

系统架构设计师（软考）核心知识点复习提纲

—— 依据 1150 道真题提取整理 • 15 模块 107 条 • 去冗余合并版 ——

目 录

模块一	质量属性与架构评估	10 条（1 - 10）
模块二	软件架构风格	7 条（11 - 17）
模块三	架构设计方法、架构模式与复用	10 条（18 - 27）
模块四	构件与软件复用	5 条（28 - 32）
模块五	开发模型、方法与项目管理	8 条（33 - 40）
模块六	结构化与面向对象设计	6 条（41 - 46）
模块七	设计模式与 UML	5 条（47 - 51）
模块八	软件测试	3 条（52 - 54）
模块九	数据库与文件系统	7 条（55 - 61）
模块十	信息安全与法律法规	9 条（62 - 70）
模块十一	操作系统与计算机组成	17 条（71 - 87）
模块十二	网络与存储	7 条（88 - 94）
模块十三	新技术与人工智能	7 条（95 - 101）
模块十四	专业英语	3 条（102 - 104）
模块十五	数学与运筹	3 条（105 - 107）

模块一 质量属性与架构评估

1. 质量属性 —— 定义与二分

质量属性（Quality Attribute）指软件系统在功能之外所表现出的可度量特性，回答的不是“系统能做什么”，而是“系统做得有多好”。架构设计的根本目的就是在功能确定后让这些属性达到可接受水平。

- 开发期：可维护性、可测试性、可移植性、可扩展性、可重用性、可理解性
- 运行期：性能、安全性、可用性、可靠性、互操作性、易用性
- 六大属性（教材口径）：可用性、可修改性、性能、安全性、可测试性、易用性——它跨了这条线

★ 动词是“用”（登录、响应、扛住、连上）→ 运行期；动词是“改、测、搬、读、复用、维护”→ 开发期。“外部可观察”就是用户或运维能直接看见。

2. 质量属性 —— 各属性度量指标

质量属性	关注点	典型度量
可用性	系统正常运行的时间比例	MTBF、MTTR、可用率
可靠性	出错时仍能维持基本功能	MTTF、失效率
性能	响应速度和吞吐量	延迟、TPS、利用率
易用性	用户学习和使用的难易程度	学习曲线、操作效率、满意度
互操作性	与其他系统交互的能力	接口标准兼容性
可修改性	改一个功能要动多少模块	修改影响面、耦合度
可测试性	发现并隔离缺陷的难易	覆盖率、可观察性

★ 吞吐量属于性能，**不属于可用性**——这是最经典的错项。

3. 质量属性 —— 易用性

系统的学习曲线和操作效率属于易用性（Usability）。它衡量人用起来顺不顺，不是系统跑得久不久。

- 四个固定指标：学习时间、操作效率、错误率、主观满意度
- 题干出现“首次任务”“新手”“学习时间”→ 答易用性
- 易用性也在运行期，但属于“人机交互的代价”，不是“外部可观察的系统行为”

★ 易用 = 好用（用户体验）；可用 = 能用（正常运行时间）。一字之差，考点不同。

4. 质量属性场景 —— 六要素

质量属性场景（Quality Attribute Scenario）是一个具体的质量属性需求，是干系人与系统交互的简短陈述，用于精确描述难以直接度量的质量属性。

- 六要素：刺激源、刺激、环境、制品、响应、响应度量
- 前四项是输入（谁、干了啥、何时、打在哪），后两项是输出（怎么应对、做到多好）
- 例：服务器硬件 / 断电 / 正常运行 / 主数据库 / 切换到备用 / 30 秒内恢复且不丢数据

★ “谁 - 干了啥 - 什么时候 - 打在哪 - 怎么应对 - 做到多好”，六项背全，顺序常考。

5. 可用性计算

可用性（Availability）是系统在给定时间内可正常提供服务的时间比例。

- 可用率 = $MTBF / (MTBF + MTTR)$
- MTBF = 平均无故障时间（越大越好）；MTTR = 平均修复时间（越小越好）
- MTTR 趋近 0 时，可用率趋近 100%

- 年停机时长：3 个 9 (99.9%) = 8.76 小时；4 个 9 = 52.6 分钟；5 个 9 = 5.26 分钟

★ “几个 9”只答可用性；可靠性问“坏了还能不能用”。分子永远比 MTTR 大。

6. 可靠性计算

可靠性 (Reliability) 是系统在给定条件下、给定时间内不失效的能力。

- 串联： $R = R_1 \times R_2 \times \dots \times R_n$ ，一个坏全坏
- 并联： $R = 1 - (1 - R_1)(1 - R_2) \dots (1 - R_n)$ ，冗余提升可靠性
- MTTF = 平均失效前时间

★ 串联 = 把坏的概率加起来（更不可靠）；并联 = 把全坏的概率乘起来（更可靠）。

7. SAAM / ATAM / AHP / CBAM

架构评估在架构落地前用场景检验其能否满足质量属性需求，真题只考“谁最早/谁提出”和“用来干什么”。

- SAAM：最早形成文档并广泛使用的架构分析方法，关注**可修改性**
- ATAM：在 SAAM 基础上发展，关注**多质量属性权衡**
- AHP：T. L. Saaty 提出的多准则决策方法（层次分析法），定性问题定量分析
- CBAM：架构决策的**成本效益**分析，算 ROI 与回收期

