南京大学
NANJING UNIVERSITY

可信智能化软件工程：

人工智能打开的空间和带来的挑战

李宣东　南京大学

2025年8月29日

SEG Nanjing University

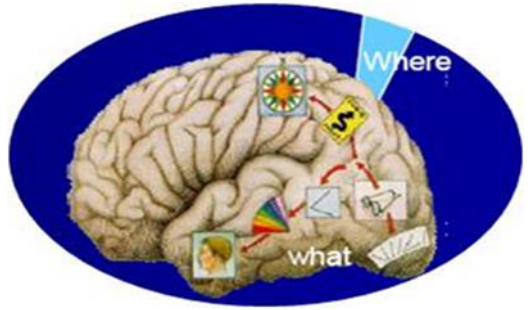
南京大学计算机科学与技术系
Department of Computer Science and Technology Nanjing University

提　纲

- 软件、软件开发与演化
- 人工智能打开的空间和带来的挑战
  - 智能化软件：确定的符号计算与非确定的概率计算相融合
    - 打开了软件解决问题的空间，带来了软件可信保障的挑战
  - 大语言模型：基于自然交互的人机协同软件开发与演化工具
    - 打开了解决软件问题的空间，带来了软件可信保障的挑战
- 进一步思考和小结

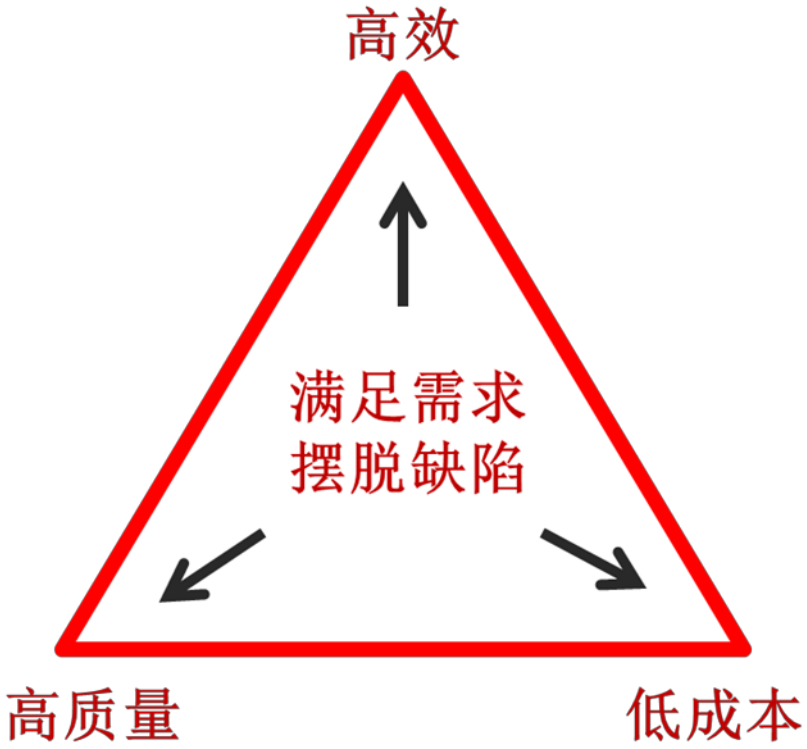
软件是人类生产的最复杂制品

- 软件开发和演化是人类针对所解决问题的创造性思维活动
  - 科学家探索世界、工程师改造世界
  - 计算机科学家、软件工程师创造世界
- 软件是人类大脑逻辑思维活动的体现
  - 认识和理解人的思维活动非常困难
    - 认识与理解软件系统非常困难
  - 管理和规范人的思维活动非常困难
    - 管理和控制软件开发过程相当困难
    - 软件开发进展情况难以衡量，质量难以评价



软件开发与演化

- 把现实世界复杂的问题模型转换为计算模型
  - 从自然语言需求到满足需求的程序代码
    - 创造性过程，决策性任务，工作难度大、人力成本高
    - 不可判定的计算问题
    - 规约、设计、编码、测试、调试、运维、……
    - 主要涉及两种语言：自然语言，程序语言
  - 目标：高效、高质量、低成本
    - 在成本一定的情况下，提高效率与保障质量通常是一对矛盾
    - 提高效率意味着质量下降，提高质量意味着效率下降
  - 核心问题：如何使得软件系统满足需求、摆脱缺陷？



