RPG：从“对话”到“蓝图”，用图谱指导AI思考

2025年10月6日　2 分钟阅读

#AI #RPG #蓝图

> *RPG：从“对话”到“蓝图”，用图谱指导AI思考*

TLDR

微软新推出的**Repository Planning Graph (RPG)** 在代码生成领域确实表现突出，它通过用**结构化的图谱替代传统的自然语言规划**，解决了大规模代码库生成中的许多核心难题。

下面的表格梳理了它的核心效果:

评估维度	RPG (ZeroRepo框架) 表现	关键对比 (如 vs Claude Code)
功能覆盖率	在RepoCraft基准测试中达到**81.5%**	比最强基线高出**27.3%**
代码规模与复杂度	平均生成**3.6万行代码**，复杂度接近人工项目	生成规模是Claude Code的**3.9倍**
代码正确率	代码通过率达到**69.7%**	比Claude Code高**35.8%**
扩展能力	支持**近线性增长**，能持续稳定添加新功能	传统方法易过早进入平台期，无法持续扩展

技术原理：图谱如何驱动代码生成

RPG的核心创新在于，它不再依赖容易产生歧义的自然语言文档（如“先设计一个数据模块，再设计一个算法模块”）作为AI生成代码的规划蓝图，而是构建了一张**机器可精确理解和执行的结构化“工程图纸”**。其工作流程主要包括三个阶段：

> **提案级构建**：将用户需求（如“构建一个机器学习库”）通过检索大规模功能树，组合成一张功能图谱，节点代表从宏观模块到具体功能的各级任务。

目录

文章信息

字数

阅读时间

发布时间

更新时间

标签

#AI #RPG #蓝图

实现级构建：将功能图谱细化为具体的文件结构、接口依赖和数据流。例如，“支付处理”功能节点会被分解为`payment_service.py`、`api/payment_api.py`等文件，并明确其与订单、库存模块的依赖关系。

图驱动生成：按照图谱的拓扑顺序（依赖关系）逐个生成文件。这个过程遵循**测试驱动开发（TDD）**原则，即先为某个功能点生成测试用例，再实现代码，并通过测试进行验证，形成“生成-验证-更新”的闭环。

这种方法确保了从高层意图到底层代码实现的**一致性**，有效避免了传统方法中常见的功能遗漏、接口错乱和实现漂移等问题。

🚀 核心优势与潜力

除了表格中展示的量化效果，RPG还展现出两大关键优势：

提升开发效率：RPG为AI智能体提供了**全局的、结构化的上下文**。当需要修改或调试代码时，智能体可以像拥有“项目地图”一样，快速定位问题、追踪依赖关系，据称其定位效率能提升30-50%。

支持持续创新：在测试中，由RPG生成的代码库不仅能覆盖现有需求，还能产生约**11-13%**的“创新功能”，即超出初始需求描述但符合项目上下文的新功能。这表明RPG具备支撑软件持续演进的潜力。

总的来说，微软的RPG技术将AI生成代码的能力从**函数级、文件级提升到了复杂的仓库级**。它通过结构化的图谱规划，在生成代码的**规模、正确性、一致性和可扩展性**方面都取得了显著突破，为未来AI辅助的大型软件工程奠定了坚实的基础。当然，这项技术目前可能仍存在一些局限，例如对预定义功能树的依赖以及测试覆盖率的进一步提升空间。

引言：为何AI能写出完美代码片段，却构建不出完整的软件？

尽管AI能轻松处理代码片段，但当我们试图让它从一个简单的想法出发，构建一个完整、连贯、可运行的软件项目（即一个完整的代码仓库）时，挑战却呈指数级增长。从一个高层级的用户意图，到一个由无数文件、类和依赖关系构成的复杂网络，这中间存在着一道巨大的鸿沟。本文将深入探讨一项突破性研究，它揭示了问题的根源，并提出了一种全新的解决方案，可能彻底改变AI驱动的软件工程。

核心洞察：我们与AI对话的方式，正是问题的根源

这项研究得出了一个违反直觉的结论：我们一直以来依赖的、用自然语言与AI进行规划和沟通的方式，恰恰是构建大型软件项目的最大瓶颈。

当项目规模扩大时，单纯依赖于文本描述或对话式的计划会暴露出三大致命缺陷：

> **模糊性 (Ambiguity)**: 自然语言的灵活性是一把双刃剑。它很容易模糊用户意图与技术实现约束之间的界限，导致AI在理解和执行上出现偏差。
>
> **缺乏结构 (Lack of Hierarchy)**: 一个纯文本的计划列表是扁平的。当软件模块之间存在复杂的依赖关系时，用自然语言很难清晰、准确地追踪这些盘根错节的联系。
>
> **脆弱性 (Brittleness)**: 在一个漫长的开发周期中，静态的纯文本计划会随着项目的演进而逐渐"退化"，导致前后不一致、组件错位和设计脆弱等问题。

