智能化软件系统与工程

AI系统设计概述

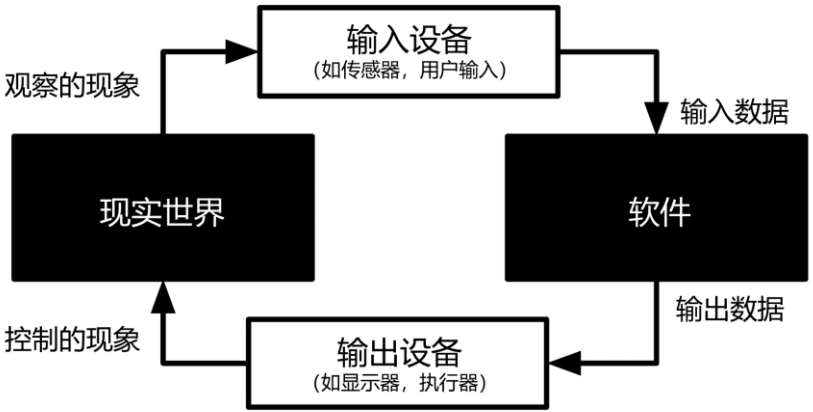
马 郓

人工智能研究院

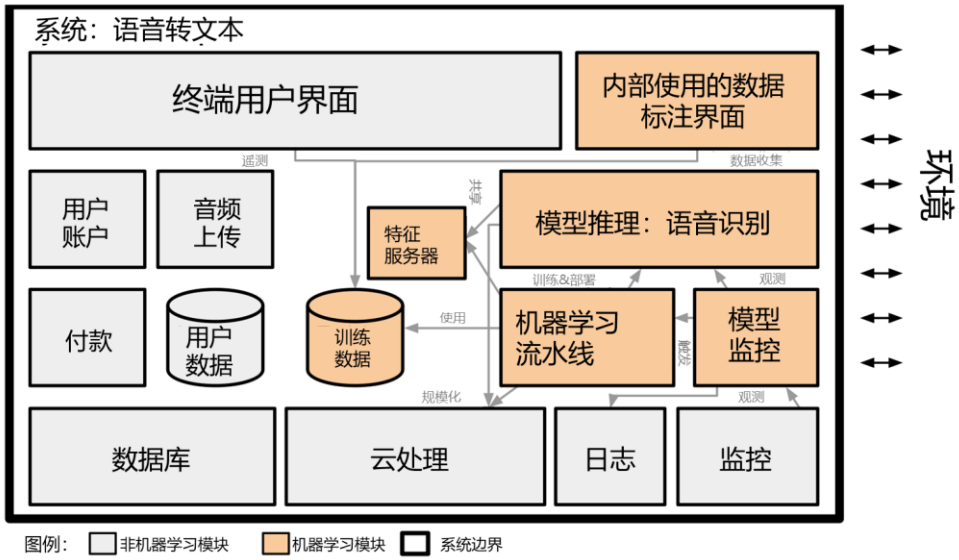
北京大学 PEKING UNIVERSITY

回顾：环境、系统、模型

系统是环境的一部分



AI模型是AI系统的一部分



系统需求（REQ）描述对环境的期望状态

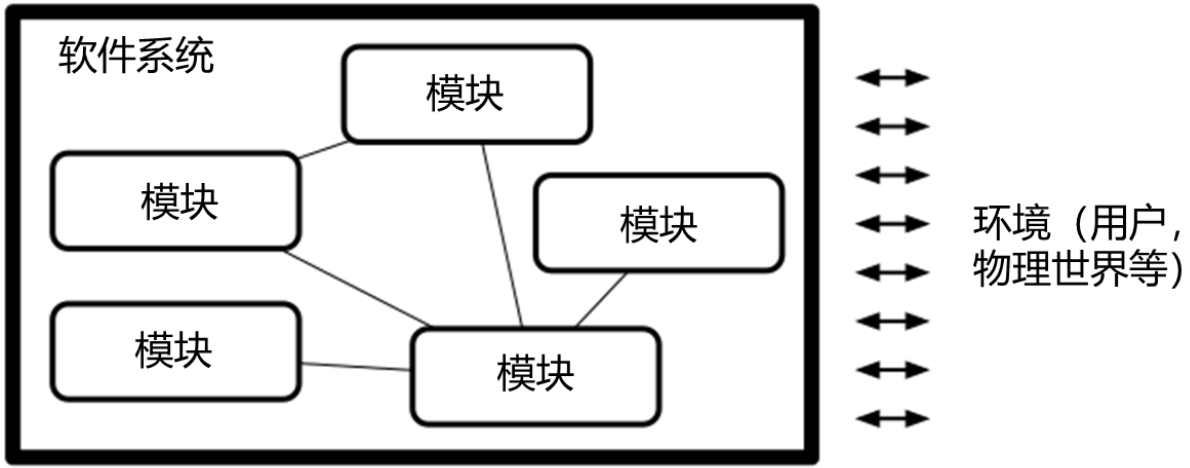
系统需求能够被实现的条件：

$$ASM \wedge SPEC \models REQ$$

$$IMPL \models SPEC$$

系统思维

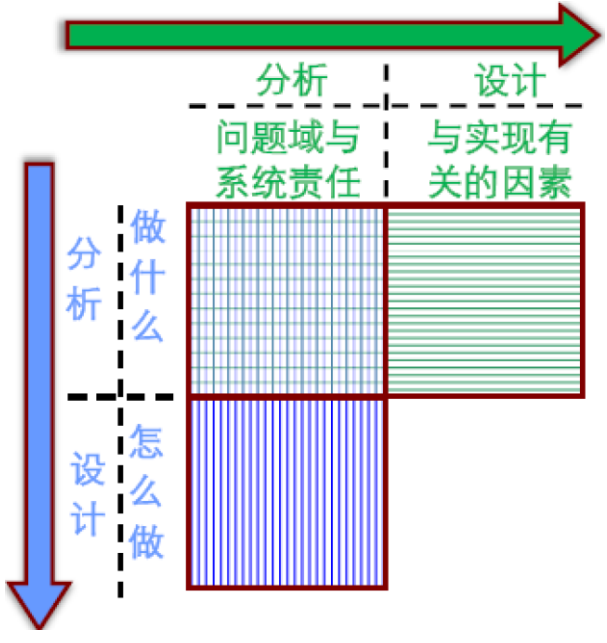
