Creative Education Studies 创新教育研究, 2026, 14(7), 380-388
Published Online July 2026 in Hans. https://www.hanspub.org/journal/ces
https://doi.org/10.12677/ces.2026.147527

Hans汉斯

# 基于共建代码仓库的编程学习参与度与学业成效的实证研究

康静雨*，罗承茜，乔　丹

重庆城市科技学院人工智能与大数据学院，重庆

收稿日期：2026年6月3日；录用日期：2026年7月13日；发布日期：2026年7月22日

## 摘　要

针对程序设计课程中长期存在的看得懂却写不出、代码阅读能力薄弱与成绩分布偏低等问题，本文报告一项以全班共建代码仓库为载体的教学改革实践，并对学生的学习参与度与学业成效之间的关系进行实证分析。改革的核心在于让学生主动参与仓库的构建：每名学生在以学号命名的独立目录中提交代码与学习笔记，仓库通过持续集成工作流校验目录权限，教学范例库则覆盖从基础语法到区块链密码学的完整示例。本文以本校《Go语言程序设计》课程同一学期51名学生为对象，将其仓库参与度指标(代码行数、文件数、提交次数、笔记字数)与期末卷面成绩进行配对分析。结果显示，四项参与度指标与期末卷面成绩均呈正相关(代码行数$r = 0.39$，$p < 0.01$；文件数$r = 0.37$，$p < 0.01$；Spearman秩相关结论一致)；以文件数划分的高参与组($n = 16$)期末卷面均分为75.88，显著高于低参与组($n = 35$)的66.77 (Welch $t = 2.64$，$p = 0.012$，Cohen's $d = 0.73$，中等偏大效应)；分组阈值的敏感性检验显示结论不依赖于特定切点。两届课程的趋势比较显示整体成绩明显上移，不及格比例由37.5%下降至17.6%，降幅接近50%，及格率由62.5%提升至82.4%。研究表明，“参与即贡献”的协作机制与编程学习成效之间存在稳定的正向关联。基于上述发现，本文在讨论部分进一步提出“同伴语料锚定的AI辅助”这一面向未来的教学设计构想，为AI时代的编程教育提供新的研究方向。作为观察性研究，参与度与成绩之间的因果关系仍需通过引入对照组的准实验设计加以检验。

## 关键词

协作学习，实践共同体，编程教育，AI辅助编程

# An Empirical Study of Programming Learning Engagement and Academic Achievement Based on a Co-Built Code Repository

**Jingyu Kang*, Chengxi Luo, Dan Qiao**

*通讯作者。

文章引用: 康静雨, 罗承茜, 乔丹. 基于共建代码仓库的编程学习参与度与学业成效的实证研究[J]. 创新教育研究, 2026, 14(7): 380-388. DOI: 10.12677/ces.2026.147527

School of Artificial Intelligence and Big Data, Chongqing Metropolitan College of Science and Technology, Chongqing

Received: June 3, 2026; accepted: July 13, 2026; published: July 22, 2026

## Abstract

**This paper reports a teaching reform practice in a programming course that uses a class-wide collaboratively constructed code repository to address long-standing problems, including students' ability to understand code but difficulty in writing it independently, weak code-reading competence, and a generally low distribution of academic performance. It further conducts an empirical analysis of the relationship between students' learning engagement and academic achievement. The core of the reform is to engage students as active contributors to the construction of the repository. Each student submits code and learning notes to an individual directory named after their student ID; the repository uses a continuous-integration workflow to verify directory permissions; and the instructional example library provides complete examples ranging from basic syntax to blockchain cryptography. Using data from 51 students enrolled in the "Go Language Programming" course at the university in the same semester, this study pairs repository-engagement indicators, including lines of code, number of files, number of commits, and word count of notes, with final-examination scores for analysis. The results show that all four engagement indicators are positively correlated with final-examination scores, including lines of code, $r = 0.39$, $p < 0.01$, and number of files, $r = 0.37$, $p < 0.01$; the Spearman rank-correlation results are consistent with these findings. The high-engagement group, divided by number of files ($n = 16$) achieved an average final-examination score of 75.88, significantly higher than the low-engagement group ($n = 35$), whose average score was 66.77 (Welch's $t = 2.64$, $p = 0.012$, Cohen's $d = 0.73$), indicating a moderate-to-large effect. A sensitivity test of the grouping threshold further shows that the conclusion does not depend on a specific cutoff point. A trend comparison across two cohorts shows a clear upward shift in overall academic performance: the failure rate decreased from 37.5% to 17.6%, a reduction of nearly 50%, while the pass rate increased from 62.5% to 82.4%. The findings indicate a stable positive association between the collaborative mechanism of "participation as contribution" and programming learning outcomes. Based on these findings, this paper further proposes, in the discussion section, the future-oriented teaching-design concept of "peer-corpus-anchored AI assistance," offering a new research direction for programming education in the AI era. As an observational study, the causal relationship between engagement and achievement still needs to be examined through a quasi-experimental design with a control group.**

