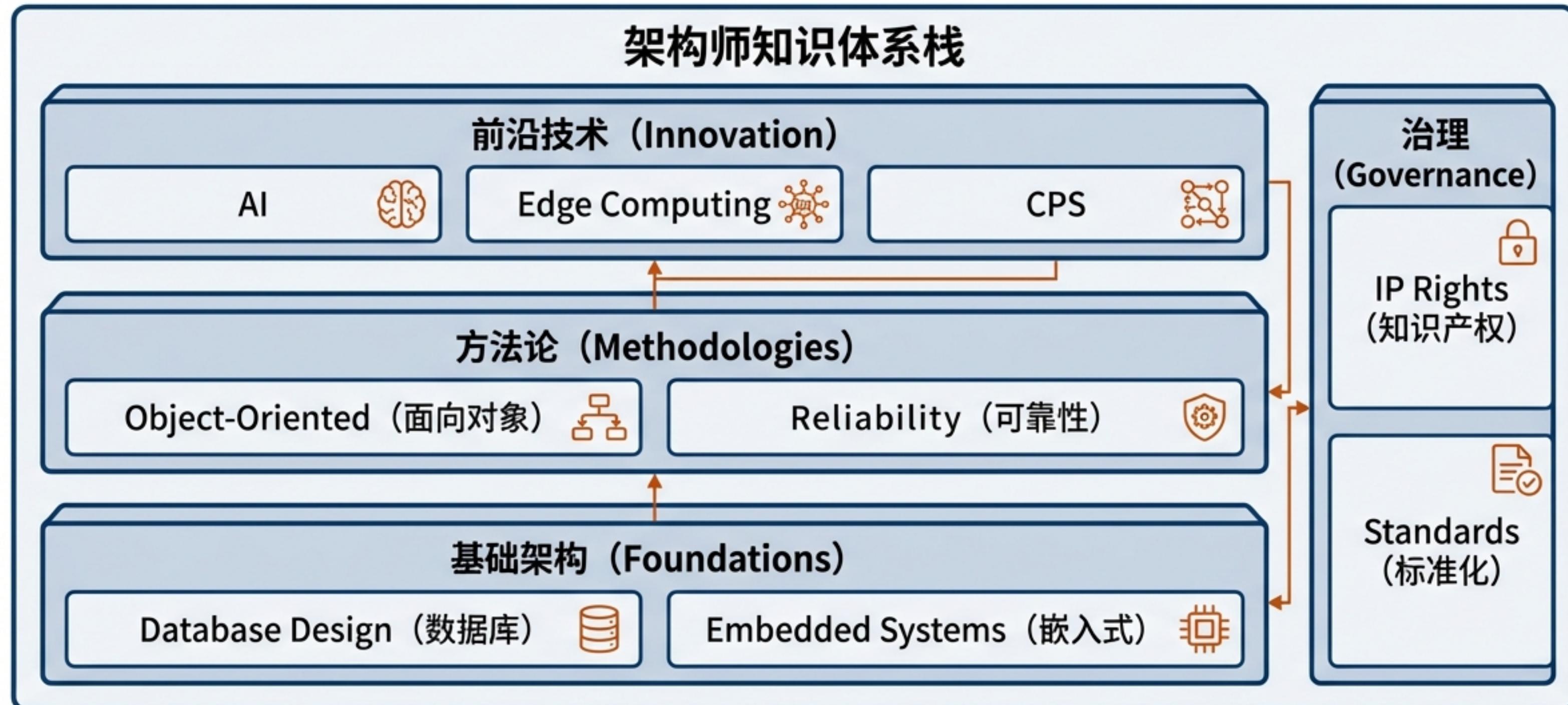


系统架构设计师核心知识精讲

进阶篇：从基础架构到未来技术与法规



架构师的知识版图与考点权重

数据库设计 (Database Design)

约 4 分 (Approx 4 pts)

- 三级模式 (3-Level Schema)
- 规范化 (Normalization)
- SQL, 事务 (Transactions)

知识产权与标准化 (IP & Standards)

约 3 分 (Approx 3 pts)

- 著作权归属
保护期限

面向对象方法 (OO Methodology)

约 4 分 (Approx 4 pts)

- UML图 (UML Diagrams)
- 设计模式 (Design Patterns)

未来信息技术 (Future Tech)

约 3 分 (Approx 3 pts)

- AI, 边缘计算 (Edge)
- CPS, 机器人 (Robotics)

嵌入式系统 (Embedded Systems)

约 3 分 (Approx 3 pts)

- 实时性 (Real-time), BSP,
嵌入式OS, Fira Code

软件可靠性 (Reliability)

约 2 分 (Approx 2 pts)

- 可靠性计算 (Calculations),
容错设计 (Fault Tolerance)

注意：数据库与可靠性涉及计算题，需掌握公式与逻辑推导。

数据架构（一）：三级模式与规范化

数据库三级模式 (The 3-Level Database Schema)

外模式 (External Schema)

用户视图 (User View)

逻辑独立性

(Logical Independence)

模式/概念模式 (Conceptual Schema)

逻辑视图 (Logical View)

物理独立性

(Physical Independence)

内模式 (Internal Schema)

物理视图 (Physical View)

数据独立性是数据库设计的核心目标。

规范化理论 (Normalization Rules)

- 1NF

- 属性原子化 (Atomic Attributes) - 不可再分

- 2NF

- 消除非主属性对码的部分依赖 (Eliminate Partial Dependency) - 每一个非主属性完全依赖于码

- 3NF

- 消除非主属性对码的传递依赖 (Eliminate Transitive Dependency) - $A \rightarrow B, B \rightarrow C$