- 系统是一组相互关联的模块，它们在特定环境中协同工作，执行实现系统目标所需的各项功能



经典软件工程中的软件系统设计

系统分析与设计

- 需求描述“软件系统为用户解决什么问题”
- 分析描述“用户使用软件系统如何解决问题”
- 设计描述“软件系统如何为用户解决问题”

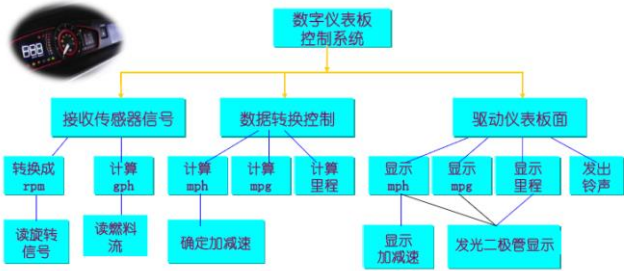


- 传统观点
  - 分析描述 what-is（玩家注册）
  - 设计描述 how-to（数据结构及其增删改查）
  - 由于均围绕目标系统，因此，分析与设计很难严格区分
  - 两者往往存在很自然的过渡，一不小心就从分析进入了设计
- 现代观点
  - 分析描述what-is和how-to（玩家注册及其数据结构）
  - 但不考虑实现因素和细节（增删改查）
  - 模型驱动的开发以及开源软件的普及