## Keywords

**Collaborative Learning, Community of Practice, Programming Education, AI-Assisted Programming**



## 1. 引言

程序设计类课程是计算机及相关专业的核心基础，但难教也难学几乎是任课教师的共识[1]。在笔者所授的《Go 语言程序设计》课程中，连续两届的课程考核质量分析均指向同一组症结：基础语法类题目

得分尚可，而程序阅读题难度系数高达 0.70 以上，成为主要失分点；学生成绩高度集中于中低分段，优秀段(90 分以上)的占比长期不足 5%。这反映出一种典型的学习困境——学生能看懂教师演示的代码，却难以独立写出，更难以阅读和推演他人的复杂逻辑。

这一困境的成因之一，在于传统编程课堂中学生接触的代码样本相对单一：要么是教材与教师提供的教学范例，要么是个人闭门编写、缺乏交流的习作。学生很少有机会系统地阅读同伴的真实代码，也缺乏在真实协作环境中迭代、修正的经验。与此同时，版本控制工具(Git/GitHub)进入课堂已被多门课程证明有助于培养协作与工程素养[2]-[4]。如何把版本控制平台从单纯的提交渠道升级为“共建–共享–共读”的学习共同体，是本研究的出发点。

笔者在课程中引入了一个全班共建的代码仓库：与传统教学方式相比，学生不再是被动接收资料，而是主动参与仓库的构建——提交自己的代码、撰写学习笔记、在统一的协作规则下迭代。这样做有两层意图。其一，共建本身构成一个学习共同体，学生在贡献过程中既阅读同伴代码、又被同伴阅读，从而把代码阅读从孤立训练变为日常协作的副产品。其二，全班共建的代码构成了一份贴近本班真实水平的语料，为后续探索“如何让自动化辅助工具的输出更贴近学习者最近发展区”等延伸课题提供了基础(详见第 6 节讨论)。

由此，本文提出三个研究问题：

• RQ1：学生在共建代码仓库中的参与度，与其编程学业成效(期末卷面成绩)是否存在关联？关联强度如何？

• RQ2：高参与度与低参与度学生的学业表现是否存在显著差异？

• RQ3：不同参与模式背后的学习过程具有怎样的特征？

本文的贡献有三：一是实施并系统呈现了“参与即贡献”的共建式代码仓库教学机制，包括其协作规则、目录权限工作流与反思笔记设计；二是基于同一学期、同一试卷、同一教师的 51 名学生样本，给出参与度与学业成效之间的剂量–反应式实证证据，规避了常见的组间不可比问题；三是通过对典型学生提交内容的案例分析，呈现参与背后的学习过程；并在讨论部分提出“同伴语料锚定的 AI 辅助”这一面向未来的延伸构想，为 AI 时代的编程教学提供新的研究方向。

## 2. 理论基础与相关工作

建构主义认为知识不是被传递的，而是学习者在主动建造外部作品的过程中习得的；Papert 的构造主义(constructionism)进一步强调，当学习者投入于建造一个对自己有意义的公共作品时，学习最为深刻[5]。共建代码仓库正是这样一个公共作品：学生提交的每一份代码都成为集体制品的一部分，这种公开的、累积的建造过程，为深度学习提供了天然的动机与场域。