- BCNF

- 消除主属性对码的部分和传递依赖 (Eliminate dependencies on keys in prime attributes) - 每个决定因素都是候选键

实体
(Entity)

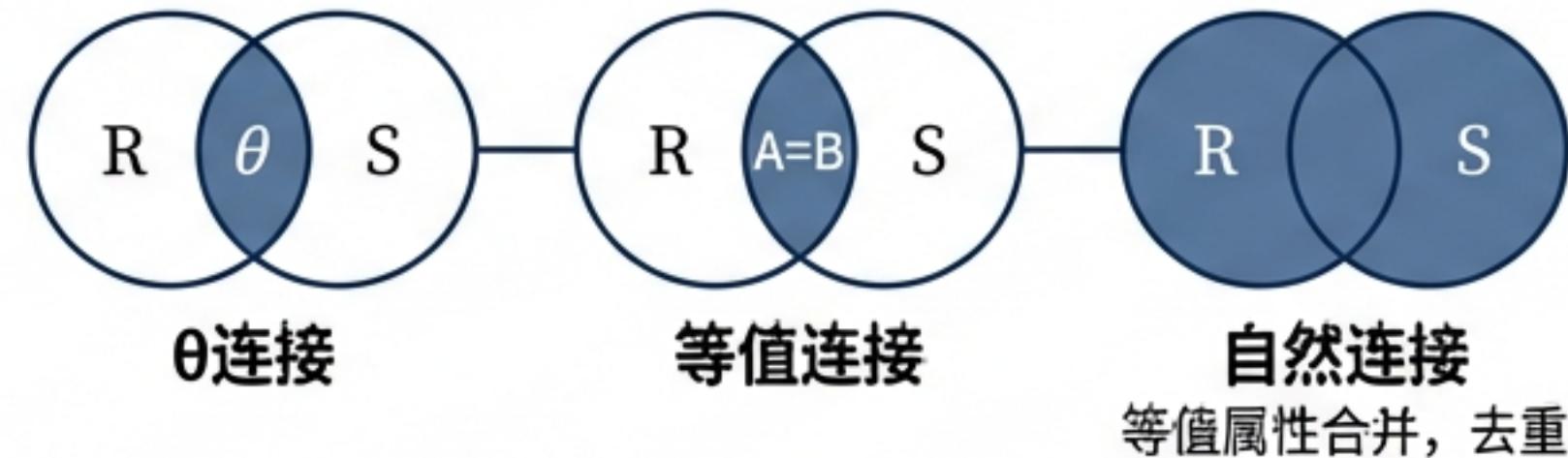
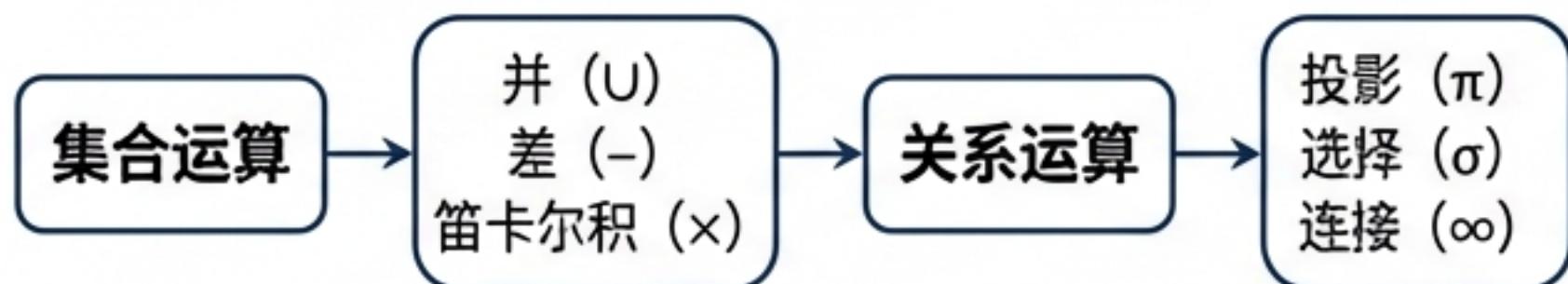
属性
(Attribute)

联系
(Relationship)

联系类型: 1:1, 1:n, m:n

数据架构（二）：关系代数与事务控制

关系代数运算 (Relational Algebra Operations)



事务管理 (ACID)

- A: 原子性 (Atomicity) - “All or Nothing” (全做或全不做)
- C: 一致性 (Consistency) - “Valid State” (有效状态)
- I: 隔离性 (Isolation) - “Independent execution” (独立执行)
- D: 持久性 (Durability) - “Permanent save” (永久保存)

并发控制 (Concurrency Control)

问题 (Problems)

丢失更新 (Lost Update), 读脏数据 (Dirty Read),
不可重复读 (Unrepeatable Read)

解决方案 (Solution)

- 封锁协议 (Locking Protocols)
 - X锁 (Exclusive Lock) : 写锁, 不允许其他锁
 - S锁 (Shared Lock) : 读锁, 允许其他S锁

数据架构（三）：分布式、数仓与大数据

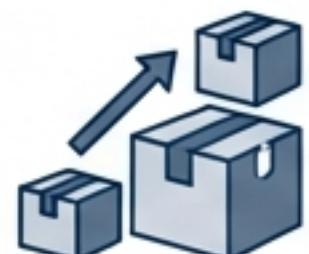
分布式数据库透明性 (Distributed DB Transparency)

1. **分片透明性** (Fragmentation Transparency):
最高层。用户无需知道数据是如何分片的
(User doesn't know how data is split).
2. **复制透明性** (Replication Transparency):
用户无需知道数据是否有副本 (User doesn't know about copies).
3. **位置透明性** (Location Transparency):
用户无需知道数据存放的物理位置 (User doesn't know physical location).

