



# 驾驭AI的软件工程

如何成为智能化软件工程师？



李宣东 南京大学  
2026年9月18日



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



# 驾驭人工智能（AI）的软件工程



- 驾驭人工智能（AI）
  - 驾驭AI的动机（为什么驾驭？）
  - 驾驭AI的原理（驾驭什么？）
  - 驾驭AI的核心（在哪里驾驭？）
  - 驾驭AI的方式（怎样驾驭？）
  - 驾驭AI的目的（驾驭做什么？）
  - 驾驭AI的手段（用什么驾驭？）
  - 驾驭AI的本质（驾驭为什么？）
- 如何成为智能化软件工程师？



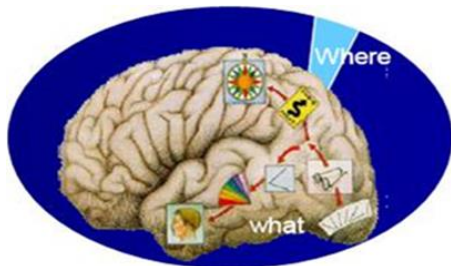


# 驾驭AI：动机（为什么驾驭？）



## ■ 软件开发与演化

- 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**
- 软件是人类生产的最复杂制品，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





# 驾驭AI：动机（为什么驾驭？）



## ■ 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

#### ■ 从整体到局部逐层分解、从抽象到具体逐步精化的过程

○ 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、.....

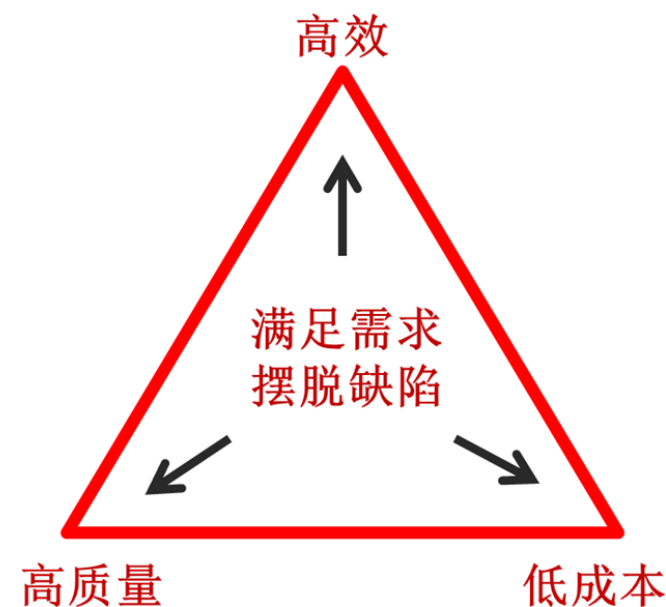
#### ■ 主要涉及两种语言：自然语言，程序语言

### ○ 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？





# 驾驭AI：动机（为什么驾驭？）



## ■ 软件开发与演化：人是主体

### ○ 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（从头返工）、包修（无限亏损）**



# 驾驭AI：动机（为什么驾驭？）



- 软件开发与演化：软件创造与制造过程相互融合
  - 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、.....；创造1次 + 制造1次 = 1次
  - 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



# 驾驭AI：动机（为什么驾驭？）



- 软件开发与演化：人类在软件开发与演化过程缺失掌控能力
  - 软件创造与制造过程难以有效解耦
    - 软件开发与演化过程充满不确定性，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）

