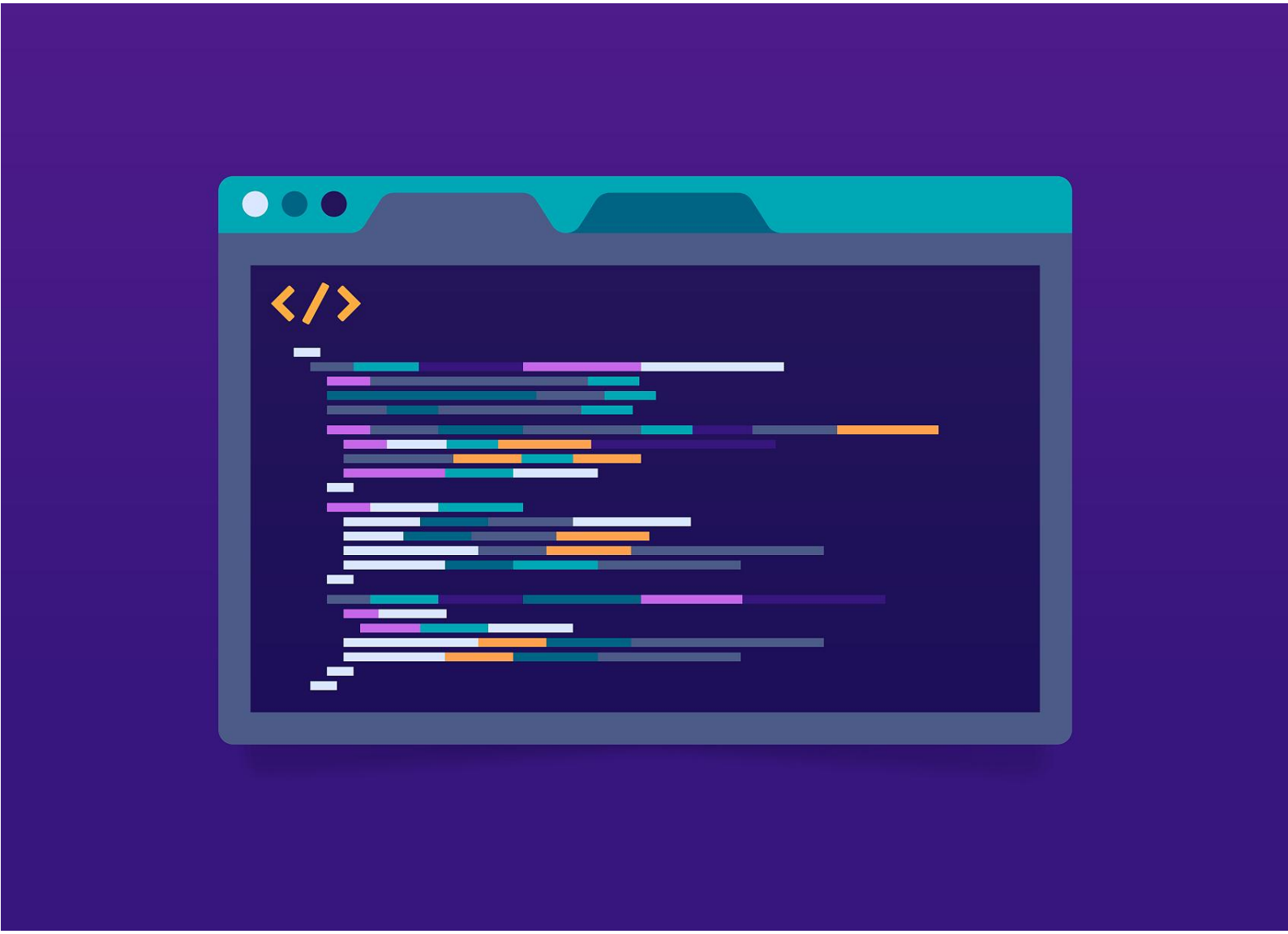


采用 AI 编程助手，发展新质生产力

「商汤日日新 • 代码小浣熊」提高企业开发高质量软件应用的能力，为开发者提供新质生产力工具

联合发布方

InfoQ、稀土掘金、SegmentFault 思否、开源中国、CSDN

DataWhale、RTE 开发者社区、FounderPark、异步社区、商汤智能产业研究院

指导委员会

祁　宁　SegmentFault 思否创始人兼 CTO，Apache Answer、Typecho 等开源项目作者

陈冀康　人民邮电出版社信息技术分社社长

石亚琼　Founder Park 研究中心负责人，变量资本投资合伙人

姜昕蔚　InfoQ 研究中心研究总监、首席分析师

林日华　开源中国总编

范晶晶　DataWhale 团队负责人

张　涛　商汤科技 Copilot 应用技术负责人

詹明捷　商汤科技 AIGC 研究与应用团队负责人

贾安亚　商汤科技 Copilot 产品负责人

田　丰　商汤智能产业研究院院长

鸣谢

王尚　徐培源　郭晓际　赵倩影　卜瑶函　陈木青

作者

刘 亮　商汤智能产业研究院战略研究主任 liuliang2@sensetime.com

商汤日日新·代码小浣熊

目录

关键发现 4

第一章 AI 编程助手是开发者应掌握的新质生产力工具 6

1. 在 AI 编程助手的赋能下，企业可以实现更全面的高质量发展 7

2. AI 编程助手遵循 KRE 理论架构不断升级满足生产力发展需求 8

第二章 借助 AI 编程助手进入“心流”状态，成为 10 倍开发者 11

1. 聚沙成塔，AI 编程助手有效提高开发者的生产力 13

2. 超越生产力，AI 编程助手还可以提高开发者的工作体验 20

3. 保持理性，穿越炒作：从 AI 编程助手工具的采用中获得最大价值 23

第三章 AI 编程助手让开发者从执行者变成了代码协调者 26

1. 保持开放心态，提升自身能力，在变革中寻找成长与机遇 26

2. 当好协调者与指挥者，开发者自身角色定位在发生转变 28

第四章 群雄并起，中国大模型厂商逐鹿 AI 编程助手时代 31

1. 中国 AI 编程助手市场发展趋势与格局 32

2. AI 编程助手评估框架：基模型、产品工程、持续迭代、市场生态 36

3. 谁塑造了这个市场？—— 商汤日日新·代码小浣熊 38

4. 商汤日日新·代码小浣熊成为软件开发新时代的创新基石 40

第五章 不可忽视的另一面：AI 编程助手的风险和“陷阱” 46

第六章 他山之石，参考行业最佳实践加速部署 AI 编程助手 50

1. 海通证券开发提效 40%+，打造“泛海言道”大模型促进创新发展 50

2. 澳新银行提高了 42%生产效率，代码质量提升了 12% 53

3. 某商业银行实现提效 35%，代码补全平均响应时间 50ms/token 55

4. 某车企通过 AI 编程助手实现软件开发现代化，效率提升超过 50% 55

第七章　建议：不要等待，行动起来，完美是优秀的敌人 57

第八章　展望：从 Copilot 到 Agent，迈向 AGI 彼岸 60

关键发现

1，**遵循 KRE 理论架构的 AI 编程助手凭借强科技属性成为开发者应该掌握的新质生产力工具，企业通过 AI 编程助手可以实现全面的高质量发展。**AI 编程助手能够解放开发者，释放更高效率的生产力，不仅可以持续推动企业的数字化转型，并且可以贡献额外的经济效益。对于企业缺少开发者人才、消弭高质量软件应用开发的障碍等挑战，AI 编程助手均可以有效解决。遵循 KRE（知识-推理-执行）理论架构使得 AI 编程助手可以伴随大模型、生成式 AI 等技术的发展而进化，具备世界知识、理性思维和互动能力，不断满足生产力发展需求。

2，**AI 编程助手可以帮助开发者提高自身生产力，改善工作体验，帮助开发者进入和维持高效率的"心流"状态，让他们聚焦关键业务，缩短产品上市时间。**AI 编程助手可以有效减少编程时间投入、减轻编写代码的压力、产出高质量代码、实现赋能软件开发生命周期的关键环节。AI 编程助手还可以最大限度地减少认知疲劳，提高解决问题的能力，加强协同效力，更好的实现知识共享。AI 编程助手不仅赋予开发者新的技能加持，还加快了开发者的学习速度，培养创新能力。最后，通过 AI 编程助手，开发者可以更好的将高效时间投入到高价值事务当中，及时满足用户需求，交付更有价值的软件应用，实现助力业务发展的最终目标。

3，**开发者正在面临角色转换的挑战，如何快速完成角色转换，从代码的执行者变成代码的协调者、AI 工具的使用者，是应对生成式 AI 发展浪潮的关键行动。**开发者的工作目标就是持续的及时的交付软件应用满足客户需求，而 AI 编程助手是在让这个目标的实现更加高效和轻松。开发者不必焦虑于 AI 编程助手是否会替代自己，而通过自身技能的提升，操控 AI 编程助手为自己服务。新工具涌现不会停止，软件开发的需求也只会越来越旺盛，开发者应该保持持续学习的习惯，让自己的身份完成华丽的转变，成为协调者、指挥者和系统思考者。

4，**AI 编程助手评估框架可以帮助开发者和企业更好的理解市场上的相关产品，并为采购选型提供参考依据。**AI 编程助手评估框架从基模型、产品工程、持续迭代、市场生态四大维度 16 个二级指标来对厂商进行整体评估和分析，帮助开发者和企业避免被厂商的宣传和媒体的渲染造成困惑和失焦。整体而言，当前中国 AI 编程助手尚处于发展阶段，各个厂商在产品发展进度上发力点有所不同，商业化模式并未形成共识，面对挑战与难题，各方需持续精进，以实现从技术理论到商业价值进而完成营收转换。

5，**商汤日日新•代码小浣熊在 AI 编程助手评估框架的打分下表现优异，整体得分高出市场平均分，体现出领先的发展优势。**商汤代码小浣熊拥有业界领先的大语言模型作为其模型基础，保证了自身产品功能的实现效果和成长空间。通过早起布局，商汤代码小浣熊提供了完善的产品工程体验，满足不同规格客户的多元需求。基于与客户的互动，及时了解客户需求，捕捉技术发展趋势，商汤代码小浣熊保持着不断创新的节奏和功能发布。最后，商汤代码小浣熊构建了较为成熟的生态体系并取得了较高量级的企业客户和开发者用户。

6，**企业组织应该立即行动起来，采用 AI 编程助手，发展新质生产力，参考最佳实践和案例的同时，识别风险和“陷阱”。**企业 CIO 和软件开发团队的领导应该及时了解前瞻性企业部署 AI 编程助手的实践心得和取得的实际效果，同时也要为部署新兴的生成式 AI 应用带来的潜在风险和“陷阱”做好预案。“完美是优秀的敌人”，面对新技术和新工具，领导型企业不要畏惧，应该大胆行动，不断调优自身组织的数字化韧性，做好迎接 AGI 时代的来临。

第一章 AI 编程助手是开发者应掌握的新质生产力工具

新质生产力主要由技术革命性突破催生而成，科技创新能催生新产业、新模式、新动能，是发展新质生产力的核心要素[1]。新质生产力依靠科技创新驱动，通常表现为更高的效率、更好的质量、更强的创新能力和更符合高质量发展要求的生产方式，它以高科技、高效能、高质量为特征，是对传统生产方式的革新[2]。

新质生产力的显著特点之一就是技术创新，当下以大模型、生成式 AI 应用为代表的新一代信息技术体系正在成为新质生产力的重要引擎之一，正成为当今世界发展的最大变量，成为推动新一轮产业变革、促进全球经济增长的核心动力引擎。加快形成新质生产力，需要依托并充分运用好这些技术。相比内容生成、营销、拟人等新兴生成式 AI 应用，从用户量级和场景落地发展来看，AI 编程助手已成为产品化和应用落地速度较快的生成式 AI 技术，大模型的不断进化所带来的积极影响，也能够及时反映到 AI 编程助手产品迭代进化中，这从各厂商高速迭代 AI 编程助手产品的频次可见一斑。

生产工具的科技属性强弱是辨别新质生产力和传统生产力的显著标志。AI 编程助手能够进一步解放开发者，削弱了客观条件对软件开发活动的限制，拓展生产空间，为软件开发领域的新质生产力发展提供了工具基础，推动软件开发跃上新台阶。AI 编程助手帮助企业催生新业务，推动现有业务的深度转型升级，促进质的有效提升和量的合理增长，进而推动生产力全面提升和经济社会高质量发展。一项研究显示，AI 编程助手对于未来经济的影响巨大，到 2030 年，保守预计 AI 编程助手的生产力效益将为全球 GDP 增长贡献超 1.5 万亿美元[3]。另外一组数据更值得关注，因 AI 编程助手带来的生产力提升，到 2030 年，相当于为全球增加 1,500 万“有效开发者”，这些新增“开发者”将会创造、贡献海量社会价值，促进生产力发展。

1．在 AI 编程助手的赋能下，企业可以实现更全面的高质量发展

数字化时代，开发者作为技术的创造者和推动者，其价值不言而喻。企业的数字化转型与发展，都离不开开发者的主导和贡献，他们的代码和软件应用、移动应用，对这个社会做出了无法比拟的贡献。依据 GitHub 数据，2021 年中国有 755 万开发者，排名全球第二；放全球开发者数量超过 7300 万，比 2020 年增长了 1700 万[4]。伴随着企业数字化转型的不断升级，企业在构建软件方面遇到了各种调整和困难：

- **开发者人才的数量依然不够，尤其是中国这种高速发展的数字化市场。**相对于全球的开发者数量和增长速度，中国的开发者人才缺口在扩大，形势也更加严峻。这背后的原因是中国企业越来越多地依靠软件来实现差异化、推动转型以及在行业中取得领先地位。但是，对于大多数中国企业而言，快速踏上软件驱动的转型之旅往往面临各种困难。其中一个关键短板，就是开发者人才不足，尤其是能够理解领域知识并资深的开发者更加短缺。尤其是近年来生成式 AI 应用项目虽然暴增，但是来自中国的生成式 AI 应用项目数量却远远落后于美国、印度、日本等国家与地区[5]。如何提高软件开发人才的密度，是当下进一步推动新质生产力发展的有效措施之一。
- **软件需求超出了大多数企业组织的能力，现有开发者已不堪重负。**开发者不仅无法快速构建软件以迎合业务高速变化，也无法在工作中获得幸福感。招募更多开发者很困难，培养和提升开发者的技能也很困难。开发者永远是最忙碌的一群人。开发者没有足够的“专注时间”来处理重要的业务问题[6]。数据显示，软件开发人员每周大约有 19.6 个小时的专注时间，而每周要花费 10.9 个小时开会，而专注时间与工作效率有直接相关性[7]。
- **即使在开发人员可以专注于编写代码的有限时间内，他们也会遇到各种各样的障碍。**除了要花费很大一部分时间来手动编写简单且大同小异的代码之外，他们还要花费大量时间了解、学习和跟上复杂且不断变化的工具和技术发展趋势，这样才能使用最新的技术

和服务来构建创新的解决方案。这进一步压缩了可用于开发创新功能、差异化功能或关键业务功能的时间。而企业招募的大量的新入职的开发者，还要花费大量的时间在了解企业代码知识库，阅读海量文档等落地繁冗事务上。

AI 编程助手无疑为解决上述挑战提供了一个全新的高速进化的方案。AI 编码助手正在成为倍增器，提高开发人员的生产力和幸福感。通过处理日常任务，AI 编程助手使开发者能够专注于更高价值的活动。这使得企业组织能够利用现有团队更快地交付更多功能，支持更多业务。Gartner 调研显示，18%的企业机构已部署了 AI 代码助手，另有 25%的企业机构处于部署阶段，20%的企业机构处于试点阶段，14%的企业机构处于规划阶段[8]。

2. AI 编程助手遵循 KRE 理论架构不断升级满足生产力发展需求

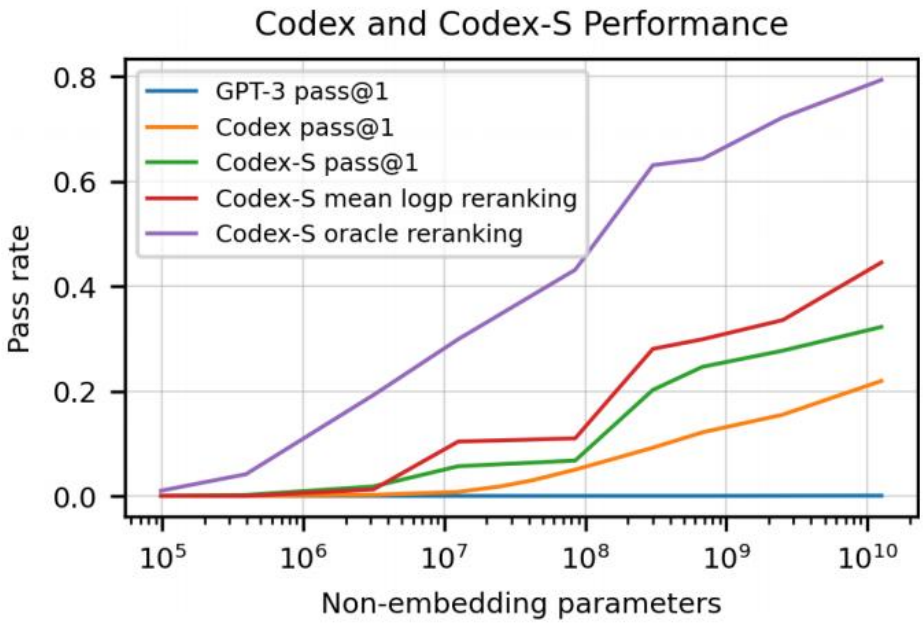
在过去几十年间，市场上一直存在不同形式的代码辅助工具和解决方案，但直到 2022 年底 ChatGPT 的发布，AI 编程助手作为一个有实际发展意义的概念才脱颖而出，取得了突破性的成功，斩获了技术厂商的巨大关注度。在此之前，历史上涌现的由不同厂商推出的大量产品工具则为我们今天拥有复杂编程工具铺平了道路[9]。这些早期的 AI 驱动的编码工具帮助塑造了软件开发的演变，并为当今人工智能驱动的编码辅助和自动化工具奠定了基础。

大模型使代码生成工具的发展迎来新趋势。2021 年 7 月，OpenAI 推出基于 GPT-3 的 AI 编程助手——Codex，其基础是大语言模型，并针对代码库进行了训练和微调。这意味着 Codex 不仅是通用的大语言模型，它还被特别训练来理解和生成代码。一项研究显示，相比未经精炼的 GPT-3 模型，Codex 在代码生成准确性和相关性方面取得了重大改进**（见图 1）**[10]。

从 GPT-3 到 Codex 的进化，对于生成式 AI 在软件开发中的应用具有重要意义，表明大语言模型不仅能生成自然语言，还能够生成高质量的、可执行的代码。2021 年 8 月，Codex

被集成到 GitHub Copilot 中，为开发者提供了一个实用的 AI 编程助手，正式开启了此领域的全球竞赛，后续 GitHub Copilot 不断迭代增加全新功能，满足开发者和企业的需求[11]。