★ “SAAM 最早、ATAM 权衡、AHP 定量、CBAM 算钱”。AHP 是决策方法，不是架构评估方法。

8. 架构评估 —— 敏感点与权衡点

敏感点与权衡点是 ATAM 的核心产出，二者区别在于影响的质量属性个数。

- 敏感点：对**某个**质量属性有显著影响的架构决策，可能涉及一个或多个构件
- 权衡点：影响**多个**质量属性的架构特性，改变会对不同属性产生相反影响
- 风险点：可能导致不良后果的决策；非风险点：经分析确认安全的决策

★ 敏感点 = “一碰就变”（单属性）；权衡点 = “此消彼长”（多属性）。“敏感点一定只关心一个构件”是错误说法。

9. ATAM 架构评估方法

ATAM (Architecture Tradeoff Analysis Method) 把质量属性纳入评估，在系统开发之前进行分析、评价与折中，产出效用树与四类点。教材口径为“4 个活动领域 / 9 个步骤”，两套说法都要认。

四个活动领域（考纲口径）	对应九步（教材口径）
1. 场景和需求收集	1 - 3 介绍与需求；7 - 8 场景头脑风暴
2. 属性模型构造和分析	5. 生成效用树、6 分析架构方法
3. 体系结构视图与场景实现	3 介绍架构、4 确定架构方法
4. 折中分析	9 报告结果，产出四类点

- ATAM 主要关注的四种质量属性：性能、安全性、可修改性、可用性
- 场景优先级：先按重要性给 H/L，再按实现难度给优先级

★ “四领域”考填空选属性模型构造和分析；“九步”考位置选第 5 步生成效用树。两套说法分别对应两种问法。

10. 质量属性效用树

效用树（Utility Tree）是 ATAM 用来对质量属性分类和优先级排序的工具，把模糊的质量要求逐层细化到可度量场景。

层级	内容
第 1 层（树根）	效用（Utility）
第 2 层	质量属性（可用性、性能、安全性…）
第 3 层	属性分类
第 4 层（叶）	质量属性场景，带优先级二元组（H, M）

★ 从根到叶依次是“树根 → 质量属性 → 属性分类 → 质量属性场景”，这四段就是标准答案的四个词。把“质量属性”当根、把“战术”当叶，都是错的。

模块二 软件架构风格

11. 软件架构风格总览

软件架构风格描述某一类系统组织方式的固定套路，由构件、连接件、配置约束三要素定义。

风格	构件	连接件	特点
管道-过滤器	过滤器	管道	数据流驱动，过滤器独立无状态
主程序/子过程	主程序、子过程	调用-返回	单线程控制，树形分解
面向对象	对象	方法调用	封装、继承、多态
事件驱动	过程/函数	事件绑定	构件不直接调用，松耦合
黑板	知识源+黑板+控制器	共享黑板	复杂非结构化问题
仓库	中央存储+处理构件	数据访问	以数据为中心
解释器	解释引擎+代码	状态存储	动态执行规则/脚本

★ 看连接件反查：管道→数据流；调用→调用返回；事件→独立构件；共享数据→仓库；解释→虚拟机。

12. 架构风格三要素

描述软件架构风格的核心表述是“构件、连接件以及配置约束”。ADL（架构描述语言）的三元素同样是这三样。

- 构件 Component：承担计算的单元（进程、对象、过滤器）
- 连接件 Connector：元素之间如何交互（管道、消息、过程调用）
- 配置约束 Constraint：组合时遵循的规则（拓扑、同步、并发）