人是软件开发与演化的主体

- 人是软件需求产生、认识、确认的主体
  - 正确认识和理解软件需求（做正确的事， build right things ）十分困难
    - 需求是人类在认识复杂世界的基础上形成的主观意图
    - 人类对复杂事物的认识不是一蹴而就的，需要循环反复、不断探求
    - 人类认知的局限性导致软件需求经常动态变化不确定
- 人在软件开发与演化过程中是会犯错的
  - 软件可信保障（正确地做事， build things right ）成本很高
    - 人类认知的局限、疏忽、遗漏等，导致软件系统缺陷难以避免
    - 软件产品没有“三包”，绝不包退（颗粒无收）、包换（再做一遍）、包修（无限亏损）

软件创造与制造过程相互融合

- 软件开发与演化：从自然语言需求到满足需求的程序代码
  - 软件创造：规约软件需求（分析、认识、理解和确认软件需求）
  - 软件制造：根据软件需求规约构造满足需求的程序代码
  - 软件创造与软件制造过程，看似分离实则相互融合
    - 规约、设计、编码、测试、调试、……
- 软件系统开发出来之前没有参考样品
  - 做正确的事和正确地做事融为一体：在创造过程中制造，在制造过程中创造
  - 制造未完成就不知道创造是否正确；创造不正确，制造再完美也没意义
  - 需求规约是否正确需要等代码产生后才能最终判定
    - 从需求到代码周期长、成本高、代价大
  - 我们经常陷入“按错误的需求开发出所谓正确的软件”的困局

人类在软件开发与演化过程中缺失掌控能力

- 软件创造与制造过程难以有效解耦
  - 软件开发（制造）过程充满不确定性，人类掌控力缺失
  - 上世纪六十年代爆发软件危机
    - 供求关系失调、开发费用失控、进度拖延
    - 可靠性差、难以维护、……
- 如何解耦软件创造与制造过程？
  - 按照软件系统功能规约需求，瀑布式开发
  - 按照问题域对象及其关系规约需求，面向对象开发
  - 遵循人类认识复杂事物的规律规约需求，迭代开发
    - 原型速成、敏捷开发、开发运维一体化（DevOps）

软件开发与演化是决策性任务

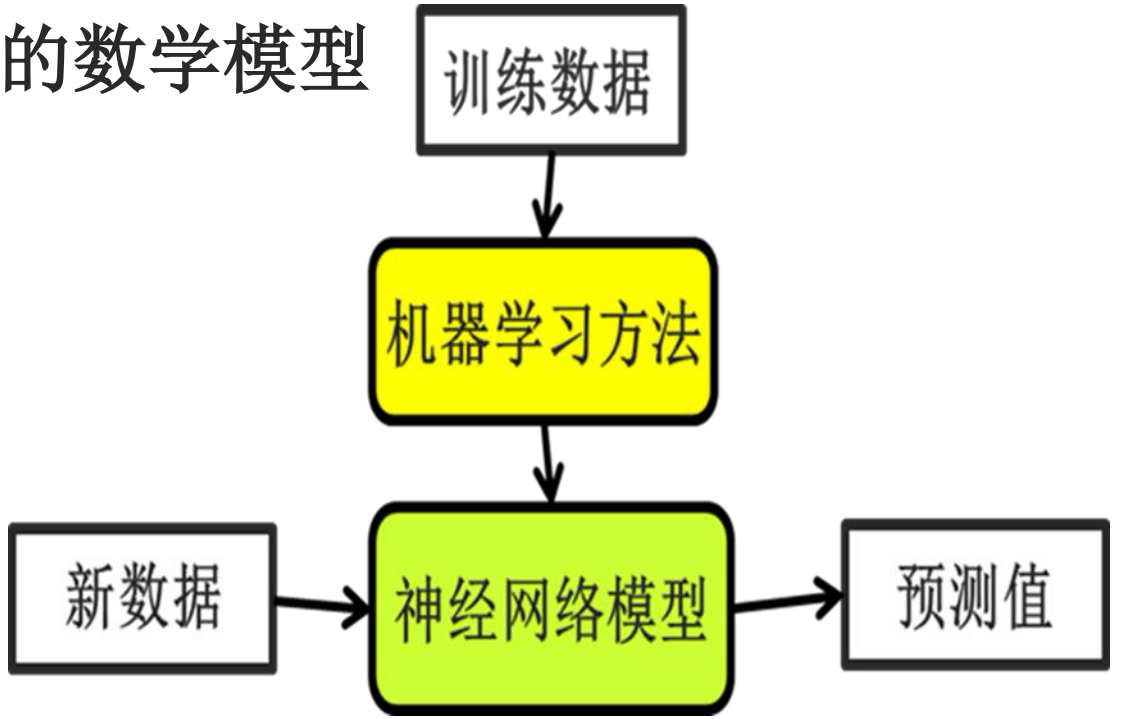
- 人类在现实世界中所承担完成的任务
  - 预测性任务
    - 通常不需要承担风险责任、没有可信保障需求
  - 决策性任务
    - 通常需要承担相应的风险责任、有相应的可信保障需求
- 人在软件开发与演化过程所完成的是决策性任务
  - 需要对任务完成正确与否做出可信性判断
    - 需要对软件系统是否满足需求、摆脱缺陷做出可信性判断
  - 承担相应的风险责任，以此构成一定的可信保障
- 人们通过软件系统完成的任务大都也是决策性任务