图 1：HumanEval 评测集显示经过代码数据微调的 Codex 通过率远高于 GPT-3 大语言模型



大语言模型的进步正在加速 AI 编程助手的能力进化，并催生出 AI 编程助手的定义：

AI 编程助手是协助开发者编写和分析软件代码的智能化工具，其本质是基于基础模型，如大语言模型，再通过特定的高质量代码数据进行微调，提供自然语言交互界面，生成应用程序代码，实现从设计到代码的转换，并增强代码测试能力。AI 编程助手赋能软件开发生命周期各环节，内嵌集成开发环境等工具中，可根据用户需求提供定制化的解决方案。

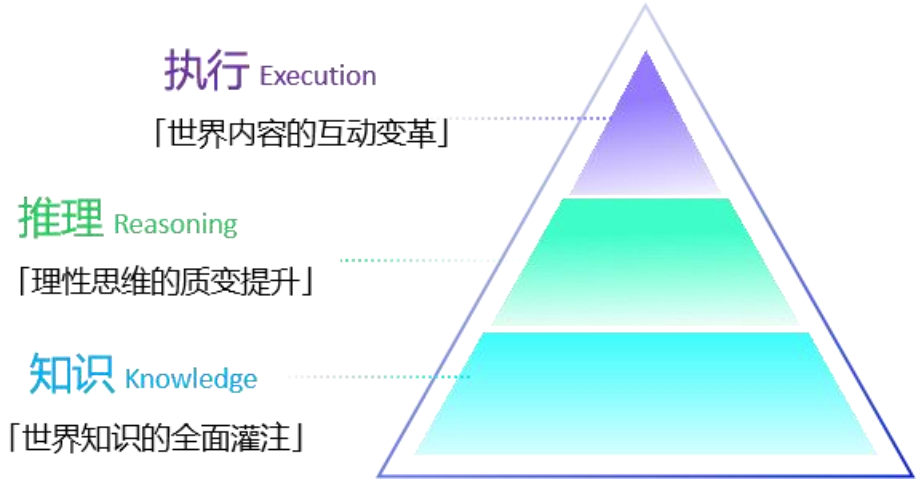
大模型能力可分为三层架构：知识（Knowledge）、推理（Reasoning）、执行（Execution）。这三层架构之间互有依赖，但又相对独立，总体来讲，这三层可以组成一个完备的对于世界提供生产力工具模型的三层能力**（见图 2）**：

- **第一层知识（Knowledge），世界知识的全面灌注。**目前，许多生产力工具解决的都是

知识层的问题，当用户提出问题时，其底层的逻辑都来自于“世上无新事”——你所面临的问题，前人可能已经遇到过并解决了，因此通过大模型通过前期的学习形成记忆在进行检索，就可以很好地完成基础性的任务。对于 AI 编程助手而言，这一层就是开源的代码数据集、合规的代码知识库，当然也包括企业自身的代码积累。

- **第二层推理（Reasoning），理性思维的质变提升。**有了世界知识之后，再往前演进，则是处理更加有逻辑和领域专业性的问题。即使不知道这件事实，也可通过思维链逐步把这个事实推理出来，给出更多的可能性。在这一层，AI 编程助手则是从简单的代码补全和生成，扩展覆盖到软件开发的全生命周期。
- **第三层执行（Execution），世界内容的互动变革，即如何跟这个世界互动反馈。**无论是前期大模型的函数调用（Function Calling）能力，还是处于高速动态发展的 AI 智能体（AI Agent），再或是为大模型装上“手和脚”的具身智能都是在执行层面实现突破的重要功能。对于 AI 编程助手而言，其“Copilot”的定位，本身就是聚焦在如何与开发者进行互动，执行开发者的要求，完成开发者下达的任务，切分到垂直场景中，以场景化智能为手段对业务产生全面赋能。

图 2：大模型能力的构建来自三个方面：知识、推理、执行



第二章 借助 AI 编程助手进入“心流”状态，成为 10 倍开发者

Gartner 调研显示，CIO 和技术领导者对 AI 编程助手及其对开发者生产力的提升效果抱有很高期望。近半数的人期望 AI 编程助手能提高生产力，超过三分之一的人认为该技术是潜在的“颠覆性因素”[12]。部署和扩展 AI 编程助手是一个复杂的技术决策，同时也是一个具有前瞻性的企业文化调整。对 AI 编程助手带来的价值的认识要有全貌，避免狭隘的视角，超越传统的技术 ROI 衡量标准，只有这样才能为 AI 编程助手捕捉到完整的企业价值故事，管理好企业内部利益相关者的预期，并获取这种技术投资的全部好处。

相对于其它生成式 AI 应用依然处于探索最佳应用场景的初级阶段而言，AI 编程助手属于最早成为有实际应用效果的生成式 AI 应用之一，也是少数让企业和用户感受到日常化影响的生产力工具。随着越来越多的企业认识到这些工具的价值并开始采用 AI 编程助手，我们可以预见到软件开发领域将继续经历变革。

AI 编程助手带来的收益在厂商的宣扬下已经“家喻户晓”，AI 编程助手的作用不仅在于提高写代码速度，其功能超出了代码生成和补全的范围。AI 编程助手可以充当助手，提高开发者的效率，帮助开发者保持专注状态，提高解决问题的能力。AI 编程助手的聊天互动功能还可以刺激头脑风暴，而其他功能则有助于重构代码、创建单元测试、调试、修复漏洞以及提高代码质量。此外，AI 编程助手还能够帮助开发者持续地提高编程语言及框架方面的技能和熟练度。

AI 编程助手节省写代码的时间仅仅是一个起点，不要止步于节省开发时间，还要看到 AI 编程助手可能减少任务切换，让开发者保持“心流”，改善开发者的工作体验，对人才的吸引和留存都有积极的意义。AI 编程助手还能够搭建企业内部统一的代码规范和定义，在企

业内部实现知识共享和标准化、提高代码质量和可维护性，安全增强、高效协同、降低企业的技术负债，这些都能够帮助开发团队将跟多的时间投入到更有价值的业务当中，或专注于高级别的设计和更复杂的创新工作。开发者能够利用 AI 编程助手生成的代码，快速实现功能，同时还能确保代码质量，从而加快产品从概念到市场的速度。这样的效率提升对于应对快速变化的技术环境和市场需求至关重要。

GitHub 的研究人员创建了用于对开发者生产力因素进行分类的 SPACE 框架**（见图 3），**包括满意度和幸福感、绩效、活跃度、沟通和协作、效率和心流五个方面[13]。该框架同样适用于去衡量和归纳 AI 编程助手给开发者带来的收益：

- **提高活跃度，执行更多的编程和测试等相关任务。**“活跃度”特指开发者在规定时间内做了多少事情，虽然关注仅仅是开发者的产出，但是单纯的计数依然关键。AI 编程助手能够帮助开发者在同样时间编写更多的代码，完成更多的测试，只有当开发者能够节省下来更多的时间，才能用来去创造价值。这个是 AI 编程助手带来的基础价值，也是衡量开发者生产力最直接有效的指标。
- **通过消除满意度和幸福感、沟通和协作的限制，维持高效率，并进入“心流”状态。**“满意度和幸福感”、“沟通和协作”、“效率与心流”这三个因素则与开发者的工作体验相关。满意度和幸福感不仅可以了解当下开发者的生产力水平，还可以预测未来的生产力发展，更好的识别开发者团队的倦态和生产力下降的趋势。软件开发是一项协作和创造性的任务，AI 编程助手可以有效促进和整合彼此的工作，盘活信息在团队内部和团队之间的流动性，集成代码文档和资料的可用性和发现性。AI 编程助手能够通过一系列自动化功能帮助开发者尽快进入“心流”状态，而不被轻易打断。
- **为产出好“绩效”奠定扎实的基础，并给予更具期待性的赋能和加持。**开发者的绩效考核一直很难量化，因为很难将个人贡献将产品成果直接挂钩，代码与营收之间依然存在

较大的不可量化的鸿沟。AI 编程助手正在逐渐帮助企业看清楚两者之间的关系，包括提高代码质量，加快产品上市、开发成本整体降低等等。而在提高生产力并提升工作体验之后，开发者将会有更多的时间投入到更高价值的业务上，让整体的工作价值聚焦到业务结果上，而非流程产出上。当然，目前 AI 编程助手的产品成熟度还处于前期阶段，厂商和企业用户都刚刚尝试用它来加强赋能业务的创新和实验，未来 AI 编程助手将会更加直接有效的为开发者带来全新的改善绩效表现的能力和模式。

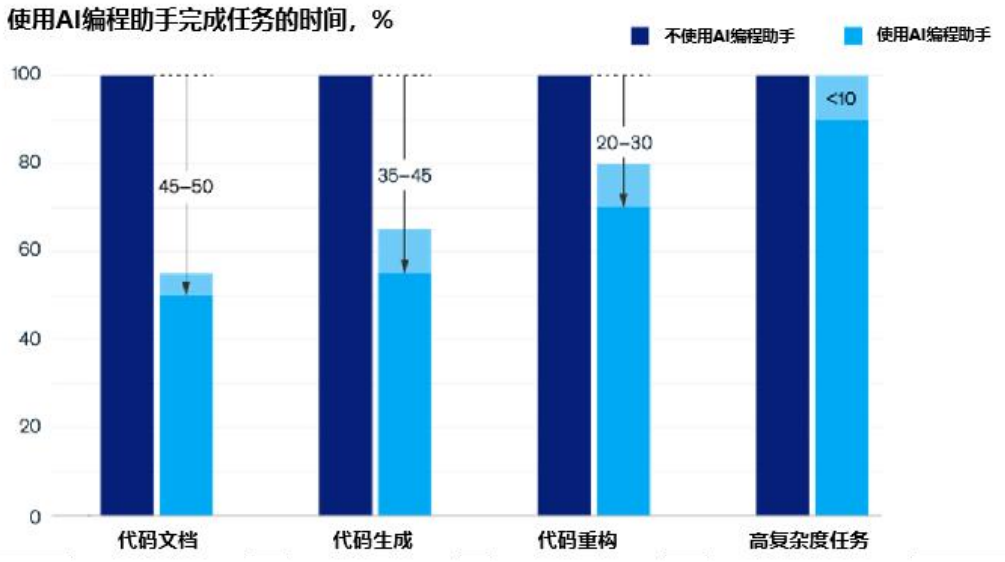
图 3：开发者生产力的构成：SPACE（来源：GitHub，商汤智研院整理）

1. 聚沙成塔，AI 编程助手有效提高开发者的生产力

一项研究显示，通过对照实验，使用 AI 编程助手的开发者组别完成任务的效率比未使用的组别要快 55.8%，表明 AI 编程助手显著提升了生产力，且经验较少的开发者反而从 AI 编程助手中获益更多，这对未来技能培训和职业转型都将会产生深远的影响[14]。在大模型、生成式 AI 趋势的驱动下，AI 编程助手等产品正在迅速获得业界的关注和采纳，其影响和普及可能会超出最初的预期，不仅为开发者提供了全新的智能化生产力工具，其对全球经济也在产生重大影响。

企业 CIO 和软件开发团队的领导者表达了利用 AI 编程助手提高开发人员生产力的浓厚兴趣。在使用 AI 编程助手提高开发人员生产力主要体现在提高开发者完成工作任务的“活跃度”**（见图 3）**，在此阶段，AI 编程助手可以显著减少开发人员完成这些任务所需的时间。McKinsey Digital 调研显示，AI 编程助手可以给予开发者令人印象深刻的效率提升，对于代码文档、代码生成等工作能够缩短 50%左右的时间，而对于代码优化和重构则可以缩短 30%的时间，这些速度的提升可以转化为生产力的提高，这种提高将超越以往由新工具和流程驱动的工程生产力的进步**（见图 4）**。

图 4：使用 AI 编程助手与不使用完成任务的时间对比



围绕软件开发生命周期，AI 编程助手通过启用多个场景用例来提高开发者整体生产力，正是这些细微的进步，汇聚成开发者生产力提升的动力**（见图 5）**。随着 AI 编程助手产品技术的不断进化，对开发者的赋能也将会呈现越来越深入，覆盖越来越广的趋势。对于企业用户而言，应该逐步关注每个用例特点，展开 POC 试点，来评估 AI 编程助手对开发者生产力的影响。

图 5：AI 编程助手的场景用例，正在逐步覆盖软件开发全生命周期的各个环节

为了充分利用 AI 编程助手的优势，应该将其应用于整个软件开发生命周期的所有阶段，而不局限于代码生成。这种做法不仅能够防止生产力瓶颈，还能利用创造价值的机会，结合优秀的 DevSecOps 实践，如使用 AI 辅助代码审查和自动化测试，以及应用现代化，提升软件开发的效率和质量。

首先，在软件程序的创建阶段，AI 编程助手可以帮助开发者减少时间投入，通过多个场景帮助开发者减轻编写代码的压力：

- **代码自动补全。**开发者使用 AI 编程助手的自动功能来提高编程速度。当他们输入时，AI 编程助手会帮助他们发现可用的代码，只需按一下键，就能帮助他们完成一行代码。AI 编程助手不仅能查看鼠标上方的代码和注释，还能扫描上下文，这使得它能够预测更加复杂额代码结果。另外，由于基模型的代码数据集训练，导致 AI 编程助手擅长从开源代码或企业代码库找出以前见过的模式，预测模板和重复代码。AI 编程助手将开发者从这些琐碎的工作中解放出来，使他们能够专注于更有挑战更有创意的方面。AI 编程助手还可以根据企业代码和文档对基模型进行提示词调整或微调，进一步提高代码建议的相关性和准确性，并帮助调用开发者不知道的企业代码知识库。AI 编程助手会

跟 IDE 工具无缝集成，不改变开发者习惯，实现实时代码补全，无需二次加工。

- **代码生成。**在大语言模型的支持下，AI 编程助手中的聊天互动界面可以让开发者通过自然语言来做提示词，从而探索和生成大段新代码，甚至是整个程序。AI 编程助手将这些信息与代码自动完成功能结合起来。这样，开发者就可以使用自由形式的自然语言来生成代码，并动态使用代码补全界面。让自然语言成为一种编程语言，成倍提高开发人员的工作效率，让更多非开发者身份的员工，如业务人员，具备开发软件的基础能力，更好的从开发者角度提出业务诉求，更好的弥补用户需求和开发设计之间的理解差距。
- **代码理解。**AI 编程助手支持开发者将代码片段粘贴到聊天界面，并获得自然语言解释，还可以在 IDE 环境中突出显示需要解释的代码片段。这些功能可以帮助开发者理解复杂和陌生的代码，甚至是陌生编程语言的代码。AI 编程助手还支持对完整程序生成代码图，进一步帮助开发者理解软件程序的结果和语义，保证产出的代码符合企业设计原则，这背后离不开基模型能力的加持。
- **代码翻译。**AI 编程助手的一个比较新兴和有用的用例场景就是将代码从一种编程语言翻译成另外一种编程语言，也就是代码转译功能。该功能可以帮助精通一种语言的开发者更快的熟悉其他编程语言进而提升工作效率，同时也有助力开发者重写程序。代码翻译功能还可以与代码生成和理解相结合，进而实现代码现代化。

其次，AI 编程助手可以提高代码质量。GitHub 研究显示 AI 编程助手对代码质量和可维护性都产生了积极效果[15]。高质量代码会提高生产力，研究显示将会使软件开发时间平均减少了 78%[16]。高质量代码还是企业最关键的业务目标之一，34 %的软件领导者认为提高软件质量是他们的三大绩效目标之一[17]。AI 编程助手在提高代码质量领域能做的：

- **代码调试。**开发者调试代码错误的时候会陷入困境，不得不依靠其他资深开发者的帮助或从互联网上搜索解决方法。AI 编程助手的基模型基本都在标记代码数据上训练过，

包括历史错误代码，这就使其能够帮助开发者以对话的形式进行代码调试，快速定位和解决问题，提升代码的整体质量和稳定性。特别是在软件开发的早期阶段，及时修复错误可以避免后续的连锁反应。

- **文档生成。**创建和更新文档是开发者工作中必不可少的一部分，也是最繁琐和最缺乏创新的工作。AI 编程助手可以帮助开发者分析代码并创建相关文档，包括代码注释、API 规范和嵌入注释等。这样的话，开发者只需做好审核工作，保证 AI 编程助手生成的文档的准确性即可，可以节省大量的时间成本。而更清晰、更一致的在线文档则能够帮助开发者构建高质量代码。
- **拉取请求汇总。**开发者在合并代码的时候会发出合并或拉取请求，以便让同行在提交分支代码之前进行审查更改。自动生成的拉取请求可以节省时间，提高沟通和审查效率，并增强对代码变更的推理能力。AI 编程助手可以记录开发者的工作并分析其差异化，创建拉取请求汇总的叙述性文本，并提供有关更新的上下文和更新推理。代码审核人员可以根据这些上下文记录，更好的理解代码变更的目的和影响。通过自动生成代码差异化总结，AI 编程助手减少了手动编写文档的工作，简化了审核流程。
- **单元测试用例生成。**开发者可以通过自然语言提示词让 AI 编程助手生成单元测试。众所周知，单元测试对于提高软件程序的可靠性至关重要，但生成单元测试却十分繁琐。因此，开发人员编写和维护单元测试的量级很低，而使用 AI 编程助手的开发者能够比以前编写更多的测试用例，极大提高了单元测试环节效率，保证软件程序的有效运行。
- **代码重构。**代码重构是指在不改变软件功能的前提下，对代码进行结构上的调整，以提高代码的可读性、可维护性、可扩展性和性能。虽然 IDE 工具可以为开发者提供主流编程语言的内置重构辅助功能，但是 AI 编程助手则更进一步，能够主动提出建议给出选择，让重构变得更加容易，甚至支持不具备重构功能的编辑器。AI 编程助手还能够让

开发者在多个程序中重构大部分的代码。厂商可以将代码图与基模型的推理能力相结合，以实现更大规模的自动化重构。

- **代码现代化。**随着 AI 编程助手的进化，一些更加复杂的功能被厂商推出，如理解跨多个程序的复杂依赖架构，该功能可以帮助企业减少“技术负债（Technical Debt）”，实现代码现代化（Code Modernization）。厂商正在推出新的用户界面，使程序依赖关系可视化，支持代码模式的搜索和变更影响分析等功能。该功能尤其对企业内部一些用老旧编程语言或版本编写的软件系统，且少有开发者了解或者积极去维护的独立系统有立竿见影的效果。对于遗留系统的现代化，AI 编程助手可以辅助代码的逆向工程，提取业务逻辑，发现领域模型，从而降低现代化过程中的成本和风险；还可以分析旧代码库，理解其结构和功能，然后提供重构建议，帮助开发者构建更加模块化、可扩展和易于维护的系统。这不仅提高了代码质量，还为未来的技术创新打下了坚实的基础。

最后，AI 编程助手还可以帮助开发者提升自身技能，并降低任务启动时面临的困难度，这些都是不容忽视的对开发者生产力有巨大影响的场景用例：

- **学习新的语言和框架。**当开发者想要学习一种新的语言或框架时，他可能会求助于多种渠道，在线示例和教程、书籍和开发人员论坛（如 Stack Overflow、CSDN、知乎），也包括企业内部的开发相关的知识库。AI 编程助手借助底层的基模型，其包含了从这些来源获得的大量相同信息，并且可以通过提供示例、生成问题和提出修复建议来提供一种教授新概念的有效方法。事实证明，与 AI 编程助手进行聊天交互是学习和研究新概念的有效方式。当然，开发者需要保持警惕，毕竟大模型的幻觉问题依然存在于 AI 编程助手的输出内容当中。
- **快速熟悉陌生的代码库、编程语言或框架，以便完成任务。**万事开头难，特别是在面对新的开发挑战时，开发者可以通过利用这类工具来获得与向经验丰富的同事请教相当的