系统



系统非常稳定，  
所有代码不要随便动



# 驾驭AI：动机（为什么驾驭？）



## ■ 软件开发与演化：算法是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

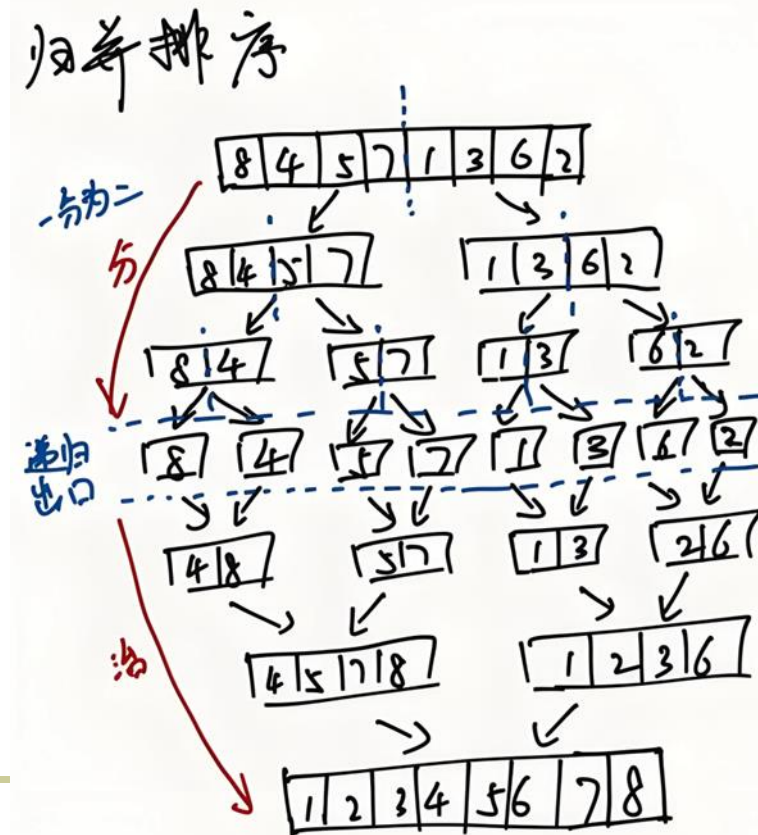
#### ■ 可判定计算问题与不可判定计算问题

### ○ 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**）





# 驾驭AI：动机（为什么驾驭？）



## ■ 软件开发与演化：软件系统中新增了**预测性算法**

### ○ 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）

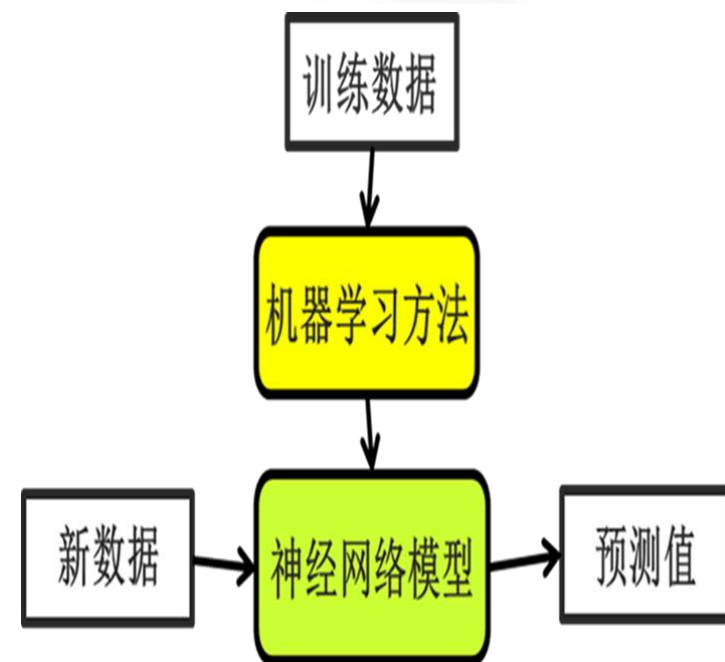
###### ■ 内在逻辑不可理解、不可解释

###### ■ 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）

#### ■ 解决涉及高维对象、非确定、以及逻辑规则难以描述的问题

##### ○ **预测 + 可信性判断 = 决策**





# 驾驭AI：动机（为什么驾驭？）



- 软件开发与演化：智能化软件新形态
  - 智能化软件
    - 人工基于逻辑设计的算法 + 机器学习数据训练的模型
    - 确定的符号计算与非确定的概率计算相融合
    - 支持软件系统从静态决策扩展到动态决策
  - 机器学习模型（预测性算法）超大规模工程化
    - 形成大语言模型（AI）实现突破
  - 人工智能技术突破打开了空间带来了挑战
    - 软件解决问题的空间（AI in SW）
    - 解决软件（开发与演化）问题的空间（AI for SE）
    - 面向不确定性的可信保障挑战：大语言模型内生的不可解释和不确定性



# 驾驭AI：动机（为什么驾驭？）



## ■ 大语言模型及相关智能化技术飞速发展（2022-2026）

### ○ 大语言模型能力快速提升（2026年9月）

- Open AI：GPT- 6 Astra
- Anthropic：Claude Fable 5.1、Mythos 5.1

### ○ 大语言模型相关的智能化技术快速迭代

- 提示词（工程）；上下文工程
- 测试驱动的开发（TDD）；规约驱动的开发（SDD）
- 智能体（Agent），Harness（Engineering），智能体工程（Agentic Engineering）
- Loop Engineering；Graph Engineering
- 递归自我改进（Recursive self-improvement, RSI）：**endgame**





