Research on the Impact of Vibe Coding on Traditional Software Development and the Improvement of AI Literacy for the New Generation of Programmers

Di Zhang[1,2]

1. School of Engineering, Sias University, Zhengzhou, Henan, 451150, China

2. Henan Intelligent Manufacturing and Digital Twin Engineering Research Center, Zhengzhou, Henan, 451150, China

Abstract

With the rapid implementation of big language model technology, the Vibe Coding paradigm characterized by natural language driven, human-computer collaborative trial and error iteration has gradually emerged, reshaping the traditional software engineering development model. Under the Vibe Coding paradigm, developers do not need to complete all code writing line by line. Instead, they rely on AI programming tools to complete code generation, debugging and reconstruction, focusing on demand disassembly, architecture design and code quality control. This paper combs the concept of Vibe Coding and compares it with traditional software development paradigm; Based on the practical operation results of Baidu Seconda, TRAE and Hermes, the capability boundary, application scenarios and defects of mainstream Vibe Coding tools are analyzed horizontally; Discuss the impact of Vibe Coding on software development process, project cost, code quality and developer capability from the positive and negative dimensions. Research shows that Vibe Coding is the inevitable direction of software engineering evolution, but there are hidden code defects, developer capability degradation, copyright and data security risks. Based on this, this paper constructs an AI literacy framework for developers, proposes an AI+X+N human-computer collaborative development system, which releases the efficiency advantages of AI tools while ensuring the engineering reliability and business value of software projects, and provides theoretical reference for human-computer collaborative software development practice.

Keywords

Vibe Coding; human-computer collaborative development; AI literacy

Vibe Coding对传统软件开发的影响及新一代程序员AI素养提升研究

张迪[1,2]

1.郑州西亚斯学院工学部，中国·河南 郑州 451150

2.河南省智能制造数字孪生工程研究中心，中国·河南 郑州 451150

摘要

随着大语言模型技术快速落地，以自然语言驱动、人机协同试错迭代为特征的 Vibe Coding 范式逐步兴起，重塑了传统软件工程开发模式。Vibe Coding 范式下开发者无需逐行完成全部代码编写，依托 AI 编程工具完成代码生成、调试与重构，自身聚焦需求拆解、架构设计与代码质量管控。本文梳理 Vibe Coding 的概念内涵，对比传统软件开发范式；基于百度秒哒、TRAE、Hermes三款工具实操结果，横向分析主流 Vibe Coding 工具的能力边界、适用场景与缺陷；从正反维度讨论 Vibe Coding 对软件开发流程、项目成本、代码质量与开发者能力带来的影响。研究表明，Vibe Coding 属于软件工程演进的必然方向，但存在隐性代码缺陷、开发者能力衰减、版权与数据安全等风险。据此本文构建面向开发者的 AI 素养框架，提出 AI+X+N 人机协同开发体系，在释放 AI 工具效率优势的同时保障软件项目的工程可靠性与业务价值，为人机协同软件开发实践提供理论参考。

关键词

Vibe Coding；人机协同开发；AI 素养

1 引言

传统软件工程长期依靠人工编码完成软件交付，需求分析、架构设计、编码、测试、运维等环节高度依赖程序员手工编写代码，大量人力消耗在模板代码、重复业务逻辑编写等低创造性工作，原型验证周期长，项目迭代成本较高。伴随代码大模型能力持续迭代，AI 编程工具从基础代码补全演进到支持项目全

【作者简介】张迪（1982-），男，中国河南淅川人，硕士，副教授，从事计算机软件与智能控制研究。

局理解、多文件重构与对话式调试，催生 Vibe Coding 这一人机协同编程范式。Vibe Coding 以自然语言交互、持续对话迭代、快速试错为核心特征，开发者通过自然语言描述需求，由 AI 生成代码，开发者持续校验、调试与优化，属于探索式开发范式。