研究论文中的一段话精准地总结了这一点：

> *"While natural language remains a flexible and human-readable medium, it can often be less efficient for large-scale repository generation. Its inherent ambiguity may blur distinctions between intent and constraints, its lack of explicit hierarchy makes dependency tracking particularly difficult, and static plans may gradually degrade over long horizons without adaptive adjustment.(译文：虽然自然语言仍然是一种灵活且易于人类理解的媒介，但在大规模知识库生成方面往往效率较低。其固有的模糊性可能会模糊意图与约束之间的界限，缺乏明确的层次结构使得依赖关系追踪尤为困难，而静态计划若缺乏适应性调整，在长期运行中可能会逐渐失效。)"*

解决方案：从"对话"到"蓝图"，用图谱指导AI思考

为了克服自然语言规划的局限性，研究人员提出了一种名为"**仓库规划图谱" (Repository Planning Graph, RPG)** 的创新方法。我们可以将其形象地理解为一份给AI使用的、详尽的软件"施工蓝图"。

与自由形式的对话不同，RPG将整个软件项目的规划编码成一个统一的、结构化的图谱。这份蓝图并非凭空产生，而是通过一个严谨的两阶段规划过程构建而成。第一阶段是**“提案级规划”（proposal-level planning）**，决定要构建什么功能模块，如同建筑师确定大楼的功能分区。第二阶段是**“实现级规划”（implementation-level planning）**，详细定义如何构建，如同绘制出具体的施工图纸。

这份“蓝图”中包含了所有关键信息：

> 功能目标 (Functional goals and capabilities)
>
> 文件结构 (File structures)
>
> 数据流 (Data flows)
>
> 函数和类 (Functions and classes)

这一转变之所以如此重要，是因为它为AI提供了一个“持久且可演进的表示形式”（persistent and evolvable representation）。通过这份蓝图，AI可以在编写任何一行代码之前，就对软件的所有组件进行连贯一致的规划。它解决了自然语言计划中因信息碎片化和一致性差而导致的核心问题，确保了整个软件系统在设计上是完整和协调的。

惊人成果：代码规模提升68倍，且真实可用

为了验证“蓝图”方法的有效性，研究团队基于RPG开发了一个名为**ZeroRepo**的框架(尚未开源，但是有人根据论文实现了一个zerorepo），并在一个名为**RepoCraft**的真实项目基准上进行了评估。结果令人震惊：

> 代码规模巨大飞跃： 使用RPG的ZeroRepo框架生成的代码仓库，平均比最强的基线模型Claude Code大3.9倍，比其他基线模型更是大出惊人的68倍，平均生成了近36,000行有效代码和445,000个代码词元（code tokens）。
>
> 功能更完整，也更准确： ZeroRepo实现了81.5%69.7%27.3和35.8个百分点。这表明，基于蓝图生成的软件不仅功能更全，而且实现也更正确。
>
> 可持续的线性增长： 研究发现，依赖自然语言的基线模型在迭代几轮后便迅速撞上了“天花板”，其增加新功能的能力很快就趋于停滞（flatlined）。相比之下，基于蓝图的方法展现出了持续、近乎线性的增长势头，在30轮迭代中稳定地扩展功能和代码规模。这有力地证明了其在应对长期、复杂项目时的卓越可扩展性。

RPG对比BMAD和SDD

下面这个表格清晰地对比了 RPG、BMAD 和 SDD 的核心异同。

对比维度	微软 RPG (Repository Planning Graph)	BMAD-METHOD	SDD(Spec-Kit)
核心定位	**仓库级蓝图工程师**：专注于从零开始，为整个项目生成结构化的宏观规划。	**多角色AI团队模拟器**：模拟一个完整的开发团队（如产品经理、架构师），进行需求分析和高层设计。	**规范驱动开发工具集**：提供一套严格的文档规范和命令，将需求转化为可执行任务。
工作核心	**结构化图谱**：用节点表示功能/文件，边表示依赖关系，生成机器可精确执行的"工程图纸"。	**角色扮演与协作**：通过定义不同AI代理的角色和指令，进行多角度、非线性的讨论与规划。	**阶段化文档**：强制执行 `/specify`（需求）→ `/plan`（技术方案）→ `/tasks`（任务分解）的线性流程。
主要产出	大规模、结构严谨的完整代码仓库框架（可达数万行代码）。	产品需求文档（PRD）、系统架构设计、经过讨论和分片后的需求描述。	四大核心文档（`spec.md`, `plan.md`, `tasks.md`）和据此生成的模块化代码。
最佳适用场景	**项目冷启动**，尤其是需要高复杂度和架构一致性的仓库级代码生成。	**需求探索和战略规划阶段**，适用于需求模糊、需要多角色专业意见进行深度分析的复杂项目。	**工程化实现阶段**，适用于需求相对明确，注重**架构严谨性、代码一致性**和**测试驱动开发（TDD）**的项目。