★ 三个常见错项都是“只见一类构件”：模块接口（代码视角）、类对象（OO 视角）、进程线程（运行视角）——都答不出“怎么连”。

13. 黑板体系结构风格

黑板风格由知识源、黑板（共享数据区）、控制器三部分组成，模拟专家围坐讨论。

- 知识源之间不直接通信，全部通过黑板交换信息
- 控制器决定何时调用哪个知识源
- 典型应用：语音识别、图像理解、模式识别

★ 黑板 = 共享区 + 求解器 + 调度器，少了控制器就不是黑板，这是最常见的错项。

14. 事件驱动（隐式调用）架构

事件驱动风格中构件不直接调用彼此，而是发布事件由总线通知订阅者，构件之间不知道对方存在。

- 解耦空间（不知订阅者是谁）与解耦时间（可异步）
- 缺点：构件放弃对计算的控制；执行顺序不确定；调试困难；无法确定何时结束
- 优点：松耦合，易复用和演化

★ 事件驱动 = “发广播”：好处是没人需要认识你，坏处是你不知道谁听见了、什么时候听完。

15. 主程序/子过程架构风格

主程序/子过程风格采用单线程控制，按调用-返回方式执行，是层次化系统的主要结构。

- 采用树形层次分解结构，主程序拥有对子过程的控制权
- 正确性依赖每个子过程的正确性
- 空间划分：主程序与子过程之间存在调用层级

★ “单线程 + 树形 + 调用返回”三个词凑齐就是主程序/子过程。

16. 进程通信体系结构风格

进程通信风格的构件是独立进程（命名过程），连接件是消息传递。

风格	构件	连接件
进程通信	独立进程	消息传递
管道-过滤器	过滤器	管道（数据流）
事件驱动	过程/函数	事件绑定
分层	各层	层间接口调用

★ 进程通信 = 进程 + 消息，关键词：命名过程、RPC、点对点、同步/异步。

17. 分层架构

分层风格把系统按抽象级别叠成若干层，每层只对相邻层提供服务。

- 严格分层：只能调用紧邻下一层，可替换性强但性能开销大
- 松散分层：可跨层调用，性能好但耦合变高
- 好处是可移植与可替换，坏处是逐层转发拖慢性能

★ 严格 = “只能叫隔壁”；松散 = “可以跳着叫”。分层核心英文词：Hierarchical / independent / stable。

模块三 架构设计方法、架构模式与复用

18. ABSD 方法（基于架构的软件设计）

ABSD 是自顶向下、递归细化的架构设计方法，由功能分解和架构风格选择共同驱动。

- 三基础：功能分解、架构风格选择、软件模板使用
- 在需求获取之前就进行架构设计（架构驱动）；递归细化直到产生软件构件和类
- 三驱动：功能需求、质量属性（非功能需求）、商业需求
- 自顶向下从需求出发；自底向上从运行数据、构件能力出发，落点是运行效率

★ ABSD = 架构为中心 + 功能/质量/商业 三驱动。看到“以架构为中心并由业务、质量属性等需求驱动”直接选它。

19. DSSA（特定领域软件架构）

DSSA 是在一个特定应用领域中为一组应用提供组织结构参考的标准软件架构，面向“一族系统”而非单个系统。

- 三阶段：领域分析（获得领域模型）→ 领域设计（获得 DSSA）→ 领域实现（开发可复用构件）
- 各阶段以并发、递归、反复（螺旋模型）方式进行

- 四类角色：领域专家、领域分析师、领域设计人员、领域实现人员
 - 领域工程三过程 = 分析 / 设计 / 实现；应用实例化不在其中。DSSA 四阶段 = 三过程 + 实例化
- ★ 分析→设计→实现 = “定词、搭架、造件”。问“定义领域词汇”答领域分析，问“形成可复用结构”答领域设计。

20. 4+1 视图模型

4+1 视图由 Kruchten 提出，用五个视角描述同一套架构，“+1”的场景把另外四个视图串起来。

视图	关注	面向
逻辑视图	功能与对象模型	最终用户
进程视图	并发、同步、性能	系统集成者
开发视图（实现视图）	代码组织、模块、包	程序员
物理视图（部署视图）	部署、硬件拓扑	运维人员
场景（+1）	用例，验证其余视图	—

★ 开发 = 实现，物理 = 部署，别名题几乎年年有。逻辑管功能、进程管并发、物理管部署、场景管验证。

21. 软件复用 —— 按时机分类

软件复用指在开发新系统时使用已有软件资产，按复用时机分为系统复用与机会复用。

- 系统复用（计划复用）：开发之前就规划好，预先决定哪些部分需要复用
- 机会复用：开发过程中偶然发现可复用构件
- 相关：构件复用、逆向工程、再工程

★ 系统复用 = “预课”（提前规划）；机会复用 = “偶遇”（偶然发现）。

22. SOA 面向服务架构

SOA 把业务能力封装成可复用的服务，通过企业服务总线编排，角色分为服务提供者、服务消费者、服务注册中心。

- 服务提供者：发布服务；服务消费者：调用服务
- 服务注册中心：提供发布和发现机制（UDDI、Zookeeper、Nacos）
- 区别于微服务：SOA 常用集中式 ESB、可共享数据库、粗粒度

★ SOA = “有总线的服务”。看到 ESB、共享库、粗粒度 → SOA；看到独立库、容器、独立部署 → 微服务。

23. 微服务架构与熔断机制

微服务把系统拆成独立部署的细粒度服务；防止下游服务延迟拖垮上游（级联雪崩），最直接有效的是熔断器（Circuit Breaker）。

- 熔断三状态：关闭（正常）→ 打开（快速失败）→ 半开（试探恢复）
- 相关模式：服务降级（兜底数据）、限流（控制速率）、重试（网络抖动）
- 实现框架：Netflix Hystrix、Alibaba Sentinel

★ 熔断 = “跳闸”，先断开保护上游，再半开试探恢复。降级是给兜底，限流是控速率，三者别混。

24. REST 架构约束

REST 是一种面向资源的架构风格，核心是统一接口、无状态、资源导向。

- 统一接口：GET/POST/PUT/DELETE 对应查增改删
- 无状态：服务端不保存会话状态
- 资源导向：URI 标识资源；其他约束：可缓存、分层系统、按需代码（可选）

★ REST = “资源 + 无状态 + 统一动词”。会话保存在服务端就不是 REST。

25. 两层 C/S 架构 —— 胖客户机

两层 C/S 按客户端承担的职责轻重分胖客户机与瘦客户机，三层 C/S 则把业务逻辑独立成中间层。

类型	客户端职责	服务器职责
胖客户机	用户交互 + 业务逻辑处理	仅数据管理
瘦客户机	仅用户交互（表示层）	业务逻辑 + 数据管理

★ 胖客户端 = 干得多 = 界面 + 逻辑；瘦客户端 = 只管显示。三层 C/S 让客户端变“瘦”。

26. CQRS 模式（命令查询职责分离）

CQRS 将写操作（Command）和读操作（Query）分离为独立模型和数据存储。

- 写模型关注数据一致性和业务规则；读模型关注查询性能和展示
- 最大挑战：数据一致性问题（最终一致性）
- 常与事件溯源（Event Sourcing）配合

★ CQRS = “读写分家”。分家容易，一致难——最终一致性是它的命门。

27. CAP 定理

分布式系统不能同时满足一致性 C、可用性 A、分区容错性 P，网络分区不可避免故 P 必选。

- CP 系统：保证一致性牺牲可用性（ZooKeeper、Consul）
- AP 系统：保证可用性牺牲一致性（Cassandra、Eureka）
- BASE：基本可用、软状态、最终一致