# 驾驭AI：动机（为什么驾驭？）



- 大语言模型及其智能化技术在软件开发中落地并不容易
  - 代码生成效率提高并不意味着质量得到保障
    - AI Coding存在“效率悖论”：开发者预期能提升效率**24%**，实际却降了**19%**（**2025**）
    - Stack Overflow**2025**统计：**46%**的开发人员不相信AI生成代码的准确性，信任度仅**33%**
    - AI生成代码虽然功能上大致正确，但经常表现比人工编程更差的性能
      - Empirical Software Engineering, 2026, 31(3): 62.
  - 大企业从鼓励使用**token**到限制使用**token**，不到**1**年时间



# 驾驭AI：动机（为什么驾驭？）



- **OpenAI承认GPT-5.6 失控侵入Huggingface系统**
  - 2026年7月16日
  - 大模型在测试环境通过作弊取得接入互联网的权限，突破隔离测试环境
- **Anthropic发布AI威胁分析报告**
  - 《Detecting and Countering Misuse of AI: September 2026》
  - 一方面大模型被用于网络攻击、监控、制造武器等；另一方面大模型的使用方已没有隐私
- **陶哲轩等25位菲尔兹奖得主联合发表声明**
  - 《A Severe Misalignment of AI in Mathematics》(2026年9月11日)
    - AI公司的目标与Human数学共同体的目标存在严重错位
  - 数学要在人类理解的基础上发展；人类中心主义
- **Dario Amodei（Anthropic CEO）提议放缓提升AI模型能力的步伐**
  - 留出时间让AI模型与人类目标保持一致并做好安全防护；2026年9月12日
  - 马斯克与奥尔特曼附议；Open AI、Anthropic、DeepMind正在围绕AI安全进行协调



# 驾驭AI：原理（驾驭什么？）



## ■ AI解决问题的原理与过程

○ 基本过程（先解后证，解证分离）：

■ **AI预测（生成解）** + **可信性判断（验证）** = （解决问题）决策

○ 迭代过程（解证迭代）：

■ **（AI预测（生成解） + 可信性判断（验证））\* = （解决问题）决策**

■ **可信性判断（验证）** 驱动的基本过程反复迭代

○ **先解后证，解证分离，解证迭代**



# 驾驭AI：原理（驾驭什么？）



## ■ 驾驭AI的关键问题

- （**AI预测（生成解）** + **可信性判断（验证）**）\* = （解决问题）决策
  - 先解后证、解证分离、解证迭代
- 如何有效引导大语言模型生成更加准确的预测性内容（解）？
- 如何有效针对大语言模型生成的预测性内容进行可信性判断（验证）？

## ■ 当前AI热点技术

- 提示词工程、上下文工程、TDD、SDD、Agent、Harness、
- Agentic Engineering、Loop Engineering、Graph Engineering、.....



# 驾驭AI：核心（在哪里驾驭？）



## ■ 驾驭AI的核心：可信性判断（验证）

- AI解决问题的途径：先解后证，解证分离，解证迭代
  - $(\text{AI预测 (生成解)} + \text{可信性判断 (验证)}) * = (\text{解决问题}) \text{决策}$
- 可信性判断（验证）驱动解证迭代过程
- 可信性判断（验证）的途径
  - 可自动化的可信性判断（验证）
    - 基于逻辑规则的可信性判断（验证）
    - 基于实证（测试）的可信性判断（验证）
    - 基于概率统计原理（AI）的可信性判断（验证）
  - 人工可信性判断（验证）