算法：软件系统解决问题的核心

- 算法及其复杂性
  - 求解问题需要遵循的、被清楚指定的简单指令的集合
    - 输入和输出、确定性、有穷性、能行性
    - 算法复杂性：P类问题；NP类问题：随机算法、近似算法
  - 可判定计算问题与不可判定计算问题
- 人工基于逻辑设计的算法（逻辑使能的算法）
  - 内在逻辑可理解、可解释，构造过程、输出结果可信
  - 正确性可证明（输出结果满足需求规约）
- 机器学习数据训练的模型（数据使能的算法）

机器学习数据训练的模型：预测性算法

- 基于概率与统计原理和训练数据形成的数学模型
  - 预测性算法：输出的是预测性结果
    - 预测性判别，预测性内容生成
    - 输出结果缺失可信性判断
  - 基本特征（对比于基于逻辑设计算法）
    - 内在逻辑不可理解、不可解释
    - 易受数据扰动；缺陷不可避免、难以预测（基于概率的拟合逼近）
    - 输出结果缺失可信性判断



机器学习数据训练的模型

- 打开了解决问题的新空间
  - 开放、动态和不确定场景下，通过预测形成解决问题的决策
  - 打开了软件系统解决问题的空间：AI（模型）预测 + 可信性判断 = 决策
    - 解决涉及高维对象、非确定、以及逻辑规则难以描述的问题
- 带来了软件可信保障的新挑战
  - AI（模型）预测若不经过可信性判断直接用于决策，必定存在风险
  - 预测性算法需要可信性判断
    - 对模型（预测准确度）进行可信性判断，从而使得模型预测结果可以直接用于决策
      - 相比于人工基于逻辑设计的算法，对机器学习数据训练的模型进行测试使其可以直接用于决策，成本很高，比如端到端的自动驾驶
    - 对模型输出的预测结果进行可行性判断，然后用于决策

智能化软件：AI（模型）预测 + 可信性判断 = 决策

- 确定的符号计算与非确定的概率计算相融合
  - 人工基于逻辑设计的算法 + 机器学习数据训练的模型
    - 采用AI预测+可信性判断=决策的方式解决开放、动态和不确定场景下的复杂应用问题
      - 类似于人在开放、动态和不确定场景下完成决策性任务
      - 先预测，再基于预测结合逻辑（知识）推理判断形成决策；预测 + 逻辑推理判断 = 决策
    - 正处于快速增长的发展态势，对满足国防和经济建设的智能化应用领域需求、提高关键软件技术创新和供给能力、抢占信息技术前沿领域的战略制高点都具有十分重要的意义
  - 软件系统的复杂性进一步加剧
    - 非确定概率计算缺乏精确度
    - 机器学习模型具有不可解释性和内生安全隐患
    - 人工基于逻辑设计的算法和机器学习数据训练的模型交互协同、组合嵌套、深度融合