★ “三选二，P 必选”。CP 牺牲 A，AP 牺牲 C——见到 ZooKeeper 想 CP，见到 Cassandra 想 AP。

模块四 构件与软件复用

28. 构件的定义与特性

构件是可独立部署、可组合、通过契约交互的软件单元，是“造”与“装”的分界。

- 构件四特性：可独立部署、可组合、有明确接口、可复用
- 构件通过**接口契约**对外提供服务，内部实现不可见
- “构件不能独立部署”“构件必须携带源码”都是**错误说法**

★ 构件的价值 = “把实现藏在接口后面”。凡强调“必须开放源码”的选项都错。

29. 主流构件模型：CORBA / EJB / COM

模型	核心	关键组件
CORBA	OMG 分布式对象标准	IDL 接口、ORB、 伺服对象 （真正实现）、POA（协调）
EJB	Java 企业级构件	会话 Bean（交互）、实体 Bean（持久化）、消息 Bean
COM/DCOM	微软对象模型	接口（IUnknown）、 包含与聚合 两种组装

★ CORBA 的“**伺服对象**”是真正实现、POA 负责协调。EJB 三 Bean：会话交互、实体持久、消息异步。

30. IDL 与接口定义语言

IDL 用与语言无关的方式描述构件接口，再由工具映射到具体语言。

- IDL 最核心的内容是**接口（interface）**
- 根模块（module）映射为 Java 的**包**或 C++ 的**命名空间**
- IDL 只描述“能做什么”，不描述“怎么实现”

★ “接口是 IDL 的核心、模块映射包/命名空间”——这两个映射年年考。

31. 逆向工程与再工程

逆向工程分析已有程序、寻求比源代码更高级的抽象；再工程在此基础上重构出新系统。

活动	做什么
逆向工程	从代码导出 更高层抽象 （不改变系统）
重构	不改变行为，改善内部结构
再工程	逆向 + 重构 + 正向，产生新系统
设计恢复	从代码恢复设计信息

★ 四层抽象：目标代码 → 程序结构 → 数据结构 → 用户文档。“各部分依赖关系”属程序结构层。

32. 遗留系统演化策略

遗留系统按“技术水平”与“业务价值”二维评价，四象限对应四种策略。

象限	策略
高价值、高技术	继承 （继续维护）
高价值、低技术	改造 （技术升级）
低价值、高技术	集成 （接入新系统）
低价值、低技术	淘汰

★ “高高继承、高低改造、低高集成、低低淘汰”——四象限八字口诀。

模块五 开发模型、方法与项目管理

33. V 模型

V 模型是瀑布模型的变体，核心特点是测试与开发阶段形成对应验证关系，左降右升。

开发阶段（左下）	测试阶段（右上）
需求分析	验收测试
概要设计	系统测试
详细设计	集成测试
编码	单元测试

★ “需验、概系、详集、编单”——左降右升，V 字对称。

34. 其他开发模型区分

- 瀑布模型：阶段严格串行，文档驱动，风险后置
- 增量模型：渐进交付、降低风险
- 螺旋模型：**风险驱动**，融合线性 + 原型迭代
- RUP：用例驱动、**以架构为中心**、迭代增量
- 原型模型：快速构建可运行版本，澄清需求

★ 看到“风险驱动”选螺旋；“以架构为中心 + 用例驱动”选 RUP；“文档驱动”选瀑布。

35. 敏捷开发与 Scrum

敏捷强调迭代、可工作软件、响应变化；Scrum 是最常用的敏捷框架。

三要素	内容
三个角色	产品负责人（PO）、Scrum Master、开发团队
三个工件	产品待办 Product Backlog、Sprint 待办、增量
四个会议	计划会、每日站会、评审会、回顾会

★ Product Backlog 排序**最主要依据是业务价值**，不是开发难度或进度压力。站会是“每天同步、不解决问题”。

36. RUP 统一过程

RUP 是“用例驱动、以架构为中心、迭代增量”的软件开发过程。

四个阶段	目标
初始 Inception	确定范围与可行性
细化 Elaboration	建立架构基线、消除高风险
构造 Construction	完成大部分实现
移交 Transition	交付用户、培训与试运行

★ RUP 三大特点“用例驱动 + 架构为中心 + 迭代增量”，三缺一就不完整。架构在**细化阶段**定型。

37. 其他开发方法

- 净室软件工程：形式化方法 + 统计质量控制，强调缺陷预防优于事后测试
- 函数式程序设计：无副作用、不可变数据，易于并行与验证
- 面向对象方法：以对象为中心，封装、继承、多态
- 结构化方法：自顶向下、逐步求精、模块化