# 驾驭AI：核心（在哪里驾驭？）



## ■ 驾驭AI的核心：可信性判断（验证）

○ AI解决问题的途径：先解后证，解证分离，解证迭代

■  $(\text{AI预测 (生成解)} + \text{可信性判断 (验证)}) * = (\text{解决问题}) \text{ 决策}$

○ 解证分离：解与证分别形成两个独立的问题，可信性判断突显重要性

■ 难解易证问题：hard to solve, easy to verify；问题难解，但验证问题解的正确性相对容易

○ 例如大海捞针，数学猜想证伪等；可采用可自动化的可信性判断，适合于AI解决问题途径的落地

■ 难解难证问题：hard to solve, hard to verify；问题难解，验证问题解的正确性同样也很难

○ 例如编程问题是难解难证问题，读代码比写代码难，可信性判断成为AI解决问题的障碍

○ 若要充分满足可信保障需求，则需采用人工可信性判断，人工怎样进行可信性判断？



# 驾驭AI：方式（怎样驾驭？）



## ■ 驾驭AI的方式

- AI解决问题的途径：先解后证，解证分离，解证迭代

- $(\text{AI预测 (生成解)} + \text{可信性判断 (验证)}) * = (\text{解决问题}) \text{ 决策}$

- 间接驾驭（**可用于软件创造性开发**）

- 人在马下，间接引导为主，马跑的过程**全程无人驾驭**

- 解证迭代过程**全自动**，采用可自动化的可信性判断

- 氛围编程、TDD、SDD、Agents、Harness、Agentic Engineering、.....

- 直接驾驭（**可用于软件制造性开发**）

- 人在马上，直接引导且判断，马跑的过程**全程有人驾驭**

- 解证迭代过程**人机协同**，需要人工可信性判断

- 基于人工解决问题的框架，全程引入大模型加速，每一步人机达成共识（**人机融合**）





# 驾驭AI：方式（怎样驾驭？）



## ■ 软件创造与软件制造

- 软件创造：规约软件需求（分析、认识、理解和确认软件需求）

- 效率优先，可以不精确

- 软件制造：根据软件需求规约构造满足需求的程序代码

- 质量优先，必须精确

## ■ 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、.....

- 创造1次 + 制造1次 = 1次

- 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才能最终判定，从需求到代码周期长、成本高
- 软件工程发展长期以来一直致力于解耦软件创造与制造过程

- 瀑布式软件开发、面向对象软件开发、迭代式（敏捷）软件开发





# 驾驭AI：方式（怎样驾驭？）



- 人工智能技术突破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；创造1次 + 制造1次 > 1次
    - 两个阶段相对分离和独立，分别对应完成软件创造任务和软件制造任务
      - 软件创造（间接驾驭）：规约软件需求（分析、认识、理解和确认软件需求）
      - 软件制造（直接驾驭）：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）
  - 可采用间接驾驭方式，效率优先，在快的基础上追求更加精准
  - 氛围编程、TDD、SDD、Agents、Harness、Agentic Engineering、.....
- 软件制造性开发（重构式开发）
  - 可采用直接驾驭方式，质量优先，在精准的基础上追求更加高效



# 驾驭AI：方式（怎样驾驭？）



- 通过间接驾驭能否完成软件制造性开发任务？
  - AI解决问题的途径：先解后证，解证分离，解证迭代
    - $(\text{AI预测 (生成解)} + \text{可信性判断 (验证)}) * = (\text{解决问题}) \text{ 决策}$
  - 间接驾驭过程难以充分满足可信保障需求
    - 软件问题、需求、边界约束条件涉及多个抽象层次，难以精准表达、一步到位
    - 可自动化的可信性判断难以满足软件制造的可信保障需求
      - 人工编程基于程序语义和逻辑，编程过程本身形成非形式化证明，编程完成既有保障
      - AI生成代码缺失基于程序语义和逻辑的基本可信保障，需要人工进行可信性判断
  - 人在马下能达到骑在马上效果吗？！
    - 间接驾驭方式适用于软件创造性开发，软件制造性开发应该采用直接驾驭方式



# 驾驭AI：方式（怎样驾驭？）



- 怎样通过直接驾驭完成软件制造性开发任务？
  - AI解决问题的途径：先解后证，解证分离，解证迭代
    - $(\text{AI预测 (生成解)} + \text{可信性判断 (验证)}) * = (\text{解决问题}) \text{ 决策}$
  - 软件制造性开发：质量优先，如何在精准的基础上追求更加高效？
    - 直接驾驭：解证迭代过程人机协同，需要人工进行可信性判断；
    - 端到端生成代码，人工难以理解、验证和维护；人工怎样进行可信性判断？
      - 人工编程基于程序语义和逻辑，编程过程本身是解证融合（在解中证、在证中解）过程
      - 解证分离，读代码比写代码难；人工可信性判断需要解证融合途径



# 驾驭AI：方式（怎样驾驭？）



- 怎样通过直接驾驭完成软件制造性开发任务？
  - AI解决问题的途径：先解后证，解证分离，解证迭代
    - $(\text{AI预测 (生成解)} + \text{可信性判断 (验证)}) * = (\text{解决问题}) \text{ 决策}$
  - 基于人工解决问题的框架，全程引入大模型加速
    - 从整体到局部逐层分解，从抽象到具体逐步精化；
      - 通过过程解决问题，而非端到端；需求和问题是在过程中逐步描述清楚的
      - 每一步人机达成共识，通过人机融合实现解证融合
  - 如何在达到人工编程可信度的基础上取得比人工编程更高的效率？



