第1章

欢迎来到AI编程时代

欢迎来到AI编程的新纪元。随着大语言模型的出现，我们正站在一场彻底改变软件开发领域的革命前沿。AI不再是遥不可及的概念，而是已经深入我们的日常编程实践中，重新定义了人机交互的方式。

本章将带你深入探索AI编程的世界，揭示它如何改变传统开发模式，降低编程门槛，并为创新开辟无限可能。无论你是经验丰富的开发者，还是对编程充满好奇的新手，这里都有让你耳目一新的见解。让我们一同踏上这段激动人心的AI编程之旅，探讨它对软件开发生态的深远影响，以及它将如何塑造我们的数字未来。

1.1 AI编程的兴起与未来趋势

1.1.1 AI编程序幕拉开: 大语言模型的编程应用

大语言模型（large language models，LLMs）的出现标志着 AI 编程时代的序幕正式拉开。这些模型通过在海量文本和代码数据上进行训练，获得了理解和生成人类语言及编程语言的能力。只需要给 AI 一段提示词，AI 就能为我们编写代码、运行代码并展示运行结果，如图 1–1 所示。

图 1–1　使用大模型开发简易的微信 APP 聊天界面

大语言模型在编程领域的主要应用包括：

◎ 代码补全：大语言模型能根据上下文预测并提供合适的代码片段，以提高编程效率。例如，OpenAI 的 Codex 模型可在开发者输入注释或部分代码后，自动生成剩余代码。

◎ 从自然语言到代码转换：开发者可使用自然语言描述需求，大语言模型能将这些描述转换为可执行代码。这一功能使编程变得更加直观，尤其对编程新手有益。

◎ 代码解释和文档生成：大语言模型能理解复杂代码结构，并生成相应注释和文档。这不仅有助于代码维护，也便于团队成员之间的沟通协作。

◎ 代码重构和优化：通过分析现有代码，大语言模型可提出重构建议或直接生成优化后的代码版本，帮助提高代码质量和性能。

◎ 编程教育：大语言模型可作为智能导师，回答编程问题，解释复杂概念，甚至提供个性化学习路径，为编程教育带来创新，如图 1–2 所示。

◎ 跨语言转换：大语言模型具备在不同编程语言之间进行代码转换的能力，这对项目迁移和多语言开发环境具有重要意义。

图 1–2　AI 编程助手回答编程问题

GPT-3、Codex 等模型的发布，以及 GitHub Copilot 等 AI 编程工具的出现，标志着 AI 编程开始进入开发者的日常工作流程。

随着 AI 技术的快速发展，编程领域甚至出现了专门针对代码生成和理解的大语言模型。这些模型在通用大语言模型的基础上，进行了针对性的优化和训练，以更好地适应编程任务的特殊需求。典型的编程专用大语言模型包括：

◎ OpenAI Codex：由 OpenAI 研发，为 GitHub Copilot 提供支持，能够理解自然语言指令并生成相应代码。

◎ DeepMind AlphaCode：由谷歌公司研发，专门用于解决编程竞赛问题，展示了在复杂算法任务中的卓越能力。

◎ Anthropic Claude：由 Anthropic 公司研发的通用模型，在代码生成和理解方面表现出色，尤其擅长解释复杂代码。

◎ CodeGeeX：由清华大学和智谱 AI 联合开发，支持多种编程语言，提供代码补全和生成功能。

这些工具为软件开发带来了显著的效率提升和创新可能。然而，当前阶段的 AI 编程工具仍存在一些局限性，如生成的代码可能存在错误或安全漏洞，以及对特定领域知识的理解有限等。因此，人类开发者的专业判断和监督仍然不可或缺。

大语言模型的出现不仅改变了代码的编写方式，也正在重塑整个软件开发行业。随着技术不断进步，AI 编程的应用范围和深度将持续扩大，为编程世界带来更多革新。

1.1.2 AI编程大众化：AI编程的产品化和普及

大语言模型的快速发展推动了 AI 编程工具的产品化和普及，使得更广泛的群体能够参与软件开发。这种大众化趋势主要体现在多样化 AI 编程产品的涌现，让人们能更好地利用 AI 加速编程过程。