数据库 vs 数据仓库 (DB vs Data Warehouse)

Database (OLTP)	Data Warehouse (OLAP)
面向事务	面向主题
实时数据	历史数据
细节数据	集成/聚合
当前值	随时间变化

大数据 4V 特征 (Big Data 4Vs)



Volume
数据量大
(Scale)



Velocity
处理速度快
(Speed)



Variety
类型多样
(Forms)



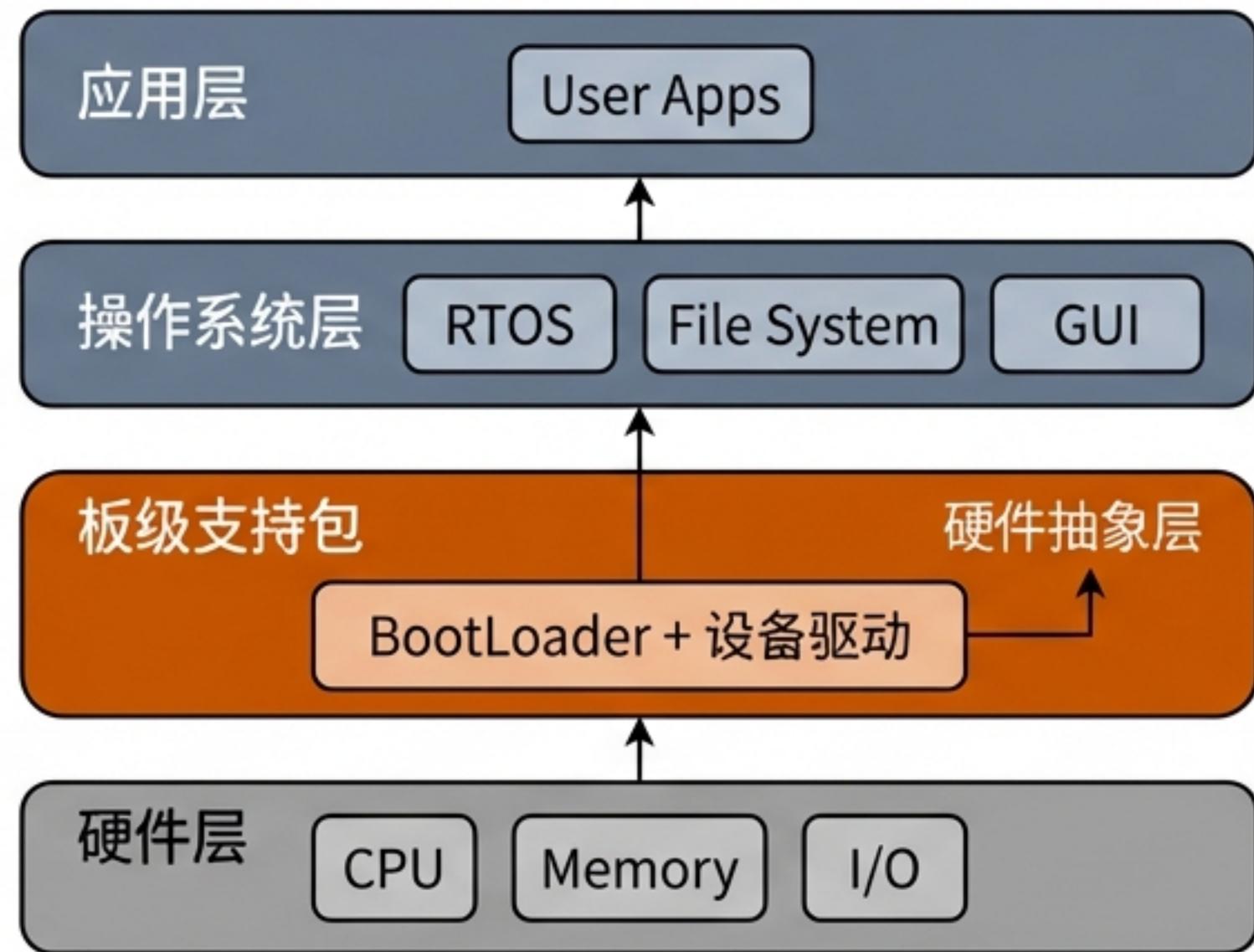
Value
价值密度低 (Low density value)