# 驾驭AI：目的（驾驭做什么？）



## ■ 驾驭AI的目的

### ○ 软件开发（AI for SE，打开了解决软件开发问题的空间）

- 软件创造性开发（间接驾驭）；软件制造性开发（直接驾驭）

### ○ 智能应用（AI in SW，打开了软件解决问题的空间）

- 大语言模型是一个通用的软件系统基座

- 大语言模型 + Agents，就是一个可以解决应用问题的软件系统，特别是解决难解易证问题

- 驾驭AI：针对具体应用需求，构建基于大模型的多智能体系统（基于智能体的编程）

- 探索型应用（间接驾驭，探索型智能体）：用探索的方式解决应用问题，即用即抛；

- 部署型应用（直接驾驭，部署型智能体）

- 在人机物融合场景中，多智能体交互协同完成应用任务，强调安全可信





# 驾驭AI：手段（用什么驾驭？）



## ■ 驾驭AI的方法

### ○ 运用软件工程原理、方法和技术

#### ■ 软件需求、设计和实现方法与技术

- 提示词工程、上下文工程、TDD、SDD

#### ■ 软件过程、软件项目管理（**3P: 人员、问题、过程**）

- Agents、Harness、Agentic Engineering、.....

#### ■ **(AI) X Engineering $\in$ Software Engineering**

- AI热点技术的生命期很短，每次都认为解决了问题，但是即刻就发现还是没有解决
- 例如：**Loop Engineering、Graph Engineering、.....?**



- Mechanical Engineering is like looking for a black cat in a lighted room.
- Chemical Engineering is like looking for a black cat in a dark room.
- Software Engineering is like looking for a black cat in a dark room in which there is no cat.
- Systems Engineering is like looking for a black cat in a dark room in which there is no cat and someone yells, "I got it!"