1. 对话式AI编程助手

基于 ChatGPT、Claude 等大语言模型开发的对话式 AI 编程助手，使得用户可以通过自然语言对话获取编程帮助。这种方式特别适合编程新手和非技术背景的用户，降低了编程的入门门槛。

2. 专业领域AI编程助手

针对特定编程领域或技术栈的 AI 编程助手正在涌现。例如，以 v0.dev 为代表的 AI 编程平台专注前端开发，可以帮助开发者快速生成 HTML、CSS 和 JavaScript 代码；而数据科学领域的 AI 助手则可以辅助数据分析和可视化代码的编写。

3. 集成式AI编程助手

主流集成开发环境（integrated development environment，IDE）正在集成 AI 编程助手功能。GitHub Copilot、文心快码、通义灵码、CodeGeeX 等作为典型代表，可以直接集成在 Visual Studio Code 等编辑器中，为开发者提供实时的代码补全和生成建议。这类工具大大提高了编程效率，使得开发者能够更快地将想法转化为代码。

4. 原生AI编辑器

以 Cursor 为代表的原生 AI 编辑器将 AI 能力深度集成到开发环境中。这类工具

不仅提供实时代码补全和生成功能，还能理解项目上下文，提供更智能的编程建议。Cursor 等工具的出现标志着 AI 编程正从辅助工具向核心开发平台演进。

5. AI编程平台。

豆包 MarsCode 等 AI 编程平台提供了更全面的 AI 辅助开发体验。这些平台整合了代码编辑、版本控制、部署等功能，并在各个环节中融入 AI 能力，不仅提高了开发效率，还为团队协作提供了新的可能性。

6. AI编程智能体

AI 编程智能体代表了更高级的 AI 编程形态。如 Replit Agent 这类编程智能体（如图 1–3 所示），不但能够理解复杂的项目需求，还能自主完成部分编程任务，甚至与人类开发者协同工作。虽然目前还处于早期阶段，但 AI 编程智能体展现了未来 AI 编程的发展方向。

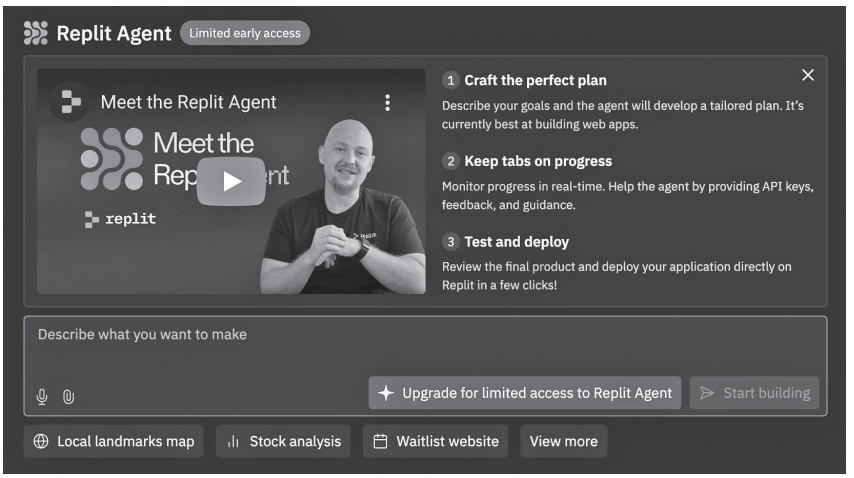

图 1–3　Replit Agent 编程智能体界面

这些多样化的 AI 编程产品显著提升了专业开发者的生产力，同时为非专业人士打开了软件创作的大门。AI 编程工具的普及正在淡化程序员和非程序员之间的传统界限，孕育出一个更具包容性和创新力的数字生态系统。