嵌入式系统：软硬件的紧密融合

嵌入式系统架构栈

BSP 的作用

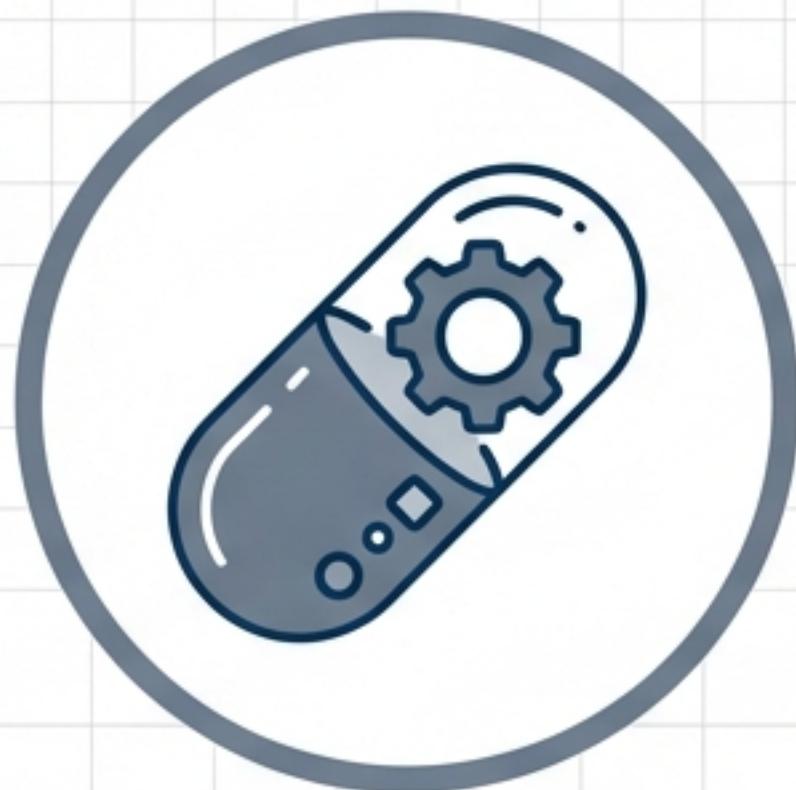
介于主板硬件和操作系统之间的一层，主要目的是为了支持操作系统，使之能够更好的运行于硬件主板。包含系统初始化和硬件驱动。



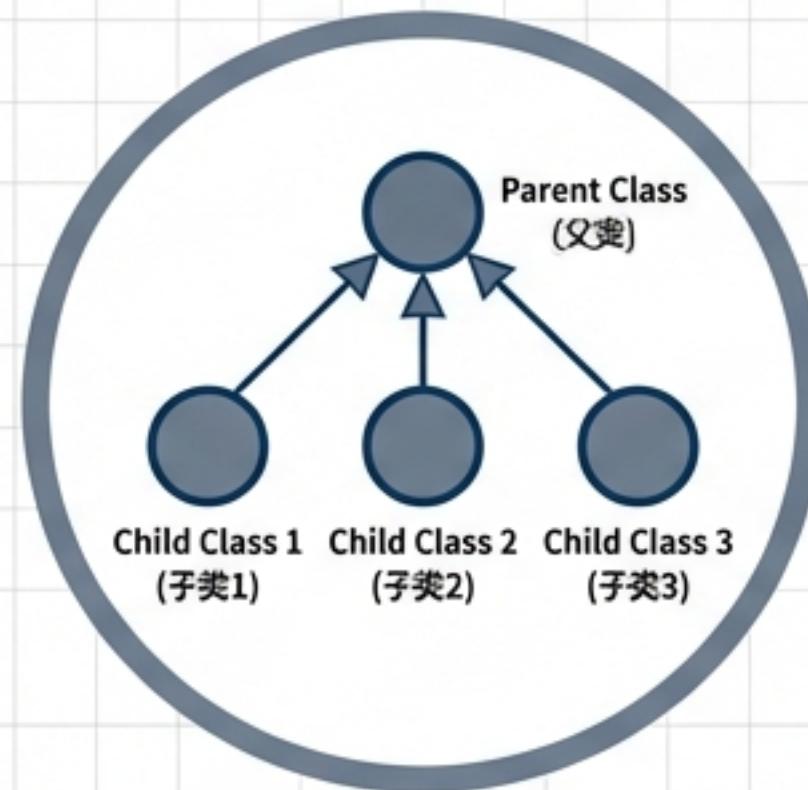
系统特点

- 专用性
- 实时性
- 资源受限
- 高可靠性

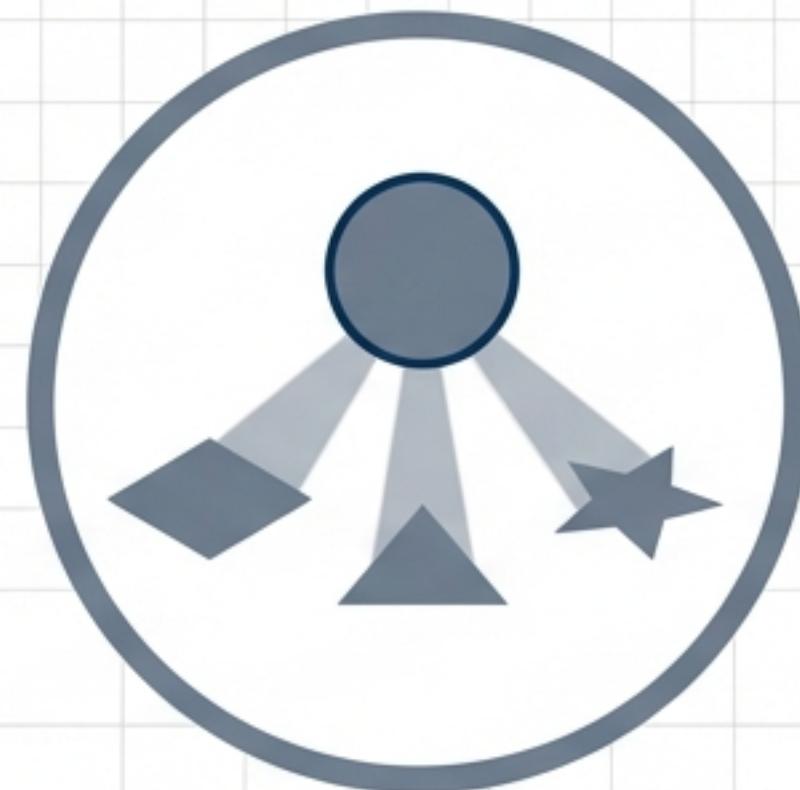
面向对象方法论：核心概念



封装 (Encapsulation)
隐藏细节，暴露接口



继承 (Inheritance)
复用代码，Is-a 关系



多态 (Polymorphism)
同一消息，不同响应

子类型：通用多态 (Universal) vs 特设多态 (Ad-hoc)