智能化软件：关键科学问题

- 如何使得智能化软件满足需求、摆脱缺陷？
  - 如何融合机器学习模型预测形成满足软件需求的决策？
    - 机器学习模型输出的是缺乏准确度的预测性结果（预测性判别、预测性生成）并且缺失可信性判断，直接用于决策必然存在失效风险
      - 如何对机器学习模型进行可信性判断？
      - 如何对机器学模型的预测结果进行可信性判断？如何综合预测结果和逻辑推理形成决策？
  - 如何有效控制机器学习模型缺陷导致的系统风险？
    - 机器学习模型缺陷所产生的影响在系统内传播和扩散将可能导致系统失效和安全风险
      - 如何揭示机器学习模型缺陷的发生规律、有效控制模型失效导致的系统风险
    - 在软件这个更大、更实际的空间解决可信人工智能难题

人工智能正在打开更大的空间

- AI预测+可信性判断=决策似乎正在成为解决所有问题的新途径
  - 可信性判断的成本决定了这种途径解决问题（落地）的可行性
    - Hard to solve，easy to verify，Asymmetry of Verification，Jason Wei
  - AI for Science √
    - 科学探索和发现： 人工预测（猜）+ 实验验证
    - AI预测 + 可信判断（实验验证）=决策：AI预测代替人工猜测，可以大幅提高效率
  - AI for MATH （Formal Methods，Theorem Proving） √
    - AI预测生成数学定理证明，定理证明器自动验证（可信性判断成本很低）
  - AI for Software Engineering ？
    - Hard to solve，hard to verify

大语言模型：基于自然语言交互的软件工具

- 通过深度学习技术突破、在汇聚丰富知识的基础上形成强大自然语言生成能力
  - 高效地利用互联网上和图书资料中积累的海量文本内容进行训练
  - 接收文本序列作为输入，产生关于下一个词的概率分布作为输出
  - 可以针对自然语言提问生成预测性内容回答
- 通过类似自然语言的处理方式汇聚积累了丰富的软件开发与演化知识
  - 通过类似自然语言的处理方式，采用了大量的程序语言相关语料进行训练
- 基于自然语言交互的人机协同软件开发与演化工具
  - 通过自然语言与软件开发与演化人员进行交互
  - 针对给定的软件开发与演化任务生成预测性内容
  - 所生成的预测性内容涉及软件开发与演化各个阶段
    - 软件需求、设计、编码、测试、调试及维护等
    - 对软件开发与演化人员完成相应任务可以提供不同程度的帮助

大语言模型（简称大模型）：预测性软件工具

- 基于自然语言交互的人机协同软件开发与演化工具
  - 基于自然语言的人机交互
    - 拓展了人机协同的工作空间，提高了人机协同的效率和灵活性
  - 针对软件开发与演化任务的预测性内容生成
    - 可以对软件开发与演化提供一定程度的支持和帮助
  - 生成的预测性内容缺失可信性判断，需要经过分析、理解和确认后才能加以使用
    - 基于概率与统计原理和训练数据所形成的数学模型
    - 具有不可解释性和内生不确定性
- 大模型打开了解决软件开发与演化问题的空间：大模型预测 + 可信性判断 = 决策
  - 软件开发与演化是决策性任务，大模型生成的预测性结果需要可信性判断才能形成决策
  - 由于软件是人类生产的最复杂制品，所以可信性判断的成本很高

大模型：打开从需求到代码的空间

- 大模型：支持从自然语言需求生成预测性代码
  - 软件开发与演化：将自然语言描述的需求转换为满足需求的程序代码
    - 不可判定的计算问题
  - 大模型生成的预测性代码不一定满足需求
- 相比于人工编程，大模型生成的代码缺失程序语义逻辑可信的基本保障
  - 程序代码：基于程序语义因果关系的逻辑制品
  - 人工编程：直接根据程序语义因果关系编写代码
    - 编程过程对应基于程序语义逻辑的非形式化证明过程
    - 程序语义逻辑可信得到基本保障
  - 大模型生成代码：通过关联（next token）关系近似和逼近因果，生成预测性代码
    - 难以判断是否达到因果、何时达到因果、或逼近因果到什么程度
    - 缺失程序语义逻辑可信的基本保障

从预测性代码到满足需求的程序代码

- 大模型放弃可信约束条件生成预测性代码
  - 打开了解决问题的新空间：大模型预测 + 可信性判断 = 决策
  - 如何基于大模型的预测性代码得到满足需求的程序代码？
    - 如何对大模型生成的预测性代码进行可信性判断？
- 通过测试手段对大模型（预测准确度）进行可信性判断？
  - 测试成本很高，现实不可接受
  - AI 测试数据集上的刷榜，离软件系统可信保障要求相差甚远
  - 大模型预测结果直接用于决策（直接使用预测性代码）风险很大