常见的分析方法

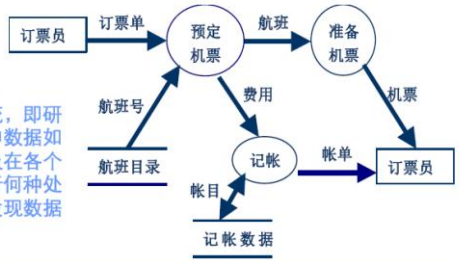
功能分解法

一层层地进行功能分解，得到由模块及其接口构成的系统模型



数据流法（结构化分析）

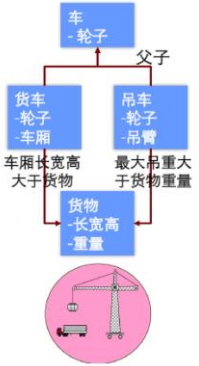
跟踪数据流，即研究问题域中数据如何流动以及在各个环节上进行何种处理，从而发现数据流和加工



信息建模法

❑实体＋属性＋关系＋父类型／子类型＋关联对象

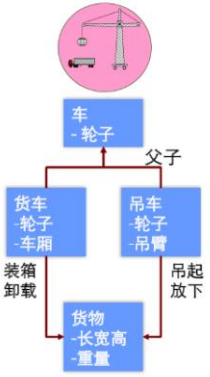
- 由实体-关系法（E-R方法）发展而来，与数据库设计有渊源
- 核心概念是实体和关系
  - 实体描述问题域的事物，含有属性
  - 关系描述事物之间在数据方面的联系，也可以带有属性
- 发展之后的方法也把实体称作对象，并使用了类型和子类型的概念作为实体（对象）的抽象描述



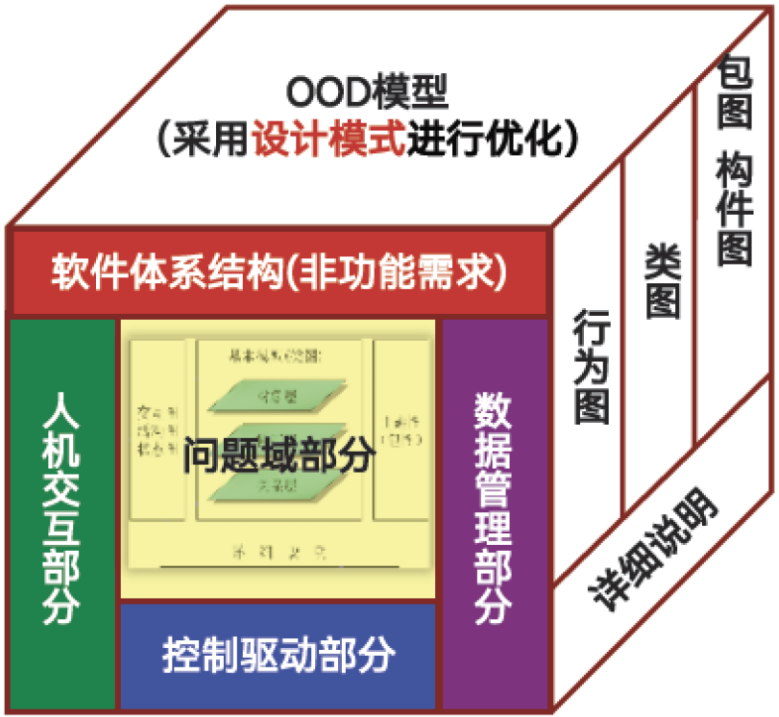
面向对象分析

❑面向对象的分析

- 运用面向对象方法进行系统分析
- 从编程领域延伸至分析领域
- OOA具有一般分析方法共同具有的内容、目标及策略
- 但强调用面向对象的概念和表示法表达分析结果
- 基本任务是：运用面向对象方法，对问题域和系统责任进行分析和理解，找出描述问题域及系统责任所需的对象，定义对象的属性、服务以及它们之间的关系



面向对象系统设计



- 基本思想：以OOA模型为基础，继续应用面向对象概念与原则，针对具体的实现条件进行系统设计
  - ➤不是转换；是调整和增补

1. 将OOA模型搬到OOD，进行必要的调整，作为OOD模型的问题域部分
2. 首先设计软件体系结构
3. 然后设计
   - 怎么与人交互
   - 怎么存储数据
   - 怎么并发执行
4. 最后根据编程实现进行优化和调整