Vygotsky 提出的最近发展区(ZPD)指学习者在他人帮助下所能达到的水平与其独立水平之间的区间[6]。这一理论对本研究尤为关键：同伴编写的代码，其复杂度与风格通常落在学习者的最近发展区之内，相较教师提供的专家级范例或工业级开源项目，更易被理解和吸收。换言之，读懂同班同学的代码比读懂标准库源码更可能发生在 ZPD 之内，这正面回应了质量分析所揭示的代码阅读能力薄弱问题。近年来，以最近发展区为理论基础的支架式教学已被引入程序设计类课程，通过为学生搭建适配其当前水平的脚手架来纾解学习难点[7]；本文的共建仓库可视为一种由同伴语料自然形成的分布式支架。

Lave 与 Wenger 提出，学习是参与实践共同体的过程，新成员通过合法的边缘性参与逐步从边缘走向核心[8]；Wenger 随后系统阐述了实践共同体的构成要素[9]。共建仓库可视为一个微型实践共同体：学生从最初仅提交模板化的初始文件(边缘参与)，到持续贡献代码、笔记与协作(趋向核心参与)，其参与轨迹本身即是一种学习。

将 Git/GitHub 作为学习对象和协作平台引入课程，已有不少实践经验。Beckman 等报告了在统计与数据科学课程中以版本控制为学习目标的实施经验与决策[2]；Tu 等总结了在课程中使用 GitHub 组织教学与协作的若干经验教训，包括权限管理与提交规范方面的实践[3]；在国内，曾少宁较早提出基于 GitHub 平台的协同式实验教学方法，以平台支撑实验任务的分发、提交与协作[4]。这些工作多聚焦于工程协作流程与实验管理效率，本文则进一步将仓库视为同伴学习语料的来源，并把学生的参与度作为可量化的研究变量。

以 GitHub Copilot 为代表的 AI 编程助手对学生编程行为的影响已成为研究热点。已有实证研究指出，AI 助手在提升编码效率的同时，也带来学生对建议不加审视即接受、削弱独立思考的风险[10]。在国内，徐宁与樊郁徽探讨了 AI 编程背景下程序设计类课程的改革，主张将 AI 编程与任务驱动教学相结合，并强化过程性评价[11]。这些研究为本文第 6 节的延伸讨论提供了背景，本文实证部分则聚焦于参与度与学业成效的关联，AI 辅助的相关构想留待讨论与未来工作部分展开。

Sweller 的认知负荷理论区分了由材料本身复杂度(元素交互性)决定的内在认知负荷与由呈现方式决定的外在认知负荷[12]。专家级范例或工业级开源代码的元素交互性往往远超初学者已有图式，带来较高的内在负荷；同伴代码所处理的问题规模与抽象层次更贴近学习者当前水平，其内在负荷相对可控，因而更易被纳入已有图式进行加工。需要指出的是，同伴代码在风格与结构上不及精心设计的范例规整，这要求教学中辅以适当的同伴互读与规范引导，以免风格噪声抬高外在负荷。总体而言，认知负荷理论为复杂度适配的同伴语料更易吸收提供了又一解释视角。

综合来看，已有研究为协作学习、版本控制教学与 AI 辅助编程分别提供了理论与实践基础，但较少有工作在同期可比样本上给出学生参与度与学业成效之间的剂量证据，也较少系统呈现由学生共建语料这一教学形态的实施细节。本文尝试在这一缺口上提供初步发现，并在此基础上探讨同伴共建语料对未来 AI 辅助教学的潜在意义。

## 3. 教学设计与实施

本课程为《Go 语言程序设计》，32 学时、2 学分，专业选修。本文实证分析所针对的本届课程共 51 名学生，分属两个平行教学班。

仓库以学生为单位组织：每名学生拥有一个以“班级_学号_姓名”命名的独立目录，目录下设 code/(代码)、note.md (学习笔记)，部分学生另有 experience/(实验)与 exam/(练习)子目录。仓库根部设 teacher/教学范例库，提供从第一章到第十三章的完整示例，内容覆盖基础语法、控制结构、数组与切片、函数、结构体等，并延伸至区块链密码学工具集(如 AES、RSA、ECDSA、Merkle 根等)，为学生提供可对照参考的教学范例。

学生被要求按照统一的协作规范提交：每次编码前先 git stash 暂存本地修改、git pull --rebase 拉取最新代码、再 git stash pop 恢复，以保持线性历史、减少合并冲突。为保障协作秩序，仓库配置了持续集成工作流(check-permissions.yml)，在每次推送或合并请求时自动校验改动是否仅限于提交者本人的目录，从机制上落实只改自己、不动他人的规则。这一设计使每名学生都对集体仓库负有明确而有边界的责任，参与的门槛与归属同时得到明确。