然而，尽管 AI 编程工具降低了编程门槛，深入理解编程原理和培养问题解决能力仍然是不可或缺的。AI 工具应被视为增强人类创造力的有力助手，而非取代人类思考的替代品。开发者需要学会如何有效利用这些工具，将其融入自己的工作流程中，以实现最大化的效率提升。

随着 AI 技术的不断进步和产品的持续迭代，AI 编程工具必将变得更加智能、直观和易用，进一步加速编程的大众化进程。这种变革不仅将重塑软件开发行业的格局，还可能对社会的数字化转型产生深远影响，推动创新型人才的培养和数字经济的蓬勃发展。

1.1.3 AI编程的智能化趋势

AI 编程工具的快速发展还呈现出明显的智能化趋势，这种趋势体现在 AI 辅助编程的能力范围不断扩大，从简单的代码片段生成逐步发展到更复杂的编程任务。以下将详细探讨 AI 编程智能化的 4 个主要阶段。

1. 从编写代码片段到生成完整的功能函数

AI 编程助手最初主要用于辅助编写简单的命令行指令或短小的代码片段。例如，GitHub Copilot 可以根据注释生成 Linux 命令，或者补全简单的变量、字符串、函数等短的代码字符，如图 1–4 所示。随着模型的改进，AI 助手现在能够理解更复杂的编程上下文，生成完整的功能函数。

这一进展极大地提高了开发效率，特别是在处理常见编程任务时。开发者只需提供简要的功能描述，AI 就能生成包含异常处理、输入验证等完整逻辑的函数。这不仅节省了时间，还有助于减少常见的编程错误。

