 DOI: https://doi.org/10.12345/bdai.v6i5.32062

Research on efficiency improvement of large-scale models in software development

Cheng Lin

China Telecom Chongqing Branch, Chongqing, 400039, China

Abstract

This paper focuses on the application of large models in software development, delving into their improvement benefits throughout the software R&D process. Leveraging their outstanding natural language processing and data analysis capabilities, large models are fully integrated into all stages including requirement analysis, planning/design, coding, testing/ 调试 , and operation/maintenance. This comprehensive involvement effectively shortens development cycles, enhances product quality, and reduces costs. However, challenges such as technical barriers, security concerns, privacy issues, and organizational adaptation remain during implementation. Through adopting various countermeasures, large models can achieve full and efficient utilization in software development, thereby providing robust support for related industries.

Keywords

large model; software development; efficiency improvement; R&D process

大模型应用于软件研发领域的效率提升研究

程林

中国电信股份有限公司重庆分公司，中国·重庆 400039

摘　要

本文重点于大模型在软件研发领域的应用，深入探究大模型在软件研发流程中的改进效益，大模型凭借其优秀的自然语言加工能力和数据分析能力，全流程融入软件研发的需求分析、规划设计、编码开发、软件测试、运维维护等各个环节，从而有效缩短软件研发周期，提升研发质量，降低研发成本，然而在应用过程中面临技术难关、安全隐私、组织人员适配等方面的问题，采取各种对策之后，大模型就能在软件研发当中得到全面而高效的运用，从而为相关产业赋予强有力的支撑。

关键词

大模型；软件研发；效率提升；研发流程

1 引言

在数字化浪潮之下，软件研发的速度与质量至关重要，传统的软件研发流程很繁琐，每个环节常因人员能力不足或沟通不畅等问题导致研发效率下降、成本增加。大模型的出现为软件研发赋予了新机遇，它可以理解复杂的请求，生成代码，改善结构，帮助测试并参与运维，有望重塑软件研发新格局，因此深入研究大模型在软件研发领域的应用，对于改良整个行业的效能，推进技术创新很有意义，这会在激烈的市场竞争里让企业抢得先机。

2 大模型对软件研发各环节效率的提升

2.1 需求分析：精准洞察与高效整理

传统软件开发时，需求分析这一环节就会碰到许多难题，客户真实需求模糊、表述不规范，导致开发团队与客户间沟通存在障碍，需求认识存在偏差，因此项目就得返工，这严重影响研发速度。Standish Group 的 CHAOS 报告指出，需求理解偏差导致的返工占项目成本的 40% 以上。

大模型的应用让需求分析发生改变，它有着很强的自然语言处理能力，能够将客户用自然语言描述的需求准确地加以解析。例如，当开发人员输入“开发一个可支撑海量用户在线协同编辑文档，还能做到随时保存和分享的功能”，大模型就能自动生成详细的用例图和时序图，就像 PlantUML 代码那样表现出来，明确地显示出用户同系统以及系统各模块之间的交互流程，还会列出具体的功能清单，明确标明各个功能的输入、输出和处理逻辑。

大模型还可以去分析大量的历史需求文档，找到现在需求中的冲突点，比如一个既需要“高实时性数据传输”又需要“低功耗离线运行模式”的项目需求，在需求分析时大模型就能发现这两个需求在技术上是冲突的，从而提醒开发团队和客户进行沟通确认，而不是在后续开发过程中才发现

【作者简介】程林（1989–），男，中国重庆人，本科，从事分布式应用软件架构优化研究。

这个问题，造成资源的浪费。

2.2 设计阶段：智能规划与架构优化

架构设计在软件项目中至关重要，且耗时较长，大概占整个项目时间的 20%，传统的架构设计依靠架构师的经验和专业知识，面对复杂系统时，易出现考虑不周或设计不合理等问题。

大模型可以实现自动化架构设计，当我们输入“创建一个基于微服务架构的电商系统”这个需求时，大模型就会立刻输出一个 SpringCloud 组件图，非常清晰地显示出各个微服务模块以及它们之间的关联，并且会生成 API 规范，很明确地显示各个模块之间的接口，还会形成数据库 ER 模型，规划好数据库的表结构和数据联系，像 IBM 的 AI 架构助手，它就能按照输入的系统需求，生成符合行业规范的 Kubernetes 部署模板，这样就极大缩短了架构设计所需的时间。

大模型还可以依靠以前项目的资料和性能分析，为架构做预先的性能优化，好比推荐恰当的缓存策略，若项目的数据读取速度要求比较高，大模型就会提议用 Redis 集群，这样可以提升数据读取速度，削减系统的反应时延，进而改善性能。