支持——比如解释新的概念、综合不同来源代码的信息（比如通过比较和对比来自不同存储库的代码）、以及提供如何逐步使用某个框架的指南等。在这种情况下，开发者期待得到 AI 编程助手的有价值输入，帮助他们克服面对“空白画布”时的恐惧感，使得开发者能够摆脱启动障碍，从而更快地开始编码工作。

- **降低新技能和专业知识的学习曲线，使开发者能够更快地掌握新技术。**这对于保持开发者团队在技术进步和行业趋势方面的领先地位至关重要。举例来说，在技能获取阶段。AI 编程助手不仅可以加速代码编写过程，还可以作为一个强大的学习工具，帮助开发者学习新的软件开发技能。通过 AI 辅助的教程、代码解释和调试工具，开发者可以更快地掌握新技术和编程语言。AI 编程助手可以提供个性化的学习路径，根据开发者的现有技能和学习进度推荐合适的资源。此外，还可以通过分析大量的代码库来提供最佳实践和设计模式的建议，从而提高开发者的技术能力和生产力。
- **加速产品上市时间，让企业在竞争中保持领先地位。**在当今竞争激烈的世界，企业将新构想推进至生产就绪状态的速度至关重要。即使工程性能仅仅提高 1%，也可能成为决定成败的关键因素。例如，企业交付一款面向客户移动应用，假设该 APP 第一年可产生 1000 万的利润。因此，将交付日期提前一个月可产生的潜在价值应接近 100 万（具体取决于对增速的假设）。此外，在竞争非常激烈的环境中，产品提前几个月上市会极大地提高一家公司的竞争地位。

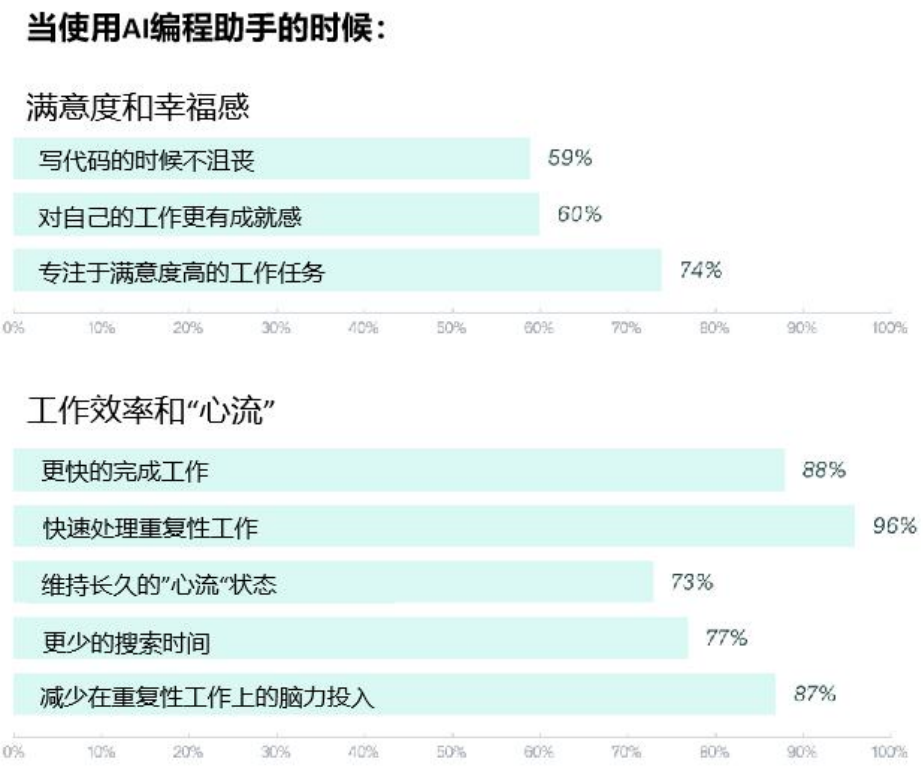
AI 编程助手还可以在软件开发的其他环节发挥作用。例如，在需求分析阶段，AI 可以通过自然语言处理技术来帮助理解用户需求，自动生成用户故事和验收标准。在部署和维护阶段，AI 可以监控应用性能，预测潜在的故障，并自动执行修复操作。总之，AI 编程助手的发展预示着软件工程领域的一个重大转变，企业 CIO 和软件开发领导者们需要认识到这一转变，并准备将其整合到自身的工作流程中，以充分利用这些技术的潜力，从而在整个软件

开发生命周期中实现生产力和效率的整体提升。

2. 超越生产力，AI 编程助手还可以提高开发者的工作体验

开发者从 AI 编程助手获得了满足感和幸福感，因为 AI 编程助手减轻或消除了开发工作中基础性的、重复性任务导致的痛苦，这对于企业留住并激发其最优秀的开发人才的积极性具有重要作用。GitHub 研究显示，超过 60%使用 AI 编程助手的开发者表示，满意度和幸福感都有所提升**（见图 6）**[18]。开发者将 AI 编程助手视为提高生产力的生成式 AI 工具，但它不仅限于此，更进一步而言，AI 编程助手减轻了开发者的认知负担。这种轻松感，会让开发者更有动力和精力去承担更具有挑战性的工作，进而获得更高级的成就感。

图 6：通过使用 AI 编程助手，开发者可以获得幸福感和满意度



首先，AI 编程助手能够直接减少开发者的重复性工作，诸如编写常见代码、企业代码库的拥有的代码、或者创建日常文档任务。一项研究显示，开发者最倾向于使用 AI 编程助

手的原因之一就在于，它们能帮助开发者减少击键次数，快速完成编程任务，并回忆起语法结构。”通过这种方式，AI 编程助手技术间接地促进了开发者的整体满意度[19]。

其次，AI 编程助手还可以加强开发者之间的协同效力，通过更快速和更清晰的沟通，开发人员可以更好地理解用户的意图并交付更有价值的软件。相比较于一些传统的开发赋能工具，AI 编程助手不仅仅是减少无效的沟通和协作，而是让现有的沟通和协作更加有效。AI 编程助手通过自然语言提示交互，正在逐步成为接入众多工程系统的接口，能够让开发者们在同一开发环境下工作，并向系统请求协助完成开发任务，例如在写代码的过程中请求生成测试数据的帮助。另外，AI 编程助手有助于改善书面沟通，其内置的语法工具可以将口头对话转换为文本并进行总结。还可以让翻译更快速、更准确的进行，跨国团队成员合作的时候会减少沟通错误。AI 编程助手甚至可以通过只言片语就理解开发者想要描述的一个复杂概念，进而将其转换成完整描述，减轻开发者在沟通过程中投入的精力。

同时，AI 编程助手可以更好的做好开发者之间的信息分享。软件开发的许多方面的知识沉淀，包括设计模式、安全指南、API、语言语法和各类文档，是企业通用的资产，当然还包括一系列的历史代码知识库和撰写代码的指导原则等等。这类沉淀的资产和知识，需要从企业的资深开发者传递给初级开发者或者新入职的开发者，而 AI 编程助手通过模型微调、提示工程、知识库对接等功能，助力这种知识转移。

然后，AI 编程助手可以减轻开发者日常工作中的认知疲劳（Cognitive Fatigue），帮助开发者进入并维持在心流状态。认知疲劳对决策产生强大的负面影响，从而影响生产力、质量和创新[20]。认知疲劳的一个原因是上下文切换，当开发者被打断或因当前任务受阻而转换到其它任务时就会发生该状况，开发者的大脑需要重新调整注意力，加载新的工作内容和相关记忆，这一过程消耗大量的认知资源，而频繁的切换则加重了疲劳程度。开发者经常会面

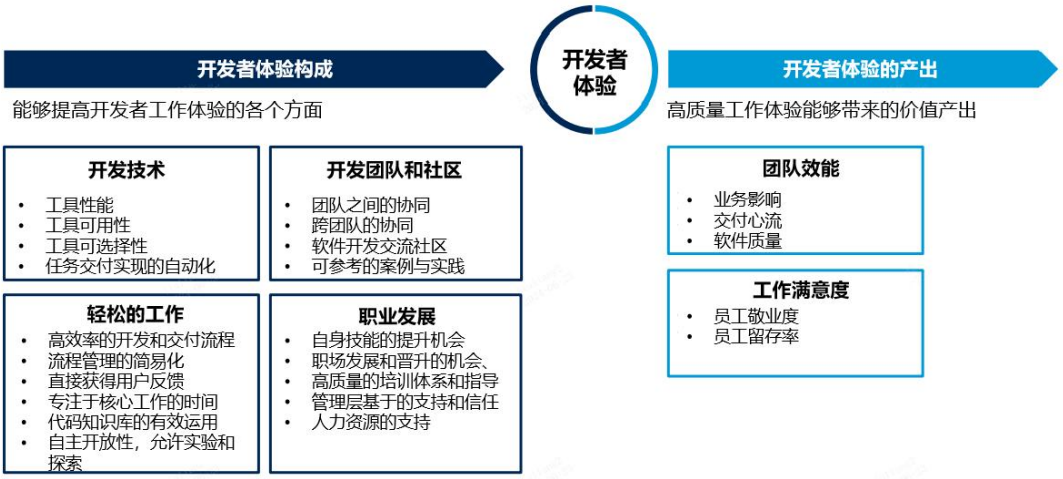
对这样的局面，这对一个处于高压下的开发者而言是非常让人苦恼的。AI 编程助手可以帮助开发者减少过多的任务切换和中断，在它们的支持下，开发者可以保持在单一的开发环境中，并提示系统帮助完成开发任务。例如，开发人员可以在编写代码的同时在相同的环境中获得帮助创建测试数据，或者进行相关的对话和查询。避免频繁切换和任务中断可以显著降低开发者的认知疲劳，提高他们的效率和流畅度。

研究显示，不需要切换上下文的开发者描述自己会更容易进入心流状态，在这种状态下他们的生产力极高[21]。开发者全神贯注于手头的任务，注意力高度集中，思维流畅，创新能力和工作效率得以最大化。

最后，AI 编程助手让开发者拥有选择推进工作方式的机会，而非一人闷头苦想。在确定解决问题或执行任务的具体步骤时也会感到很大压力，因为这项活动本质上是开放式的，充满了未知与不确定性。开发者在评估不同行动方案、尝试权衡利弊、思考影响和短期聚焦（马上解决这个问题，快刀斩乱麻）与长期视角（现在投入更多精力以产出更持久解决方案，一劳永逸）之间的相对重要性时，也同样会承受巨大压力。AI 编程助手可以通过自然语言交互，让开发者在开发环境中输入提示来寻求建议，AI 编程助手一般都可以为开发者提供多个候选计划，并对每个计划的优缺点进行论证，还可以帮助开发者跳过最初的头脑风暴环节，直接从候选选项中评估选择，或者将其作为进一步头脑风暴的提示。

虽然工作体验的改善是一个相对比较复杂的的感受和经历，但是其可以让生产力获得成倍提升。这种乘数效应意味着 AI 编程助手的整体影响可能远超最初的预期**（见图 7）**。因此，前瞻性企业正在谋求通过使用 AI 编程助手来直接改善开发者团队的体验，进一步增加生产力优势。

图 7：Developer Experience Assessment Framework



3. 保持理性，穿越炒作：从 AI 编程助手工具的采用中获得最大价值

企业对于 AI 编程助手为开发者带来的生产力提升和工作体验改善充满期待，这促使企业迅速采用这些工具。但是，这并不意味着企业在部署 AI 编程助手以及在未来扩展部署的时候，都能看到其能够带来如厂商宣传那样的积极影响。围绕生成式 AI 的炒作往往会让公司领导层对部署 AI 编程助手可能实现的目标有不切实际的期望，为此，企业要有一个行之有效的方法，让 AI 编程助手带来的价值能够满足企业利益相关者的诉求。开发者的经验水平、开放性文化、工程熟练度、交付压力和领导期望都会影响他们如何尝试使用 AI 编程助手以及他们所能够获得的价值。另外，许多组织在衡量 AI 代码助手的真实影响时面临挑战，因为缺乏基准指标、明确目标和预期结果（如提高质量或速度）。

要想从 AI 编程助手获得更大的业务价值，需要将它们应用在正确的地方和面向正确的任务上，还要控制不切实际的期望和误解，具体措施如下：

- **确保企业已经拥有项目成功部署的先决条件。**首先，AI 编程助手在执行其预训练数据里出现过的任务效果最佳，应先将其用于使用常见编程语言的领域，如 Python、Java、

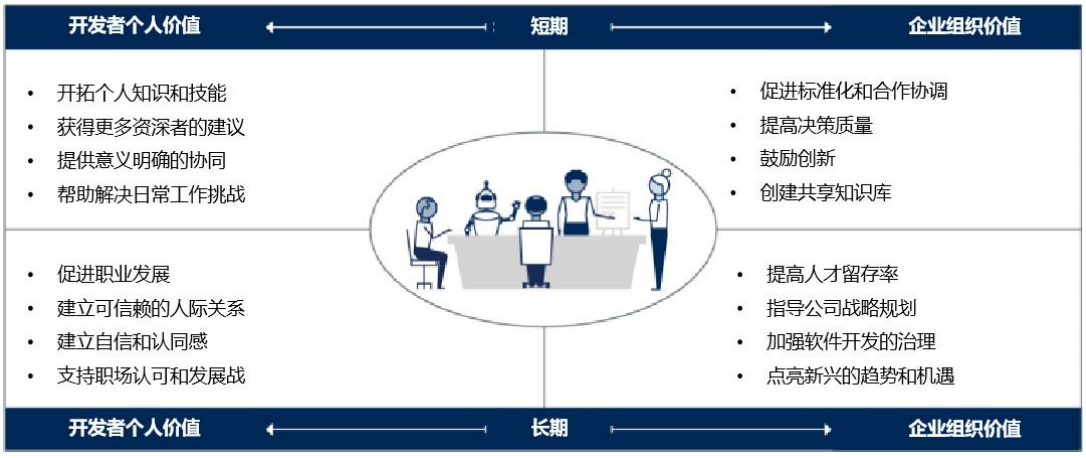
C++、Go 等。其次，企业内部要已经运行了一个良好的软件开发流程，包括代码审查、质量检查等，保证 AI 编程助手的输出可用性，避免因为新工具影响正常的开发流程。然后，将 AI 变成助手用于相对见效快的领域，如编写基础代码、测试环节或者创建文档，从而加快实现收益的速度。最后，将 AI 编程助手优先提供给资深开发团队使用。

- **定义企业部署 AI 编程助手能够带来的预期效果，并使其指标化。**在向更广泛的开发者推出之前，通过 POC 先在一个小型团队里使用进而确定行之有效符合自身组织的方法论。根据经验，将 POC 的目标定位成节省开发者时间、提高生产力是好的开始。为此，企业要为之定义好团队的生产力指标以及如何衡量它，这样才能观察和记录下来 AI 编程助手带来的影响。参考 SPACE 框架来定义全面指标，每天/每周开发者节省时间，业务交付周期的改善、使用者满意度反馈、以及软件质量提升等等，避免依赖单一指标。除此之外，周期性对使用者进行定性调研，了解使用后的对比感受、满意度、以及改进建议。除了为了证明 AI 编程助手的实际价值以外，POC 的目标还包括获取有关收益和风险的实证数据，确定使用者需要的技能，以及总结成功经验作为使用指南。
- **为收益指标建立一个基准，才能有的放矢的衡量 AI 编程助手带来的收益指标。**开发者生产力的量化是复杂的、困难的，每个维度的量化数据很难捕捉准确，所有的应用情况都有所不同，没有两个应用情况所体现的商业价值会一致。通过建立基准可以看清 AI 编程助手带来的收益。在确定好衡量指标之后，再基于这些指标去建立基准。如果当前企业内部已经有一个敏捷管理类似的工具，则可以将其作为比较基准。还可以将企业内另外一个开发团队的表现作为基准，实行对照实验，并通过内部调研获取定性参考值。
- **管理企业管理层的预期。**生成式 AI 的炒作依然在持续，不仅仅是公共媒体的持续报道，还有厂商基于最佳案例的营销，让企业管理层对 AI 编程助手的收益产生了一些不切实际的期望。当然，这对于一家组织去拥抱创新技术的转变来说是一个好事儿。但是，作

为执行部署 AI 编程助手的团队来说则要冷静判断、给出预期。“没有任何一项技术或管理方法的进展，能够独立许诺十年内使生产力、可靠性或简洁性获得数量级上的进步[22]。”该判断依然适用于生成式 AI 时代，至少目前来看是这样的。所以 CIO 或软件开发领导者在部署 AI 编程助手时，要向企业管理层明确收益预期，市场上的客户案例或最佳实践都是最理想的情况，不能生搬硬套到自身企业上。

- **建立一个供企业内部开发者们交流 AI 编程助手使用心得的社区。**社区是让企业内部的开发者们随时交流、分享、咨询、反馈使用 AI 编程助手的信息、经验、问题的平台，相当于建立了一个相互学习的机制。还可以定期收集开发者们提出的建议和想法，反馈给 AI 编程助手厂商，使产品更加完善。虽然是企业内部社区，但依然可以参考通用社区的建设方针，梳理清晰长期和短期价值，并明确员工和企业的不同角度**（见图 8）**[23]。

图 8：企业内部 AI 编程助手社区能够带来的价值（来源：Gartner，商汤智研院整理）



第三章　AI 编程助手让开发者从执行者变成了代码协调者

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.（我们最重要的目标，是通过持续不断地及早交付有价值的软件使客户满意。）

Manifesto for Agile Software Development, Principle No.1[24]

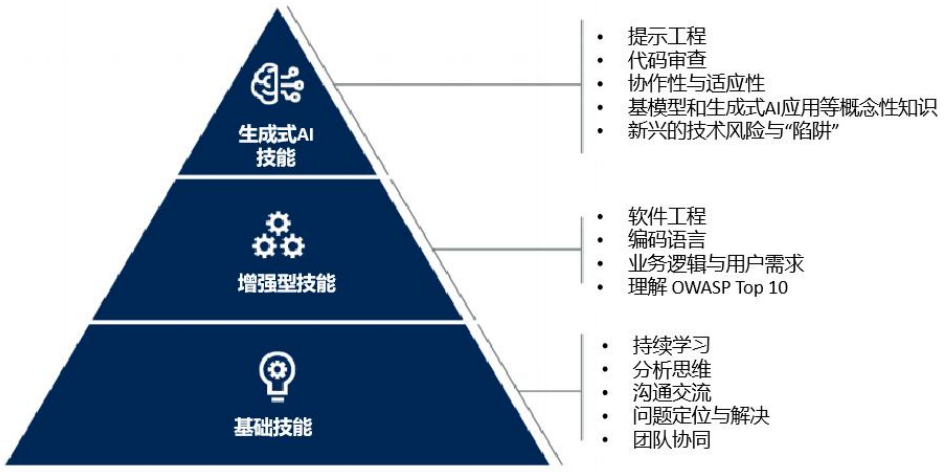
AI 编程助手显著降低了软件开发的入门门槛，这一变革不仅为新手开发者提供了快速上手的机会，越来越多的“年轻”开发者将会进入市场，也为经验丰富的专业人士提供了提高工作效率的可能。另外，由于软件开发变得更加容易，那些原本缺乏技术背景的个人和小型团队现在也有能力开发和推出自己的软件产品。这种趋势可能会加剧市场竞争，迫使现有的企业在创新速度和利润率方面面临更大的压力。