类与对象 (Class & Object)

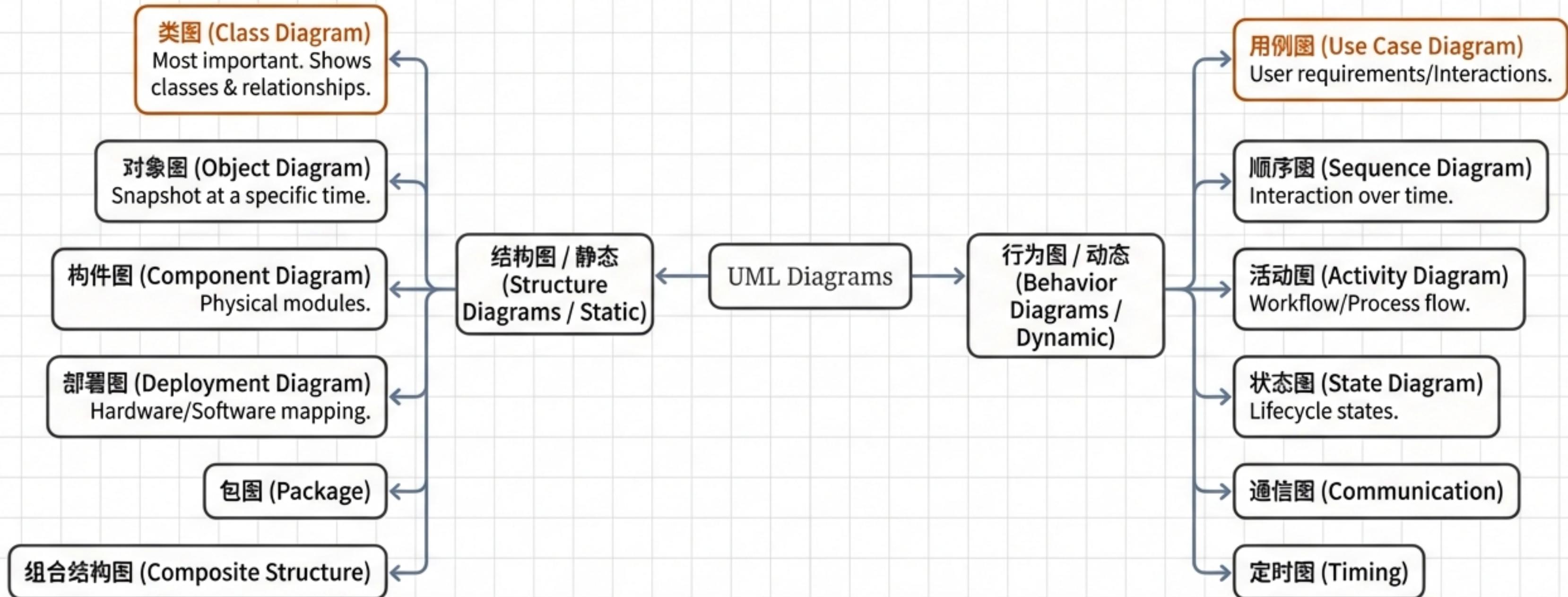
概念

- **对象 (Object)**: 系统中用来描述客观事物的一个实体
- **类 (Class)**: 对象的抽象定义

类的类型

- **实体类 (Entity Class)**: 保存需要存储在永久存储体中的信息 (Stores data)，如 Employee
- **控制类 (Control Class)**: 用于控制用例工作的类，一般是动宾短语 (Logic)，如 Manager
- **边界类 (Boundary Class)**: 用于系统接口与外部进行交互，如界面/API

UML 统一建模语言：可视化的艺术



设计模式：可复用的架构方案

创建型 (Creational) - 对象创建	结构型 (Structural) - 类/对象组合	行为型 (Behavioral) - 交互与职责
<ul style="list-style-type: none">• Factory Method: 定义创建对象的接口• Abstract Factory: 创建一系列相关对象• Singleton: 保证一个类仅有一个实例• Builder: 构建复杂对象• Prototype: 复制现有对象	<ul style="list-style-type: none">• Adapter: 接口转换• Bridge: 抽象与实现分离• Decorator: 动态增加职责• Facade: 子系统统一入口• Composite• Proxy• Flyweight	<ul style="list-style-type: none">• Observer: 一对多通知• Strategy: 算法替换• Command: 请求封装• State: 状态改变行为• Mediator• Memento• Template Method

软件可靠性（一）：度量与计算

可靠性指标 (Reliability Metrics)

- MTTF: 平均失效前时间 (Mean Time To Failure) - Reliability
- MTTR: 平均修复前时间 (Mean Time To Repair) - Maintainability
- MTBF: 平均故障间隔时间 (Mean Time Between Failures) = MTTF + MTTR

$$\text{可用性} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} \times 100\%$$