★ 净室的精髓是“把正确性证明放在测试之前”，不是“完全不做测试”。

38. 配置管理

配置管理对软件生命周期各阶段产物的版本与变更进行控制，目标是可追溯与一致性。

- 五大活动：配置标识、版本控制、变更控制、配置审计、状态报告
- 配置项 = 经审批后纳入配置管理的产物，形成**基线**
- 变更流程：申请 → 评估 → 审批（CCB）→ 实施 → 验证

★ “标、版、变、审、状”五活动。变更必须经**变更控制委员会 CCB** 审批，不能开发自行决定。

39. CMM / CMMI 五级

级别	名称	一句话
1 级	初始级	过程混乱，靠个人英雄
2 级	已管理级	项目级有计划与跟踪
3 级	已定义级	组织级标准过程
4 级	已量化管理级	用数据定量管理
5 级	优化级	持续改进过程

★ “初、管、定、量、优”——从人到组织到数据到持续改进。

40. 软件开发工具分类

开发工具按服务的过程活动分类，考试只考“某个工具属于哪一类”。

工具	归属	代表
需求分析工具	需求阶段	基于自然语言 / 基于图形描述
设计工具	设计阶段	建模工具
编码与排错工具	实现阶段	编译器、调试器
测试工具	测试阶段	自动化测试框架
版本控制工具	管理工具（不是开发工具）	SCCS、CVS、Git
软件评价工具	管理工具	度量工具

★ 四段流程“需求 → 设计 → 编码 → 测试”各配一类工具；版本控制、评价、配置管理都归“管理工具”，不是开发工具。

模块六 结构化与面向对象设计

41. 信息隐蔽

信息隐蔽（Information Hiding）指把可能变化的设计决策封装在模块内部，对外只暴露稳定接口。

- 主要目的：提高可修改性（其次可测试性、可理解性）
- 实现手段：模块化 + 高内聚低耦合，不是加密或权限
- 隐藏的对象：将来的变化，不是数据本身

★ 信息隐蔽 = “把会变的东西包起来”，包起来是为了以后好改，所以答可修改性。

42. 内聚与耦合

内聚衡量模块内部元素的紧密程度，耦合衡量模块之间的依赖程度，目标是高内聚低耦合。

内聚（低 → 高）	耦合（高 → 低）
偶然 → 逻辑 → 时间 → 过程 → 通信 → 顺序 → 功能 （最高）	内容 → 公共 → 外部 → 控制 → 标记 → 数据 （最低）

★ “功能内聚最高，数据耦合最低”。题干说“相关且按特定次序执行”→ 过程内聚。

43. 环形复杂度（McCabe）

环形复杂度衡量程序控制流复杂度，数值等于独立路径数。

- 公式一： $V(G) = E - N + 2$ （E 边数，N 节点数）
- 公式二： $V(G) = \text{判定节点数} + 1$ （最常用）
- 公式三： $V(G) = \text{闭合区域数} + 1$

★ 数判定节点然后 +1 最快。三条公式结果必然一致，可互相验算。

44. 结构化分析与设计

阶段	核心工具 / 产物
结构化分析	DFD（功能）、E-R 图（数据）、状态转换图（行为）、数据字典
结构化设计	模块结构图、结构图、HIPO 图
结构化程序设计	自顶向下、逐步求精、模块化

★ 分析阶段“画图找需求”，设计阶段“分模块定接口”。数据字典贯穿三个模型。

45. 面向对象分析、设计与 UML 关系强度

OOA 建立问题域对象模型，OOD 细化为可实现的类与接口。

- OO 三大特性：封装、继承、多态
- UML 类关系耦合由弱到强：依赖 < 关联 < 聚合 < 组合 < 泛化
- 聚合是“弱拥有”（整体散了部件还在），组合是“强拥有”（同生共死）

★ “依赖最弱、泛化最强”。聚合与组合的区别只在“生命周期是否绑定”。

46. 用户界面设计

界面设计的目标是让用户高效、少错地完成任务，遵循一致性、反馈、简约等原则。

- 常用命令应易见易得，不藏进深层菜单
- 系统状态要有及时反馈，长操作显示进度
- 出错给可理解的提示与恢复路径，不是只报错误码

★ 错误项常是“以功能齐全为先、以开发方便为先”，把用户放到后面就是错的。

47. 设计模式三大分类

GoF 按目的把 23 种设计模式分为三类：创建型 5 种、结构型 7 种、行为型 11 种。

类型	数量	关注点
创建型	5	对象怎么创建
结构型	7	对象怎么组合
行为型	11	对象怎么交互

★ “创建型管造东西”——单例、工厂、建造者都是在造对象。5-7-11 这组数字要记牢。

48. Singleton 单例模式

单例模式确保一个类只有一个实例，并提供全局访问点，属于**创建型**模式。

- 实现要点：私有构造方法 + 静态私有实例 + 静态公有获取方法
- 线程安全写法：双重检查锁定、静态内部类、枚举
- 单例归创建型是 GoF 原分类，不因线程安全而改归并发型