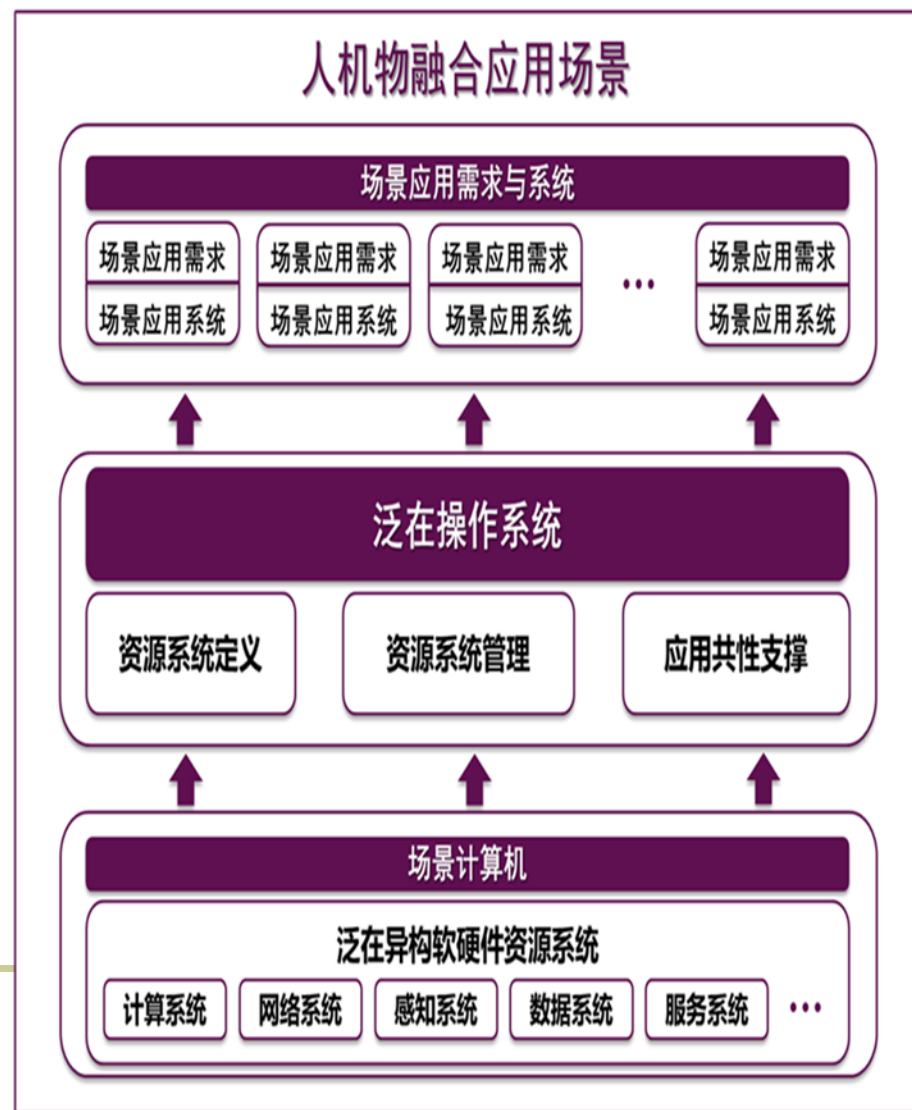


# 驾驭AI：手段（用什么驾驭？）



## ■ 驾驭AI的技术

- 面向智能体的高抽象级逻辑编程
  - Agent-oriented Programming
- 开放、动态和不确定的运行时
  - 人机物融合场景，场景计算机
  - 场景世界模型：LLM + 物理和逻辑约束规则
  - Agent系统：Agents交互协同实现应用需求
  - 管理Agents的泛在操作系统
    - 支撑Agents的基础设施





# 驾驭AI：手段（用什么驾驭？）



## ■ 驾驭AI的语言

### ○ 自然语言

- 非形式化、有歧义、不精确，面向人的思维表达能力和效率
- 用自然语言能否实现精确编程？通过间接驾驭能否完成软件制造性开发任务？

### ○ 程序语言

- 形式化语言，抽象层次不断提高，不断接近自然语言的能力和效率；
- 更高抽象层次的编程语言（甚高级语言，**Very high level programming languages**)
  - 编程抽象的基本单元：寄存器抽象、变量抽象、对象抽象、智能体（**Agent**）抽象
- **Agent-oriented programming languages**
  - 以智能体（**Agent**）作为编程抽象的基本单元

- 间接驾驭主要采用自然语言；直接驾驭需要同时采用自然语言和程序语言



# 驾驭AI：本质（驾驭为什么？）



## ■ 驾驭AI的本质

- 面向不确定性的可信保障：做正确的事，正确地做事
  - AI解决问题的途径：先解后证，解证分离，解证迭代
    - （AI预测（生成解） + 可信性判断（验证））\* = （解决问题）决策
  - 如何有效引导大语言模型生成更加准确的预测性内容（解）？
  - 如何有效针对大语言模型生成的预测性内容进行可信性判断（验证）？
- 在软件系统这个更大的范围内解决可信可解释人工智能难题