为了提高效率和创新能力，以保持竞争优势，企业会主动投资于 AI 编程助手，不久将来，AI 编程助手将成为开发者日常工作的标配。开发者不再仅仅是编写代码的执行者，而是变成了协调者和指挥者。开发者不仅要保持开放心态去迎接这个变化，更要利用这个变化去完成自身身份的华丽转变。

1．保持开放心态，提升自身能力，在变革中寻找成长与机遇

AI 编程助手对开发者的工作起到了补充作用，同时为他们提供了更多专注于编程和培养技能的机会。当开发者在日常工作中开始使用 AI 编程助手，相关的技能要求也在不断增加。这也就是意味着，开发者除了要具备一些基础性技能以外，还要培养应学习的增强型技能，同时开拓能力的生成式 AI 技能**（见图 9）**。Gartner 数据显示，超过 52%的 IT 领导者预计所在企业机构将使用 AI 编程助手。然而，60%的 IT 领导者表示其团队并不具备所需的技能[25]。帮助团队掌握所需的技能，也是企业 CIO 和软件开发领导者当前的重要任务。

图 9：面对生成式 AI 趋势，开发者要不断精进现有技能，并学习新的技能



生成式 AI 技能是其中的关键，因为这类技能无论对于初级开发者还是资深开发者，都是全新的领域，也是急需补充的技能。这些技能也在不断地进化，而且也不断地出现新的技能要求，开发者需要关注这些技能的发展，当前，值得关注的有三个方面。

第一方面，需要掌握提示工程能力。大语言模型的出现带来全新的技能要求之一，就是如何写好提升词，与模型进行更好的自然语言交互，进而拿到优异的反馈结果，减轻幻觉。因此提示工程也成为一个全新的学科。AI 编程助手在首次交流之后输出内容质量不高，但可以通过高质量的提示词，不断修正输出的信息，且随着“调教”越来越体系化，输出质量和风格也更符合企业要求。提示工程并不是简单的向 AI 编程助手进行问话，需要包括精心的提示或指令设计，如思维链，以确保其产生高质量输出。另外，提示词要明确且具体，能够给大模型一定的上下文，并根据输出内容迭代提示词，让输出质量更优异。

第二个方面，开发者需要强化代码审查能力。AI 编程助手输出的代码依然存在瑕疵，甚至错误。开发者需要依靠他们的编码能力和结构化的知识来确保不会将错误的代码投入生产，开发者需要具备扎实的基础知识。在这种场景下，AI 编程助手成为了开发者的“实习生”，开发者不仅要让它输出代码，还要审查代码，学会如何审核 AI 编程助手输出的代码会让编

程事半功倍。审核 AI 编程助手产出的代码，不仅要看其基本的愈发错误，事实性幻觉，还包括执行、逻辑问题、数据隐私、性能表现、可用性、伦理道德等一系列角度[26]。

第三方面，开发者的协作性与适应性变得更加重要。随着开发者开始利用 AI 编程助手来进行软件开发等工作，拥有与同事良好沟通和协作等技能，变得比以往更加重要。开发者一直通过例会、结对编程、与产品经理了解客户需求等一系列方式进行沟通，那现在同样要将 AI 编程助手作为助力拉入到日常沟通中，如何定义项目需求、分享使用 AI 编程助手的经验都是沟通的关键点。同时，AI 编程助手产品快速进化，开发者要具有适应能力并愿意接受新的工具、方法和新的工作模式，同时是自身的能力与生成式 AI 技术的发展同频共振。

除此之外，AI 编程助手本身也可以助力开发者学习新技能，因为其对代码的解释、重构以及不同编程语言之间的代码转译，对开发者来说都是一个可以用于学习新技能的功能。Gartner 预测，到 2025 年，75%的开发者将使用 AI 编程助手学习新的软件开发技能，而 2023 年这一比例为 30%[27]。AI 编程助手的很多功能还可以弥补传统开发者的技能缺陷，帮开发者消除面临的不断提高跨学科技能的压力。举例来说，越来越多的开发者开始自主学习提高安全技能，而 AI 编程助手可以为开发者提供指导性的修正帮助，以解决代码中的安全漏洞。

2. 当好协调者与指挥者，开发者自身角色定位在发生转变

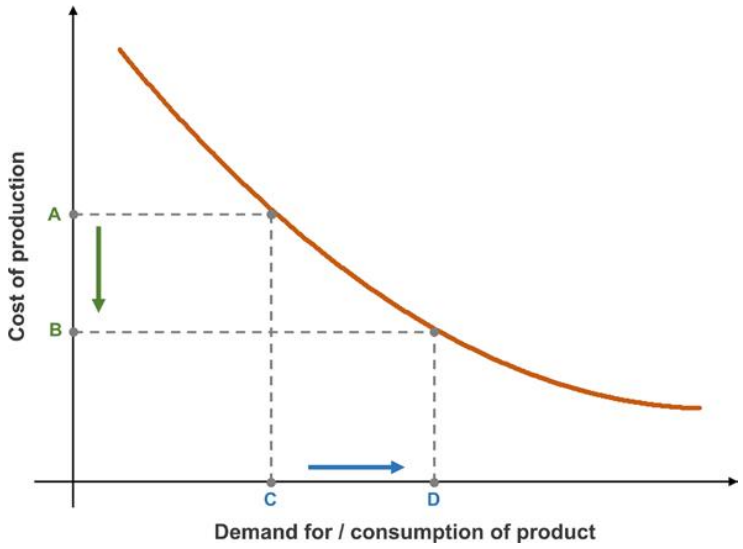
AI 编程助手并不会替代开发者。科学技术的本质就是降低成本提高生产力，AI 编程助手在降低软件开发门槛、完成一些基础性工作的当前，并不会替代开发者，相反，AI 编程助手正在逐渐改变开发者自身的价值定位，甚至是将会拔高开发者在社会发展中的叙事逻辑。

AI 编程助手带来了的这场变革成功与否或者发展是否如人所愿，都离不开开发者。AI 编程助手可以更快的写代码，但是无法离不开人类的监督，也不具备人类的业务创新能力。

对待 AI 编程助手输出的代码，就像对待论坛或互联网讨论中的建议一样。它向你展示了一个可能的方向，并侧重于你所询问的领域，但是你依然不能保证它是可放入生产环境中的，开发者有责任要负责它的安全性和适用性，需要开发者进行验证和确认。软件开发难度最高的环节不是写代码，而是需求理解和设计，以及用户体验和后续迭代。

面对新出现的任何提高生产力的工具，杰文斯悖论（Jevons Paradox）都显得尤为重要**（见图 10）**。悖论指出，当完成任务的成本降低时，人们对这些任务的需求就会增加。在微观经济学中，成本下降导致需求量成比例增长，从而在效率提高的同时，也增加了消费量。软件开发成本的下降会带来更多的数字化转型需求，将会有更多的业务领域需要全新的软件应用去支撑，不仅仅是传统的大型软件，也包括互联网应用，甚至生成式 AI 应用。

图 10：杰文斯悖论示意图：成本下降反而导致需求的爆发

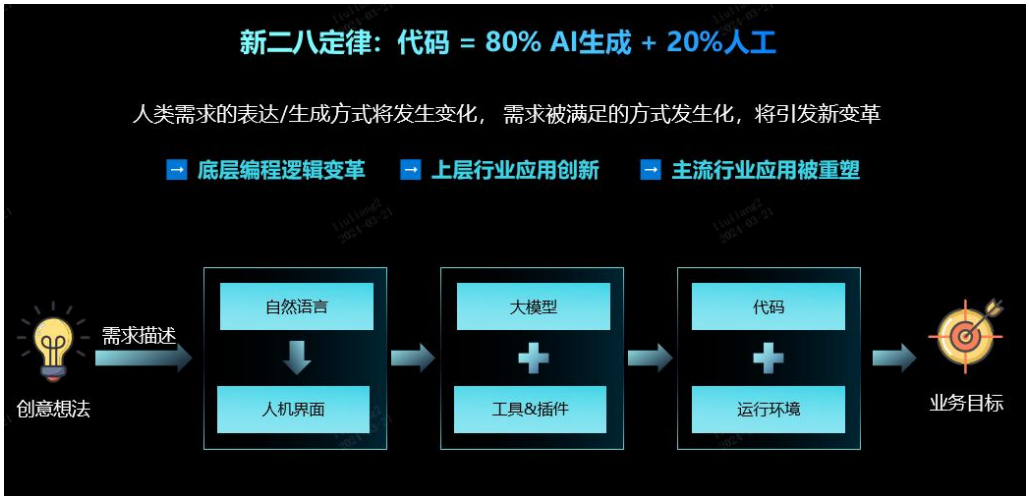


与此同时，软件开发行业存在二八定律，大概有 80%的代码量都需要开发者自行编写；在生成式 AI 时代，AI 编程助手将为软件开发行业带来“新二八定律”，即 80%的工作由 AI 完成，20%的工作才由开发者来做**（见图 11）**。AI 编程助手对于软件的底层编程逻辑将会不断进行冲击，进而使得上层的软件应用的创新模式发生质变。AI 编程助手让开发者的工作

重心发生了转移，AI 编程助手承担部分基础工作，而开发者则转向更高层次的架构设计和业务规划，专注于更复杂、更创造性的任务，如架构设计、算法优化或用户体验创新，此时的开发者更像是一场复杂任务的编排负责人，而 AI 编程助手是帮助他完成这场编排的智囊。

更进一步，开发者可能逐渐成为“系统思考者（Systems Thinkers）”，专注于描述解决问题和把控整个软件开发生命周期的运转，指导并监督 AI 工具产生的成果，并在必要时进行干预，而非深陷到具体的代码撰写任务当中。系统性思维，一直是优秀开发者必备的优秀品质，但通常被认为是具备足够经验且用于丰富开发历练的开发者才拥有的。随着软件开发的机械化工作由开发者转移到 AI 编程工具中，系统性思维将成为开发者在其职业生涯早期就能锻炼的技能，加速其成长过程。

图 11：AI 编程助手为软件开发带来“新二八定律”



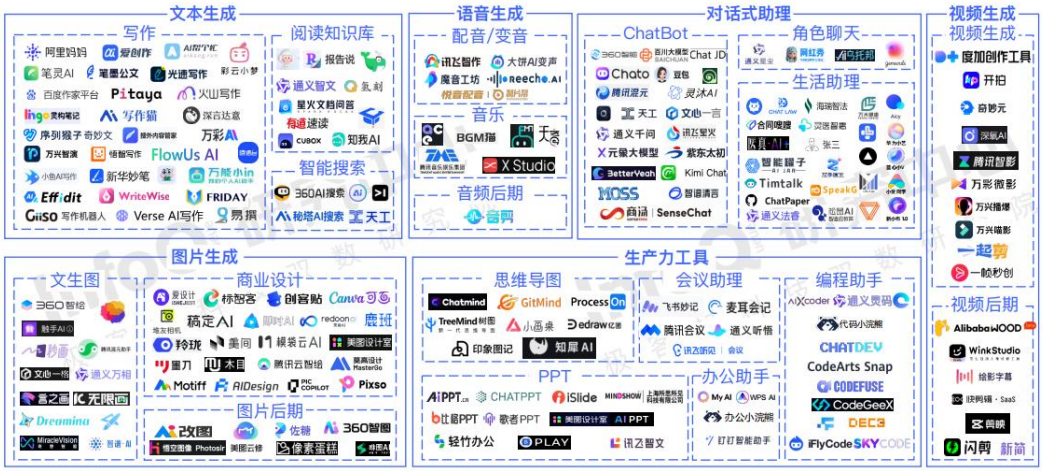
最后，企业 CIO 和软件开发领导者也要配合。开发者们希望使用 AI 编程助手，使用尖端技术对开发者很有吸引力。Gartner 调研显示，留住优秀人才和吸引高素质应聘者是企业最难完成的任务[28]。采用 AI 编程助手可以增强开发人员体验，并帮助组织吸引和留住人才。

第四章　群雄并起，中国大模型厂商逐鹿 AI 编程助手时代

在国际上，OpenAI 与 GitHub 的合作推动了 AI 驱动的编程辅助工具的跃迁式发展，受到了海量开发者群体的青睐。GitHub Copilot 通过不断提供新的功能去满足开发者的需求，逐渐由单一的辅助功能发展成一个面向开发环境的 AI 编程助手解决方案。截止 2024 年 5 月，GitHub Copilot 的付费订阅用户超过 180 万[29]。

在中国，基于大模型的整体生成式 AI 应用生态发展迅猛，大模型厂商均推出相关应用产品，涉及文本生成、语音生成、对话式助理、视频生成、生产力工具等领域，超过 200 款应用产品已经面世**（见图 12）**。其中，编程助手则是竞争更加激烈的领域，中国大模型厂商相继进入该市场，推出基于自身大语言模型或开源大语言模型微调研发的编程助手产品，希望将市场需求货币化，开拓全新的业务市场。厂商之间的竞争推动着快速创新，加速了产品能力提升，还推动了市场采用率。InfoQ 研究中心的评测数据显示，大模型产品因在开发者广泛使用，编程能力提升迅速，在其 2024 年的大模型评测终，编程题平均得分 87%，较上次测试得分率提升 49.45%；同时，开发者对大模型的整体认可程度高，付费率高达 63.5%[30]。

图 12：大模型应用产品开始规模化涌现，超 200 款产品面世（来源：InforQ 研究中心）



1. 中国 AI 编程助手市场发展趋势与格局

用户对于生成式 AI 工具的积极使用态度和需求也是中国 AI 编程助手市场百花齐放的重要推动原因之一**（见表 1）**。AI 编程助手市场格局竞争依然处于持续竞争的阶段，大量厂商进入该市场提供相关产品，既包括传统云厂商，如阿里云、百度云，也包括 AI 原生厂商，如商汤科技、智谱 AI 等。大部分情况下，这些厂商都会基于自身的大语言模型对外提供 AI 编程助手产品。麦肯锡调研显示，大中华区成为使用生成式 AI 技术的用户增长最快的区域，主要得益于技术环境、原生数字人口等优势[31]。基于大语言模型的 AI 编程助手的发展时间还不足两年，但是中国厂商的市场动态正在迅速演变，市场呈现以下几个现状与趋势：

- **各厂商推出的 AI 编程助手基本都支持主流编程语言，同时兼容大部分主流编程工具。**绝大部分厂商的 AI 编程助手都支持了当前主流的编程语言。同时，为了更好的让开发者无缝使用 AI 编程助手，对于大部分的 IDEs 编程工具都做了兼容，让开发者在不离开代码编写环境的前提下就能调用 AI 编程助手，并且适配相关 IDE 的原生主题、交互模式，保证体验的一致性。全面迎合开发者的使用习惯，避免开发者跳出原本的编程环境，符合开发者的日常编程习惯。
- **各厂商均提供灵活的收费模式，满足从个人到企业的使用需求。**当前普遍采取的是向个人开发者群体提供免费服务，面向企业客户则提供更高级别的版本并采取收费模式，多元化的服务模式可以满足开发者、创业团队、企业组织不同群体的需求。用户可以根据自身的业务场景选择最合适的商业化版本，随着版本收费的升级，其所包含的服务也更加全面，如增加更丰富的管理模式、交付模式，以及更加高级别的服务支持等。商汤代码小浣熊除了提供免费的“个人免费版”，服务于独立开发者以外，在收费版本上还提供了“个人升级版”、“团队版”、“企业版”等选择。
- **从代码生成到测试用例再到聊天交互功能，AI 编程助手的产品功能在快速完善中。**围

绕软件开发生命周期，厂商提供的 AI 编程助手所覆盖的功能模块越来越完善，不仅包括基础性的代码生成与补全、代码翻译、重构与纠错，还会提供单元测试用例的生成。一些领先的 AI 编程助手，为了更好的提升开发者日常工作体验，还会提供问答聊天功能，基于海量研发文档、产品文档、通用研发知识进行问答训练，为开发者答疑解惑。领先的 AI 编程助手还会提供知识库对接功能，如果企业客户有特定的编程规范和代码撰写方式，可以讲这些知识沉淀成知识库，与 AI 编程助手产品进行功能性结合，使其输出的代码符合内部流程标准，提升代码质量。

- **积极提供安全协议，对于客户关心的数据安全问题企业有明确的承诺机制。**代码安全依然是用户关心的关键热点。AI 编程助手厂商明确承诺不会存储和分析用户的代码数据，也不会将用户的代码数据用于 AI 编程助手未来的训练和微调，相关代码片段也不会成为其他用户的建议代码，也不会共享给其他用户。对于训练数据，也会明确是基于自身企业的代码以及开源代码，避免用户陷入可能的版权风险。同时也赋予了用户相关权利来处理自己的数据，用户可以自由的访问、更正、管理和删除相关代码数据，拥有完全的控制权。中国厂商在数据安全、风险隐私方面已经在积极布局，相关的隐私政策和保护措施也在不断完善和出台中。

表 1：中国 AI 编程助手代表性厂商

厂商	AI 编程助手产品	简介
阿里巴巴	通义灵码	基于通义大模型的智能编码辅助工具，提供行级/函数级实时续写、自然语言生成代码、单元测试生成、代码优化、注释生成、代码解释、研发智能问

		答、异常报错排查等能力，针对阿里云服务使用场景调优，助力开发者高效、流畅的编码。
百度	Baidu Comate	基于文心大模型，结合百度积累多年的编程现场大数据和外部优秀开源数据，生成更符合实际研发场景的优质代码，提升编码效率。
非十科技	Fitten Code	非十大模型驱动的 AI 编程助手，它可以自动生成代码，提升开发效率，调试 Bug，节省开发时间，对话聊天，解决编程碰到的问题。
京东	JoyCoder	基于大语言模型、适配多种主流 IDE 的智能编程助手，可以为研发人员提供代码预测续写、注释生成代码、一键生成单元测试、一键生成接口文档、AI 代码评审、代码解释、报错分析、智能问答等智能化编程功能。
华为	CodeArts Snap	基于华为云盘古研发大模型的智能编程助手，重塑智能化软件研发的新范式，基于智能生成、智能问答 2 大核心能力，覆盖代码生成、研发知识问答、单元测试用例生成、代码解释、代码注释、代码翻译、代码调试、代码检查等开发场景。
科大讯飞	iFlyCode	智能编码助手插件，可以在程序员编程过程中沉浸式交互生成代码建议，助力程序员提升编码效率和企业敏捷开发。