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3  import pandas as pd
4
5  def plot_two_variables(data, col_a, col_b):
6      """
7      Plot a scatter plot of two variables.
8      """
9      plt.scatter(data[col_a], data[col_b])
10     plt.xlabel(col_a)
11     plt.ylabel(col_b)
```

图 1–4　AI 辅助编写短小的代码片段

2. 从编写功能函数到完整的小项目

随着 AI 模型的进一步发展，它们开始展现出构建完整小项目的能力。这包括生成多个相互关联的函数、类，甚至简单的用户界面。例如，AI 可以根据需求描述生成一

个简单的网页爬虫程序，或者一个基本的待办事项应用，如图 1–5 所示。

这一阶段的进展使得快速原型开发成为可能。开发者可以利用 AI 快速构建概念验证或最小可行产品，大大缩短了从想法到实现的时间。对于初创企业和创新项目来说，这是一个重要的优势。

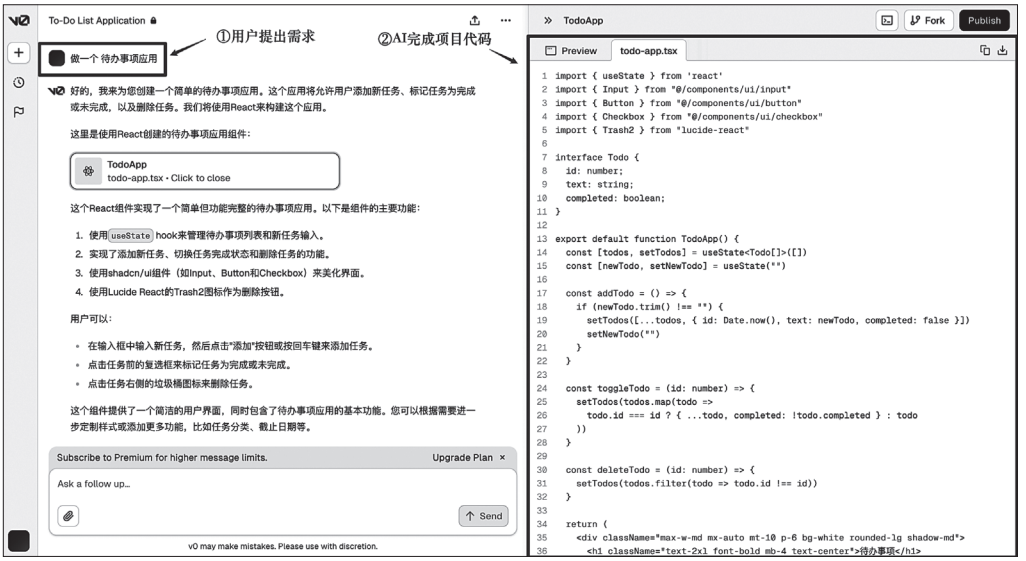

图 1–5　AI 实现一个待办事项应用

3. 从小项目到完成中大型项目

AI 编程工具正在朝着能够协助完成更大规模、更复杂项目的方向发展。虽然目前 AI 还无法独立完成大型项目，但它已经能够在项目的各个环节提供有力支持。例如，AI 可以生成项目框架、编写单元测试、生成文档，甚至提供架构建议。

在这个阶段，AI 更多地扮演高级编程助手的角色。它可以帮助开发者处理烦琐的编码任务，让开发者将更多精力集中在高层次的设计和问题解决上。这种协作模式正在改变软件开发的流程和效率。

4. 从人机协同到AI独立自主完成项目

尽管目前还处于早期阶段，但 AI 独立完成整个项目的前景已经开始显现。一些研究项目和实验性工具已经展示了 AI 在理解复杂需求、设计系统架构、编写和测试代码等方面的潜力。

这一趋势可能导致软件开发范式的根本性转变。在未来，开发者的角色可能更多

地转向需求分析、系统设计和 AI 输出的审核与优化。这不仅会大幅提高软件开发的效率，还可能催生新的商业模式和职业机会。

AI 编程的智能化趋势正在快速推进，从简单的命令生成到独立完成项目。这一发展轨迹不仅提高了编程效率，还正在重新定义软件开发的过程和方法。随着 AI 能力的不断提升，开发者的角色可能会更多地转向高层次的系统设计和创新思考，而将更多常规编码任务交给 AI 助手。这种趋势预示着软件开发行业可能迎来一个全新的时代，其中人机协作将成为主导模式，推动编程效率和开发者的创新能力达到新的高度。