系统可靠性计算模型 (System Calculation)

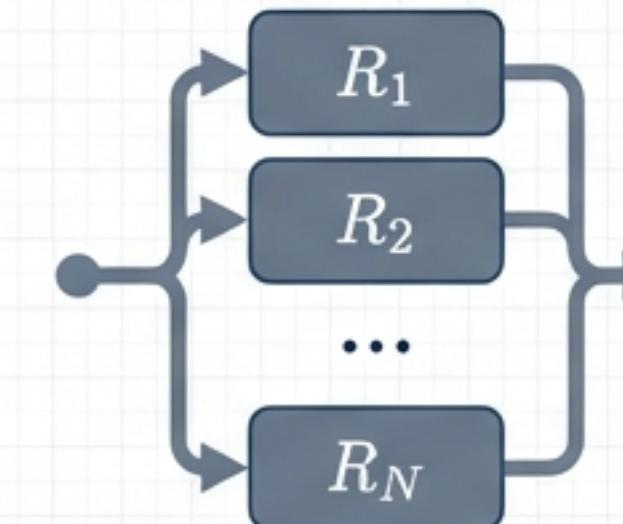
Series System (串联)



$$R = R_1 \times R_2 \times \dots \times R_n$$

任何一个失效则全系统失效

Parallel System (并联)