2.3 编码阶段：智能化生成与实时优化策略

编码是软件研发中耗时最长的环节，传统编码依赖程序员手动编写大量代码，不仅效率低下，且易产生错误，然而大模型的代码生成能力让编码变得很方便，比如电信 CodeFree、阿里通义灵码、GitHubCopilot、Cline 等。当开发者用 Python 代码进行数据清洗工作，然后在代码中写注释，说明要干什么，比如读取 csv 文件里的数据，去掉重复行，清洗某个字段的数据，Copilot 就能根据它学过的 Python 编程知识和常见的数据清洗模式，自动写出相应的 pandas 操作代码，通过实际测试，生成的代码正确率能达到 65.2%。

在编码的过程中，大模型还会对代码进行实时的优化，若开发者在编码时使用了比较低效的算法，例如线性搜索算法，大模型会根据代码的上下文，为出一个建议，使用哈希表算法，通过实测，哈希表算法在数据量大的情况下，搜索的速度可以是线性搜索的 100 倍。根据数据统计，大模型帮助开发者编码，平均编码速度提高了 250%，新手学习周期缩短 75%。

腾讯云通过 6 个月工业级项目实践发现，采用 LLM（大语言模型）与人工校验协同的工作流，代码产出效率提升 40%。GitHub Copilot X 的数据显示，资深开发者使用 AI 辅助编程后，代码产出效率提升 2.3 倍，而初级开发者的效能提升可达 5.8 倍。

2.4 测试过程：智能化覆盖与高效定位策略

传统人工测试难以覆盖软件所有边界场景，根据调查，人工测试只能覆盖到软件的 40% 边界场景，因此有些缺陷在测试时是无法被发现的。大模型在测试阶段的应用，提高了测试的效率和质量，对于测试用例的生成，输入一个登录功能的需求，大模型可以自动生成针对该功能的各种测试脚本，包括针对 SQL 注入攻击场景的测试，针对并发请求场景的性能测试等，涵盖了各种测试需求。缺陷定位上，大模型可以大量分析历史错误日志，学到不同缺陷对应不同的日志特征，能很快准确定位现在测出的日志是内存泄漏还是空指针异常等缺陷位置并给出合理的修改建议。

2.5 维护阶段：智能运维与自动修复机制

软件交付后的维护工作同样繁琐，传统的维护方式需要开发人员花费大量时间精力去查找问题，修复漏洞。大模型可实现智能运维，实时监控软件系统的运行数据，如网络流量、CPU 负载、磁盘占用、响应时长、异常日志等，根据大模型学来的正常运行模式和异常模式，它就能预估系统大概会发生哪些异常状况，比如说系统流量忽然发生异常波动，大模型靠之前的数据和流量变化趋势，可以预估系统即将发生的性能瓶颈，然后提前预警，并给出合理的扩容建议。故障出现之后，大模型可以自动故障治理，以 Kubernetes 集群为例，某一个微服务集群出现故障，大模型依靠故障特征以及前期修复经验，自动触发指定 POD 回退到上一个稳定版本，这样一来就节省了人工定位及处置时长，降低了 RTO 时长。

3 大模型应用于软件研发的优势与成效

3.1 效率提升显著

从软件研发全流程来看，大模型在每一个环节都极大地缩减了时间消耗，在需求分析环节，大模型的快速准确解析能力，使需求分析时间极大缩短，比传统方式节省 40% 以上时间。在架构设计环节，自动化设计和性能预优化功能，使设计周期极大缩短，效率提高 50%。在编码环节，代码生成和实时优化功能，使开发人员编码速度极大提高，开发周期平均缩短 30% ~ 50%。在测试阶段里，智能测试用例生成和高效缺陷定位达成测试周期缩减至 50% 以上，维护环节的效率较传统方式提升 40%。从上述各个方面来看，采用大模型辅助软件研发，整个项目研发过程的开发时间缩短了 20% ~ 60% 左右。

3.2 质量大幅改善

大模型在缺陷检测与测试上有着强大的功能，软件质量得到了提升，在编程的过程中，大模型可以实时检测代码缺陷，可以减少 30% 以上的代码缺陷，在测试的过程中，可以生成大量的测试用例，可以准确的定位到缺陷，使测试的覆盖率和缺陷检出率都得到了很大的提升，像 CypressCopilot 可以提升缺陷检出率 180%，这样就可以降低软件发布之后的故障率，根据 GitHub 调研数据显示，Copilot 可以帮助开发者在编码过程中修复三分之二以上的漏洞，高质量的软件不仅可以减少后期的维护成本，同时可以提升用户的使用体验，提高软件产品的市场竞争力。

3.3 成本有效降低

效率的提升和质量的提高，直接带来了成本的降低。一方面，研发周期缩短，人力成本、时间成本、硬件资源等成本降低。另一方面，软件质量提升，后期维护、修复缺陷的成本降低。麦肯锡研究表明，大模型辅助软件研发，整体研发成本可降低 20%-40%，大模型减少了对开发人员经验、技能的过度依赖，新开发人员借助大模型工具，可以很快上手并开发出符合规范的代码，节省人力资源培训成本。