如何对大模型生成预测性代码进行可信性判断

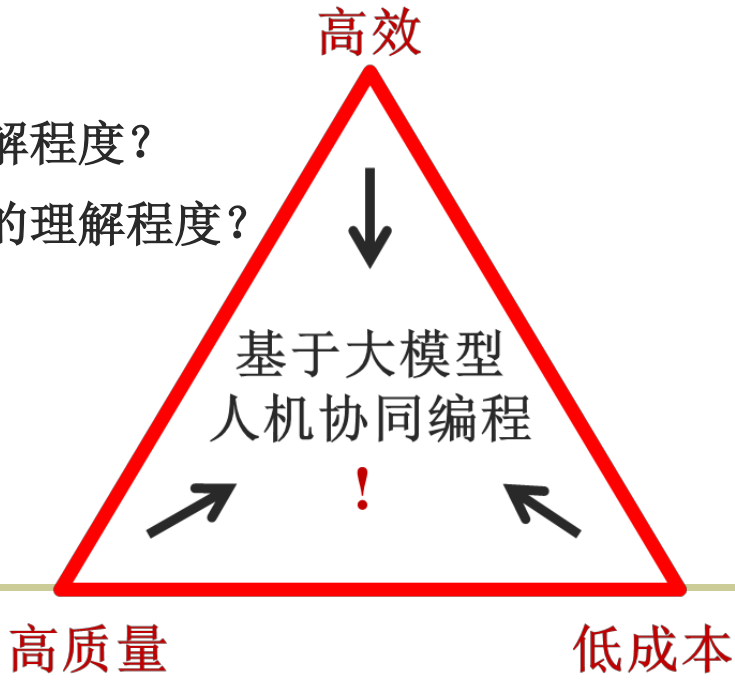
- 通过测试手段对大模型生成的预测性代码做出可信性判断？
  - 人工编程+测试，是目前工业界软件可信保障的主要途径
    - 人工编程 = 人基于程序语义和逻辑编写代码 + 人基于程序语义和逻辑分析确认代码（非形式化证明）
    - 人工编程的可信度是软件可信的重要基础
    - 即使基于人工编程的可信基础，软件测试成本仍然很高
  - 通过测试难以判断大模型生成代码的可信度是否达到人工编码的可信度
    - 程序是逻辑制品，基于程序语义的逻辑可信性需要得到基本保障
    - 测试难以达到基于程序语义和逻辑的非形式化证明效果
    - 通过测试难以保障基于程序语义的逻辑可信性
  - 可信度（预测性代码 + 测试 ）＜ 可信度（人工编程 + 测试）
    - 人工基于算法逻辑和程序语义编码，蕴含基于逻辑的非形式化证明，程序逻辑可信性得到基本保障
    - 大模型生成预测性代码，基于概率关联关系而非基于逻辑和语义因果，缺失程序逻辑可信性的基本保障

人工可信性判断：基于大模型的人机协同编程

- 基于大模型的人机协同编程（大模型预测+人工可信性判断）
  - 编程人员引导大模型生成预测性代码
  - 编程人员对预测性代码做出可信性判断和决策
    - 基于程序语义和逻辑分析、理解、修改和确认预测性代码（基于逻辑的非形式化证明）
  - 编程人员完成其它相关的可信保障活动
    - 静态分析、动态测试、形式化验证等
- 人工编程
  - 编程人员基于程序语义和逻辑编写代码和文档
  - 编程人员分析与理解代码、修改和确认代码（与编写代码融合一体）
    - 基于逻辑的非形式化证明，支持编程人员对所写代码做出可信性判断并承担相应的风险责任
  - 编程人员完成其它相关的可信保障任务
    - 静态分析、动态测试、形式化验证等