★ “单例是创建型的”——这是最常被记错的一条，GoF 原分类里没有“并发控制模式”。

49. 其他高频设计模式速查

模式	类型	一句话
Strategy 策略	行为型	定义算法族，使其可互换
Adapter 适配器	结构型	将一个接口转换为客户期望的接口
Factory Method 工厂方法	创建型	定义创建接口，由子类决定实例化哪个类
Proxy 代理	结构型	为其他对象提供代理以控制访问
Facade 外观	结构型	为子系统提供统一高层接口
Command 命令	行为型	将请求封装为对象，支持撤销和队列
Decorator 装饰器	结构型	动态给对象添加职责，比继承更灵活

★ “适配器改接口、装饰器加职责、代理控访问、外观做简化”——四个最容易混。

50. UML 用例关系

UML 用例之间有包含、扩展、泛化三种关系。

关系	符号	含义
包含 include	虚线箭头 + «include»	A 必须执行 B，B 是 A 的必要部分
扩展 extend	虚线箭头 + «extend»	A 在 B 基础上插入附加动作，B 可独立存在
泛化	实线三角箭头	子用例继承父用例

★ “包含是必须（must），扩展是可选（optional）”。A 比 B 多 → A extend B。

51. UML 图与视图对应

图	类型	描述
用例图	静态	系统与外部参与者的交互
类图	静态	类及关系，是核心
对象图	静态	类图的实例快照
顺序图	动态	强调消息的 时间顺序
通信图	动态	强调对象间的 链接关系
状态图	动态	状态转换
活动图	动态	流程与并发

★ 顺序图看“时间”，通信图看“链接”。状态图管单个对象，活动图管整个流程。

模块八 软件测试

52. 测试层次与方法

测试按层次分四层，按是否看内部结构分黑盒与白盒。

层次	依据文档	关注
单元测试	详细设计	模块内部逻辑
集成测试	概要设计	接口与模块组装
系统测试	需求规格	整体功能与性能
验收测试	用户需求	是否满足业务

★ 黑盒六法：等价类、边界值、判定表、因果图、错误推测、正交试验。按题干动词选方法。

53. 黑盒测试方法

黑盒测试不看内部结构，只按输入输出关系设计用例。

- 等价类划分：把输入分成有效与无效类
- 边界值分析：取边界及其两侧，最易出错
- 判定表 / 因果图：处理多个条件的组合
- 错误推测：凭经验猜易错点

★ “边界值必选”——只要题干有取值范围，边界值就是标配，它发现的缺陷最多。

54. 测试左移与冒烟测试

测试左移指把测试活动前移到需求与设计阶段；冒烟测试是版本准入的最小验证。

- 测试左移：往前挪的是**活动**，不是机房或环境
- 冒烟测试：新版本的基本功能能否跑通，不通过则拒绝进入详细测试
- 回归测试：确认修改没有破坏原有功能

★ 左移 = “把把关的关口往前挪”。冒烟管“能不能测”，回归管“改坏没有”。

模块九 数据库与文件系统

55. 范式理论

范式是关系数据库中表结构规范化的等级，每一级消除一类函数依赖带来的冗余与异常。

范式	消除的依赖
1NF	属性不可再分（原子性）
2NF	消除非主属性对候选键的 部分 函数依赖
3NF	消除非主属性对候选键的 传递 函数依赖
BCNF	每个决定因素都是候选键

★ “1 原子、2 消部分、3 消传递、BC 全候选”。有传递依赖 → 不满足 3NF → 最高 2NF。

56. 候选键与闭包求解

候选键是能唯一标识元组的最小属性集，求解依靠属性闭包而不是试错。

- 只在左部出现过的属性：必在候选键里
- 只在右部出现过的属性：必不在候选键里
- 取这些属性求闭包 X^+ ，等于全集即为候选键；不是全集就补一个中间属性再试

★ “左必在、右必不在、中间再试”。先求候选键再查范式，顺序一乱必算错。

57. 自然连接与关系运算

自然连接 $R \bowtie S$ 按公共属性自动匹配并去掉重复列，元组数取决于公共属性的匹配情况。

- $R(A, B)$ 有 5 个元组， $S(B, C)$ 有 10 个元组， B 是 R 的主码
- S 中每个 B 值最多匹配 R 中一个元组（ B 为主码，唯一）
- 元组数范围：0 到 10（ S 中元组可能找不到匹配）

★ 上限看“多”的那张表，下限可能是 0。 σ 选行、 π 去重、 $\bowtie$ 匹配。

58. E-R 模型基数约束

基数表示为 $(\min, \max)$ ，描述一个实体集中实例参与联系的次数上下限。

- “每个学生必须属于一个班级” → 学生基数 $(1, 1)$
- “每班至少 10 人至多 50 人” → 班级基数 $(10, 50)$

- 一个实体集中实例与同集合多个实例相关 → 递归联系 / 自联系

★ 把选项读成句子：谁必须至少几个、最多几个，(min, max) 一对就出来了。

59. 数据库事务 ACID

事务是数据库不可分割的工作单位，必须同时满足四个特性。

- 原子性 A：全做或全不做
- 一致性 C：执行前后状态一致
- 隔离性 I：并发事务互不干扰
- 持久性 D：提交后结果永久保存

★ ACID 记成“原子一致隔离持久”。隔离性靠锁或 MVCC，持久性靠日志。

60. 文件存储空间分配方式

文件分配决定文件在磁盘上如何占据物理块，核心取舍是“随机访问”与“增长”能否兼得。

方式	随机访问	增长
连续分配	✓ 起始块 + 块号	✗ 需连续空间
链接分配（隐式）	✗ 要顺链走	✓ 任意空闲块
链接分配（显式/FAT）	△ 查表较快	✓
索引分配	✓ 直接查表	✓ 加索引块

