🥏 开源项目核心维护者参与度变动对 Bug 修复效率与沟通成本的影响分析 ——以 Django（django/django）为例的实证研究

1. 基本信息

课程：开源软件基础（2025）

小组：第 12 组

完成日期：2026-02-02（按实际修改）

组员与分工

杨帆：组长、数据获取（CSV）

陈羽飞：数据清洗与绘图分析（CSV、图片）

魏汉启：代码质量分析（文档）

康维哲：成果总结补充与报告书写（文档）

2. 摘要（Abstract）

开源项目往往依赖少数核心维护者（Core Maintainers）承担关键修复、评审与合并工作。当核心维护者参与度发生变化（下降/波动）时，项目 Bug 修复效率与协作成本可能随之变化。本文以 GitHub 上 django/django 项目为研究对象，使用 GitHub API 抓取已关闭的修复类 Issue/PR 数据，构建**修复时长（duration_days）与沟通成本（comments_count）**等指标，并以"核心成员 vs 社区贡献者"进行对比分析。

在清洗后数据集中，我们获得 1602 条有效修复记录，其中社区贡献者 1410 条、核心成员 192 条。结果显示：核心成员修复时长长尾更弱，整体更稳定；两类贡献者评论数中位数一致（均为 1），但社区贡献者更易出现高评论长尾。进一步地，按月汇总后获得 29 个有效月度点，核心参与比例（core_ratio）与月均修复时长存在负相关（Pearson r = -0.3247，Spearman ρ = -0.4784），提示核心参与越高的月份整体修复越快。

关键词：开源协作；核心维护者；Bug 修复；沟通成本；GitHub API；Django

3. 研究背景与问题定义

3.1 背景动机

开源项目的治理结构通常呈现"核心—外围"特征：少数核心成员参与关键决策和合并，大量社区成员贡献补丁和讨论。当核心成员参与度降低时，可能出现：

评审/合并排队，修复周期变长

社区贡献者反馈变慢，讨论回合增加

项目维护风险上升（技术债、质量问题）

3.2 研究目标

本项目希望回答：

RQ1：核心成员修复 Bug 是否更快?

RQ2：核心成员与社区贡献者沟通成本是否不同?

RQ3：核心参与度变化是否与修复效率存在关系?

4. 数据来源与指标设计

4.1 数据来源

数据来自 GitHub 公共仓库：django/django

通过 scraper.py 使用 GitHub API 获取已关闭 Issue/PR

4.2 修复类任务识别规则

满足以下任一条件即判为"修复类任务"：

标题包含 fix / bug / fixed

label 中包含 bug

☑ 优点：实现简单、可复现 ⚠ 缺点：可能误判/漏判（将在有效性部分讨论）

4.3 核心成员定义

将仓库贡献者列表中的前 20 名贡献者定义为"核心成员（Core Member）"

其余定义为"社区贡献者（Community Contributor）"

4.4 关键指标定义

duration_days：修复时长（关闭时间 - 创建时间，单位天）

comments_count：评论数（沟通成本近似）

core_ratio（月度）：每月核心成员贡献占比 = core_fixes / total_fixes

5. 数据清洗方法

清洗脚本 数据清洗.py 主要处理以下噪声：

过滤测试 ticket：标题包含 #99999 或 my-feature

过滤自动化 PR：标题以 [Stitch Remote SWE] 开头

过滤零评论记录：comments_count > 0

过滤缺失关键字段的记录：issue_id / title / fixer_login 不为空

清洗完成后数据保存为：

data/django_bugs_filtered.csv

6. 描述性统计结果

6.1 样本量

Overall 总计：1602 条

Community Contributor：1410 条

Core Member：192 条

6.2 表1：数据统计表（N / 均值 / 中位数 / 四分位数）

（表1已保存在 tables/table1_summary.）。

Member Type N Duration_mean Duration_median Duration_Q1 Duration_Q3 Comments_mean Comments_median Comments_Q1 Comments_Q3 Community Contributor 1410 45.09 2.30 0.21 31.58 2.71 1.0 1.0 3.0 Core Member 192 16.74 2.99 0.71 13.42 2.41 1.0 1.0 3.0 Overall 1602 41.69 2.35 0.26 26.26 2.67 1.0 1.0 3.0 ☑ 表1关键解读（建议直接写进结论）

修复时长存在明显长尾：中位数约 2~3 天，但均值被长尾拉高（社区贡献者更明显）

核心成员更稳定：Q3（第三四分位数）核心成员为 13.42 天，显著低于社区贡献者 31.58 天

典型沟通成本相似：评论数中位数均为 1，Q1=1、Q3=3 一致，但社区贡献者均值更高，暗示长尾更强

7. 图表结果与分析

☑ 图1：修复时长对比（箱线图）

🔍 观察

社区贡献者组存在大量极端长尾离群点（几百天级别）

核心成员组离群点更少，上界更低，分布更集中

☑ 结论（对应 RQ1）

核心成员的修复更稳定、长尾更弱，整体效率更优。

☑ 图2：沟通成本对比（小提琴图）

🔍 观察

两组在低评论区高度集中（中位数=1）

社区贡献者更容易出现高评论长尾（沟通成本更高的少数案例）

☑ 结论（对应 RQ2）

典型沟通成本差异不大，但社区贡献者更可能触发高沟通成本情形。

☑ 图3：核心参与比例趋势（按月）

🔍 观察

核心参与比例随月份波动明显

部分月份接近 0，说明存在阶段性“社区主导修复”的情况

☑ 结论（对应 RQ3-趋势）

核心参与度并非恒定，存在显著波动，有必要进一步检验其与效率之间的关系。

☑ 图4：核心参与度 vs 月均修复时长（相关性）

你本次运行结果：

Monthly points used：29

Pearson r = -0.3247

Spearman ρ = -0.4784

🔍 解释

负相关意味着：核心参与比例越高的月份，月均修复时长越低（整体更快）

Spearman ρ绝对值更大，说明这种关系在“单调趋势”上更明显，且对离群点更稳健

☑ 结论（RQ3-相关性）

核心参与度提升与更快的整体修复效率存在中等强度的负相关关系。

8. 复现步骤

8.1 安装依赖

pip install PyGithub pandas matplotlib seaborn

8.2 数据采集

python scraper.py

8.3 数据清洗

python 数据清洗.py

8.4 生成图表与表格

python visualization_v2.py

9. 有效性威胁与局限性

核心成员定义是近似：用贡献者 Top20 代替真实 Maintainer 角色，可能偏差

修复类识别是启发式：关键词与标签可能误判/漏判

修复时长不是开发工时：duration_days 是 issue 生命周期（包含等待与排队）

相关性不等于因果：可能受版本发布窗口、重大版本周期等混杂因素影响

10. 结论与成果总结

10.1 结论总结

核心成员修复效率更稳定：长尾更弱，Q3 明显低于社区贡献者

沟通成本典型值相近：评论数中位数一致，但社区贡献者长尾更强

核心参与度与效率存在负相关：

Pearson r = -0.3247

Spearman ρ = -0.4784 核心参与比例越高，月均修复时长倾向越低

10.2 工程成果

可复现数据管线：采集 → 清洗 → 绘图 → 统计表输出

产出文件：

data/django_bugs_analysis.csv

data/django_bugs_filtered.csv

figures/*.png（4张）

tables/table1_summary.csv / .md

11. 未来工作

采用更严格的核心成员定义（review/merge 权限、组织成员等）

加入控制变量回归（样本量、版本周期、issue 类型）

将“代码质量分析”指标加入（复杂度、静态分析），形成效率+质量双维度结论

扩展到多项目对照，提高普适性