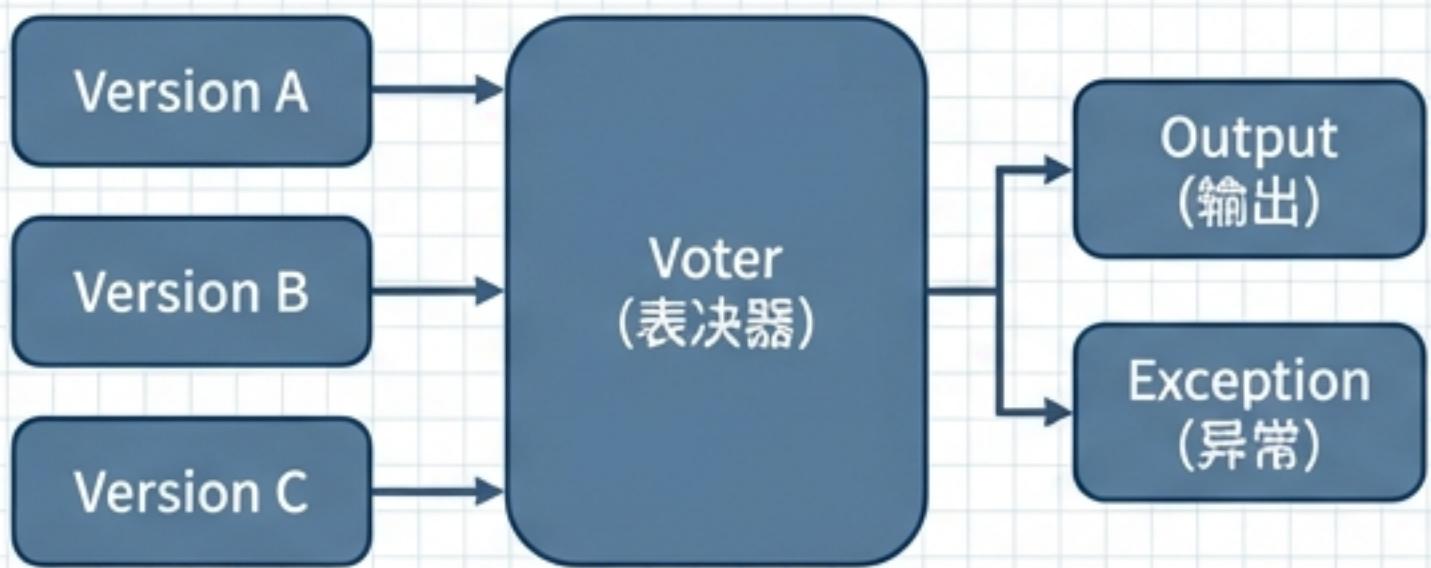


$$R = 1 - [(1 - R_1) \times (1 - R_2) \times \dots]$$

所有失效才导致系统失效

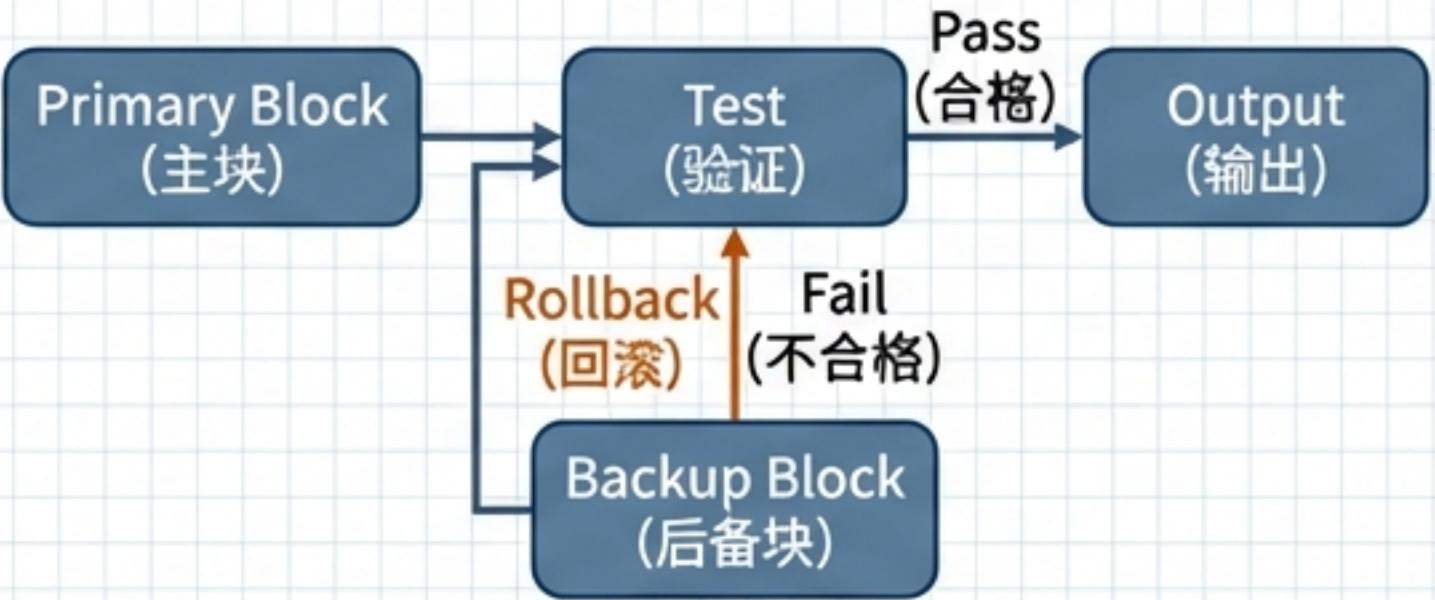
软件可靠性（二）：容错设计技术

N版本程序设计 (N-Version Programming)



- Type: 静态冗余 (Static Redundancy)
- Mechanism: 表决 (Voting)
- Recovery: 前向恢复 (Forward Recovery) - 继续执行
- Constraint: 各版本必须独立开发

恢复块设计 (Recovery Blocks)



- Type: 动态冗余 (Dynamic Redundancy)
- Mechanism: 回滚与重试 (Rollback & Retry)
- Recovery: 后向恢复 (Backward Recovery) - 回到前一状态