每名学生的 note.md 采用四段式模板，分别记录课程笔记、作业记录、项目经验与学习心得。笔记与代码并行提交，使仓库不仅沉淀写了什么代码，也沉淀如何理解与反思，为同伴阅读提供了代码之外的语境。

在上述共建仓库的基础上，本课程允许学生使用主流的编辑器进行编码，包括具备智能补全与 AI 辅助能力的编辑器(如 VS Code、AI 原生编辑器等)。教学过程中并未强制规定具体的编辑器或 AI 使用方式，仅要求所有提交均通过 Git 推送到共建仓库的个人目录。本届课程未采集学生使用 AI 辅助工具的过

程数据；关于“全班共建语料能否进一步提升 AI 辅助质量”这一延伸问题，将在第 6 节讨论。

## 4. 研究方法

本研究为观察性研究，以单门课程的真实教学数据为分析对象。主分析采用同一学期内的组内设计：所有被试来自同一学期、由同一教师讲授、参加同一份期末试卷，从而在最大程度上排除了试卷难度、教师、学期等混淆因素，使参与度与成绩的关联建立在可比样本之上。两届课程之间的成绩比较仅作为背景趋势呈现，不作为改革效果的因果证据。

分析样本为本届课程中能够在共享仓库与成绩表之间完成配对的 51 名学生。仓库中以学号命名的学生目录共 52 个，与成绩名册取交集后得到有效样本 51 人。

本研究从仓库中提取四项参与度指标作为自变量：代码行数(学生目录下全部.go 文件的总行数)、文件数(.go 文件个数)、提交次数(与该学生目录相关的提交计数)、笔记字数(note.md 等 Markdown 文件的字符数)。这四项指标分别从代码产出的量、产出的广度、参与的频次与反思的投入四个侧面刻画学生对仓库建设的参与程度，均为参与程度的客观代理变量；考虑到代码行数等可能受范例对照、提交粒度等因素影响，本研究以多指标交叉印证而非单一指标定论，并以非参数方法对结论作稳健性核验(见分析方法)。各指标的分布呈右偏特征：文件数中位数为 1、最大为 24；提交次数中位数为 1、最大为 9、合计 92 次。参与度在学生间存在较大差异，这一跨度恰好为考察参与度与学业成效的关联提供了充分的变异区间。

本研究以期末卷面成绩为主要结果变量，而非课程总评。原因在于，总评中的平时成绩部分本就可能依据仓库参与情况评定，若以总评为结果变量，将造成用包含参与度的成绩去解释参与度的循环论证。期末卷面成绩为独立闭卷考核，与平时参与的评定相互独立，更能稳健地反映学习成效。课程总评仅作为参考一并报告。

对四项参与度指标与成绩分别计算 Pearson 积差相关系数及其显著性；考虑到参与度指标呈右偏分布，另以 Spearman 秩相关作稳健性对照。鉴于同时检验多项指标，采用 Benjamini-Hochberg 法控制错误发现率(FDR)。在此基础上，以文件数是否大于 2 为界，将学生划分为高参与(活跃)组与低参与(模板)组，采用独立样本 t 检验比较两组的期末卷面成绩；考虑到两组样本量不等(16 对 35)且方差不齐，t 检验采用 Welch 校正(不假定方差齐性)，并以 Cohen’s d 报告效应量，另以 Mann–Whitney U 检验作非参数稳健性对照。为检验分组阈值的敏感性，对文件数大于 1、大于 2、大于 3 三种切分分别复算。统计分析使用 Python 的 SciPy 库完成。

本研究仅使用课程教学过程中自然产生的代码提交记录与考核成绩，分析时学号、姓名均作脱敏处理，不涉及额外的个人信息采集。

## 5. 结果

如前所述，四项参与度指标均呈右偏分布。以文件数计，中位数为 1、最大为 24；以提交次数计，中位数为 1、合计 92 次。据此以文件数大于 2 为界划分，活跃组 16 人、模板组 35 人。两组在参与程度上的明显落差，为后续比较两类学生的学业表现提供了清晰的对照。