昆仑万维	天工智码 SkyCode	开源编程大模型，支持多种主流代码语言，智能高效补全代码，提升工作效率，节省开发时间，目前还在测试阶段。
蚂蚁集团	CodeFuse	基于蚂蚁集团自研的基础大模型，具备代码补全、添加注释、解释代码、生成单测，以及代码优化功能，以帮助开发者更快、更轻松地编写代码。
深度求索	DeepSeek Coder V2	开源的智能代码助手，帮助用户快速编写程序、修改用户界面、测试程序错误以及进行数据分析，学写 SQL 等。
商汤科技	代码小浣熊	基于商汤大语言模型的软件智能研发助手，覆盖软件需求分析、架构设计、代码编写、软件测试等环节，满足用户代码编写、编程学习等各类需求。在实际应用中，代码小浣熊可帮助开发者提升编程效率超 50%。
腾讯	AI 代码助手	基于混元代码大模型，提供技术对话、代码补全、代码诊断和优化等能力。为你生成优质代码，帮你解决技术难题，提升编码效率。
智谱 AI	CodeGeeX	基于智谱 AI 大模型的智能编程助手。它可以实现代码的生成与补全、自动添加注释、代码翻译及智能问答等功能，帮助开发者显著提高工作效率。

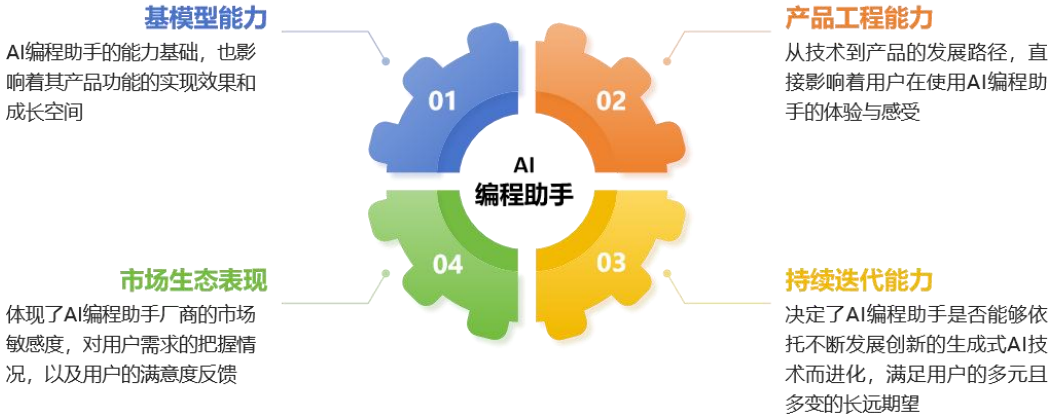
2. AI 编程助手评估框架：基模型、产品工程、持续迭代、市场生态

如今，企业 CIO 和软件开发领导者在 AI 编程助手方面面临着大量选择。厂商不断改进功能，使企业决定如何采用这些工具的能力变得更加复杂。各个厂商的 AI 编程助手产品本身也在不断进化中，基本是每 2-3 个月就会有一个较大版本的更新，这无非给企业 CIO 和软件开发团队的认知学习带来一定的压力。为此，通过本白皮书提出中国首个 AI 编程助手评估框架，帮助企业用户综合判断 AI 编程助手的产品能力和厂商实力**（见图 13）**。该评估框架包含五大维度指标，全面展现 AI 编程助手厂商全貌：

- **基础模型能力。**AI 编程助手的基础能力是来自底层的基础大模型，强劲且可靠的大模型基础能力依然是实现应用有效落地的技术基石。因此厂商是否具备自研大模型，且大模型能力具备领先性，是构建 AI 编程助手产品竞争力的关键，以及未来支持多模态大模型能力也是必要的评估因素。
- **产品工程能力。**基于大模型的基础能力，如何提供良好的产品体验，并能够与企业当前 IT 应用环境和业务场景实现良好兼容是 AI 编程助手成为企业必要选择的关键。如何赢得企业用户的信任，使得他们允许产品部署到生产环境，则进一步考验厂商能否提供灵活的交付模式、完善的安全风险保障、产品最终的交付和定价模式、产品的提供的版本和配套部署模式、以及从用户角度出发的产品设计理念。
- **持续迭代能力。**生成式 AI 依然属于新兴技术，如何保证自身产品能力能够处于竞争格局中的第一阵营，同时还能够在竞争环境中保证产品能力的不断升级，以及获得企业级客户的验证积累最佳实践是厂商能够脱颖而出、持续保证竞争力的关键。该指标重点评估厂商通过论文发布、功能领先性、产品规划路线图以及产品能定制化能力等二级指标。
- **市场生态表现。**这一指标反映了 AI 编程助手的市场接受程度和客户反馈。通过客户案例、行业覆盖、营收质量、市场响应以及行业专业知识的积累来体现厂商在市场中的认

可度，以及其在行业中的领导力和影响力。

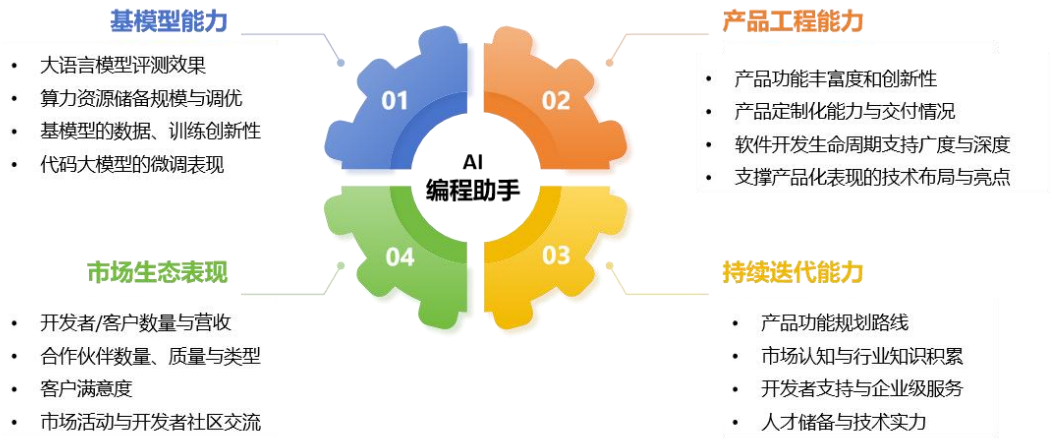
图 13：AI 编程助手评估框架：四大维度



AI 编程助手评估框架的重要性在于其可以为企业选择当前市场上最适合自己的 AI 编程助手产品提供了明确的方向和依据。基于该评估框架，企业可以更加全面的分析不同 AI 编程助手在基模型、产品工程、持续迭代、市场生态等方面的表现，从而选择最适配自身发展需求的供应商伙伴。与此同时，该评估框架也是开发者和企业用户能够系统化理解市场上 AI 编程助手厂商能力的有效工具，避免被厂商的宣传和媒体的渲染造成困惑和失焦。

为了更好的评估中国市场 AI 编程助手各个厂商的发展水平，本白皮书基于 AI 编程助手评估框架构建了一套有针对性的细分评估指标体系，该体系是基于评估框架的四大维度细化出 16 个二级指标，全方位覆盖了 AI 编程助手的关键领域，并紧密结合开发者和企业组织的需求特点，实现了精细化的指标拆解**（见图 14）**。整体而言，当前中国 AI 编程助手尚处于发展阶段，仅有部分大模型厂商实现了四大维度的全面布局，现阶段各个厂商在产品发展进度上发力点有所不同，商业化模式并未形成共识。各个厂商不仅要加强基模型能力的提升，还要打造深厚的产品工程能力，进而保持持续迭代能力的同时，应对瞬息万变的市场格局和技术迭代，以及用户需求的进化，以保持竞争力并实现市场价值最大化。

图 14：AI 编程助手评估框架：四大维度下的 16 个二级指标



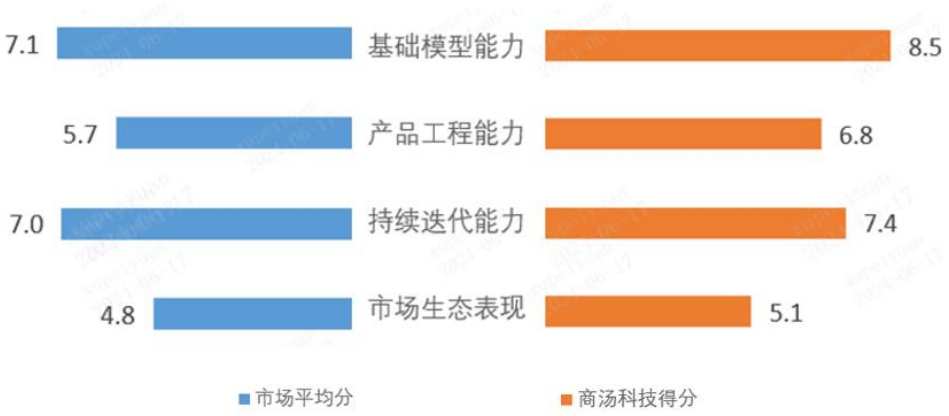
3. 谁塑造了这个市场？—— 商汤日日新·代码小浣熊

根据 AI 编程助手评估框架，得出中国 AI 编程助手市场四个维度的得分，进而得出市场平均得分。打分机制采用 1-10 分制，得分越高说明表现越好，最后再根据权重计算出每项维度的平均分。另外，针对商汤在 AI 编程助手市场的布局和能力进行了专门评估，从本次评估结果来看，商汤科技的日日新·代码小浣熊在 AI 编程助手市场具有一定优势，其评估结果整体领先市场平均水平**（见图 15）**。具体来看商汤在 AI 编程助手评估框架中的表现：

- **基模型能力，8.5 分。**商汤在 AI 编程助手的基模型能力维度具有较为突出的表现。这得益于其在大模型的前瞻布局和技术优势，为其在 AI 编程助手的布局奠定了基础。自去年 4 月首次发布，商汤日日新 SenseNova 大模型体系已正式推出五个大版本迭代。在尺度定律（Scaling Law）的指引下，商汤大模型不断经历重要的技术升级，每一次升级都稳居行业前沿，最新发布的商汤日日新 5.0 实现了强大的多模态感知理解和生成能力，并带来了出色的用户体验。强有力的基模型为代码小浣熊带来领先的生成能力，保证了代码输出内容的质量，同时也为后续的微调和产品化奠定了基础。

- **产品工程能力，6.8 分。**在产品工程能力维度，商汤取得了 6.8 分，稍微领先市场平均水平。作为新兴产品，AI 编程助手的产品工程依然是各个厂商在努力搭建和完善的组合能力，而客户对该产品的认知和要求也在不断地变化和升级。商汤的 AI 编程助手产品能力通过一系列基准测试获得领先结果，并在国家机构的评估中拿到高度认可。此外，产品体验和定制化能力也是商汤主要发力方向，为开发者的使用提供了全面的支持。
- **持续迭代能力，7.4 分。**商汤在持续迭代能力维度获得了 7.4 分，体现了其对新兴技术高速发展的把控力，无论是对于编程助手基本功能的覆盖，对软件开发生命周期的全阶段支持，还是对于代码解释器、收益评估指标等功能的开拓，都体现出商汤在 AI 编程助手领域拥有较为成熟的产品发展路线图，对客户需求也能够及时定位和识别。同时，对于开发者用户和企业用户提供不同的交付模式和版本型号，并对其进行深耕体现出差异化和定制化，提供强化的安全防护和完善的数据隐私保护措施。
- **市场生态表现，5.1 分。**商汤是国内最早推出 AI 编程助手产品的厂商之一，得分略高于市场平均值。商汤在 AI 编程助手的思想领导力和品牌推广力度上持续投入，展现了强劲的增长势头，并积极在不同行业进行布局，如金融、科技、汽车等行业。同时，积极通过 MaaS、定制化等模式开拓商业边界，扩大用户数量，提升解决方案的定制化支持，聚焦服务行业头部客户打造标杆效果，推动市场完成从技术到商业化落地的转变。

图 15：AI 编程助手评估结果与分析



4. 商汤日日新·代码小浣熊成为软件开发新时代的创新基石

代码小浣熊是基于商汤自研大语言模型的 AI 编程助手，提供集成开发环境插件形态的产品服务及 Web 端的代码对话服务。其不但具备卓越的代码生成能力，还拥有强大的语言理解能力，让开发者更好地通过自然语言交互，降低编程开发门槛，提升工作效率。代码小浣熊在 HumanEval 测试集中一次通过率高达 76.8%。在实际应用中，可帮助开发者提升编程效率提升最高达到 78%，代码采纳率超过 30%，单日人均生产 300+行代码。代码小浣熊还成为首批通过中国信通院可信 AI 代码大模型评估的产品，在通用能力、专用场景能力、应用成熟度三个维度 100 多个能力指标中表现优秀，获得当前最高评级**（见图 16）**[32]。截止 2024 年 6 月，商汤代码小浣熊已经拥有超过 20 万开发者用户，200 多个企业客户，每天 Token 数超 4 亿。具体来看，代码小浣熊拥有以下优势特点：

- **强大的基模型能力支撑代码小浣熊产品应用的有效落地。**基模型是保障 AI 编程助手有长远发展的关键条件。商汤自研的“日日新 5.0”大模型体系通过训练数据的全面升级与训练方法的有效提升，获得了出色的语言能力及多模态能力。为了突破数据层面的瓶颈，使用了超过 10T 的 tokens 确保了高质量数据的完整性。大规模采用思维型的合成数据（数千亿 Tokens 量级），极大提升了推理、计算工具使用方面的能力。在训练阶段，采用了自研的多阶段训练链路，包括三阶段预训练、双阶段监督微调和在线人类反馈强化学习。这一系列创新措施保障了商汤日日新 5.0 的领先能力，使其成为首个超越 GPT-4-Turbo 的中文大模型**（见图 17）**，并在中文主观综合能力上超过 GPT-4o[33]。借助大模型，代码小浣熊把代码理解、生成、优化等能力无缝集成到研发流程的各个环节，为开发者提供了一个 7*24 小时的 AI 助手。
- **面向企业级 BizDevOps 全过程，覆盖软件开发全生命周期，支持主流语言与 IDE。**代码小浣熊实现了软件开发生命周期全赋能，从需求分析、架构涉及、代码编写、软件测试

再到部署上线和维护等环节，助力开发者代码生成&补全、代码翻译&优化、代码重构、代码纠错、代码问答、代码翻译、代码添加注释、测试用例生成以及技术文档生成等多个应用场景，同时还满足用户数据分析、编程学习等各应用场景**（见图 18）**。借助日日新大模型体系最新的函数调用（function calling）能力，代码小浣熊与 IDE 深度集成，实现了大语言模型与 IDE 的互操作，为开发者带来了更顺滑的编程体验，支持 Visual Studio Code、IntelliJ IDEA、Android Studio 等主流 IDE；支持 Python、Java、JavaScript、C#、C、C++、Go、SQL 等 90+主流编程语言。

- **打造企业级功能，满足企业客户的高要求和针对性需求，实现企业代码资产有效融合。**企业部署 AI 编程助手不仅仅从开发者使用的角度出发，还会从组织层面关注是否符合业务规范和属性，从组织的管理和运营角度去考量 AI 编程助手的能力。代码小浣熊充分将大模型与 RAG 技术结合运用，使企业已有代码库、内部规范、过程数据等资产能够充分融合运用，进而将企业自身的代码管理规范，代码输出格式、文档输出的格式，产品文档、技术文档、企业代码仓库等现有知识与代码小浣熊打通，使生成代码更加贴合企业业务、更符合企业内部的管理要求。企业自有的代码和规范、自主设计的编程风格等，产品如何与之适配和调优，真正为企业所适用**（见图 19）**。代码小浣熊与企业内部的工程化流程逐步结合，实现资产沉淀、规范落地、知识共享，让相关的赋能经验和最佳实践能够被更多开发者复用。举例来说，代码小浣熊支持输入企业内部统一模板要求的产品需求文档，并与相关功能绑定，使得用户在与小浣熊进行交互生成的产品需求文档符合企业内部的格式和颗粒度，通过代码小浣熊约束生成的产品需求文档格式。其次，通过代码小浣熊，企业也可以更加便捷的将高价值的“内部知识库”让软件开发团队更加便捷的访问、学习理解、以及关联应用，让企业的初级开发者迅速成长，让开发者日常工作无需大量翻阅文档类资料，实现一键式编程。

- **提供端到端的数据安全与隐私保护解决方案，打消企业风险顾虑。**企业对于自身的代码安全一向看得很重，包括代码、文档资料都是企业的核心资产。代码小浣熊通过提供端到端的安全措施，保障数据安全与隐私保护。代码小浣熊通过用户身份验证、授权和权限管理确保只有授权人员能够访问敏感数据，并对敏感数据如身份信息、登录信息，保密数据如设备安装包或升级包、License 和密钥等，都使用可靠机制加密后存储。另外，代码小浣熊还提供传输加密、漏洞管理、审计和监控等保障措施。在数据隐私方面，小浣熊支持建立数据中心内所有运维和用户操作行为的日志记录制度，并使用审计系统进行管理和分析，及时发现和制止异常行为。对于数据进行分级处理，明确审批、存储和传输要求，确保敏感数据湖得到适当的保护和管理。对于个人信息的处理，符合个人信息保护相关法规，并对个人信息进行脱敏处理，以实现去标识化。
- **支持多样化部署和交付模式，实现国产化适配。**除了公有云 SaaS 的部署模式以外，代码小浣熊还提供了包括软硬一体机和私有化等更多类型的部署模式支持。SaaS 部署模式提供企业版、团队版、个人版等多个选项，供客户根据自身需求进行不同权益的选择。对于依然想追求公有云弹性，但是安全可控要求较高的企业，可以使用公有云专区，开辟独有的隔离的资源供客户使用。私有化部署则包括完整版和轻量版，前者针对的是自身有较强算力的企业客户，配置全套的代码小浣熊服务和参数量最大的基础模型；后者则针对的是自身算力不强的中小企业客户，保证模型能力的同时采取轻量化配置。代码小浣熊提供一体机的部署模式，企业客户无需专门采购硬件就能私有化部署 AI 编程助手，并且更能保护好企业的代码等核心资产，数据不出域安全有保障。针对大语言模型设计了一个高性能推理后端，该后端对显存进行了最细粒度的规划，极大的减少了推理时的显存碎片化，从而保证系统能够支持更高的并发，在同等硬件配置和同样参数的模型条件下，能够比友商支撑更多的用户使用，进一步提升企业部署的整体 ROI。提供基

于国产化芯片的一体机，实现全栈国产化，满足企业自主可控的部署要求。

- **提供原子指标体系和效能评测方法，供企业用户随时量化监控部署效果。**如何量化收益是企业采用 AI 编程助手后最关心的问题，尤其是拥有庞大软件开发团队且开发流程复杂的大型企业，也是 CIO 决策扩大 AI 编程助手应用规模的参考依据。但是，企业很难实现 AI 编程助手收益的量化监控，因为内部用户开发水平不一导致效果反馈无法拉齐相关书指标进行衡量。代码小浣熊提供了可量化的效能评估体系，帮助企业在使用产品过程中对开发环节、测试环节、交付环节的效能提升情况形成数据洞察**（见图 20）**。代码小浣熊自带指标埋点功能，通过量化代码采纳率、日活、使用时长、代码生成数量、人机交互频次、以及用户对生成的代码复制和收藏的次数等数据，形成指标跟踪，为企业提供系统性评估提供数据基础。还支持根据企业客户的需求进行定制对比功能，对比使用代码小浣熊和不使用代码小浣熊之间的差别，观测对比之下的效能和质量。

