Thoughtworks 技术雷达揭示：颠覆你认知的 5 个 AI 开发新规则

于 2025-12-29 21:07:01 发布

引言：当旧地图遇上新大陆

在人工智能飞速发展的今天，我们赖以生存的软件开发法则是否依然有效？这已不再是一个遥远的哲学问题，而是摆在每位技术从业者面前的现实挑战。Thoughtworks 最新发布的第 33 期技术雷达（一份着眼于 2025 年及未来的前沿指南），如同一份前沿指南，它不仅追踪着那些闪耀的新技术，更深刻地揭示了我们必须立即适应的思维模式转变。本文将从这份详尽的报告中，为您提炼出 5 个最具颠覆性、甚至有些反直觉的 AI 开发新规则，帮助您在这片由 AI 塑造的新大陆上，重新校准航向，保持领先。

1. 规则一：警惕“AI 加速的影子 IT”等新兴反模式

人工智能的加速应用不仅带来了前所未有的效率提升，也催生了新的陷阱——即“新兴 AI 反模式”。这些模式乍看之下似乎是捷径，但长期来看却可能将组织引向技术债和治理混乱的深渊。技术雷达特别指出了几个值得警惕的例子：

1. **AI 加速的影子 IT (AI-accelerated Shadow IT)**：AI 极大地降低了非技术人员构建和集成软件的门槛。这使得业务人员能够快速创建应用来解决燃眉之急，但也可能导致未经治理的应用呈爆炸式增长。这种现象类似于过去电子表格的无序扩散，虽然解决了短期问题，但长期会形成难以维护的技术债，加剧安全风险和数据分散。
2. **天真的 API 到 MCP 转换 (Naive API-to-MCP conversion)**：随着 AI 智能体（Agent）的兴起，许多团队试图将内部 API 直接转换为模型上下文协议（MCP，一种让 AI 智能体与外部工具和数据交互的开放标准），以便智能体调用。这是一个危险的捷径。这些为人类设计的 API 往往包含大量细粒度操作，直接暴露给 AI 会导致 Token 过度消耗和上下文污染。更严重的是，对于人类开发者，此类风

险尚可通过架构模式和代码审核来缓解，但这些人类中心的安全防线，对于自治的 AI 智能体却几乎无效，从而构成了一类全新的、难以管控的风险。

这些反模式的共同点在于它们“乍看有效”，但从长远来看，会导致系统适应性差、反馈迟缓或责任不清等严重问题。在拥抱 AI 带来的便利时，保持警惕是第一要务。

2. 规则二：对 AI 代码的自满是新型技术债

随着 AI 编码助手的普及，“对 AI 生成代码的自满”正成为一个日益严重的问题。尽管这些工具能显著提升开发速度，但多项研究证据敲响了警钟：

- GitClear 在 2024 年的研究发现，AI 辅助开发导致了重复代码和代码反复变动的增多，而关键的重构活动则有所下降。
- 微软的研究也指出，AI 助手带来的自信心增强，可能正以牺牲开发者的批判性思维能力为代价。

这个问题之所以严重，是因为随着编码智能体开始推动更大范围的自动化变更，由 AI 生成的大量代码变更集将变得更加难以评审。简单地加速编码环节，只会将压力转移到代码审查、测试和维护等下游环节，正如技术雷达的作者所指出的：

正如任何系统一样，流水线中的某一环提速后，其他环节的压力也会随之增加。我们的团队发现，要在生产环境中高效安全地使用 AI，必须重新关注和强化代码质量。

3. 规则三：从“提示工程”到“上下文工程”的思维跃迁

在 AI 应用开发中，“提示工程”（Prompt Engineering）已广为人知，但技术雷达指出，我们亟需一次思维升级：从关注提示词本身，跃迁到关注整个“上下文工程”（Context Engineering）。

上下文工程，是在推理过程中对提供给大语言模型的信息进行系统性设计与优化，以便让模型的内部层处于最优状态，从而稳定可靠地产出期望结果。它与提示工程的核心区别在于，后者更像是将模型视为黑盒并关注提示的措辞，而前者关注的是上下文的**整体配置**，即如何组织与传递相关的知识、指令以及先前的交互历史，以实现最有效的结果。其关键方面包括：

- **上下文设置**：通过最小系统提示词、规范的少样本示例等策略，为模型设定清晰、高效的初始状态。
- **上下文管理**：面对有限的上下文窗口，采用上下文摘要、子代理架构等方法来管理和持久化长期记忆。
- **动态信息检索**：让智能体在需要时才自主加载外部数据（即时检索），而非一次性灌输所有信息，从而最大化效率与准确性。

对于构建可靠、可控的生产级 AI 应用而言，掌握上下文工程，远比单纯优化几句提示词更为关键。

4. 规则四：AI 时代，基础设施的“拓扑结构”成为头等大事

在 AI 时代，我们必须重新审视基础设施。GPU 和 LPU（语言处理单元）不再是孤立的计算设备，而是一个个紧密耦合的加速器网络，其整体性能高度依赖于它们的物理放置位置和网络**拓扑结构**。

这一转变催生了“**拓扑感知调度**”（Topology-aware scheduling）这一核心概念。它要求调度系统在分配计算作业时，必须同时考虑硬件的物理布局和性能波动。忽略拓扑结构的简单调度器，可能会随意地将一个多 GPU 工作负载分散到网络延迟较高的不同位置，例如跨越了高速 NVLink/NVSwitch 互联的边界，导致整体效率大幅下降。

无论是需要高带宽、低延迟的分布式训练，还是对响应时间敏感的推理服务，都必须将硬件拓扑作为资源调度的首要考量因素。

我们认为，GPU 感知编排正成为基本要求——拓扑结构如今已成为首要的调度考量因素。

5. 规则五：别让你的数据工程团队成为瓶颈

AI 的发展对组织结构也提出了新的挑战。技术雷达明确指出，“**独立数据工程团队**”（Isolated data engineering teams）是一种反模式，必须予以警惕。

这种结构重复了过去隔离 DevOps、测试或部署功能的错误，不可避免地会创造出知识孤岛和沟通瓶颈。将数据工程师组织成一个独立的、与业务领域团队分离的中央团队，会导致数据工程师由于缺乏业务和领域上下文，很难设计出真正有价值的数据产品。

正确的做法是：让数据平台团队专注于维护共享的基础设施，而数据产品的构建和落地则应由遵循数据网格原则的跨职能业务团队来负责。在 AI 应用对领域信息丰富、高质量数据的需求持续增长的今天，打破这种组织孤岛，让数据能力回归业务一线，比以往任何时候都更加关键。

总结：AI 浪潮中的新航海图

Thoughtworks 技术雷达清晰地表明，拥抱 AI 远不止是采用新工具那么简单。它要求我们对开发实践、架构思维甚至组织结构进行一次根本性的重塑。旧的地图在新大陆上已然失效，我们需要的是一套全新的航海法则。从警惕新出现的反模式，到重塑我们对代码、上下文、基础设施和团队协作的认知，每一步都是对未来的关键投资。

当 AI 智能体成为我们团队的新“同事”时，我们面临着两种技术债：一种是它们在“影子 IT”中创造的混乱，另一种是我们因自满而接受的劣质代码。您认为哪一种对组织的长期健康更具威胁？