★ 两件事唯一的交集是**索引分配**：既随机访问又支持增长。

61. 索引位数计算

索引位数题表面是算术，实际是单位换算——先字节换位，再看编址单位。

- 第 1 步：索引项 Y 字节 $\times 8 = n$ 位，可区分 2^n 个编号
- 第 2 步：按字节编址 $\rightarrow 2^n$ ；按字编址且字长 k 字节 $\rightarrow 2^n \div k$
- 第 3 步：一个文件占一个索引项，文件数上限 = 编号总数

★ “按字节编址”四个字是“不用除”的许可证。文件名长度、目录项总长都不进公式。

模块十 信息安全与法律法规

62. 信息安全 CIA 三要素

信息安全的核心是 CIA 三要素，分别对应“保密、保真、保通”。

属性	英文	含义	典型攻击
机密性	Confidentiality	信息不泄露给未授权实体	窃听、社工
完整性	Integrity	信息不被未授权篡改	中间人攻击
可用性	Availability	授权用户能正常使用	DDoS 攻击
不可否认性	Non-repudiation	不能否认已发生的操作	抵赖

★ CIA = 保密、保真、保通。DDoS 让系统“不可用” = 可用性受损，最经典的考点。

63. 加密技术体系

算法	类型	密钥长度	说明
DES	对称	56 位	已不安全
3DES	对称	112 位 (2×56)	加密-解密-加密
AES	对称	128/192/256 位	当前主流对称加密
RSA	非对称	1024/2048 位	基于大数分解难题
MD5 / SHA	哈希	无密钥	摘要、完整性校验

★ 3DES = “加解加”三步，两把钥匙，112 位。对称快、非对称慢但解决密钥分发、哈希不可逆。

64. 数字签名技术

数字签名 = 非对称密钥加密技术 + 数字摘要技术，不涉及对称加密。

- 1. 发送方用哈希函数对原文生成数字摘要
- 2. 发送方用自己的私钥对摘要加密，生成数字签名
- 3. 接收方用发送方的公钥解密签名，得到摘要
- 4. 接收方对收到的原文重算摘要，比对两者是否一致

★ “私签公验”——私钥签名、公钥验证。作用：完整性、身份认证、不可否认性，不保证机密性。

65. SSL/TLS 协议

SSL/TLS 位于传输层与应用层之间，提供数据加密、服务器认证、消息完整性检查、客户端认证（可选）。

- 两个子协议：握手协议（协商密钥）、记录协议（数据传输）
- HTTPS = HTTP + SSL/TLS，默认端口 443
- 同时使用对称和非对称加密：非对称协商密钥，对称加密数据

★ “SSL 仅用对称加密”是错误的。非对称管协商，对称管传输，这是效率与安全的折中。

66. 访问控制矩阵与访问控制模型

访问控制矩阵（ACM）行 = 主体（用户/进程），列 = 客体（文件/资源）。

模型	决策方式
DAC 自主	资源所有者自主决定权限
MAC 强制	系统根据安全级别强制控制
RBAC 角色	权限赋予角色，用户通过角色获权
ABAC 属性	基于多维属性动态决策，最灵活

★ 行 = 我能做什么（能力表 CL），列 = 谁能访问我（访问控制列表 ACL）。

67. 认证技术

三道问答区分认证三兄弟：你是谁、你能做什么、你做了什么。

- 认证（Authentication）：你是谁？——口令、数字证书、生物特征、USB Key
- 授权（Authorization）：你能做什么？
- 审计（Accounting）：你做了什么？

★ “确保会话对方的资源与其声称的一致” = 身份认证。认证在前、授权在后、审计在尾。

68. 专利权

专利类型	保护期限	优先权期限
发明	20 年	12 个月
实用新型	10 年	12 个月
外观设计	15 年	6 个月

- 保护期限自**申请日**起算，不是授权日
- 职务发明的专利权归单位所有
- 不授予专利权：科学发现、智力活动规则和方法、疾病诊断治疗方法
- 外观设计专利中**相似设计最多 10 个**

★ “发 20、实 10、外 15”，优先权“发实 12、外 6”。起算日是最常考的陷阱。

69. 软件著作权

著作权自开发完成之日起自动产生，不以登记为必要条件；保护的是**表达形式**，不保护思想、算法、处理过程与操作方法。

- 保护期：自然人**终生 + 死后 50 年**；法人**首次发表后 50 年**
- 委托开发：合同约定归属，**无约定归受托方**
- 软件著作权保护**程序及其文档**，不保护开发所用的思想、处理过程、操作方法、数学概念

- 不适用著作权法的：法律文书、时事新闻、历法通用数表、公式

★ “自动产生、不用登记、保表达不保思想”十二个字通吃这一类题。无约定时委托方拿不到著作权。

70. 信息网络传播权

以有线或无线方式向公众提供作品，使公众可在其个人选定的时间和地点获得作品的权利，称为信息网络传播权。

权利	含义
信息网络传播权	网络使公众自主选择时间地点获取作品
发表权	决定作品是否公之于众
复制权	印刷/复印等制作一份或多份
发行权	出售/赠与提供原件或复制件（实体交付）