四项参与度指标与期末卷面成绩均呈正相关(见表 1)。其中代码量类指标的关联最强且最稳健：代码行数与期末卷面的 Pearson 相关系数为 0.392 ($p = 0.0045$, 95% CI [0.13, 0.60])，文件数为 0.370 ($p = 0.0075$)，提交次数为 0.366 ($p = 0.0082$)；以 Spearman 秩相关作稳健性对照，结论一致甚至更强(代码行数 $\rho = 0.390$，文件数 $\rho = 0.435$，p 均 $< 0.01$)，说明右偏分布未对结论造成实质影响。经 Benjamini-Hochberg 法控制 FDR 后，上述三项仍在 0.05 水平上显著；笔记字数的相关相对较弱($r = 0.284$, $p = 0.0435$)，在更严格的 Bonferroni 校正下不再显著，故对其解读宜审慎。若以课程总评为结果变量，各指标的相关系数更高(代码行数达

0.504)，但如方法部分所述，这一数值因可能的循环论证而仅作参考。以独立的期末卷面为准，参与度与成效之间稳定地呈现中等强度的正相关，构成本研究的核心发现。

**Table 1.** Pearson correlations between engagement indicators and final examination scores
**表 1.** 参与度指标与成绩的 Pearson 相关

| 参与度指标 | 与期末卷面 | 与课程总评(参考) |
|---|---|---|
| 代码行数 | r = 0.392, p = 0.0045** | r = 0.504, p = 0.0002** |
| 文件数 | r = 0.370, p = 0.0075** | r = 0.469, p = 0.0005** |
| 提交次数 | r = 0.366, p = 0.0082** | r = 0.484, p = 0.0003** |
| 笔记字数 | r = 0.284, p = 0.0435* | r = 0.398, p = 0.0038** |
| 代码行数 | r = 0.392, p = 0.0045** | r = 0.504, p = 0.0002** |
| 文件数 | r = 0.370, p = 0.0075** | r = 0.469, p = 0.0005** |
| 提交次数 | r = 0.366, p = 0.0082** | r = 0.484, p = 0.0003** |

注：*表示 $p < 0.01$，表示 $p < 0.05$。经 Benjamini–Hochberg 法控制 FDR 后，前三项与期末卷面的相关仍显著；笔记字数在 Bonferroni 校正下不显著。以 Spearman 秩相关复核两项代码量指标与期末卷面的关联，结论一致(代码行数 $\rho = 0.390$，$p = 0.0047$；文件数 $\rho = 0.435$，$p = 0.0014$)。

活跃组(n = 16)的期末卷面均分为 75.88，模板组(n = 35)为 66.77，两组相差约 9 分。独立样本 t 检验表明这一差异具有统计显著性(t = 2.64, p = 0.012)，效应量 Cohen's d = 0.73，属中等偏大水平；非参数的 Mann-Whitney U 检验给出一致结论(p = 0.010)。在课程总评上，两组分别为 77.81 与 69.91，差异方向一致。需说明的是，分组所用的文件数本身即表 1 中已检验的预测变量之一，因此该组间比较并非对相关分析的独立印证，而是同一关系在二分尺度上的粗化呈现。为检验分组阈值的影响，对文件数大于 1、大于 2、大于 3 三种切分分别复算，组间差异均保持显著(Welch t = 2.36~2.64, p = 0.012~0.025, Cohen's d = 0.66~0.73)，表明结论不依赖于特定阈值的选取。

作为背景参照，本届与上一届同名课程的卷面成绩比较显示：卷面平均分由 62.06 提升至 69.63，及格率由 62.5%提升至 82.4%，不及格比例由 37.5%下降至 17.6%，降幅接近 50%；良好段(80~89 分)占比由 3.1%升至 23.5%，成绩分布整体向中上段集中。需说明的是，两届为不同自然班、试卷亦不相同，属非等组前后比较，上述差异反映总体趋势，不作为单一教学干预的因果证据。