当前行业与学术界对 Vibe Coding 的认知存在分化：部分开发人员过度依赖 AI 生成代码，缺少代码审查流程，引入工程隐患；另一部分群体排斥 AI 辅助工具，错失生产效率提升机会。现有国内外相关研究多聚焦代码大模型本身的技术评测，或是 AI 辅助编程工具的效能统计，缺少从开发者实操视角对多款 Vibe Coding 工具的横向对比，对该范式带来的综合性风险讨论不够充分，面向人机协同场景的开发者能力体系研究仍有待完善。本文结合实操分析 Vibe Coding 范式的双重影响，构建人机协同开发模型与开发者 AI 素养框架。

2 Vibe Coding 范式与传统软件开发模式对比

Vibe Coding 是依托代码大模型实现的人机协同编程范式，核心特征包含自然语言交互、快速原型生成、沉浸式对话开发、试错迭代、人机分工协同。AI 承担重复编码、语法纠错、基础调试等机械性编码任务；开发者负责需求拆解、系统架构设计、代码审查与业务逻辑把控。Vibe Coding 的本质并非以 AI 替代程序员，而是将程序员的工作重心从代码编写转向项目设计、质量审核与业务价值决策。

传统软件工程经过长期发展形成瀑布、敏捷等成熟开发流程，其优势在于流程标准化，代码可维护性强，适合大型复杂项目，系统稳定性可控；但存在固有短板，重复编码工作量大，需求变更成本高，原型验证周期漫长。传统范式强调前置完备设计，追求方案一次性落地；Vibe Coding 强调快速产出最小可用版本，边开发边调整。二者不存在绝对替代关系，可相互融合，在 Vibe Coding 开发流程中引入传统软件工程的编码规范、测试与评审机制，能够缓解快速试错带来的工程质量风险。

本文提出 AI+X+N 人机协同开发框架，用于规范 Vibe Coding 开发过程，平衡开发效率、代码质量与项目价值。其中 AI 作为开发底座，负责代码生成、补全、语法校验、简单 bug 定位、代码重构与文档编写；X 代表开发者掌握的技术栈、底层原理与行业业务知识，决定项目架构方向与业务合理性，属于不可被 AI 替代的核心要素；N 涵盖代码质量管控、安全合规、版权风险、项目可维护性、成本管控、持续迭代能力与商业价值等约束维度。该框架核心思路为借助 AI 提升编码效率，依靠开发者的专业能力把控架构与业务逻辑，通过多维度约束保障软件交付的合规性、稳定性与实用价值。

3 主流 Vibe Coding 工具实操对比与共性缺陷

本文选取百度秒哒、TRAE、Hermes三款主流AI编程工具开展对照实验，测试覆盖Web原型搭建、业务功能开发等六类场景，从编码速度等八个维度开展评估。

百度秒哒是基于文心大模型打造的国内首个对话式无代码应用开发平台，用户无需写代码，只用自然语言描述需求，多智能体协同就能快速生成H5、小程序等可直接使用的应用。它的核心优势是零门槛上手，非程序员也能快速落地创意。在针对 Django、Spring Boot等国内常用开发框架代码生成效果较好，开发和执行效率都不错。局限在于复杂大型定制应用需要多轮调整，极端高并发场景下运行效率仍有优化空间，部分垂直行业深度功能对接需要少量学习成本。笔者通过该平台完成了多个原型开发实践，发现其在与部分Skill的协同上仍有上升空间。

TRAE以IDE插件或客户端形式提供服务，基于VS Code改造，面向个人与中小项目轻量化开发，配置简单，中文适配好，全自动开发模式效率高，上手零成本，适配性强，性价比高；但处于迭代期，偶发卡顿，生态薄弱，缺少企业级能力，不适合涉密及大型项目，通过实践开发，笔者认为其效率和效果表现较好。

Hermes作为 AI 编程Agent，需要在本地主机上搭建Linux运行环境，但可以根据需求配置AI大模型和Skill，基于Powershell进行需求提示对话，其核心亮点是持久化跨会话记忆与自进化技能机制，完成任务后自动提炼可复用技能，后续同类任务直接调用并持续优化，核验代码需要通过相关的IDE如Pycharm进行，IDE提供的错误提示能迅速理解，并修正代码缺陷。缺点是配置过程较为繁琐，上手门槛高，无原生代码编辑器，编码效果依赖接入的AI大模型，并且无法预览开发效果，依赖相关IDE工具代为执行。