💡 核心理念的深层解读

尽管三者都致力于提升AI辅助开发的效率和质量，但它们的底层哲学和实现路径各有侧重。

> **RPG：用"图谱"实现精确的规模扩张** RPG的根本创新在于，它用一张**机器可无歧义理解的结构化图谱**，取代了容易产生模糊性的自然语言规划。这张"工程图纸"定义了从功能模块到具体文件，再到它们之间的依赖关系和数据流。这使得AI能够像遵循精确算法一样，按图索骥地生成大规模代码，从根本上解决了代码仓库生成中的**一致性**和**可扩展性**难题。它的目标是实现从"想法"到"完整代码库"的**自动化、保真度高的翻译**。

> **BMAD：用"角色模拟"激发集体智慧** BMAD的核心思想是**社会分工与协作**。它不追求单一的、固定的输出格式，而是通过模拟一个完整团队中的不同角色（如`*analyst`业务分析师、

`*architect`系统架构师），让AI从多个专业视角对项目进行“头脑风暴”和“辩论”。这个过程更接近人类团队的决策方式，擅长处理不确定性和复杂性，其价值在于**探索过程本身**，旨在产出一份经过多轮锤炼、共识度高的高质量规划文档。

SDD(Spec-Kit)：用“规范”保障工程的严谨 Spec-Kit 的理念是**工程化与标准化**。它认为，清晰、阶段化的规范是高质量代码的基础。通过强制推行线性的三阶段流程（定义需求→技术规划→任务分解），它确保了从需求到代码的每一步都有据可依。特别强调的是，其任务分解阶段会遵循**测试驱动开发（TDD）**原则，优先生成测试任务，这为代码质量提供了坚实保障。它的目标是成为一套可靠的“施工管理手册”。

🤝 协同价值：从战略到战术的完美闭环

重要的是，**这三者并非互斥，而是可以形成强大的互补关系，构成一个从宏观战略到微观实现的完整工作流。**

BMAD 先行，进行战略探索：在项目初期，利用BMAD的多代理团队对模糊的需求进行深度分析，产出高质量的PRD和架构设计。

RPG 或 Spec-Kit 接力，进行工程实现：

对于**全新且复杂**的项目，可以将BMAD的产出作为输入，交由RPG生成整个仓库的结构化蓝图和核心代码框架。

对于**大多数项目**，可以将BMAD产出的清晰规划交给Spec-Kit，由其完成严谨的任务分解和代码实现。这种方式结合了BMAD的灵活性优势和Spec-Kit的工程化优势。

💎 总结与选择建议

简单来说，可以这样理解它们的角色：

BMAD 像是你的**战略顾问团队**，负责回答“我们要做什么以及为什么这样做”。

RPG 是一位**顶尖的蓝图建筑师**，负责回答“整体的结构应该如何科学地搭建”。

Spec-Kit 则是一位**严谨的工程监理**，负责回答“如何按照标准一步步高质量地完成施工”。

如何选择？

如果你的项目处于**从0到1的探索阶段**，需求不明确，需要深度规划，**BMAD** 会非常有用。

如果你需要**生成一个极其复杂且结构严谨的全新代码仓库**，那么 **RPG** 所代表的范式潜力巨大。

如果你的项目**需求已经相对清晰**，你追求开发的**规范性、可预测性和代码质量**，那么 **Spec-Kit** 是最直接、最实用的选择。

对于大型复杂项目，采用 **BMAD（规划） + RPG/Spec-Kit（实现）** 的组合策略，往往能取得最佳效果。

希望这个详细的对比能帮助你更好地理解这些前沿理念，并在你的项目中做出合适的选择。

结语：开发者的新角色——从“编码者”到“架构师”

这项研究预示着一个重要的转变：未来AI驱动的软件工程，可能不再仅仅依赖于我们与AI的“对话”和“提示”，而是更多地转向我们与AI“共同设计”结构化的规划蓝图。AI将扮演高效、精准的“施工队”，而人类开发者的角色则越来越趋向于项目的“总架构师”。

当AI能够遵循蓝图独立建造整个软件帝国时，人类开发者的新角色会是什么？我们是否正在从“砌砖工”转变为真正的“架构师”？

参考文献

论文：RPG: A Repository Planning Graph for Unified and Scalable Codebase Generation

分享这篇文章

相关文章推荐

智谱AI(GLM)

智谱AI(GLM)产品线收集整理分析

强化学习的奠基人的...

强化学习的奠基人惊人警告：...

腾讯AI产品线收集

腾讯AI产品线收集整理分析