图 16：商汤小浣熊代码大模型获中国信通院代码大模型评估最高评级

CAICT 中国信通院

证书编号： 2024VK002920
报告编号： 24V01K000131-001

代码大模型检验证书

兹 证 明

上海商汤智能科技有限公司

单位地址：上海市徐汇区虹梅路 1900 号 6 楼 6-75、6-76、6-79 单元
产品名称：商汤小浣熊代码大模型
产品版本号：V4.0
符合标准： AIIA/PG 0110-2023
《智能化软件工程技术和应用要求 第 1 部分：代码大模型》（2023/V1.0）

经检验，贵单位的“商汤小浣熊代码大模型”，于中国泰尔实验室开展的首轮代码大模型能力评估中，获得具体评测结果如下：

4+级

【该评测是针对 2024 年 6 月 7 日前（产品型号：V4.0）该产品服务情况的披露，用户依然应保留在具体使用过程中可能的风险（非必要项）。因 AI 相关技术、产品不断升级迭代，代码大模型标准规范随行业发展不断更新等原因，当产品或评测标准发生较大变化时，建议该产品重新检验。】

中国泰尔实验室
中国信息通信研究院
授权签字人：
签发日期：
2024年06月07日

图 17：商汤日日新 5.0 在 SuperCLUE 评测中成为首个超越 GPT-4 Turbo 的国产大模型

SenseChat V5在SuperCLUE4月基准表现				
模型名称	机构	总分	使用	日期
SenseChat V5	商汤	80.03	API	2024年5月11日
GPT-4-Turbo-0125	OpenAI	79.13	API	2024年4月30日
GPT-4-Turbo-0409	OpenAI	77.02	API	2024年4月30日
GPT-4(官网)	OpenAI	75.32	网页	2024年4月30日
Claude3-Opus	Anthropic	74.47	API	2024年4月30日
Yi-Large	零一万物	74.29	API	2024年5月11日
Baichuan3	百川智能	73.32	API	2024年4月30日
GLM-4	智谱AI	72.58	API	2024年4月30日
通义千问2.1	阿里巴巴	72.45	API	2024年4月30日
腾讯Hunyuan-pro	腾讯	72.12	API	2024年4月30日
文心一言4.0	百度	71.90	API	2024年4月30日
MoonShot(Kimi)	月之暗面	70.42	网页	2024年4月30日
从容大模型V1.5	云从科技	70.35	API	2024年4月30日
MiniMax-abab6.1	稀宇科技	70.18	API	2024年4月30日
山海大模型	云知声	69.51	API	2024年4月30日
讯飞星火V3.5	科大讯飞	69.43	API	2024年4月30日
Llama-3-70B-it(poe)	Meta	68.77	网页	2024年4月30日
阶跃星辰step-1-32k	阶跃星辰	68.69	API	2024年4月30日

图 18：日日新·代码小浣熊覆盖软件开发生命周期的各个环节，定义全链条辅助功能

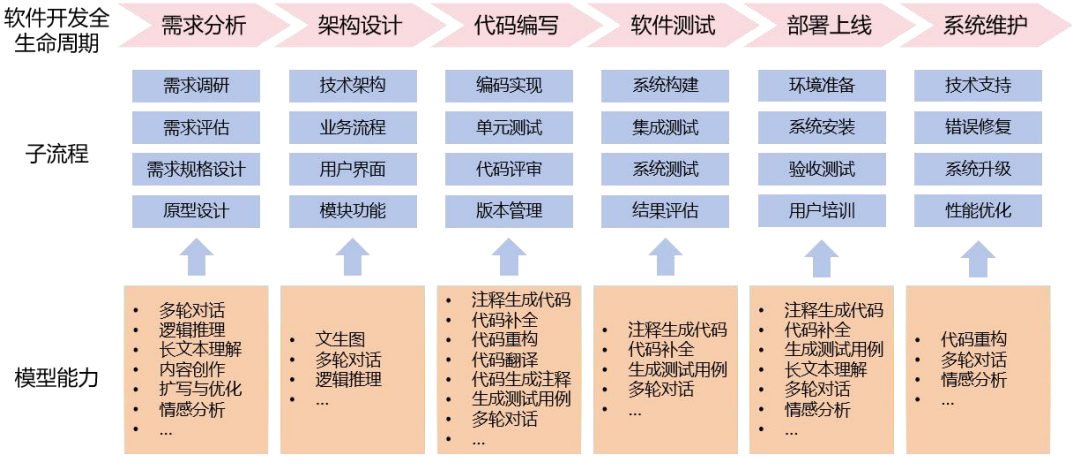


图 19：企业自有代码和规范资产的融合运用与调优

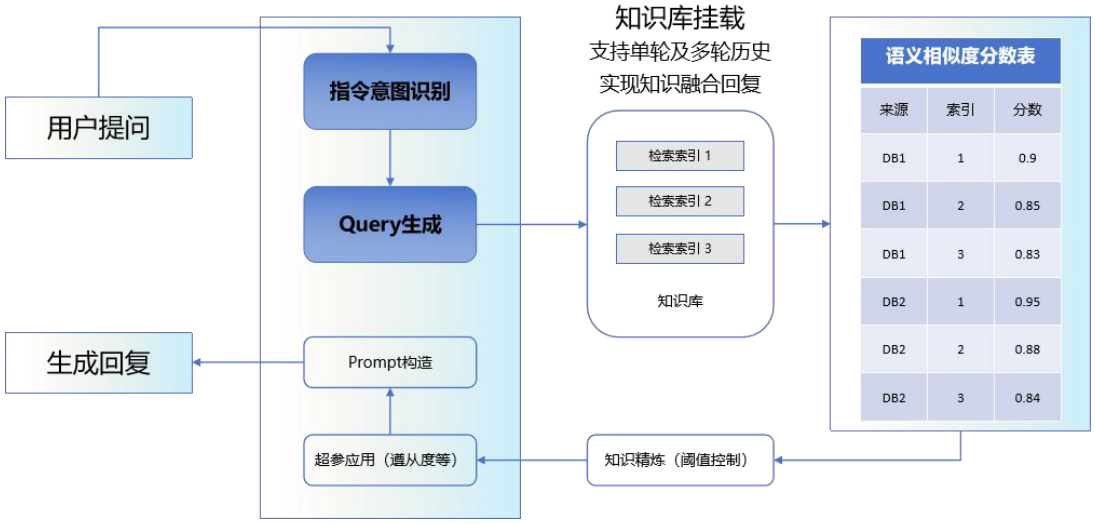


图 20：代码小浣熊收益量化评估方法典型示例

环节	子环节	指标	评估方法
开发环节	代码编写速度	完成同一功能代码所需时间对比	让两组开发人员（使用小浣熊和不使用小浣熊）分别完成相同或相近的任务，记录完成任务所需的时间。
		每小时内编写的代码行数	在任务周期，跟踪两组开发人员（使用小浣熊、不使用小浣熊）在平均每小时写的代码行数。
	代码质量	代码的复杂度比较	使用静态代码分析工具来测量每个任务的代码复杂度。
		模型生成有效代码占总代码比例	对于使用小浣熊的组，记录模型生成的代码行数与手工编写的代码行数。
		代码重复率的变化	使用代码重复检测工具来分析两组的代码。
	代码的维护性和可读性	代码审查时指出问题或改进数量	在代码审查过程中，记录由于小浣熊生成的代码导致的问题或建议的数量以及总的代码问题或建议的数量。
		代码的文档完整性和清晰度	让两组开发人员（使用小浣熊、不使用小浣熊）编写文档。请独立的评审团队对两组的代码文档进行评分，检查代码中的注释、字符串等相关内容的完整性和清晰度。
测试环节	缺陷发现率	测试阶段发现的缺陷数量	记录在测试阶段发现的缺陷数量。
		生产环境中发现的缺陷数量	记录在生产环境中发现的缺陷数量。
	测试覆盖率	代码测试覆盖率	让两组开发人员（使用小浣熊、不使用小浣熊）编写测试用例，再测量两组的测试覆盖率。
交付环节	交付速度	从代码完成到生产环境部署时间	记录代码从完成到部署到生产环境所需的时间。
	交付的稳定性	生产环境中因新代码导致故障率	跟踪因新代码导致的生产环境中的故障数量。

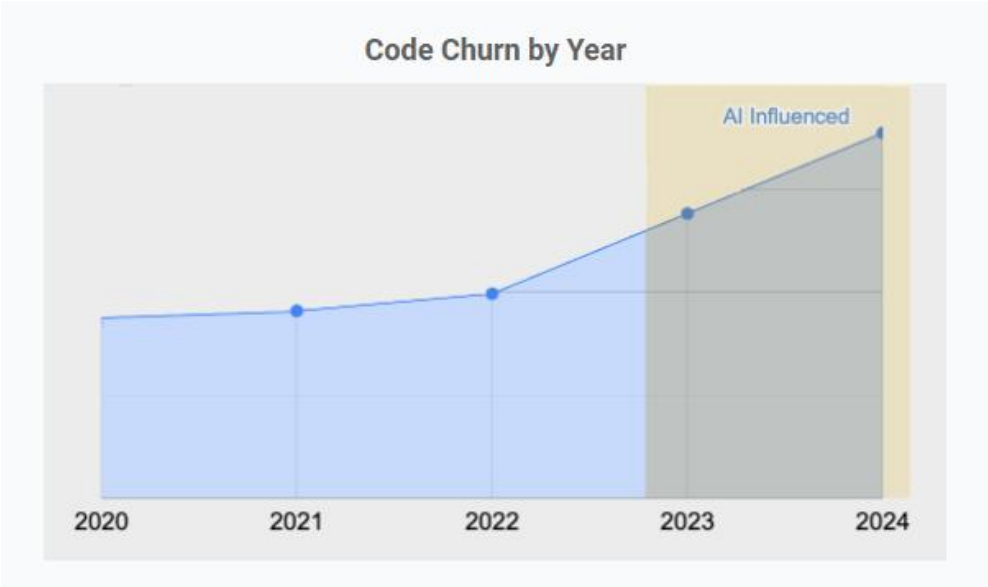
第五章　不可忽视的另一面：AI 编程助手的风险和“陷阱”

The Main Challenge with AI assisted programming is that it becomes so easy to generate a lot of code which shouldn’t have been written in the first place. (人工智能辅助编程的主要挑战在于它很容易生成大量本不该编写的代码)

Adam Tornhill, author of Your Code as Crime Scene

任何新生事物的兴起和发展，都是收益和挑战并存，AI 编程助手也在所难免。AI 编程助手所带来的影响，在当前阶段，依然有不同的看法和声音。有的观点认为生成式 AI 带来的主要是挑战是它很容易生成大量本来就不该编写的代码。还有的观点认为，更快地写出低质量代码，意味着后续阅读代码的人将面临更多困难，代码的阅读时间是编写时间的十倍[34]。GitClear 收集了 1.5 亿行代码变更记录，分析 AI 编程助手如何影响代码质量。分析发现代码变更率显著上升（指在编写后不到两周就被修改或撤销的代码行所占的比例），而代码复用率则不断衰退**（见图 21）**。从这个角度来看，AI 生成的代码更像是一位频繁更换工作的合同工写的临时代码，不断在违反 DRY（Don’t Repeat Yourself，不重复自己）原则[35]。

图 21：代码变更率——预计在 2024 年将是 2021 年 AI 生成代码出现之前的两倍



除此之外，还存在频繁收到增加代码的建议，却很少有关于更新、移动或删除代码的建议；评估代码建议可能耗费大量时间；代码建议的优化并非基于和代码维护者相同的激励机制等一系列挑战。企业不会拒绝生产力工具的迭代，上述这些使用过程中的问题势必会伴随 AI 编程助手功能的不断完善，以及开发者在使用成熟度上的不断提高而得到解决。但是，在产品发展完善之前，产业生态建设还不够成熟的当前，企业在采用 AI 编程助手时要充分考虑如下风险和"陷阱，为之做好应对措施**（见图 22）**：

- **在使用过程中面临知识产权相关的风险。**AI 编程助手的训练数据来源，以及生成的代码作品，均可能会涉及相关知识产权的争议，给企业用户带来风险[36]。虽然 AI 编程助手厂商提供了技术性和合同性保护，可以降低企业失去对知识产权控制的风险以及意外侵犯他人权益的风险。例如，微软出台了新版 Copilot 版权承诺（Copilot Copyright Commitment）方案，将微软现有的知识产权赔偿范围扩展适用到其商业 AI 驱动的 Copilots 付费版本（包括 GitHub Copilot 和 Bing Chat Enterprise），客户因使用微软服务而被起诉侵权，微软将会支付相关赔偿费用[37]。OpenAI 也推出了类似方案，命名为 Copyright Shield，意在缓解人们对生成式 AI 潜在侵权风险的担忧[38]。但是，这些风险仍然存在，尤其是在 AI 编程助手使用的规模和人数不断攀升的过程中。为此，企业在决定部署 AI 编程助手之时，不仅要审核厂商在此领域的承诺和服务，还要让法务、安全等团队参与到相关的决策制定过程，了解相关变更，来减轻这种风险，并出台避险方案。
- **AI 编程助手输出的内容可能会存在安全漏洞或有害信息。**大模型的"幻觉"问题依然没有得到有效解决。AI 编程助手是大语言模型基于代码数据集训练的，如果这些训练数据集没有过滤掉不安全代码、非许可代码、有毒数据或者有偏见的内容，则生成的代码、测试数据、文档、API 规范难免有可能包含错误以及潜在的安全漏洞。如果直接按照原样接受 AI 编程助手输出的内容，则会在企业内部的软件应用中引入额外的安全风险。

同时，缺乏经验的开发者可能会过度依赖 AI 编程助手，无法及时准确的识别错误，导致风险扩大。康奈尔大学（Cornell University）研究发现，经验不足的开发者在使用 AI 生成的代码时，往往比没有使用 AI 工具的开发者犯更多的错误[39]。确保培训到位，将 AI 编码助手的使用限制在那些完成培训的人身上。同时，为高素质开发者提供反馈机制，允许他们可以审查 AI 编程助手输出的代码，并能够进行评价、打分或，并可以在必要时剔除不恰当的内容。另外，辅助使用安全测试工具，如静态应用安全测试（Static Application Security Testing, SAST），帮助发现生成代码中的漏洞。

- **开发者群体由于认知偏差和确认偏误导致新增额外使用成本。**过度信任 AI 编程助手，也会带来额外的成本和陷阱，尤其是当前仍处于技术本身不成熟，以及企业部署和应用经验缺乏的初级阶段。自动化偏差（Automation bias）是当前人类普遍存在的情况，人们会更倾向于接受来自自动化系统（如 AI 编程助手）的建议，而不是来自其他人的建议。这可能导致开发者认为错误一定出在自己的代码上，而不是生成的代码，并无形中浪费时间去寻找解决方案[40]。AI 编程助手提供服务的模式，会导致确认偏误（Confirmation bias），尽管用户在解决问题时走上了错误的道理，AI 编程助手依然会持续提供建议，会让用户在错误的道路上越走越远，更加浪费时间。企业在部署 AI 编程助手的同时，要提供完善的内部培训来避免这种陷阱，确保开发团队了解 AI 工具的局限性。
- **谨慎审视 AI 编程助手带来的时间节省方面的量化数据。**避免轻信那些过分夸大 AI 编程助手对生产力提升幅度的炒作。通过代码生成效率实现时间节省是 AI 编程助手的最基础价值，但是也不要忽略，这种受控实验和小规模概念验证（PoC）所取得的结果，在大规模推广部署时未必能完全体现出来。换句话说，你在 PoC 的时候能够体现出来优异的时间节省效果，但是当你扩大部署范围，让更多的开发人员使用，并覆盖更多项目的时候，其时间节省的量化数据不会是简单的叠加，其效果的增长也不一定呈正相关。

不同开发项目的特性，开发人员素质的不同水平，都会影响实施效果，要为管理层设定好心理预期，并落实严谨的应用效果监控和规划路线图。

- **代码质量的审查与交付工作将会面临全新的挑战，需要提前衡量变化并预防风险。**采用 AI 编程助手之后，意味着企业的软件交付形态和代码库更新方式将发生变化。虽然生成式 AI 可以帮助企业快速创建代码，但是企业的审查工作以及代码交付通道都需要提前做好准备。一般情况下，AI 生成的代码更加难以审查，因为开发者并没有完全自主编写代码，也就无法对其进行合理的审查。构建一个跟踪指标体系至关重要，使用返工率、采纳率、AI 生成代码占比等指标确保生成式 AI 代码得到适当的审查，跟踪整体质量趋势，识别开发流程中的瓶颈，衡量团队效率和开发人员体验，然后在根据数据做出相应的调整和改善。

第六章　他山之石，参考行业最佳实践加速部署 AI 编程助手

具有前瞻性思维的企业和软件开发团队，为了加快创新的速度，正在迅速采用 AI 编程助手，将其作为自身生成式 AI 战略投资的重要部分，以便在未来更好的为客户提供服务，开拓新的业务发展模式。Gartner 预测，到 2028 年，90% 的企业软件开发者将使用 AI 编码助手，而 2024 年这一比例还不到 14% [41]。无论你是否已经采用 AI 编程助手，关注以下最佳实践案例，可以帮助你更好的理解 AI 编程助手的落地价值。

1．海通证券开发提效 40%+，打造“泛海言道”大模型促进创新发展

海通证券股份有限公司成立于 1988 年，截至 2023 年末，公司总资产达 7545.87 亿元，归属于母公司净资产达 1632.44 亿元。已基本建成涵盖证券期货经纪、投行、自营、资产管理、私募股权投资、另类投资、融资租赁、境外投行等多个业务领域的金融服务企业，在中国境内拥有 341 家证券及期货营业部，境内外拥有超 2400 万名客户。