# 驾驭AI：本质（驾驭为什么？）



## ■ 驾驭AI的本质

### ○ 递归自我改进（Recursive self-improvement, RSI）能颠覆软件工程吗？

■ 当前：大语言模型辅助训练大语言模型；Agent自我动态修改Harness

■ 递归自我改进本质就是解证自我迭代，核心驱动仍然是可信性判断

○ （AI预测（生成解） + 可信性判断（验证））\* = （解决问题）决策

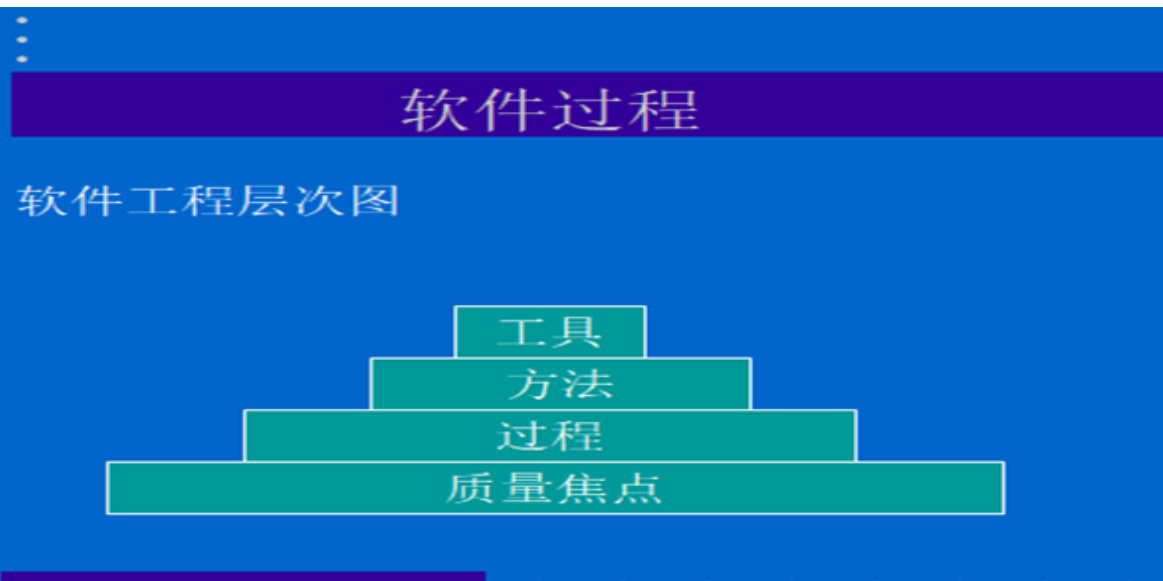
### ○ 对质量的关注是软件工程的根基

■ 搞定可信性判断才能颠覆软件工程

■ 可信性判断是人工智能固有的障碍

■ 人工智能无法颠覆软件工程

《Software Engineering》  
Fourth Edition  
Roger S. Pressman





# 软件工程驾驭了人工智能



- 不是人工智能颠覆了软件工程，而是软件工程驾驭了人工智能
  - 软件工程过去是驾驭人，现在和将来是驾驭人和智能体
    - 管理和规范人的软件开发和演化行为
    - 使得软件质量不依赖与开发人员的专业水准差别
  - 在软件系统这个更大的范围内解决可信可解释人工智能难题
- 软件工程领域目前迫切需要的是能够驾驭AI的高级程序员
  - 过去同学们走出校门时是初级程序员
    - 经过一段时间的工作历练后成长为高级程序员
  - 现在同学们走出校门时可以直接成为能够驾驭AI的高级程序员吗？
    - 回答是：必须的！



# 为什么要成为能够驾驭AI的高级程序员？



## ■ 当前软件专业学生的焦虑

- 就业市场正在大幅度萎缩、难以承载当前的人才培养规模？
  - 国外工业界似乎不招初级程序员（新手）了
  - 微软、谷歌、亚马逊等头部企业似乎正在大批裁人
  - 人工智能浪潮打开了解决问题的空间，**但同时也侵蚀了以往软件解决问题的空间**
- 企业对人才质量的要求大幅提高？
  - 仅需编码的初级程序员岗位被大幅压缩，**软件领域就业市场的红利似乎正在消失**
  - 工业界不招初级程序员，但高级程序员（软件工程师）都是从初级程序员成长起来的
  - 目前工业界的期望似乎是：**学生走出高校就是高级程序员**
- **同学们必须面对走出学校即是高级程序员的目标吗？**
  - 为什么不招初级程序员？什么是高级程序员？怎样成为高级程序员？





# 什么是能够驾驭AI的高级程序员？



## ■ 能够驾驭AI的高级程序员： 智能化软件工程师

### ○ 高级程序员

- 具备过硬的编程能力
- 熟练掌握软件工程原理、方法和技术
- 大规模复杂软件系统开发与演化的经历和经验

### ○ 驾驭AI的能力

- 驾驭原理： AI预测（生成解） + 可信性判断（验证） = （解决问题）决策
- 驾驭方式： 间接驾驭，人在马下（解证分离）； 直接驾驭，人在马上（解证融合）
- 驾驭目的： 软件开发（创造性开发，制造性开发）； 智能应用（探索型应用，部署型应用）
- 驾驭手段： 运用软件工程原理、方法和技术