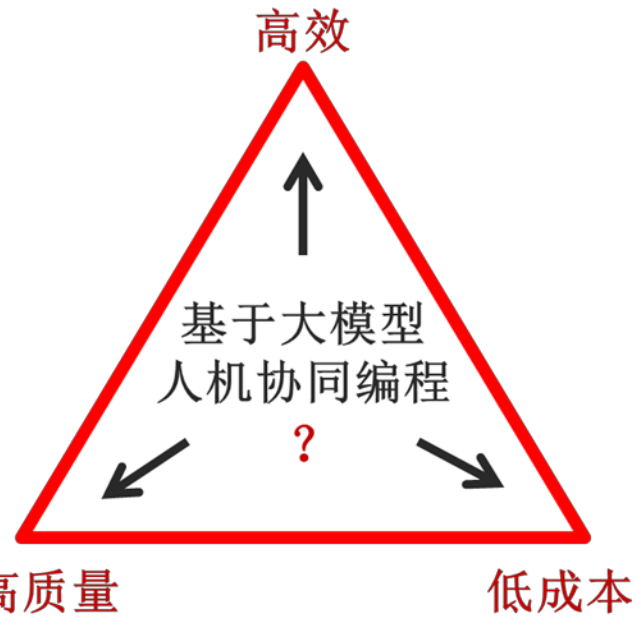
基于大模型的人机协同编程

- 人工编程
  - （程序员基于程序语义逻辑编写代码 + 理解确认代码） + 可信保障活动
- 基于大模型的人机协同编程
  - （大模型生成代码 +程序员理解确认代码） + 可信保障活动
- 可信度（人机协同编程）＝ 可信度（人工编程）？
  - 编程人员理解代码的难度通常高于自己写代码的难度
    - 程序员对大模型生成代码理解程度如何达到如同自己写代码的理解程度？
    - 如何判断程序员对大模型生成代码理解程度达到如同自己写代码的理解程度？
  - 人性所致的程序员松懈和偷懒
    - 不认真分析与理解预测性代码、直接提交预测性代码应付差事等
- 效率（人机协同编程）≥ 效率（人工编程）？
  - 理解代码需要比写代码更强的专业能力
  - 理解代码的效率未必高于写代码的效率



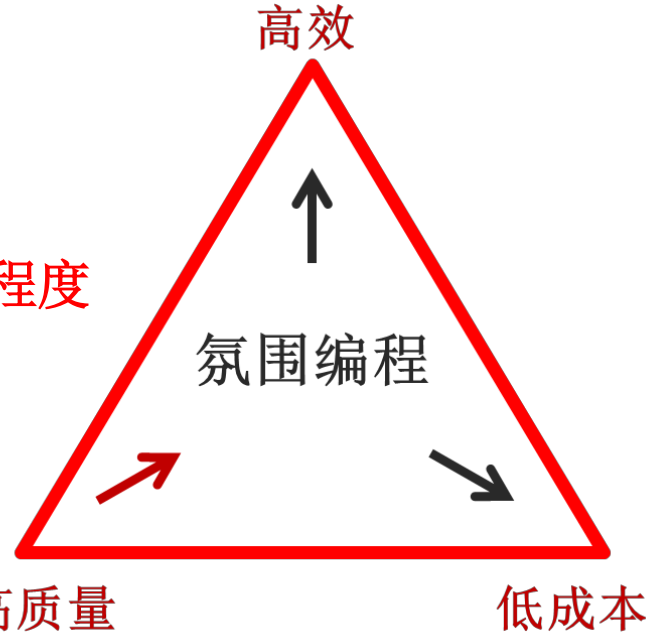
基于大模型的人机协同编程：目标与挑战

- 基于大模型的人机协同编程目标
  - 可信度（人机协同编程）＝ 可信度（人工编程）
  - 效率（人机协同编程）≥ 效率（人工编程）
- 问题与挑战
  - 编程人员理解代码的难度通常高于自己写代码的难度
    - 程序员对大模型生成代码理解程度如何达到如同自己写代码的理解程度？
    - 如何判断程序员对大模型生成代码理解程度达到如同自己写代码的理解程度？
  - 人性所致的程序员松懈和偷懒
  - 理解代码需要比写代码更强的专业能力，理解代码的效率未必高于写代码



基于大模型的氛围编程（vibe coding）

- 降低基于大模型的人机协同编程目标
  - 可信度（人机协同编程） ＜ 可信度（人工编程）
    - 程序员对大模型生成代码的理解程度低于如同自己写代码的理解程度
  - 效率（人机协同编程） ≥ 效率（人工编程）
- 氛围编程（vibe coding）
  - 人 + 大模型 + Agents
    - Agents：部分代替人做可信性判断，调用工具做部分可信性判断
    - Cursor，Claude Code，GitHub Copilot，Trae， Qwen3-Coder，GPT-5，……
  - 放松可信约束条件（质量降低），代码生产力得到极大提高（代码白菜价！）
  - 以降低质量的代价代替人工编程从而提高软件开发效率，有什么用？