AI系统的体系结构设计

软件体系结构（Software Architecture）

北京大学 PEKING UNIVERSITY

- **在开发初期做出的核心设计决策，侧重于关键性的系统质量**
  - 体系结构用于指导实现以满足系统的质量需求
  - 体系结构的决策一旦做出，后续更改将非常困难
- **为什么需要设计体系结构?**
  - 与利益相关者沟通： 在其能理解的层面展示系统以确认是否满足需求
  - 确定实现的限制条件： 设计决策形成应用的“承重墙”
  - 决定团队的组织结构： 按照不同模块进行分工
  - 权衡不同的属性和需求
  - 预测成本、质量和进度：通过预测每个模块的信息来间接预测整体
  - 辅助软件演化： 对成本、设计和变更效果进行推断

质量驱动的体系结构设计

- **设计体系结构需要识别最重要的质量需求，例如：**
  - 开发成本，运维成本 ，发布时间
  - 可扩展性，可用性，响应时间，吞吐量
  - 安全性，可用性，保密性
  - 易于修改和更新
  - 机器学习相关：准确率，收集数据的能力，训练延迟

AI系统的一般组成模块

- **模型推断服务**：使用模型对输入数据进行预测
- **机器学习流水线**：训练/更新模型的基础设施
- **监控**：观察模型和系统的运行情况
- **数据源**：手动/众包/日志/运行数据采集/...
- **数据管理**：存储和处理数据，通常是大规模数据
- **特征存储**：可复用的特征工程代码、特征缓存

- **非机器学习的普通软件模块**

AI系统相关的常见质量属性

北京大学 PEKING UNIVERSITY

模型的质量属性

- 准确性
- 推断延迟
- 推断吞吐量
- 可扩展性
- 模型大小和内存占用
- 能耗
- 确定性
- 每次预测的成本
- 可解释性
- 公平性、安全性、隐私性
- 鲁棒性

训练算法的质量属性

- 训练延迟
- 扩展性
- 分布式
- 内存占用
- 训练成本
- 能耗
- 增量训练
- 每次预测的成本
- 实验可行性
- 训练确定性

系统的质量属性

- 自动化
- 稳定性
- 可复现性
- 可观测性
- 部署延迟
- 持续学习和持续实验
- 数据监控
- 隐私保护

AI系统体系结构设计的共性挑战

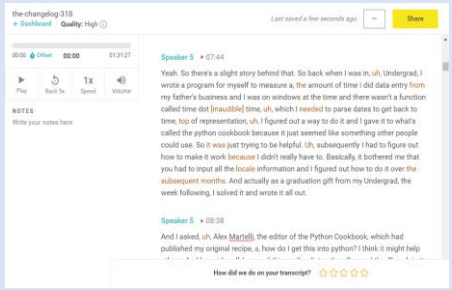
北京大学 PEKING UNIVERSITY

- **分而治之，了解相互模块之间的关系**
  - 例如，模块之间的需求冲突
- **易于实验和更新**
  - 例如，使用容器可以使系统模块独立的频繁更新
- **分离训练和推断**
  - 例如，采集运行数据，从用户与系统的交互中学习，定期更新模型
- **大规模或资源有限的运行条件**
  - 例如，云部署、嵌入式设备