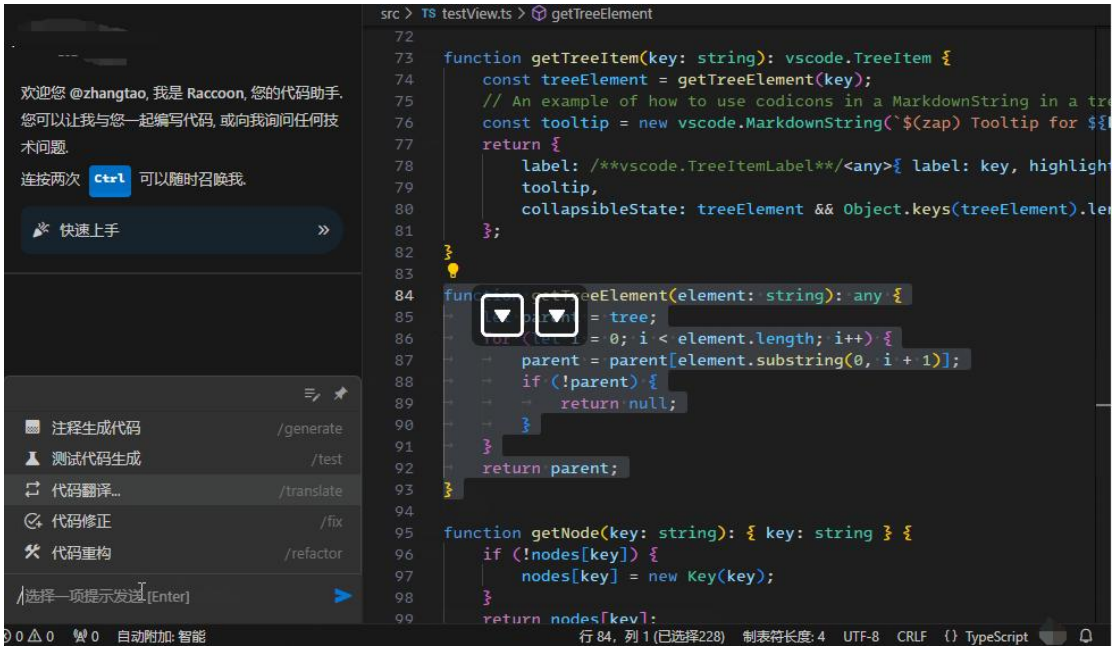
随着业务规模的不断增长，海通证券在日常的软件开发工作中，面临开发任务重、重复性工作占比高、开发者水平参差不齐、内部资料查询不变等一系列挑战。同时，虽然软件项目越来越复杂，开发团队整体的开发效率逐渐成为业务发展的瓶颈。如何利用新的生成式 AI 工具赋能软件开发中心团队的日常工作，成为海通证券面对市场的快速变化和技术的持续革新，提升研发效率和确保技术创新成为金融行业高质量发展的新动力。

经过前期的合作伙伴遴选，商汤代码小浣熊全面满足了海通证券在安全防护、代码数据信息保密等要求，以及基模型能力要大幅领先开源模型并能够提供定制化的企业级服务等技术层面的额外诉求。海通证券联合商汤代码小浣熊在企业内部推出了智能研发助手，辅助研发人员进行编程**（见图 22）**。

基于海通证券丰富的数据基础，商汤代码小浣熊构建完整思维链，深入理解业务逻辑，为开发者提供代码智能补全与对话问答服务，可辅助生产代码，降低开发技术门槛，有效提高开发效率。在减少开发者重复工作同时，还能帮助团队更早发现并修正开发中的错误，提升软件交付质量。

经过前期的试用，智能研发助手已在多个项目中展现出其卓越能力。通过智能化的代码补全、错误预测和自动修正，它不仅减少了开发者的重复工作，而且帮助团队更早地发现并修正开发中的错误，有效加速了项目进度，提高了软件交付的质量。在商汤大模型的支持下，智能研发助手能够在编码时预测光标下一行的代码，实现精准的代码补全。更为重要的是，它能通过对话问答的方式，提供包括但不限于生成测试用例、代码翻译、代码重构等多种功能，大幅拓宽了开发者在项目开发、测试及优化中的能力。

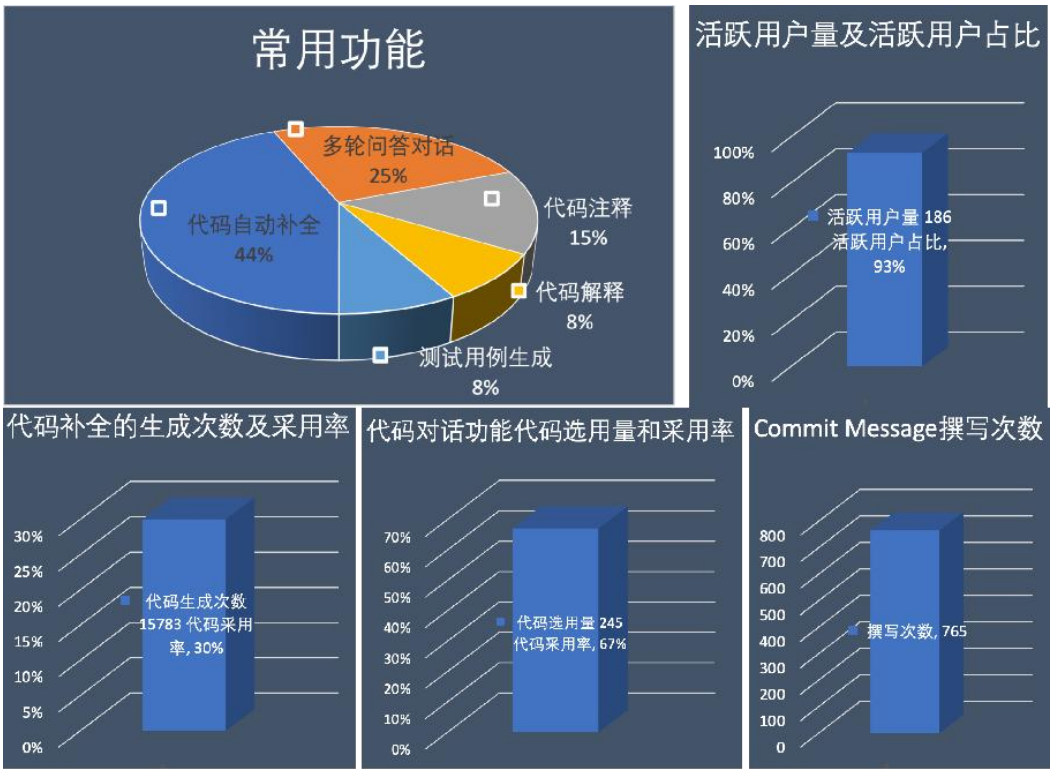
图 22：基于商汤代码小浣熊，海通证券面向企业内部软件开发团队推出了智能研发助手



基于商汤代码小浣熊的智能研发助手帮助海通证券节省了大量重复性开发工作，统一了

开发行为、代码规范和注释风格，并基于大语言模型的领先能力实现了技术问题一站式解决，最终帮助海通证券实现了开发工作效率提升超过 40%**（见图 23）**。

图 23：海通证券日常使用商汤代码小浣熊数据一览



在证券行业的数字化转型中，大模型技术正逐渐成为推动创新和效率的核心动力。海通证券与商汤合作，除了推出智能研发助手之外，还推出了智能问答、智能研报等生成式 AI 应用。智能问答，结合互联网检索引擎，汇聚海量知识，为外部客户和内部员工提供即时、高质量的问答服务，为内部员工提供即时、高质量的问答服务，提升工作效率。智能研报，通过样例示范学习能力，深度解读、分析财报数据，生成研报初稿，为研究人员提供专业可靠的写作服务。

在 2024 年 4 月 23 日，商汤与海通证券联合发布业内首个面向金融行业的多模态全栈式大模型。双方将围绕智能问答、合规风控、代码辅助、办公助手等业务领域推动大模型技术

的落地应用，深入探索智能投顾、智能投研、智能投教、舆情监控等“AI+金融”行业前沿场景，覆盖证券交易的前、中、后期各个环节，为金融行业数字化转型注入全新活力[42]。

海通证券表示，通过与商汤深度合作，我们将结合全栈式 AI 能力，共同推动证券行业的业务流程、交互变革与数智化业务系统重构，为行业垂直领域大模型的落地探索经验。

2. 澳新银行提高了 42%生产效率，代码质量提升了 12%

澳新银行（ANZ Bank）是全澳四大银行之一，为超过 850 万零售和商业客户提供银行和金融产品和服务，业务遍及 30 多个市场。澳新银行雇佣超过 5,000 名软件开发人员，涵盖软件开发生命周期的各个阶段。

澳新银行对 1,000 名软件开发人员进行了为期六周的 AI 编程助手试用实验。实验发现，当软件开发人员使用 AI 编程助手时，平均生产率提高 42%，代码质量提高 12%。开发人员的工作满意度也显著提高[43]。

澳新银行评估了参与者的情绪以及 AI 编程助手对生产力、代码质量和安全性的影响。最初，参与者将 AI 编程助手用于所建议的用例场景，并通过定期调查收集反馈。在第二阶段，采取 A/B 测试方式，参与者被分为控制组（不使用 AI 编程助手）和 Copilot 组（使用 AI 编程助手），每组都面对相同的 Python 挑战，并再次对他们的体验和结果进行调研。结果显示，AI 编程助手加持的 Copilot 组的生产力和代码质量方面有显著提升，花费的时间最少**（见图 24）**。测试还显示，AI 编程助手对于初级开发者的效率提升更加显著**（见表 2）**。该测试证实了 AI 编程助手在大规模软件工程环境中的有效性。另外，来自 1,000 名软件开发人员的调研反馈也表明工作满意度显著提高。

图 24：解决问题的总时间：控制组 vs Copilot 组

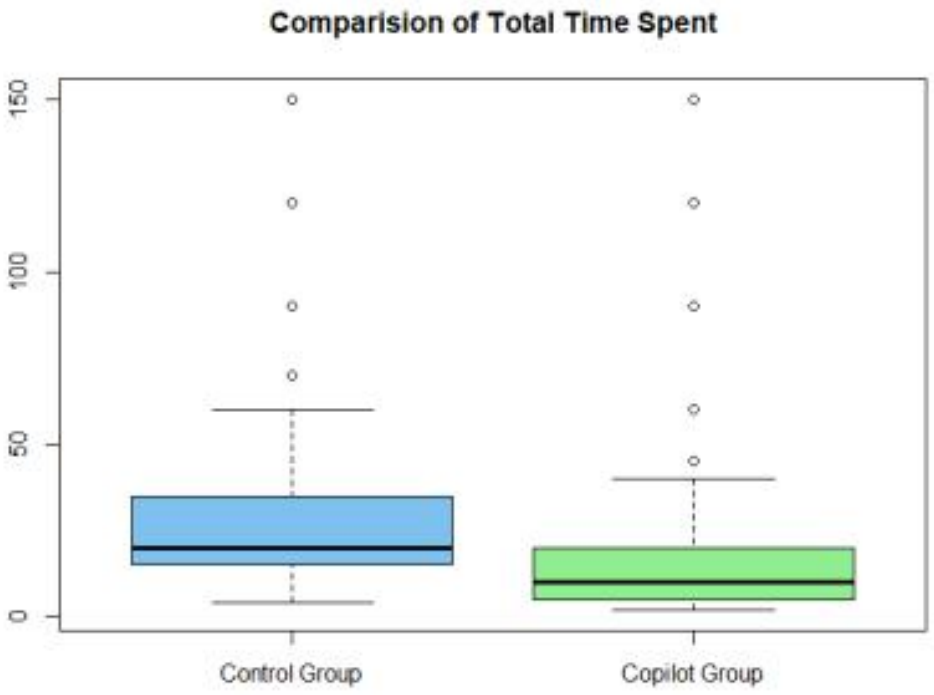


表 2：不同水平的开发者的生成效率提高情况

Table 3: Productivity increase across different proficiency levels

Python proficiency	Mean Total_Time_Spent (Control Group)/per problem (in minutes)	Mean Total_Time_Spent(Copilot Group)/per problem (in minutes)	Productivity Improvement
Beginner	20.07	9.58	52.27%
Intermediate	28.60	16.70	41.6%
Advanced	39.82	23.70	40.48%

为了分析软件开发人员在使用 AI 编程助手的生产力、工作质量和情绪的数据，澳新银行使用了多个数据来源指标，包括提供建议的次数、建议被接受的次数、接受的代码行数量，以及所使用的编程语言计其对建议的接受率、接受率百分比等。另外，针对使用体验，澳新银行保持了周期性的在线调研，以获取参与者的生产率效率和情绪相关的数据指标。

通过前期的使用和调研，有足够令人信服的证据，证明了 AI 编程助手对澳新银行工程实践的变革性影响。该工具的采用标志着工作范式的转变，使工程师能够更加专注于创造性和设计任务，同时减少花在重复编码上的时间。澳新银行正在逐步扩大 AI 编程助手在生产环境中的应用，当前有超过 1000 名软件开发人员在日常工作中已经在使用 AI 编程助手。对于 AI 编程助手带来的收益的监控和分析也依然在持续运营中。

3. 某商业银行实现提效 35%，代码补全平均响应时间 50ms/token

该银行的区域软件开发中心拥有开发人员 500 人。其面临的挑战包括，内部代码规范程度低，代码编写良莠不齐，开发团队花费大量时间在代码合规、注释编写、安全涉及等方面，同时外包人员较多，工作产出水平不一，基准难对齐。客户也曾经尝试部署开源大模型，但是发现并不能满足功能和性能的要求。

经过遴选，客户发现商汤代码小浣熊能够满足其对数据安全与信息安全的高要求，同时能够提供可量化的效能提升评估指标，并可以直接集成到客户内部的 IDE 工具，保证体验一致性。更关键的是商汤代码小浣熊的模型能力经过测试领先开源模型的表现。

客户选择部署小浣熊之后，提升了代码的规范性和质量，通过可量化的埋点统计，完善了效能评估方法，并通过私有化的部署规避了安全风险，且小浣熊的功能丰富度和性能表现均超越之前的开源大模型选择。该银行客户的代码补全接受率超过 30%、有效率超过 50%，最终实现开发提效超过 35%。

4. 某车企通过 AI 编程助手实现软件开发现代化，效率提升超过 50%

该车企内部拥有较多代码规范和定义，但是在执行过程中面临着规范化落地难的痛点，

同时面临着大量重复性开发工作导致效率提升不起来的挑战。对于过往积累的数据、自有资产也没有形成有效的沉淀与融合，无法挖掘更有价值的信息支撑开发决策。

在选择商汤代码小浣熊之前，客户也评测了市场上的一些 AI 编程助手，但是在能力表现不佳，而小浣熊则能够与客户内部自有规范、自有代码仓库进行适配，完成与企业资产的融合。更关键的是，小浣熊具备全面覆盖客户 BizDevOps 流程中的各个场景的能力，有效支撑客户对数据采纳率高的诉求，体现出极强的基础模型能力和代码生成性能。

最终，客户内部的代码标准化得到提升，并简化了执行流程，让开发者能够更便捷的执行落地，消除了大量重复性的开发工作，提升了开发团队的编写效率。通过赋能开发生命周期的各个环节，实现了代码补全有效率超过 60%，代码补全响应时间低于 50ms/token，开发效率整体提升超过 50%。

第七章　建议：不要等待，行动起来，完美是优秀的敌人

生成式 AI 是自互联网以来最具影响力的技术进步，相关技术的爆发使得大量组织进入市场并产生持续性的投入。进入 2024 年，可以看到炒作正在逐渐消退，一个更加实用的阶段已经开始。Gartner 预测，到 2028 年，在 2023 年采用 AI 编程助手的企业，其开发者团队的生产力的年度复合提升幅度将达到至少 36%[44]。

现在是尝试 AI 编程助手的最佳时机，如果你还没有行动，那么你已经落后了。不要追求完美主义，通过积极的探索来推动自身组织的进步。更何况，在使用 AI 编程代码助手方面，开发者已经比他们的企业雇主跑得更快，当他们无法访问企业批准的版本时，就会使用未经批准的版本来完成工作，这反而会平添一些不可控的风险[45]。为了更快的迎合技术发展趋势，利用好 AI 编程助手带来的技术优势，加速业务发展，企业的 CIO 及软件开发团队应该采取以下行动：

- **捕捉完整的可发展的 AI 编程助手价值体系，构建衡量收益的关键指标来评估收益。**节省时间是 AI 编程助手带来的价值之一，而非全部，要合理预估 AI 代码助手可节省的时间，使企业内的利益相关者的期望保持在合理水平。其次，实施全面的价值分析，量化企业可实现的回报，构建一个多元衡量指标组成的收益量化体系，并设计实际操作的试点来衡量 AI 编程助手所带来的影响。而非一味地听取和接受厂商所宣扬的收益，避免被“忽悠”。除时间节省和成本削减之外，扩大分析范围，充分衡量生产力的提升、软件质量提高、改善人员体验改善和产品上市速度加快的实际效果。最后，不要承诺减少开发者人员，这种狭隘的视角无法充分体现 AI 代码助手的价值，反而会在无形中让开发者心生厌恶，导致 AI 编程助手作为新的生产力工具得不到积极的响应，平添部署阻碍。在启动 AI 编程助手项目时，让企业内部的利益相关方明确地认识到此类开发不会取代

开发者，而是提高其能力。

- **建立一个跨职能团队，从评估阶段到企业推广，识别和降低 AI 编程助手相关的风险。**团队成员不仅仅是 AI 专家和业务负责人，还要包括众多职能部门的专家，如法务、安全风险、风控合规、HR 以及拥有足够话语权的高级管理人员。该团队的关键责任就是在部署 AI 编程助手之前，以及在使用过程中，总结、定位甚至穷尽潜在的风险，包括准确性、信息时效性、模型幻觉、透明度、安全和数据、道德合规等一系列风险；与此同时，该团队还应监控生成的输出，以防出现偏差、露骨材料和其他道德挑战。该团队需要打造一个可执行的框架，基于该框架将这些风险映射到相应的缓解措施之上，使风险可预防、可控制并能够可解决。当然，这个团队的设立也不仅仅是为了 AI 编程助手，更宏观的则是为了应对生成式 AI 爆发的时代。
- **创建一支小型、高技术素养的生成式 AI 代码小队，来支撑 AI 编程助手的扩大化部署。**在 AI 编程助手的部署初期，企业内部的开发者团队将会呈现不同的适应度，无论是使用的成熟度还是内心接受程度，还包括对新鲜事物的兴趣度，不同的开发者将会呈现不同的状态。在部署一定时间之后，遴选并组建一支由熟练掌握 AI 编程助手，拥有开放好学心态的开发者组成的平台团队，总结他们的使用经验，开发和发布提示工程（Prompt Engineering）模式和最佳实践，将其及时的发布和共享给整个软件开发团队，从而实现提示工程的质量一致性。通过发挥“精英小组”的积极性，形成可参考复制的最佳样本，进而带动其余开发者的兴趣，拉齐经验不足开发者的使用效果，扩大 AI 编程助手在企业内部的应用范围，并保证其收益也是正向增长的。
- **建设可进化的培训体系，缩短开发者上手到熟练使用 AI 编程助手的时间。**随着企业普遍采用 AI 编程助手来提高开发人员的生产力、丰富其开发经验，CIO 或软件开发领导者必须帮助开发人员掌握相关技能，发挥此类技术的潜在优势。虽然企业非常欢迎和期待

使用生成式 AI，但是大部分企业开发者并没有做好准备。Gartner 关于软件工程领域采用生成式 AI 的 IT 领导者投票中，60%的 IT 领导者表示其团队并不具备所需的技能[46]。帮助团队掌握所需的 GenAI 技能，是软件工程领导者当前的重要任务。但是，仓促无效的培训只会降低 AI 编程助手的投资回报。向开发人员提供自助学习资料，开通在线的自学平台，内容要包括教程和最佳实践，以及对 AI 编程助手价值的深度阐释资料。将已掌握使用 AI 编程助手的资深开发者认证为企业内部的导师，由他们提供周期性的分享与培训，验证所需 AI 编程助手技能对实现业务目标的益处。在开发者团队中尝试发展讨论小组或社区，促进开发人员之间的交流与学习，在企业内部开展结对编程（Pair Programming）、黑客松（Hackathon）、创新冲刺（Innovation Sprints）等活动，在企业内部推动社会化和体验式学习，帮助开发者团队掌握和提升 AI 编程助手技能。