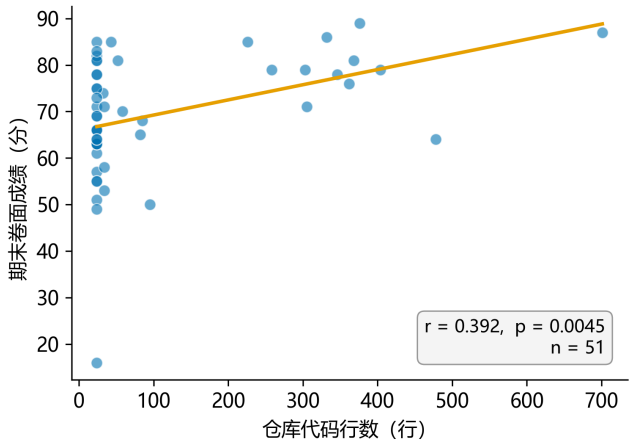


**Figure 1.** Relationship between lines of code and final examination scores (n = 51)
**图 1.** 代码行数与期末卷面成绩的关系(n = 51)

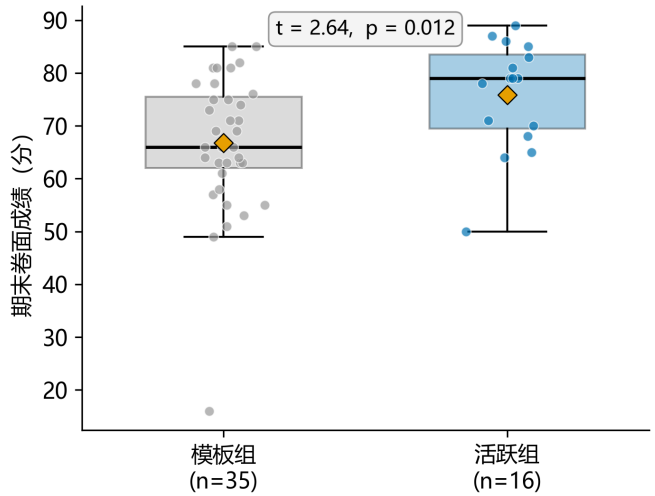


**Figure 2.** Comparison of final examination scores between the active group and the template group
**图 2.** 活跃组与模板组期末卷面成绩对比

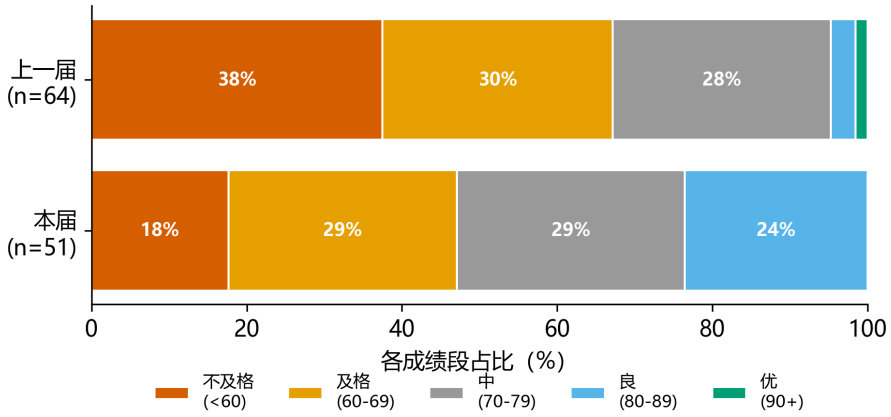


**Figure 3.** Comparison of segmented distributions of final examination scores across two cohorts
**图 3.** 两届课程卷面成绩分段分布对比

为更直观地呈现上述结果，本文提供三幅图：图 1 为代码行数与期末卷面成绩的散点图及拟合线，展示参与度与成效的剂量－反应关系；图 2 为活跃组与模板组期末卷面成绩的箱线图，展示组间差异；图 3 为两届课程成绩分段分布的堆叠条形图，展示总体趋势的变化。

## 6. 讨论

本研究在可比样本上发现，学生对共建仓库的参与度与其期末卷面成绩呈稳定的正相关，且高参与组显著优于低参与组。这一结果与“参与即贡献”的设计初衷相吻合：当学生持续提交代码、撰写笔记、在协作规则下迭代时，他们不仅在产出制品，也在反复经历阅读、修改与表达的认知过程。从理论上看，同伴编写的代码大多落在学习者的最近发展区之内[6]，其问题规模与复杂度更贴近学习者已有图式，所施加的内在认知负荷[12]低于专家级范例，因而更易被理解和内化。这为质量分析所揭示的代码阅读能力薄弱问题提供了一条可能的纾解路径：与其一开始就要求学生阅读标准库或工业级开源代码，不如先从读懂同班同学的真实代码起步。