1.2 AI如何改变编程生态

1.2.1 编程门槛的降低和编程群体的扩大

人工智能技术的进步正在显著降低编程门槛，扩大参与编程的群体范围。AI 编程的发展模糊了“技术”和“非技术”岗位的界限，提高了整体工作效率，促进了跨部门协作。这种变革主要体现在两个方面。

（1）AI 编程工具首先影响了与技术部门紧密合作的非技术岗位。产品经理现可使用无代码平台（如 Bubble）快速构建产品原型，加速产品开发周期。运营人员借助 Jupyter Notebook 等工具，能更轻松地进行数据分析，实现数据驱动决策。设计师通过 Figma to React 等工具，可将设计直接转化为代码，缩小了设计和开发的差距。

（2）基本的编程能力正成为更多岗位的必备技能，类似于办公软件使用能力的普及。这种趋势体现在以下几个方面：

◎ 办公自动化：越来越多的职场人员开始使用 Python 脚本自动化日常任务，AI 助手进一步简化了这一过程。

◎ 数据素养：AI 辅助的数据分析工具使非专业人士也能进行复杂的数据处理和分析。

◎ 业务定制化：各行业特定需求推动了编程学习。例如，金融从业者开发交易算法、教育工作者创建交互式教学内容。

◎ 创业和副业领域：AI 编程工具降低了独立开发应用的门槛，使创意转化为现实变得更加容易。

AI 技术的发展使更多不同背景的人群能够参与到编程活动中，实现“人人皆程序员”。这些人群包括：

◎ 非技术背景的专业人士：营销、设计、金融等领域的专业人士能够创建简单的

应用程序或自动化脚本。

◎ 学生和教育工作者：AI 辅助的编程学习平台可提供更直观、交互式的学习体验。

◎ 创业者和小企业主：低代码和无代码平台使快速开发原型或简单的商业应用成为可能。

◎ 设计师和前端开发者：AI 驱动的从设计到代码的转换工具减少了对前端开发技能的依赖。

◎ 领域专家：各行各业的专家可以将专业知识转化为实用的软件解决方案。

◎ 业余爱好者和创客：AI 工具使编程成为一种更易接触的爱好。

◎ 转行人士：AI 编程工具为希望进入 IT 行业的人提供了更平缓的学习曲线。

“人人皆程序员”的趋势将推动编程生态系统的进一步变革。未来需要采取多方面措施，包括开发新的最佳实践和管理框架以适应更广泛的编程参与者，重新定义编程教育体系以满足不同背景人群的需求，建立跨学科合作模式促进技术与各领域知识的融合，以及设计更智能、更直观的编程辅助工具以进一步降低编程门槛。

1.2.2 “所见即所得”的编程过程

AI 技术的进步正在改变传统的编程范式，将复杂的编程过程隐藏在用户友好的界面之后。这种转变使得最终用户能够直接获取所需的结果，而无需深入了解底层的技术细节。

现代 AI 驱动的工具正在实现“所见即所得”的编程体验，这种体验主要体现在以下几个方面。