案例分析

个性化音乐推荐

在云端部署微服务，记录用户活动，每晚批量处理推断，定期更新模型，定期实验，方便回退

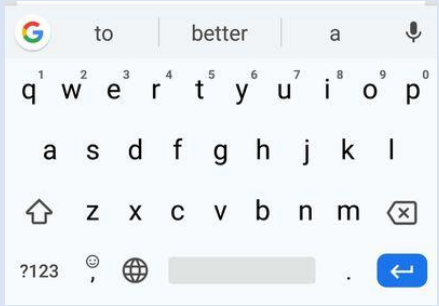


音频转录系统

用户交互不规则，模型庞大，推断成本高昂，推断延迟不关键，模型更新少见

自动驾驶汽车

车载硬件设置限制，实时需求，安全至关重要，更新的必要性，实践中有限的实验，可能离线



手机输入法

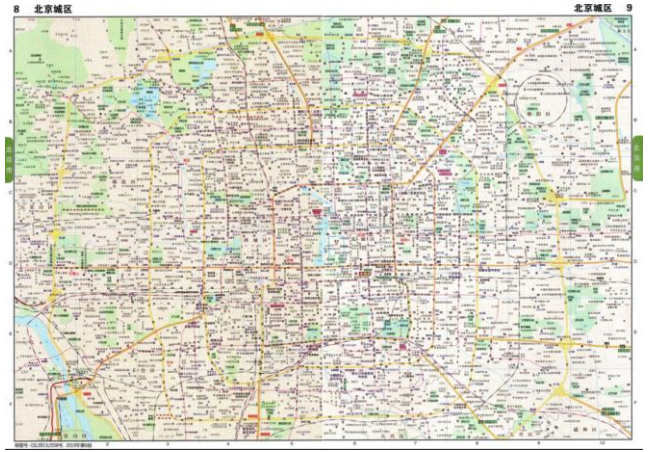
注重用户隐私，模型较小，用户设备上运行联邦学习，可采集的运行数据有限

体系结构的表达和呈现

- **软件体系结构倾向于关注重要决策，对不太重要的细节进行抽象**
  - 抽象：忽略与目标无关的部分

北京旅游地图

北京轨道交通图

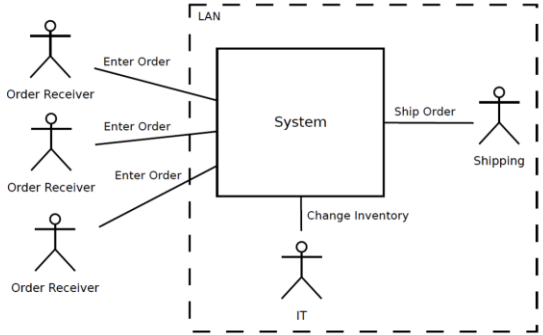


北京城市道路地图

体系结构的表达和呈现

不同的抽象得到软件的不同视图

- 软件架构师通常使用非正式的图形符号绘制软件体系结构图



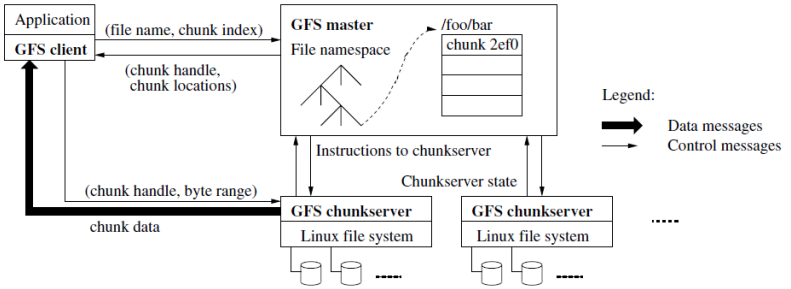
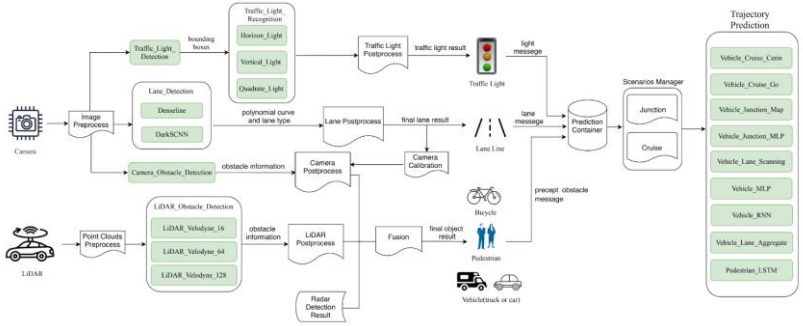
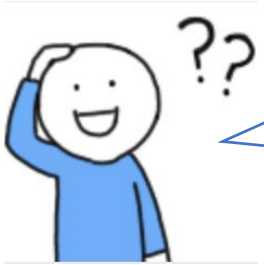


Figure 1: GFS Architecture





从图中可以推断出什么信息?