同期样本中，参与度与成绩的正相关在中低分段学生中体现得较为充分，提示共建仓库与协作机制

的受益面可能主要集中于原本处于困难与中段的学生；两届趋势中不及格率的下降与良好段(80~89 分)占比由 3.1%升至 23.5%，与这一观察方向一致，可作为背景参照(但因属非等组比较，不能据此断定机制的因果效应)。在此基础上，改革可沿两个方向继续深化。其一是进一步降低深度参与的门槛：在课程早期加强版本控制工具的训练，并配合过程性评价的引导[11]，把更多学生从一次性的提交者引向持续投入的贡献者。其二是面向高阶能力拓展任务设计：在保障大多数学生达到基本要求的同时，为学有余力的学生增加复杂逻辑推演、深度代码审计等更具挑战性的阅读与设计任务，使协作机制的牵引力有望进一步延伸至高分段。这两个方向都顺应本研究观察到的正向关联，是对现有机制的自然延展。

本研究的实证部分关注的是参与度与学业成效的关联。但在 AI 编程助手日益普及的背景下，本研究的发现自然引出一个面向未来的研究构想：当全班学生共建一份贴近本班真实水平的代码语料时，这份语料可否作为 AI 编辑器的上下文，使其建议更贴近学情？这一设想在理论上颇具吸引力：已有研究表明，学生容易不加审视地接受 AI 的建议[10]，而若 AI 的输出能够参考同伴的真实写法，便有望降低“建议过于超前、脱离学情”的风险，使 AI 从一种孤立的工具转变为同伴语料生态中的协作伙伴。需要说明的是，上述构想超出了本文的实证范围，仅作为讨论性的研究方向提出；其有效性需要后续以专门设计的研究加以检验。

## 7. 局限与未来工作

作为一项观察性研究，本文的结论仍有一定局限性。其一，参与度与成绩之间为相关关系，尚不能据此作因果推断。这一相关之中存在不可忽视的自选择偏差(self-selection bias)：学习能力较强、学习动机较积极、对工程实践更有兴趣的学生本就更倾向于深度参与共建仓库，因而其较高的成绩可能并非完全由参与本身所导致，而部分源于这些先在的学生特质。这意味着观察到的高参与–高成绩组合，至少部分反映了“更努力的学生既参与得多、也学得好”的共变结构。同期可比样本的设计已尽力排除试卷、教师与学期等混淆，但严格的因果检验仍有待在后续研究中通过准实验设计加以规避，例如：1) 以平行教学班为对照组，一组实施共建仓库教学，另一组采用传统的程序设计教学方式，比较两组在控制相同试卷与考核条件下的学习成效；2) 在学期之初通过问卷采集学生的编程基础、学习动机、自我效能感、版本控制工具使用经验等先在变量，并将其作为协变量纳入回归模型，以分离参与机制本身与学生既有特质的独立贡献；3) 在条件允许时，考虑分阶段引入或随机化协作分组，以更接近随机对照实验的证据强度。其二，研究基于单一学校单门课程的 51 名学生，样本规模与来源限制了结论的外部推广，跨校、跨学期、跨课程的重复验证将有助于检验其稳健性。其三，本研究主要采用代码行数、文件数、提交次数、笔记字数等量化指标刻画参与度，这些指标固然客观可测，但难以反映参与的深度与质量。

这些边界同时指明了未来工作的方向。第一，引入更全面的效度评价体系。在量化指标之外，可借助代码质量评估工具(如静态分析工具、复杂度度量、规范性检查等)刻画学生代码的工程质量；可对学生的反思笔记进行内容分析或编码，以评估其反思深度；可记录并分析学生间的互动数据(如 issue 区的提问与回答、合并请求中的代码评审讨论)来揭示协作的真实质量；并通过学期末的问卷或半结构化访谈，收集学生对该教学模式的主观感知与学习体验，包括感知的难度、参与意愿、协作收获等维度。这一多维评价体系有助于将参与从单一的数量维度拓展为“数量–质量–感知”的立体框架。第二，强化研究设计的因果识别能力。在前述准实验设计的基础上，可考虑在仓库与编辑器中嵌入合理的过程数据采集机制，记录学生的编码节奏、版本控制工具使用熟练度等过程变量；并以专门设计的研究检验“同伴语料锚定的 AI 辅助”这一构想——例如，比较开放的同伴语料与无同伴语料两种条件下 AI 辅助建议的贴近度、学生对建议的采纳行为及最终学习成效的差异。第三，扩大样本与教学情境。跨校、跨课程、跨学期的重复研究有助于检验本研究结论的可推广性，并为后续的元分析积累证据基础。