◎ 直接呈现结果：用户输入需求后，AI 系统能够直接生成最终结果，而非提供中间步骤或代码。

◎ 自然语言交互：用户可以使用日常语言描述需求，无需学习特定的编程语法。

◎ 实时可视化：结果以直观的形式即时呈现，如图表、网页或可交互的界面。

◎ 迭代优化：用户可以通过简单的反馈和指令快速调整结果，无需修改底层代码。

AI 驱动的“所见即所得”编程正在多个领域应用和发展，以下是几个典型的应用场景。

◎ 数据分析：使用智谱 AI 的大语言模型，用户可以直接获得数据分析结果和洞见，而无需编写复杂的统计代码。例如，用户可以直接询问“过去三个月的销售趋势如何，绘制趋势折线图”，AI 将分析相关数据并生成报告，如图 1–6 所示。

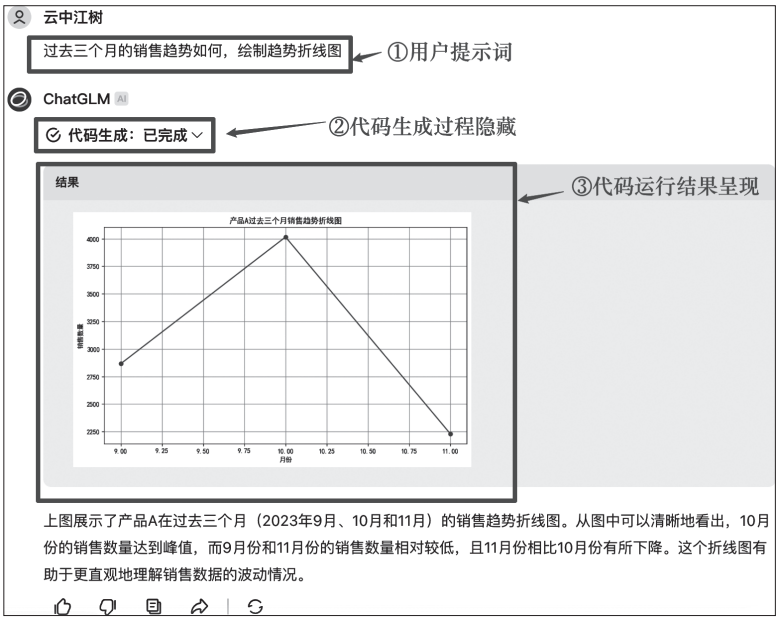

图 1–6　使用智谱 AI 数据分析

◎ 网页设计。通过 Claude 等 AI 助手，用户能够描述所需的网页布局和功能，系统直接渲染出完整的网页，无需手动编写 HTML 和 CSS。用户可以输入提示词“创建一个现代的美观大方的登录页面，包含用户名和密码输入框”，AI 就能生成相应的网页，如图 1–7 所示。

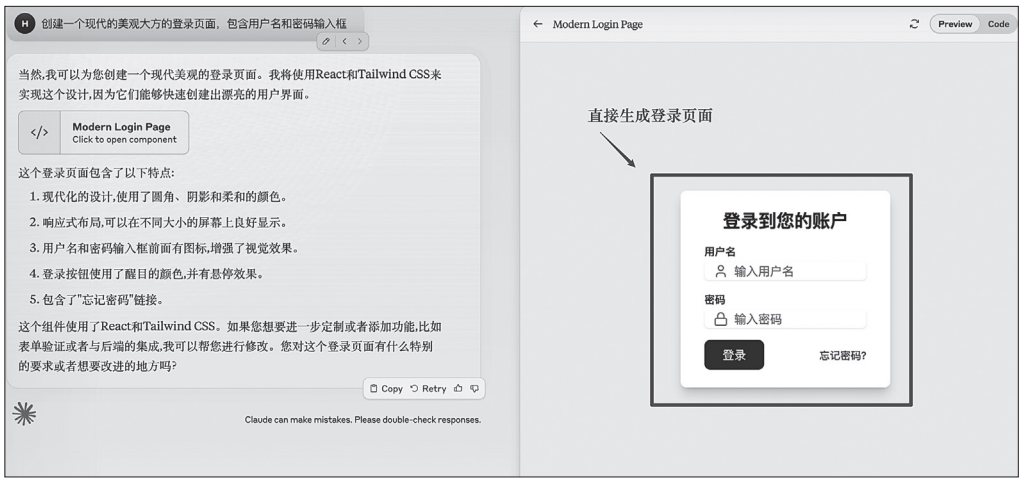

图 1–7　AI 生成登录页面

◎ 流程图绘制：使用 Kimi 等工具，用户只需描述业务流程，AI 就能自动生成相应的流程图，省去了学习专业绘图软件的需求。例如，描述“生成“客户订单处理流程”，直接生成一个包含订单接收、处理、发货等步骤的流程图”，如图 1–8 所示。

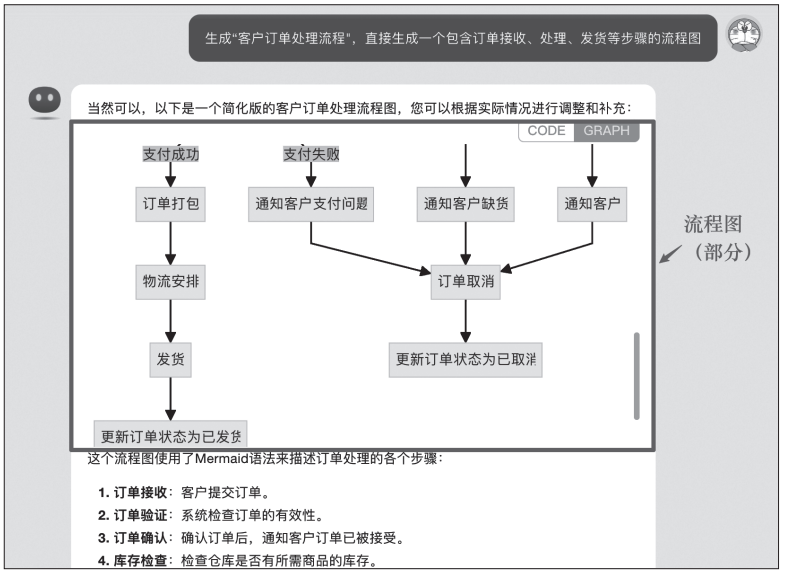

图 1–8　AI 绘制流程图

◎ 自动化脚本：用户描述日常重复任务，AI 能够生成并执行自动化脚本，无需用户了解具体的编程语言。如“15 点，推送有关财经新闻的最新消息”，AI 可以创建并设置相应的自动化任务，如图 1–9 所示。

图 1–9　使用文小言自动订阅信息

“所见即所得”的编程过程对技术行业和社会都产生了深远的影响。通过直接生成结果，大幅减少了从需求到实现的时间，加速了项目进度。更多人能够将创意快速转化为现实，有助于激发和实现更多创新想法。越来越多的岗位开始强调问题解决和创意思维，而非纯粹的编码技能。AI 技术的不断进步预示着“所见即所得”的编程过程将变得更加普遍。可以预见，未来将出现更多直观、智能的创作工具。

1.2.3 提出问题和结果验收的能力更加重要

AI 编程的快速发展正在重塑软件开发流程。随着编写代码成本的大幅降低，AI 自主解决问题能力的提高，在软件开发流程中提出问题和结果验收的能力变得越来越重要。

1. 编写代码的成本大幅降低

