。[4] L. Yin、Z. Chen、Q. Xuan 和 V. Filkov，“Apache 孵化器项目的可持续性预测”，丛书：ESEC/FSE 2021。美国纽约州纽约市：计算机协会，2021年，第1056–1067页。

[5] [ ] 10 1145 3468264 3468563 在线。获取地址：https://doi.org/。/. I. Chengalur-Smith、A. Sidorova 和 S. L. Daniel，“自由/Libre 开源项目的可持续性：一项纵向研究”，《信息系统协会期刊》，第11卷，第11期，第5页，2010年。[6] J. L. C. Izquierdo 和 J. Cabot，“在开源系统中实现治理规则的定义与执行”，载于*2015*年 *IEEE/ACM* 第*37*届 *IEEE* 国际软件工程会议，第2卷。IEEE，2015年，第505–514页。[7] R. Li、P. Pandurangan、H. Frluckaj 和 L. Dabbish，“GitHub 开源软件项目中的行为准则讨论”，《ACM人机交互论文集》，第5卷，CSCW1，第1–31页，2021年。[8] A. Amrollahi、M. Khansari 和 A. Manian，“开源软件如何取得成功？开源软件成功研究综述”，2014年。[9] A. H. Ghapanchi、A. Aurum 和 G. Low，“衡量开源软件项目成功与否的分类体系”，《First Monday》，第16卷，第8期，2011年。[10] N. Ducheneaut，“开源软件社区中的社会化：一项社会技术分析”，《计算机支持的协同工作（CSCW）》，第14卷，第4期，第323–368页，2005年。[11] C. Bird、N. Nagappan、H. Gall、B. Murphy 和 P. Devanbu，“整合各方面：利用社会技术网络预测故障”，载于*2009*年第*20*届国际软件可靠性工程研讨会。IEEE，2009年，第109–119页。[12] C. Bird，“开源软件中的社会技术协调与协作”，载于*2011*年第*27*届 *IEEE* 国际软件维护会议（*ICSM*）。IEEE，2011年，第568–573页。[13] J. D. Herbsleb，“全球软件工程：社会技术协调的未来”，载于软件工程的未来（*FOSE'07*）。IEEE，2007年，第188–198页。[14] W. Scacchi，“自由/开源软件开发过程中的社会技术交互网络”，载于软件过程建模。Springer，2005年，第1–27页。[15] 2022年4月。 [在线]。获取地址：https://www.osgeo.org/ [16] L. Yin、M. Chakraborti、Y. Yan、C. Schweik、S. Frey 和 V. Filkov，“开源软件的可持续性：结合制度分析与社会技术网络”，《ACM人机交互论文集》，第6卷，CSCW2，第1–23页，2022年。[17] L. Yin、Z. Zhang、Q. Xuan 和 V. Filkov，“Apache 软件基金会孵化器项目可持续性数据集”，载于*2021*年 *IEEE/ACM* 第 *18* 届软件仓库挖掘国际会议（*MSR*）。IEEE，2021 年，第595–599页。[18] [在线]。获取地址：https://www.linuxfoundation.org/ [19] S. Due˜nas、V. Cosentino、G. Robles 和 J. M. Gonzalez-Barahona，“Perceval：随心所欲地获取软件项目数据”，载于第*40*届国际软件工程会议论文集：伴随论文集。ACM，2018年，第1–4页。

[20] G. Gousios，“GHTorrent 数据集与工具套件”，载于第十届软件仓库挖掘工作会议论文集，丛书：MSR '13。美国新泽西州皮斯卡塔韦：IEEE 出版社，2013年，第233–236页。

[21] [ ] =2487085 2487132 在线。获取地址：http://dl.acm.org/citation.cfm?id。在线。获取地址：https://chaoss.community/ [22] [在线]。获取地址：https://docs.github.com/en/rest?apiVersion = 2022 11 28 -[23] [在线]。获取地址：https://docs.github.com/en/graphql [24] D. Spadini、M. Aniche 和 A. Bacchelli，用于挖掘软件仓库的 *Python* 框架，2018年。[25] C. M. Schweik 和 R. C. English，互联网成功：开源软件公地研究。麻省理工学院出版社，2012年。[26] S. Frey 和 R. W. Sumner，“大量自治社区中综合制度的形成”，*PLOS ONE*，第14卷，第7期，文章编号 e0216335，2019年。[27] E. Ostrom，治理公地：集体行动制度的演进，丛书：Canto Classics。剑桥大学出版社，2015年。[28] S. Frey、P. Krafft 和 B. C. Keegan，“这个地方正是为其用途而建”：为参与式变革设计数字制度”，《ACM人机交互论文集》，第3卷，CSCW，第1–31页，2019年。[29] N. Schneider，“利用 communityrule 设计社区自治”，2020年。[30] A. X. Zhang、G. Hugh 和 M. S. Bernstein，“PolicyKit：在在线社区中构建治理机制”，载于第*33*届 *ACM* 用户界面软件与技术年度研讨会论文集，2020年，第365–378页。