综上，三款工具适用场景各有侧重：百度秒哒适配新项目原型探索与中小规模代码重构；TRAE适合成熟项目日常增量编码；Hermes因其适配灵活性，适合较大型项目开发，开发者可根据任务类型组合选用。通过原型开发实践也发现三款工具存在共性问题：AI 生成代码常忽略边界条件，隐性缺陷较多；生成代码普遍存在代码冗余度大，代码整洁度与性能欠佳；AI 无法自动识别项目专属业务规则，易产出不符

业务规范的逻辑；若开发者直接使用不做代码审查，长期会削弱自身独立调试与排错能力。

4 Vibe Coding 对软件工程的双重效应

4.1 正向赋能效应

第一，缩短项目交付周期，提升开发效率。传统开发中大量时间用于模板代码编写与基础语法调试，AI工具可快速产出基础代码、定位简单错误，压缩原型开发周期，加快产品想法验证速度。

第二，优化开发流程，增强需求响应能力。Vibe Coding支持快速构建最小可用版本，持续迭代优化，可降低需求变更改造成本，提升项目对业务变化的响应能力。

第三，降低入门门槛，优化项目人力投入。AI工具可为初级开发者提供代码示例与语法解释，降低编程入门难度；企业可将基础编码工作交由AI承担，减少底层编码人力投入。

第四，转变开发者工作内容，释放创造性工作空间。程序员可减少重复性编码工作，更多投入需求挖掘、业务创新、架构设计等高价值工作，提升软件产品的业务价值。

4.2 潜在风险与负面冲击

其一，代码质量与系统稳定性风险。AI生成代码并非天然正确，常缺失异常处理与边界校验，未经人工评审、单元测试直接上线易引发线上故障；且AI产出代码风格杂乱、缺少注释，长期维护成本高。

其二，开发者能力退化风险。开发者若直接复制AI代码，不校验底层逻辑，长期会弱化独立编码、原理理解、故障排查能力，面对AI无法解决的问题时会丧失独立解决能力。

其三，冲击软件工程规范。Vibe Coding的快速迭代模式易诱导开发者跳过代码评审、单元测试等标准环节，形成“先保运行”的开发习惯，造成代码质量失控，增加长期维护难度。

其四，知识产权与数据泄露风险。大模型训练使用海量开源代码，AI生成代码可能与开源代码重合，引发知识产权纠纷；将企业业务代码、敏感需求输入第三方AI工具，存在数据外泄隐患。

总体而言，Vibe Coding不会取代传统软件工程体系，是软件工程发展新趋势，可提升生产效率，但架构设计、代码评审、软件测试、工程规范管控等核心环节不可省略，软件工程需在效率、质量安全、人才培养间建立平衡机制。

5 Vibe Coding 场景下开发者 AI 素养框架与 AI+X+N 体系落地路径

面向 Vibe Coding人机协同开发场景，新一代软件开发者需要构建五层 AI 素养体系：①工具应用素养：掌握主流 AI编程工具选型，熟练编写结构化提示词，精准传递开发需求；②代码甄别与审查素养：读懂 AI生成代码，识别逻辑漏洞、性能缺陷与安全隐患，完成代码重构、精简与规范化改造；③底层基础素养：扎实掌握数据结构、算法、操作系统、计算机网络、数据库等计算机基础理论，这是校验 AI代码正确性的根本；④业务与架构素养：具备需求拆解、模块划分、系统架构设计能力，把控业务逻辑，为 AI划定开发框架；⑤规范与风控素养：熟悉软件工程规范与测试方法，具备版权、数据安全意识，评估 AI代码带来的合规风险，建立常态化代码质量管控习惯。