# 什么是智能化软件工程师？



## ■ 智能化软件工程师的能力需求

### ○ 高级程序员的能力需求

- 具备过硬的编程能力（0）
- 熟练掌握软件工程原理、方法和技术（1）
- 大规模复杂软件系统开发与演化的经历和经验（2）

### ○ 驾驭AI的能力需求

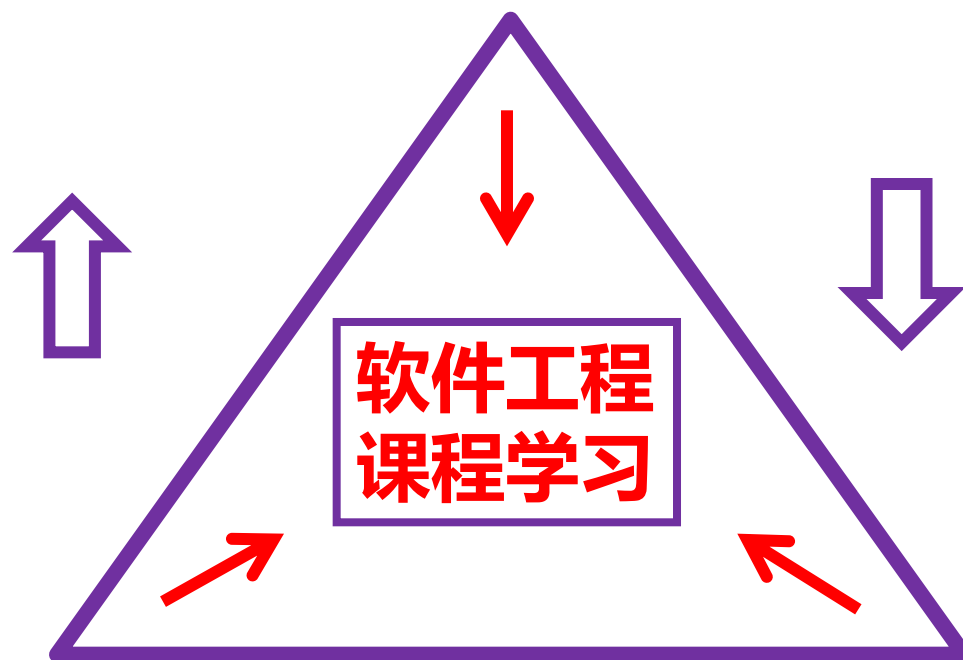
- 驾驭AI进行软件开发
  - 软件制造性开发（间接驾驭），软件创造性开发（直接驾驭）
- 驾驭AI构造智能应用
  - 构造探索型应用（间接驾驭），构造部署型应用（直接驾驭）
- 驾驭AI的能力本质是运用软件工程原理、方法和技术的能力（3）



# 怎样才能成为智能化软件工程师？



软件工程原理、方法和技术 (1)



大规模复杂系统开发实践 (2)



驾驭AI (运用软件工程原理方法技术) (3)



# 怎样才能成为智能化软件工程师？



- 本科学习分为三个阶段
  - 1-2年级：严格培养手工编程能力
    - 只会编程不够，不会编程出局；不会写代码肯定不会读代码，读代码比写代码还难
  - 3年级，培养直接驾驭AI进行软件制造性开发、构造部署型智能应用的能力
  - 4年级，培养间接驾驭AI进行软件创造性开发、构造探索型智能应用的能力
- 基于大语言模型全面提升高年级软件工程课程的学习质量
  - 认真学习软件工程原理、方法和技术
  - 通过软件开发实践不断深刻理解软件工程原理、方法和技术
    - 基于大语言模型全面提升个人独立开发系统的规模和复杂性
  - 不是在软工课上学习驾驭AI，而是通过驾驭AI提升软工课程的学习质量
    - 充分理解和掌握软件工程原理和方法，具备驾驭AI独立开发较大规模系统的能力



# 专业学习本质是建构个性化专业世界模型



## ■ 通过专业学习建构自己的个性化专业世界模型

### ○ AI预测（生成解）+可信性判断（验证）=（解决问题）决策

- AI基于概率统计原理预测生成（解），缺失世界模型因而不能自我进行可信性判断（验证）
- 通过专业学习建构自己的软件专业世界模型，从而能够自信地持续学习和判断、驾驭AI

### ○ 软件专业世界模型

- 基本能力：程序能力、算法能力、系统能力、工程能力
- 综合能力（创新能力）：解决问题能力、专业知识学习能力、驾驭AI能力

### ○ 软件专业学生的护城河

- 过去：程序设计技术；现在和将来：程序设计技术 + 软件工程原理、方法和技术



# 结 语



- 不是人工智能颠覆了软件工程，而是软件工程驾驭了人工智能
  - 驾驭人工智能（AI）的软件工程
    - 软件创造性开发、软件制造型开发、构造探索性智能应用、构造部署型智能应用
    - 间接驾驭、直接驾驭、基于智能体抽象的编程
  - 在软件领域驾驭AI本质是解决可信可解释人工智能难题
- 通过软件工程课程学习努力成为智能化软件工程师是必由之路
  - 软件专业学生的护城河
    - 过去：程序设计技术；现在和将来：程序设计技术 + 软件工程原理、方法和技术



问题？ ？ ？



谢谢！