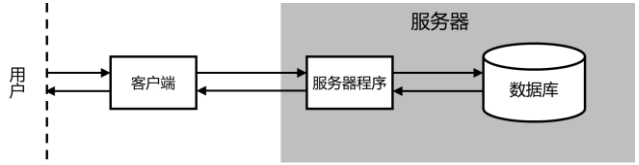
体系结构的表达和呈现

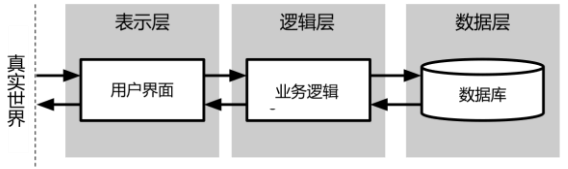
关于图形化的建议

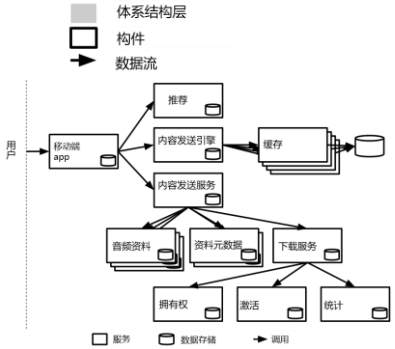
- 图中的文字要易于理解
- 在图例中明确解释图中“框”和“边”的含义
  - 例如，框代表模块或功能单元，边表示通信、依赖或控制流等关系
  - 注意同一种线/框应代表同一种含义
- 可以使用图形或文本表示
  - 白板草图通常已足够：不一定要使用复杂工具，关键是表达清晰，足以支持推理与沟通
- 可选用形式化的图形语言
  - 如果需要严谨分析，可采用如 UML（统一建模语言）、AADL（体系结构分析与设计语言）等形式化方法

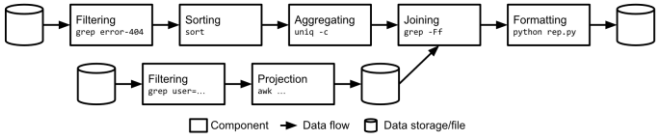
常见的软件体系结构

- **客户端-服务器体系结构**
  - 服务器通过网络连接向多个客户端提供功能
- **多层体系结构**
  - 计算和数据存储分为多个层，客户端向服务器发出请求，服务器向其他服务器发出请求
- **面向服务的体系结构和微服务**
  - 系统被组织成多个独立的服务（进程），这些服务远程调用其他服务
- **数据流体系结构**
  - 采用顺序流水线，其中一个模块生成的数据用作下一个模块的输入









设计模式

对已知设计问题的通用解决方案

- 面向对象设计模式
  - 观察者模式、单例模式、工厂模式
- 体系结构设计模式
  - 心跳策略、冗余模式、缓存策略
- 机器学习系统设计模式
  - 特征存储模式、联邦学习模式、在线学习模式
- 大语言模型设计模式
  - 检索增强生成模式、链式思维模式、知识蒸馏模式

软件设计模式

- GoF总结了三大类、23种设计模式
- 创建型模式（Creational Patterns）5种
  - 关注对象的创建，提供在创建对象的同时隐藏创建逻辑的方式，可以更加灵活地创建对象
  - 工厂方法模式、抽象工厂模式、单例模式、建造者模式、原型模式
- 结构型模式（Structural Patterns）7种
  - 关注类和对象的组合，通过改变代码结构来达到解耦的目的，使得代码容易维护和扩展
  - 适配器模式、装饰器模式、外观模式、代理模式、桥接模式、组合模式、享元模式
- 行为型模式（Behavioral Patterns）11种
  - 关注类之间的通信关系，将职责划分清楚，使得代码更加清晰
  - 策略模式、模板方法模式、观察者模式、访问者模式、状态模式、备忘录模式、迭代器模式、责任链模式、命令模式、中介者模式、解释器模式

投入多少精力进行系统设计?