## 8. 结论

本文报告了一项以全班共建代码仓库为载体的程序设计教学改革，并基于同一学期 51 名学生的可比样本，对“参与即贡献”协作机制的学习成效关联进行了实证检验。研究发现，学生对共建仓库的参与度与其期末卷面成绩呈显著正相关，高参与组的成绩显著高于低参与组(Cohen’s d = 0.73)，支持了协作共建机制与编程学习成效正相关的判断；两届课程的趋势比较亦显示整体成绩明显上移、不及格比例下降幅度接近 50%，为改革方向提供了背景印证。通过对四类典型学生的案例分析，本文进一步呈现了参与背后差异化的学习路径与质量。在讨论部分，本文还提出了“伴语料锚定的 AI 辅助”这一面向未来的延伸构想，主张以学生共建的真实语料作为 AI 辅助工具的上下文，使其建议更贴近学习者所处的最近发展区；这一构想为 AI 时代的编程教学提供了一个值得后续研究检验的方向。作为观察性研究，本文的结论受限于自选择偏差与单一样本来源等因素，相关因果机制仍需通过准实验设计与多维评价体系在后续工作中加以验证。

## 基金项目

2025 年重庆城市科技学院院级高等教育教学改革研究项目(202510802)。

## 参考文献

[1] 刘阳, 陈峰. C 语言程序设计课程教学改革探索[J]. 安徽工业大学学报(社会科学版), 2025, 42(3): 55-57.

[2] Beckman, M.D., Çetinkaya-Rundel, M., Horton, N.J., Rundel, C.W., Sullivan, A.J. and Tackett, M. (2021) Implementing Version Control with Git and GitHub as a Learning Objective in Statistics and Data Science Courses. *Journal of Statistics and Data Science Education*, **29**, S132-S144. https://doi.org/10.1080/10691898.2020.1848485

[3] Tu, Y.-C., Terragni, V., Tempero, E., Shakil, A., Meads, A., Giacaman, N., *et al*. (2022) GitHub in the Classroom: Lessons Learnt. *Proceedings of the* 24*th Australasian Computing Education Conference*, Virtual Event, 14-18 February 2022, 163-172. https://doi.org/10.1145/3511861.3511879

[4] 曾少宁. 基于 GitHub 平台的协同式实验教学方法[J]. 计算机教育, 2016(12): 144-148.

[5] Papert, S. (1980) Mindstorms: Children, Computers, and Powerful Ideas. Basic Books.

[6] Vygotsky, L.S. (1978) Mind in Society: The Development of Higher Psychological Processes. Harvard University Press.

[7] 李薇, 黑新宏, 杨媛, 等. 程序设计类课程的支架式教学模式[J]. 计算机教育, 2024(10): 207-210.

[8] Lave, J. and Wenger, E. (1991) Situated Learning. Cambridge University Press. https://doi.org/10.1017/cbo9780511815355

[9] Wenger, E. (1998) Communities of Practice. Cambridge University Press. https://doi.org/10.1017/cbo9780511803932

[10] Shihab, M.I.H., Hundhausen, C., Tariq, A., Haque, S., Qiao, Y. and Mulanda, B.W. (2025) The Effects of GitHub Copilot on Computing Students’ Programming Effectiveness, Efficiency, and Processes in Brownfield Coding Tasks. *Proceedings of the* 2025 *ACM Conference on International Computing Education Research* V.1, Charlottesville, 3-6 August 2025, 407-420. https://doi.org/10.1145/3702652.3744219

[11] 徐宁, 樊郁徽. AI 编程背景下程序设计类课程教学改革研究[J]. 创新教育研究, 2026, 14(3): 595-602.

[12] Sweller, J. (1988) Cognitive Load during Problem Solving: Effects on Learning. *Cognitive Science*, **12**, 257-285. https://doi.org/10.1207/s15516709cog1202_4