未来技术（一）：信息物理系统（CPS）与机器人

信息物理系统（CPS - Cyber-Physical Systems）

计算、通信、控制（3C）的深度融合（Integration of Computation, Communication, Control）

四大核心要素

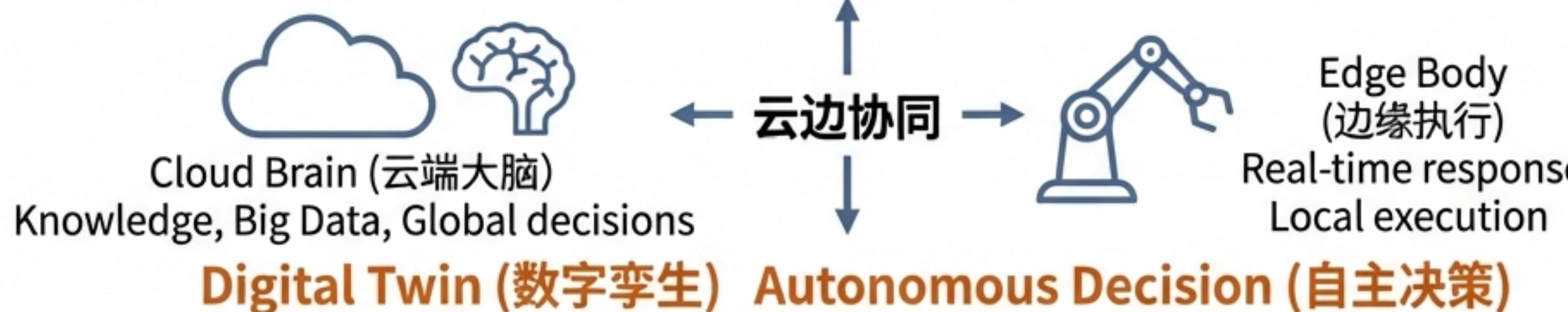
一硬：感知和自动控制
硬件支撑

一软：工业软件
核心

一网：工业网络
互联

一平台：工业云与智能服务平台
支撑

机器人 4.0（Robotics 4.0）



未来技术（二）：边缘计算与人工智能

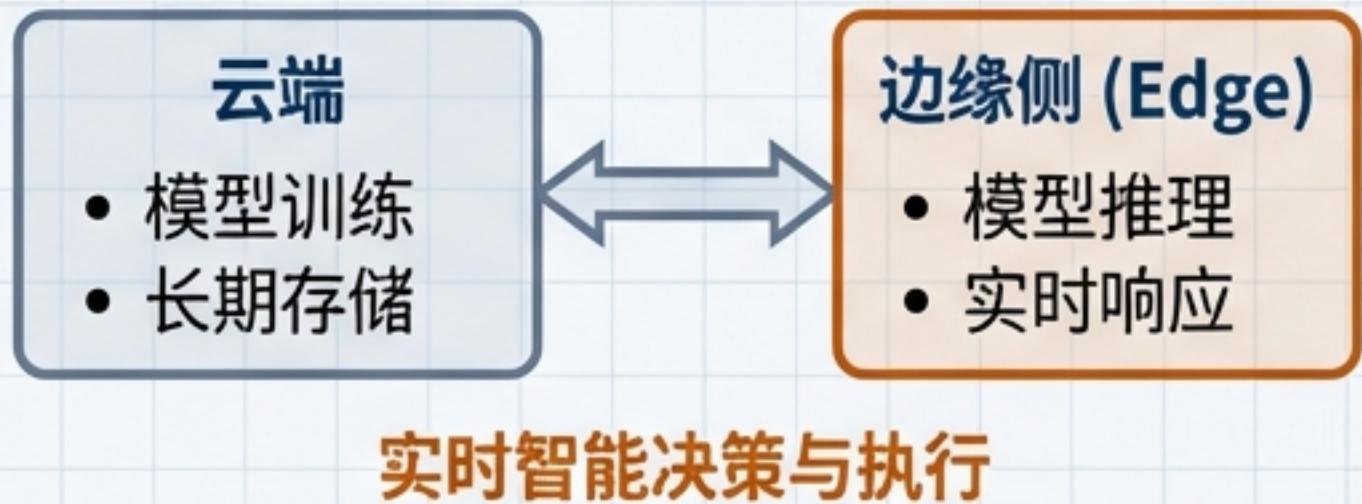
Future Tech II: Edge Computing & AI

边缘计算 (Edge Computing)

概念：在靠近数据源头（边缘侧）进行计算
(Processing near the source)

优势：低延迟 (Low Latency)
节省带宽 (Bandwidth Saving)
隐私保护 (Privacy)

边云协同：



人工智能 (Artificial Intelligence)

学习类型：

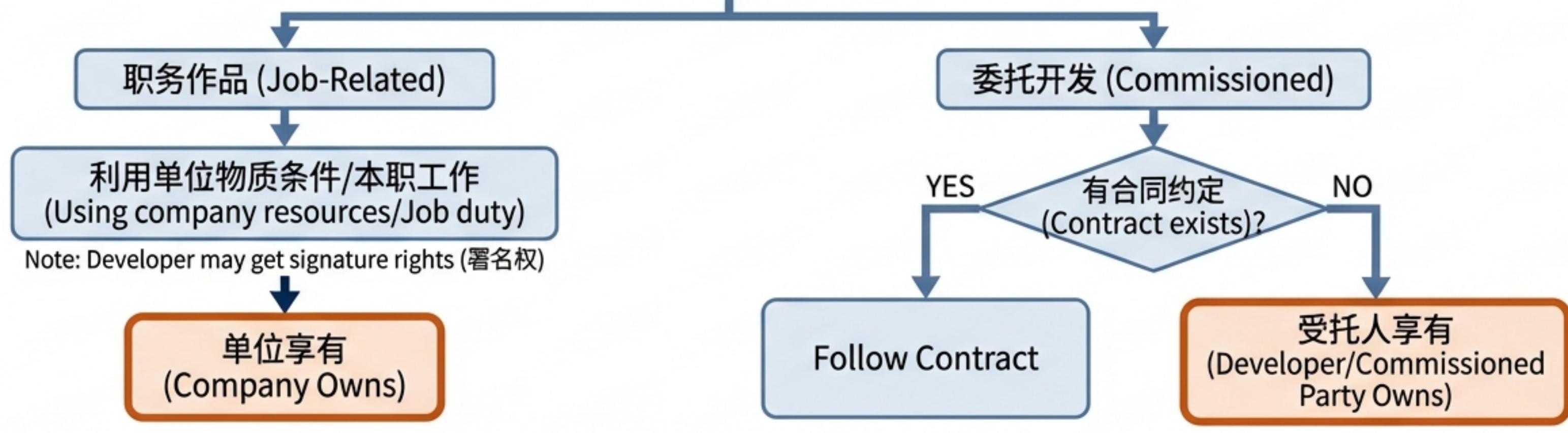
- 监督学习 (Supervised) : Labeled 数据 (有标签)
- 无监督学习 (Unsupervised) : 无标签 (无标签), 集结
- 强化学习 (Reinforcement) : 奖励和/或惩罚 (奖惩反馈)

AI芯片：

- GPU : Graphics, 平行处理 (Parallel processing)
- FPGA : Programmable, Low latency
- ASIC : Dedicated circuit, 高效率 (e.g., TPU)

法规与标准：知识产权 (IPR) 与 标准化

软件著作权归属判定 (Software Copyright Ownership)



保护期限 (Protection Duration)

- 公民 (Citizen): 终生 + 死亡后50年 (Life + 50 yrs)
- 单位 (Entity): 发表后50年 (50 yrs after publication)
- 专利 (Patents): 发明20年 (Invention 20y),
设计/实用新型10年 (Design/Utility 10y)

标准化 (Standardization)

- GB: Mandatory (强制性)
- GB/T: Recommended (推荐性)
- ISO: International

考前冲刺：核心速记清单

Final Sprint: The 'Must-Memorize' Checklist)

必背逻辑 (Logic to Master)

Normalization

1NF (Atomic)

2NF (No Partial)

3NF (No Transitive)

Reliability

Series ($R_1 \times R_2$)

Parallel ($1 - (1 - R_1)(1 - R_2)$)

ACID

Atomicity, Consistency,
Isolation, Durability

必背列表 (Lists to Memorize)

CPS 4 Elements

一硬 (Sense), 一软 (Software)
一网 (Net), 一平台 (Cloud)

Big Data 4Vs

Volume, Velocity,
Variety, Value

UML

13 Diagrams (Class/Object/
Component/Deploy vs
UseCase/Seq/Act/State)

必背法规 (Laws to Know)

IP Ownership

No contract = Developer
owns it

Copyright Term

Life + 50 years

“构建知识体系，掌握架构之道。”