大模型：打开软件创造与制造过程分离的空间

- 软件创造与制造过程分离
  - 软件开发过程：软件创造性开发 + 软件制造性开发
    - 软件创造：规约软件需求（分析、认识、理解和确认软件需求）
    - 软件制造：根据软件需求规约构造满足需求的程序代码
  - 软件创造性开发（原型式开发）
    - 大模型快速生成预测性代码，极大提高了软件原型开发的效率、降低了成本
    - 人 + 大模型 + Agents，快速构造软件原型，将需求创意快速转化为代码
    - 用于需求分析确认、设计仿真验证、“即用即抛、自用”、其它产品创造性活动
    - 创造性开发可以不精确，可以不达到人工编程的可信度
  - 软件制造性开发（重构式开发）
    - 针对确认的软件需求规约，以人为主导开发高质量的软件系统
    - 制造性开发必须精确，必须至少达到人工编程的可信度

大模型：提高了人类对软件开发与演化过程的掌控力

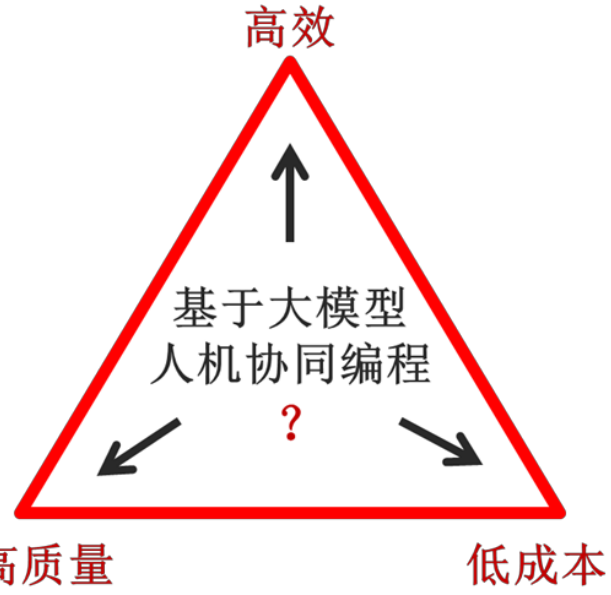
- 大模型通过建立自然语言与程序语言之间的非精确关联关系，拉近了软件需求与代码之间的距离
- 根据自然语言需求描述，快速、非精确地构建软件原型系统以分析和确认需求，打开了软件创造与制造过程分离的空间
- 降低了软件开发过程的不确定性、有效提高了人类对软件开发过程的掌控能力

发展趋势：软件创造更准、软件制造更快

- 如何使得软件创造在高效的基础上更精准？
  - 基于大模型的氛围编程（效率优先，在快的基础上更准）
    - 人 + 大模型 + Agents；将出现一批专门从事软件创造性开发的企业
    - 可信度（人机协同编程）＜ 可信度（人工编程），效率（人机协同编程）≥ 效率（人工编程）
    - 支撑工具与平台：Cursor，Claude Code，GitHub Copilot，Trae，Qwen3-Coder，GPT-5，……
  - 困难与挑战
    - 专业能力更强、更全面的开发者（超级个体）才能驾驭AI工具高效地创造出更精准的软件原型
- 如何使得软件制造在精准的基础上更高效？
  - 基于大模型的人机协同编程（质量优先，在准的基础上更快）
    - 以人为主，大模型辅助；经典软件工程原理、方法和技术在制造过程中将进一步发挥作用
    - 可信度（人机协同编程）＝ 可信度（人工编程），效率（人机协同编程）≥ 效率（人工编程）
    - 方法学、支撑工具与平台？
  - 困难和挑战？

软件制造更高效的困难与挑战

- 基于大模型的人机协同编程
  - 大模型生成代码 +程序员理解确认代码 + 可信保障活动
    - 可信度（人机协同编程）＝ 可信度（人工编程）
    - 效率（人机协同编程） ≥ 效率（人工编程）