AI 编写代码的时间和成本花费已经接近于零。现代 AI 系统能够在几秒钟内生成复杂的代码段，这种速度远超人类程序员。例如，阿里的通义千问模型能够快速将自然语言描述转化为功能完整的代码。关于成本方面，一旦 AI 模型训练完成，生成代码的边际成本几乎可以忽略不计。这种高效率使得软件开发的瓶颈不再是代码编写本身，而是转移到了开发的其他环节。

2. AI自主解决问题能力的提高

AI 系统在自主解决编程问题方面的能力正在不断提升。这种进步体现在以下 3 个方面。

（1）错误检测和修复。AI 能够快速识别代码中的语法错误、逻辑漏洞，并提供修复建议。例如，百度推出的文心快码不仅可以生成代码，还能指出潜在的安全隐患。

（2）代码优化。AI 系统能够分析现有代码，提出性能优化建议。如阿里的通义灵码可以使用 AI 大模型技术分析代码瓶颈，给出多种解决方案优化代码执行效率。

（3）自动化测试。AI 可以生成全面的测试用例，并自动执行测试过程。这大大提高了软件质量保证的效率。

3. 新的关键能力需求

随着 AI 在编程中的应用日益广泛，人类在软件开发过程中的角色正在发生变化。问题定义、结果验收等能力变得越来越重要。

（1）问题定义能力：准确定义问题和需求成为关键技能。开发人员需要清晰地描述所需功能，向 AI 正确下达指令，以便 AI 系统能够准确理解并生成相应代码。

（2）结果验收能力：评估 AI 生成代码的正确性和适用性变得至关重要。这要求开发人员具备深入的领域知识和系统思维。

（3）创新思维：AI 擅长于已知问题的解决，而人类则需要专注创新性思维，提出新的问题和解决方案。

（4）跨学科整合能力：将不同领域的知识与编程需求相结合，成为人类开发者的独特优势。

这种趋势要求对软件开发人员的教育和培训进行调整。培养学生准确分析复杂问题和明确需求的能力。强化整体系统设计的能力，而不仅是编码技巧。加强对软件质量评估和测试方法的训练。

软件开发行业的实践也需要相应调整，开发流程可能更加注重需求分析和结果验证阶段。代码审查的重点可能从语法和风格转向更高层次的架构设计考量。项目管理方法也需要适应 AI 辅助开发的特点，变得更加灵活和迭代化。

1.3 本章小结

AI 编程技术的快速发展正在彻底改变软件开发的格局。大语言模型的出现推动了 AI 编程工具的产品化和普及，从简单的代码补全到复杂的项目生成，AI 正在逐步提升其在编程领域的能力。这一趋势不仅提高了专业开发者的效率，还降低了编程门槛，使得更广泛的群体能够参与软件开发。

随着 AI 编程的智能化程度不断提升，传统的编程范式正在发生转变。“所见即所得”的编程体验和 AI 自主解决问题的能力正在成为新的发展方向。然而，这也对开发者提出了新的要求，使得提出问题和结果验收的能力变得更加重要。未来的软件开发将更加注重创新思维、跨学科整合和系统设计，而编码本身的重要性可能相对降低。