- 与需求工程一样，软件设计也被广为诟病
- 但与需求也类似，开发前期对软件设计的投入能规避后续问题并节省总成本
- 敏捷视角：早期聚焦最核心的质量需求，同时保留灵活性

- AI写代码对系统设计的影响?

度量

什么是度量（Measurement）

- **度量是根据从模型或理论中推导出的规则，将数字以经验性、客观的方式赋予对象或事件的属性，目的是对它们进行描述**
  - Craner, Bond, “Software Engineering Metrics: What Do They Measure and How Do We Know?"
- **基于一次或多次观察，所表达出的对不确定性的定量减少**
  - Hubbard, “How to Measure Anything …"

一切皆可度量

- **如果 X 是我们关心的事情，那么根据定义，X 必须是可被探测的**
  - 如果某些事物（如质量、风险、安全或公众形象）是完全不可探测的（无论是直接还是间接），那我们又怎么可能关心它们呢？
  - 我们之所以关心某个未知量，是因为我们认为它在某种程度上与期望或非期望的结果相关联
- **如果 X 是可探测的，那么它一定可以被探测为某种数量**
  - 如果能观察到某件事物，那么就可以观察到它更多或更少
  - 如果能以某种数量来观察它，那么它就必然是可度量的

并非每一个度量都是精确的，也并非每一个度量都成本可控

一切皆可度量

北京大学 PEKING UNIVERSITY

- **思考：度量什么？怎么度量？**
  - 度量模型质量（语音转文本模型）
  - 度量系统质量（语音转文本系统）
  - 度量用户满意度（使用语音转文本系统的学术会议组织方）
  - 度量组织目标
  - 度量数据科学家的能力
  - 度量软件工程师的“码力”

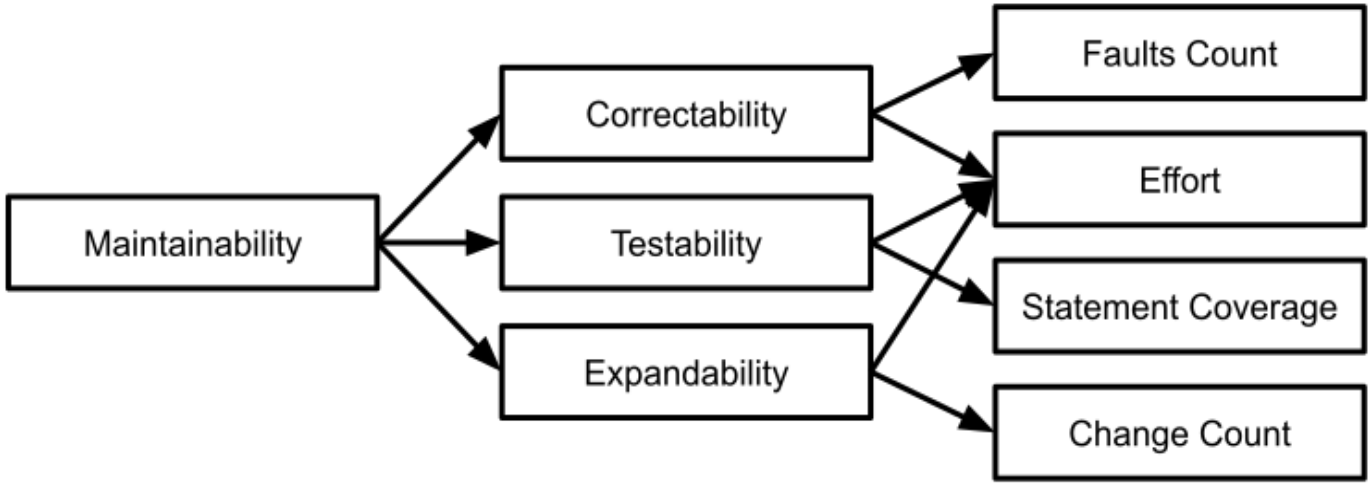
相关概念

- **量化（Quantification）**
  - 将观察结果转化为数字的过程
- **度量标准/指标（Metric）和度量方法（Measure）**
  - 用于测量某物的方法或标准形式
  - 可以认为是同义词，但有人认为度量标准是由多个测量方法推导而来，或是指标准化的测量方法
- **操作化（Operationalization）**
  - 确定并实施某一方法，以对特定因素进行测量