- 困难与挑战
  - 编程人员理解代码的难度通常高于自己写代码的难度
    - 程序员对大模型生成代码理解程度如何达到如同自己写代码的理解程度？
    - 如何判断程序员对大模型生成代码理解程度达到如同自己写代码的理解程度？
  - 人性所致的程序员松懈和偷懒
    - 程序员不论高级还是低级都会松懈和偷懒，需要方法学引导、工具和规范进行有效的约束和管理
  - 理解代码需要比写代码更强的专业能力，理解代码的效率未必高于写代码的效率

软件企业如何应对挑战

- 实施软件开发和演化的智能化转型
  - 软件创造与制造过程分离、提升软件开发过程掌控能力，实现降本增效
  - 35岁以上程序员从事软件创造性开发，35岁以下程序员从事软件制造性开发
  - 按专业能力等级制定使用大模型工具的规范（初级少用，高级多用）
- 在软件制造过程中保留人工编程流程和岗位
  - 无论软件创造还是制造，都需要有更强专业能力的程序员
  - 通过大量编程实践才能够培养出驾驭大模型工具的程序员
  - 建议30岁以下程序员禁用大模型编程，不会写代码肯定读不好代码
- 大模型代替不了程序员，不要简单淘汰初级程序员
  - 有程序员、没有大模型，软件企业可以生存；有大模型，没有程序员，软件企业不能生存
  - 既有程序员，又有大模型，软件企业才可以生存得更好

软件工程专业人才培养如何应对挑战

- 软件开发与演化人员具备更强专业能力才能驾驭大模型工具
  - 更强的专业能力：能够对大模型生成的预测性内容进行可信性判断
  - 大模型（预测性软件工具）只能使得强者更强，不能使得弱者变强
- 学生刚毕业不可能就是高级程序员，但应该具备成长为高级程序员的基础和潜力
  - 基本能力：程序能力、算法能力、系统能力、工程能力
  - 综合能力：解决问题能力，专业及相关知识学习能力，驾驭AI能力
  - 帮助学生建构自己的软件工程专业世界模型，使其能够自信地学习和判断，具备驾驭AI的能力
- 软件创造与制造过程分离，将大幅提升软件生产力
  - 软件创造效率提升将拓展出更大的软件需求空间，软件制造需要更多的专业人才
  - 本硕贯通培养准高级程序员

人工智能正在打开更大的空间：进一步思考

- AI预测+可信性判断，可能需要不断反馈和多次迭代才能形成决策
  - AI预测+可信性判断 + AI预测+可信性判断 + AI预测+可信性判断 + …… = 决策
    - 不断判断反馈和多次迭代即形成所谓的思维链、推理链、搜索链等，新维度Scaling Law
  - 可信性判断是形成各类思维、推理、搜索链的关键核心驱动
    - 基于逻辑规则的可信性判断：是否违反必要条件？是否达到充分条件？
      - 可实现自动化，但充分完备的逻辑规则难以构造，难以判断代码逻辑可信性
    - 基于模型预测的可信性判断：是否概率性违反必要条件？是否概率性达到充分条件？
      - Agents，强化学习，……
      - 预测判断预测、预测叠加预测，难以形成充分的可信性判断，难以判断代码逻辑可信性
    - 人工可信性判断：基于逻辑的非形式化证明
      - 代码逻辑可信性判断离不开人：读代码（对代码进行可信性判断）的成本不低于写代码的成本

结　语

- 软件是人类制造的最复杂制品，软件创造过程与制造过程相互融合、难以解耦
- 人工智能打开了解决问题的新空间：AI预测 + 可信性判断 = 决策
  - 智能化软件：融合确定的符号计算与非确定的概率计算
    - 人工基于逻辑设计的算法 + 机器学习数据训练的模型
    - 解决开放动态不确定场景下的问题
  - 大语言模型：打开了软件创造与制造过程分离的空间
    - 软件开发过程：软件创造性开发 + 软件制造性开发
    - 人+大模型+Agents，根据自然语言需求描述快速构建软件原型系统以分析和确认需求
    - 降低了软件开发过程的不确定性、有效提高了人类对软件开发过程的掌控能力
- 人工智能带来了软件可信保障的难题与挑战
  - AI预测 + 可信性判断 = 决策；软件工程问题：hard to solve, hard to verify
  - 驾驭人工智能需要更强的软件工程专业能力

问题？？？

谢谢！