配套的开发者能力提升策略包含四点：持续夯实计算机底层理论，避免对 AI工具形成依赖，保留独立编码、自主排错的训练；系统学习 AI编程工具特性与提示词工程，建立多工具组合使用策略；转变开发思维，从 “手动编写全部代码”” 转向 “设计、指挥、审核、优化代码”；建立强制 AI代码审查流程，所有 AI产出代码必须人工阅读、编写单元测试、校验边界条件并重构优化。

AI+X+N 人机协同体系落地分为三个阶段：①前期规划阶段（X 主导）：开发者完成需求分析、系统架构设计、模块划分，定义项目技术规范与业务约束，不直接向 AI交付模糊需求；②原型与编码阶段（AI+X 协同）：按模块拆分开发任务交由 AI生成代码，开发者逐段审查、单元测试、修正 AI错误，迭代优化；③质量与价值管控阶段（N 约束）：执行代码评审、集成测试、安全审计，评估项目长期维护成本，管控版权风险，保障交付项目具备稳定的业务价值。该机制兼顾开发速度与工程质量，避免快速迭代产出不可维护、存在安全隐患的软件系统，实现可持续的人机协同开发。

6 结论与展望

综上所述，Vibe Coding作为基于代码大模型形成的人机协同编程范式，改变了程序员的工作内容，但无法替代程序员；百度秒哒、TRAE、Hermes三款主流 Vibe Coding工具定位差异化，需要根据项目场景合理选型；Vibe Coding范式对软件工程兼具正向赋能与多重风险，可显著提升开发效率、降低项目人力成本，但同时存在代码质量隐患、版权与数据安全等问题；适配 Vibe Coding开发环境，软件开发者需要建立完整 AI素养体系，依托 AI+X+N 人机协同开发模型，通过明确人机分工，在提升开发效率的同时保障软件项目的工程可靠性与业务价值。

技术层面，未来代码大模型的逻辑推理能力、大型项目全局上下文理解能力将持续提升，Vibe Coding

工具对复杂业务架构的理解能力增强，代码错误率逐步下降。行业层面，人机协同开发将逐步成为软件工程的主流模式，企业会陆续制定面向 AI 生成代码的开发规范与自动化评审流程，将 Vibe Coding纳入标准化软件工程体系。人才层面，AI 人机协同开发素养将成为软件开发者能力评估的重要指标，掌握 AI+X+N 协同开发能力的复合型开发者将具备更强行业竞争力。

本研究的实践原型局限于中小型软件项目原型开发，后续研究可针对大型分布式企业级软件项目开展实证研究。随着代码大模型持续迭代，可持续跟踪新版本工具能力变化，更新工具对比结论。

参考文献

[1] 任志磊,高越,周志德等。基于大语言模型的代码生成智能体综述 [J]. 软件学报，2025,36 (08):3121-3145.

[2] 谭舒孺,肖宏彬,李智等.RD2ESC：多智能体嵌入式代码生成框架 [J]. 计算机研究与发展，2026,63 (04):789-804.

[3] 朱少民,王千祥.软件工程 3.0:大模型驱动的研发新范式 [M]. 北京：人民邮电出版社，2025.

[4] 张铭,李娜.AI 赋能软件工程场景下开发者核心素养研究 [J]. 计算机教育，2025 (09):112-118.

[5] Drosos I. Vibe coding: programming through conversation with artificial intelligence [J]. arXiv preprint arXiv:2506.-23253, 2025.

[6] Koch C. Agentic Agile-V: From Vibe Coding to Verified Engineering in Software and Hardware Development [J]. arXiv preprint arXiv:2605.20456, 2026.

[7] Gonçalves C A, Gonçalves C T. Assessment on the Effectiveness of GitHub Copilot as a Code Assistance Tool: An Empirical Study [M]//Lecture Notes in Computer Science. Springer, 2025:27-38.

[8] Li H, Zhang H, Hassan A E. AIDev: Studying AI Coding Agents on GitHub [J]. arXiv preprint arXiv:2602.09185, 2026.