- **将 AI 编程助手厂商发展成为你的成长伙伴，而非单纯的签署合同的技术供应商。**不可否认的是，生成式 AI 技术依然在高速发展进化当中，相关的生态体系也在不断波动，当前拥有优势的 AI 编程助手更多的体现在其先发优势，并未构建不可逾越的能力壁垒，技术本身将会呈现不断突破，功能不断完善甚至出现惊喜的局面。这意味着当你采购和部署了某一款 AI 编程助手，可能会很快发现市场上出现新的能力更强的替代品。积极关注 AI 编程助手的技术发展创新，让自己的眼界跟上时代发展步伐，还要积极的与 AI 编程助手厂商探索更加多元化深入的合作关系，不能再是单纯的甲乙方关系，一起探索生成式 AI 应用的实践与创新，让厂商成为你在生成式 AI 时代的伙伴，通过一系列的互动形式，如技术交流、战略合作甚至是共同研发等，一起捕捉生成式 AI 带来的价值。

第八章　展望：从 Copilot 到 Agent，迈向 AGI 彼岸

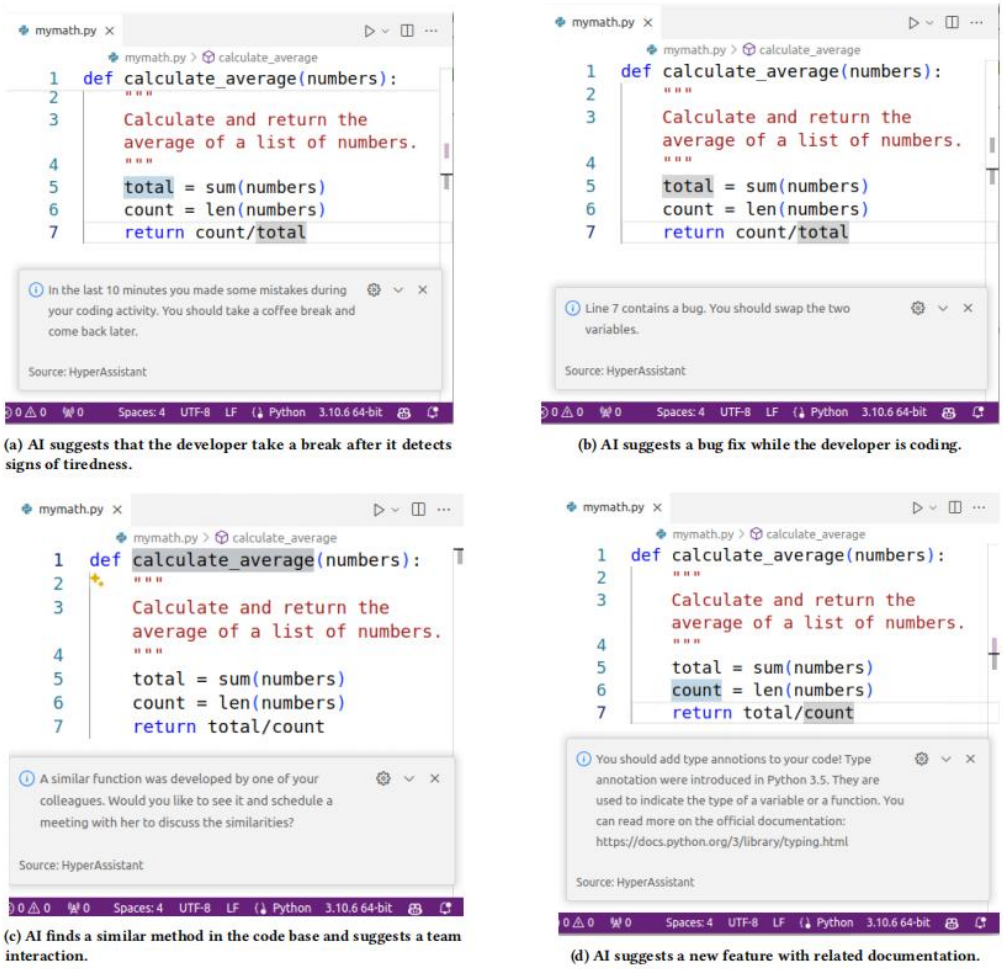
AI 编程助手的发展基本是由伴大模型技术进化所驱动的，加强 AI 编程助手的“副驾驶（Copilot）”能力，并在未来逐渐演进实现“代理（Agent）”形态。整个演进过程，除了大模型本身的技术以外，也让很多系统工程能力成为了必要部分。基于预训练语言大模型，通过继续训练和数据微调，得到代码大模型，同时针对落地需要，还需要解决 RAG、工具使用等能力，以及代码执行等问题。如何完善相关的技术能力，进而让整个演进顺利且符合开发者的需求是 AI 编程助手未来发展的核心**（见图 25）**。

图 25：从通用场景大语言模型，到代码 Copilot，再到代码 Agent

当前，AI 编程助手更多扮演着“副驾驶（Copilot）”的角色，与以前的类似系统（如确定性聊天机器人）相比，已经体现出适应性更强，可以处理更复杂的任务。然而，它依然需要人来的监督，依赖开发者清晰明确的提示词才能更好的发挥作用。这背后的主要技术原因，是大模型依然无法实现在极少人为监督的情况下的自主行动能力，以及在复杂的实际环境中适应和成功执行任务的能力。

一项研究正在尝试探讨极致的“副驾驶”能力，到 2023 年，AI 编程助手将会进化成 HyperAssistant（超级助手），能够为开发者提供全方面的支持[47]。HyperAssistant 通过自动的支持和优化开发者的工作环境而积极影响他们的心理健康，如及时提醒开发者需要休息，或者通过个性化视觉元素来优化 IDE 环境；提供先进的错误检测和错误修复功能，以及高质量的代码优化，即使是复杂的软件系统；重塑开发者之间的互动，完全集成到企业的软件开发全流程，辅助所有开发者，促进所有开发者之间的交流，如在企业代码库发现类似实践给出建议等等**（见图 26）**。

图 26：HyperAssistant 在 2030 年改进开发者的日常工作



(a) AI suggests that the developer take a break after it detects signs of tiredness.

(b) AI suggests a bug fix while the developer is coding.

(c) AI finds a similar method in the code base and suggests a team interaction.

(d) AI suggests a new feature with related documentation.

无论是厂商，还是作为用户的开发者，都愿意看到 AI 编程助手进化到 AI Agent 的形态。AI Agent 不仅能够对话，还具备反思和规划能力，能够为了完成任务目标而自主调用相关的工具，并且智能体与智能体之间也能协同配合一起完成相关任务。这对于开发者的影响将会是彻底颠覆性的。

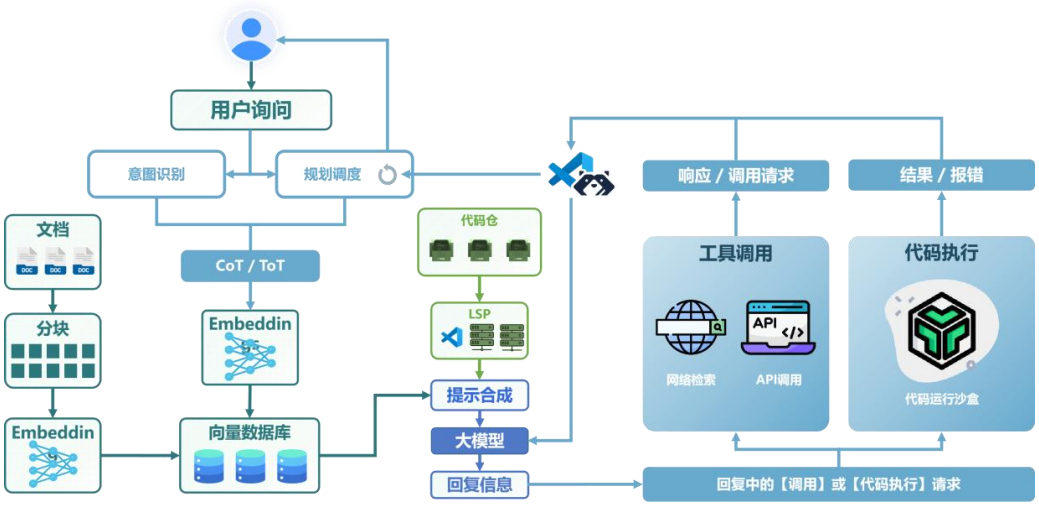
AI Agent 可以缩小当前生成式 AI 与 AGI 愿景之间的距离**（见图 27）**。AI Agent 是自主或半自主的人工智能系统，在数字或物理环境中感知、决策、采取行动并实现目标，具有自动执行任务、做出明智决策以及与周围环境进行智能互动的能力，因此有可能彻底改变各行各业和各种环境。AI Agent 的核心是由控制端、感知端、行动端三部分组成，基于此不断与环境交互并获得反馈[48]。AI Agent 可以配置在任何数字化环境中运行。虽然不能草率的去说 AI Agent 是实现 AGI 的唯一途径，毕竟技术的发展是多元和不确定的，但是从某种层面来看，AI Agent 无疑是当前生成式 AI 继续升级发展的关键方向之一。

图 27：AI Agent 帮助生成式 AI 无限接近 AGI 彼岸

以商汤日日新·代码小浣熊为例，其一直致力于发展代码智能体，不断加强任务理解和规划调度的能力，以及让自身的代码大模型学会使用工具。外围系统帮助大模型，按照预期来调用这些工作，反馈给大模型进行后续的工作**（见图 28）**。AI Agent 对于代码大模型的能力要求，提升了很多，除了代码编写能力，还需要模型在意图理解、指令遵从上十分优秀，

需要在训练过程中，就接入沙盒环境，对运行结果，报错信息等有深刻理解，能够顺利进行反思和自我修正。

图 28：适合 Agent 场景的 AI 编程助手（以商汤日日新•代码小浣熊为例）



除此之外，多智能体系统（Multi-Agent System，MAS）也得到了技术关注[49]。MAS 是一种人工智能系统，由多个独立、互动的代理组成，每个代理都能感知环境并采取行动。多个代理可以朝着一个共同的目标努力，而这个目标超出了单个代理的能力范围。多个代理的联合应用可以解决单个代理无法完成的复杂任务，同时创造出适应性更强、可扩展性更高和更稳健的解决方案。对于 AI 编程助手而言，自身将会作为开发智能体而成为更广泛的多智能体系统的一部分，全面赋能开发者的所有日常工作，这样的愿景也会逐渐到来。

另外，完全自主代理（Autonomous Agent）的概念也逐渐兴起，并将其作为最终实现 AGI 的前景广阔的方法[50]。自主代理是自我管理的人工智能系统，执行领域限定的任务，并表现出三个基本特征：自主性（在没有外部协助的情况下自主执行自己的决策和任务）；学习性（根据经验、不断变化的条件或目标修改自己的行为和内部操作）；代理性（对自己的内部状态和目的有感知，从而指导自己如何学习、学习什么，并使自己能够独立行动），

虽然自主代理的概念还处于非常新型的阶段，但是其正在成为一种重要趋势，因为它们能够实现传统人工智能技术无法达到的业务适应性、灵活性和敏捷性水平。在运行环境不可预测、实时监控不现实的情况下，自主代理的灵活性非常宝贵。

已经开始有厂商推出了自主代理（Autonomous Agent）的 AI 编程助手，宣称不仅掌握了全栈技能，还可以自主学习相关技术，自主端到端的部署应用程序、修改错误代，甚至自主训练和微调模型[51]。虽然后续得到了很多质疑，但是至少体现了业界所期待的 AI 编程助手未来的发展方向。

“这是最好的时代，也是最坏的时代”。大模型、生成式 AI 带来的技术颠覆远未停止，任何大胆的畅想在当前都属于可以理性讨论的预测。AI 编程助手的影响也在不断地进化，无论其产品形态如何改变，其宗旨依然是服务于开发者，解放和发展生产力，其价值体现则是进一步推进各个企业的数字化转型，改变这个世界。

[1] 发展新质生产力是推动高质量发展的内在要求和重要着力点，求是网

[2] 科技创新是发展新质生产力的核心要素，求是网

[3] Sea Change in Software Development: Economic and Productivity Analysis of the AI-Powered Developer Lifecycle. [arXiv:2306.15033]

[4] The 2021 State of the Octoverse, GitHub.

[5] Octoverse: The state of open source and rise of AI in 2023, GitHub.

[6] 2023 Planning Guide for Application Architecture, Integration and Platforms, Gartner.

[7] For Developers, too many meetings, too little focus time, Computerworld. 专注时间的定义是不少于 2 小时的不间断地专注于工作的时间。

[8] 2023 年 Gartner 大型企业技术采用路线图调研。这项调研参考了 IT 领导者的集体智慧，了解 205 项技术的部署计划、采用时间表、价值和风险。

[9] IBM 在 20 世纪 50 年代就推出了最早的自动编码工具 Autocoder，算是简化计算机编程任务的早期探索之一[Bitsavers, IBM 1410 Autocoder]。1958 年，约翰·麦卡锡（John McCarthy）发明了 LISP，引入了符号处理和递归函数，为人工智能编程奠定了基础并成为人工智能研究中最受欢迎的编程语言[Lisp Programming Language Guide: History, Origin, and More, History Computer]。

麻省理工学院的特里·威诺格拉德（Terry Winograd）在 1970 年开发了 SHRDLU，是一种早期的自然语言理解程序，可以解释和响应有限的英语子集中的命令，展示了 AI 理解和生成人类语言的潜力[SHRDLU, Stanford HCI Group]。

进入 20 世纪 80 年代，代码生成器（例如 The Last One）作为可以根据用户规范或预定义模板自动生成代码的工具出现。虽然不是现代意义上严格的人工智能驱动，但它们为后来代码生成和自动化的进步奠定了基础[The Last One, Personal Computer, David Tebbutt]。20 世纪 90 年代，逐渐成熟的神经网络预测模型越来越多地应用于与代码相关的任务，例如预测程序行为、检测软件缺陷和分析代码质量[A Concise History of Neural Networks, Medium]。

具有 AI 功能的代码助手类工具在 2000 年代开始出现，如微软 2018 年推出 Visual Studio IntelliCode，为重构和改进代码提供自动化帮助[Re-imagining Developer Productivity with AI-assisted tools]。这

些工具使用人工智能技术来分析代码模式，识别重构机会，并向开发人员建议适当的重构。

[10] Evaluating Large Language Models Trained on Code. [arXiv:2107.03374]

[11] 2023 年 3 月，GitHub 发布 GPT-4 驱动的版本，增加了聊天和语音功能，提供更加个性化的开发人员体验。2024 年 2 月，GitHub Copilot Enterprise 发布，能够与企业的代码知识库无缝融合，可以根据企业需求进行定制。4 月，GitHub Copilot WorkSpace 发布，这项新服务旨在提供一个能与 Copilot AI 助手原生协同工作的开发者环境，开发人员现在可以用自然语言进行头脑风暴、规划、构建、测试和运行代码。5 月，推出 GitHub Copilot Extensions，项服务使得开发者能够在 IDE 或 GitHub.com 中，使用自然语言和他们偏好的工具及服务进行构建和云端部署。不再需要离开熟悉的开发环境。

[12] 2023 年 Gartner AI 调研：CIO 和技术领导者观点。本调研旨在了解 CIO 和技术领导者对 AI 的看法，同时也为了更好地了解企业机构因最近的 AI 变革和公告而正在采取的行动。

[13] The SPACE of Developer Productivity: There's more to it than you think, ACM Queue.

[14] The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. [arXiv:2302.06590]

[15] Research: Quantifying GitHub Copilot's impact on code quality, GitHub.

[16] Code Red: The Business Impact of Code Quality -- A Quantitative Study of 39 Proprietary Production Codebases. [arXiv:2203.04374]

[17] Top 5 Strategic Technology Trends in Software Engineering for 2024, Gartner.

[18] Research: Quantifying GitHub Copilot's Impact on Developer Productivity and Happiness, GitHub.

[19] A Large-Scale Survey on the Usability of AI Programming Assistants: Successes and Challenges. [arXiv:2303.17125].

[20] A Neuro-Metabolic Account of Why Daylong Cognitive Work Alters the Control of Economic Decisions, Current Biology.

[21] Developer Flow State and Its Impact on Productivity, Stack Overflow.

[22] 《没有银弹，软件工程的本质性与附属性工作》，Frederick Phillips Brooks.

[23] Community of Practice Essentials, Gartner.

[24] Manifesto for Agile Software Development, The Agile Manifesto.

[25] 2024 Gartner Technology Adoption Roadmap for Large Enterprises Survey, Gartner.

[26] https://github.com/mgreiler/code-review-checklist?tab=readme-ov-file. This code review checklist helps you be a more effective and efficient code reviewer.

[27] Predicts 2024: Generative AI Is Reshaping Software Engineering, Gartner.

[28] Top 5 Strategic Technology Trends in Software Engineering for 2024, Gartner.

[29] Microsoft Fiscal Year 2024 Third Quarter Earnings Conference Call, Microsoft.

[30] 大语言模型综合能力测评报告，2024，InfoQ 研究中心。

[31] The state of AI in early 2024: GenAI adoption spikes and starts to generate value, McKinsey.

[32] 首批通过！商汤小浣熊代码大模型通过可信 AI 代码大模型评估，获当前最高等级，信通院

[33] OpenCompass 2.0 大语言模型阅读榜单，2024 年 4 月榜

[34] Clean Code: A Handbook of Agile Software Craftsmanship.

[35] Coding on Copilot: 2023 Data Suggests Downward Pressure on Code Quality, GitClear.

[36] OpenAI, Microsoft want court to toss lawsuit accusing them of abusing open-source code, Reuters.

[37] Microsoft announces new Copilot Copyright Commitment for customers, Microsoft.

[38] OpenAI promises to defend business customers against copyright claims, TechCrunch.

[39] Do Users Write More Insecure Code with AI Assistants?, Stanford University.

[40] How do people react to AI failure? Automation bias, algorithmic aversion, and perceived controllability, Oxford Academic.

[41] Top 5 Strategic Technology Trends in Software Engineering for 2024, Gartner.

[42] 金融行业首个！商汤科技联合海通证券发布多模态全栈式大模型，商汤科技

[43] The Impact of AI Tool on Engineering at ANZ Bank An Empirical Study on GitHub Copilot within Corporate Environment. [arXiv:2402.05636]

[44] Innovation Guide for AI Code Assistants, Gartner.

[45] 2023 Developer Survey, Stack Overflow. 超过 80%开发者在日常的编程工作中使用 AI 编程助手。

[46] 2024 Gartner Technology Adoption Roadmap for Large Enterprises Survey.

[47] From Today’s Code to Tomorrow’s Symphony: The AI Transformation of Developer’s Routine by 2030. [arXiv:2405.12731]

[48] The Rise and Potential of Large Language Model Based Agents: A Survey. [arXiv:2309.07864]

[49] LLM Multi-Agent Systems: Challenges and Open Problems. [arXiv:2402.03578]

[50] A Survey on Large Language Model based Autonomous Agents. [arXiv:2308.11432]

[51] Meet Devin: The First AI Software Engineer Redefining Code, Future Tools.