度量的分解

高层的度量通常是由低层的度量构成

- 应当能够清晰地追溯从具体的低层度量到高层指标的过程



“可维护性”度量的分解示例

精确的描述度量

- **精确说明想要传递的内容！**
  - “衡量准确性” → “用MAPE评估准确性”
  - “评估测试质量” → “用Jacoco测量分支覆盖率”
  - “测量执行时间” → “在正常负载下，计算API x 的平均响应时间和 90% 分位响应时间”
  - “评估开发者技能” → “测量该开发者每天平均编写的代码行数，以及其代码中被报告的缺陷数量”
  - “衡量客户满意度” → “对 5%（随机抽样）客户展示问卷，并报告回复率和客户平均评分”
- **理想情况下，独立第三方应能自主搭建用于衡量结果的基础设施**

如何开展度量

第一步，明确度量目标

- 界定清楚要捕捉的具体对象或指标，比如“用户满意度” “产品故障率”，这是后续所有动作的起点

第二步，收集数据

- 聚焦 “数据来源与方式”。既要确定收集哪些数据（如问卷反馈、设备日志），也要明确如何收集（如线上问卷、传感器实时采集）

第三步，计算结果

- 给出“数据计算方法”。需定义具体公式或规则，将收集到的原始数据转化为可衡量的结果，比如用 “（满意反馈数 / 总反馈数）×100%” 计算用户满意度，汇报过去7天的平均满意度

度量的潜在问题

- **错误的统计量**
  - 对度量理论及其测量对象的基本误解，如均值和中位数
- **错误的决策**
  - 对测量数据的错误使用，导致意想不到的副作用
- **糟糕的激励机制**
  - 忽视人为因素，或忽视度量带来的文化变化将造成何种影响

度量的有效性

北京大学 PEKING UNIVERSITY

结构有效性（Construct validity）

- 是否正在测量原本打算测量的东西?
- 抽象概念是否与所使用的具体量表/测量方式相匹配?
- 例如：IQ（智商）——到底在测什么?
- 其他例子：疼痛、语言能力、人格……

预测有效性（Predictive validity）

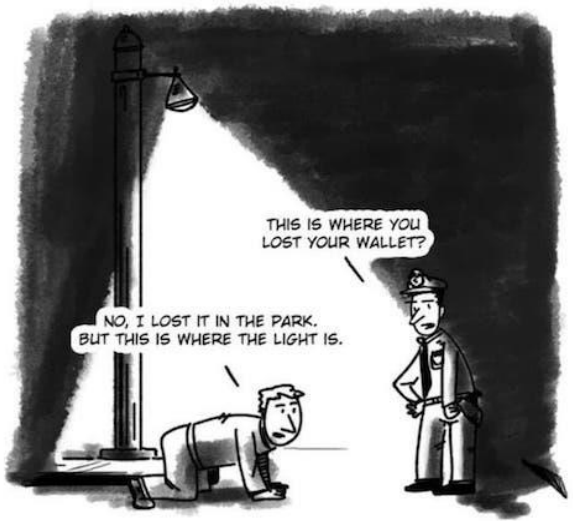
- 该测量在多大程度上可以用来解释被测对象的其他特征?
- 例如：更高的绩点是否意味着更高的学术优秀程度?

外部有效性（External validity）

- 关注研究结果能否推广到被研究环境之外的其他情境
- 例如：某药物在实验组中的效果，是否在普通大众中也成立?

避免“路灯效应”

- **人们倾向于在最容易观察的地方寻找答案**
- **使用廉价的代理指标（proxy metrics），这些指标与真正的目标只有弱相关性**
  - 例如：以每日活跃用户数（DAU）作为预测收入的衡量标准



本讲小结

- 软件体系结构侧重于早期关键设计决策，关注核心质量需求
- 体系结构设计处于需求与实现之间，负责将系统分解为各个模块
- 万物皆可度量

谢 谢

欢迎在填写问卷反馈

北京大学
PEKING UNIVERSITY