4 大模型应用于软件研发面临的挑战

大模型在软件研发应用中面临多方面挑战，技术上的代码正确性问题和性能问题并存，生成代码时经常会出现“代码幻觉”，即生成一些不存在的 API 调用，或者生成代码出错，复杂业务场景错误率达到 10% ~ 20%，生成代码还可能因算法选择不当、代码结构冗余导致运行效率低下，部分代码的响应时间较人工优化版本慢 30% ~ 50%；安全与隐私风险，数据泄漏风险高，训练数据中存在敏感信息，若安全措施不当就会造成泄漏，带来声誉和法律风险，模型学习开源代码时无意识地复制了受版权保护的部分也会产生版权纠纷。组织与人员适应难题体现为开发人员需掌握 Prompt 工程、AI 调试等新兴技能。

5 大模型应用于软件研发面临的挑战及应对策略

5.1 技术层面挑战与应对

大模型生成代码时存在准确率和效率问题，因为模型接受的训练数据有限，同时对于语言的理解存在歧义，因此会出现“代码幻觉”现象，就是模型生成的代码片段看上去很合理，但实际上存在着逻辑错误。这种问题在像复杂的金融系统交易逻辑、医疗系统数据校验规则等业务环境中，一旦出现了逻辑上的错误，就有可能引发非常严重的业务事故。针对这种情况，可以采用 JUnit 这种动态检测方法和 SonarQube 这种静态分析方法，通过为代码做单元测试以及整合测试来检验代码的逻辑是否正确。

利用 JUnit 可以自动运行单元测试测试案例，快速找到代码中的语法错误和逻辑漏洞，SonarQube 是从代码结构、代码质量的角度出发，做静态分析，找出代码异味、安全隐患，建立代码审查机制，开发人员要根据业务需求对代码再校验一遍，保证代码逻辑符合实际业务场景，针对性能瓶颈，使用 AI 优化器 TensorFlowModelAnalysis 来对代码做性能剖析，找出低效的算法和结构，为出优化建议，在模型训练时学习性能指标，让大模型产出更高效的代码，降低响应时延。

5.2 安全与隐私风险应对

数据安全和版权风险要重点防范，大模型训练时，含有大量的代码数据，用户数据以及业务数据，若这些数据外泄或者被滥用，会为企业以及用户带来极大损失，企业应当按照最小必要原则来搜集数据，只获取业务有关的必要数据，缩减数据外露的风险，利用 AES 加密存储，HTTPS 传输保障数据安全，AES 加密算法可以对存储的数据执行高强度加密，防止数据在存储期间被窃取，HTTPS 协议可以保证数据在网络传输期间的安全，防止数据被篡改或者被监听。采用 RBAC(基于角色的访问控制)策略来控制访问权限，根据不同用户角色和职责分配不同数据访问权限，防止越权访问，利用联邦学习技术，实现数据‘可用不可见’，在不泄露原始数据的前提下，完成多机构间的模型协同训练，提高模型训练效果。

5.3 组织与人员适应策略

开发人员技能转型与工作流程重塑十分关键，大模型在软件研发范畴应用之后，传统的开发模式和技能需求产生了极大的变化，开发人员要掌握 AI 工具的使用方法，明确模型生成的逻辑等新的技能，构建起分层技能培训体系，面向所有人执行基本的认知培训，使得大模型的基本原理，应用场景及潜在价值能够被人所知，针对核心人员展开更深入的技能培训，加强他们在模型调优，代码优化等层面的专业技能，按照高校资源制订课程，依靠高校的科研优势和教育力量，为企业培养适合的人才，并形成考核制度以确保培训效果。工作流程调整上，采用渐进式方法，先从非核心项目开始尝试“人机协同”，让开发人员逐渐习惯、适应大模型辅助开发的方式，成立流程改进小组收集反馈后改进，按实际项目经验不断改善工作流程，优化开发速度，形成跨部门合作机制，通过经验交流会实现知识共享，增进团队合作。

6 结语

大模型在软件研发的各个部分都有提高效率的巨大潜力，能大幅度缩短开发时间，改良软件品质，缩减成本，然而也碰到技术准确度，性能，安全隐私，人员组织适应等难题，采用专门的技术方式，强化安全管理，促使人员技能转型，改变流程等策略可以解决这些难题，以后随着技术发展和经验积累，大模型会在软件研发方面起更大作用，推动整个行业迈向高效，智能的新阶段。

参考文献

[1] 王松,王晓霞.持续改进方法在软件研发效率与质量提升中的实践[J].企业改革与管理,2025,(05):166-168.

[2] 王亦成.敏捷开发模式在大规模软件项目中的应用[J].集成电路应用,2025,42(01):124-125.

[3] 程琳.基于RFID的大规模设备信息监控软件设计与应用[J].软件,2024,45(03):110-112.

[4] 吴超.大规模项目软件测试机制研究与应用[J].中国新通信,2023,25(14):67-70.