★ “选定时间和地点” + “有线或无线” = 信息网络传播权。看到“实体交付”就是发行权。

模块十一 操作系统与计算机组成

71. 进程状态与转换

进程只会在就绪、运行、阻塞三态之间流转，合法转换只有四条。

转换	原因	合法
就绪 → 运行	被调度器选中	✓
运行 → 就绪	时间片用完	✓
运行 → 阻塞	等待 I/O 或事件	✓
阻塞 → 就绪	等待的事件发生	✓
阻塞 → 运行	—	✗
就绪 → 阻塞	—	✗

★ 阻塞只能回到就绪，永远不能直接到运行（得重新排队等调度）。

72. PV 操作与信号量

PV 操作是信号量的两个原子操作：P 使信号量减 1（申请资源），V 使信号量加 1（释放资源）。

- P 操作： $S = S - 1$ ，若 $S < 0$ 则阻塞等待
- V 操作： $S = S + 1$ ，若 $S \leq 0$ 则唤醒一个等待进程
- 信号量 S 的初值 = 资源总数（互斥时通常为 1）

★ “P 减 V 加，减到负就等”。 S 为负数时，绝对值 = 等待的进程数。

73. 死锁四条件

死锁指多个进程互相等待对方持有的资源，导致都无法推进，成立需四个必要条件同时满足。

- 互斥：资源一次只能被一个进程占用
- 保持并等待（占有且申请）：占着资源还要申请新的
- 不可剥夺：资源只能由持有者主动释放
- 循环等待：存在进程-资源环形等待链

★ “互、占、不、环”，破坏任意一个即可防死锁。银行家算法属于**避免**，不是预防也不是检测。

74. 页面置换算法

页面置换算法决定内存满了以后淘汰哪一页，Belady 异常指增加页框数反而缺页增多。

- FIFO：可能产生 Belady 异常
- LRU 和 OPT 属栈算法，不会出现 Belady 异常
- OPT 理论最优但无法实现（需预知未来）

★ “只有 FIFO 会 Belady”。LRU 是实用中最接近 OPT 的算法。

75. 中断处理

中断是 CPU 暂停当前程序转去处理紧急事件的机制，处理分五步。

- 中断请求 → 中断判优 → 中断响应 → 中断服务 → 中断返回
- 保存现场（断点）在中断服务之前，恢复现场在中断返回时
- 中断屏蔽字决定中断优先级，可屏蔽中断在关中断时不响应

★ “请、判、响、服、返”五步。保存现场不是第五步，别放错位置。

76. 嵌入式系统实时性与 DO-178B

硬实时指超时即灾难，软实时指超时只是降级；DO-178B 是航空机载软件的安全等级标准。

- 硬实时：必须在截止时间内完成，否则后果严重（飞控、刹车）
- 软实时：偶尔超时可接受，只是体验下降（视频播放）
- DO-178B 分 A（灾难级）到 E（无影响）五个等级，A 级要求最严

★ 硬实时 = “超时 = 灾难”；软实时 = “超时 = 降级”。DO-178B 的 A 级最高，别记反。

77. 主频、外频与倍频

CPU 主频由外频与倍频共同决定，三者关系固定。

- 主频 = 外频 × 倍频（外频 200MHz × 倍频 13 = 2600MHz）
- 外频是系统总线频率，倍频是 CPU 内部相对外频的倍率
- 调外频会连带影响内存与总线，调倍频只影响 CPU

★ “主 = 外 × 倍”，一步乘法。给三个数只缺一个，除法回推。

78. 存储器分类与分级存储

计算机采用分级存储体系的目的是在容量、速度、成本之间折中。

- 分级层次：寄存器 → Cache → 主存 → 外存，越靠上越快越贵越小
- ROM 分类：PROM 一次可编程、EPROM 紫外线擦除、EEPROM 电擦除
- 存取速度最快的是寄存器，不是 Cache

★ 分级存储的目的永远是“用便宜的容量换贵的高速度”。PROM 只能写一次。

79. CISC 与 RISC

维度	CISC 复杂指令集	RISC 精简指令集
指令数量	多	少而精简
指令长度	不固定	固定
寻址方式	多	少
控制方式	微程序控制为主	硬布线控制为主
流水线	难	易，适合流水线
代表	x86	ARM、MIPS、RISC-V

★ RISC 的特点不包括“指令数量多”“寻址方式丰富”——这两个是 CISC 特征，最常作为错误选项。

80. 流水线计算

流水线把指令执行拆成多阶段并行处理，提高吞吐量，但单条指令延迟不变。

- 执行时间 = $(k + n - 1) \times \Delta t$ ，k 为阶段数，n 为指令条数， Δt 为流水线周期
- 吞吐率 $TP = n / \text{总时间}$ ；最大吞吐率 = $1 / \Delta t$
- 加速比 = 顺序执行时间 / 流水线执行时间；流水线周期 = 最长那一段

★ “ $k + n - 1$ ”——第一个任务要 k 个周期打满，之后每周期出一个结果。

81. 总线与 DMA

总线是部件间传输信息的公共通道；DMA 让外设与主存直接交换数据、绕过 CPU。

- DMA 在外设与主存之间建立直接通路，CPU 只在起止时介入
- 串行总线：一次一位，线少距离远（USB、PCIe）

- 并行总线：一次多位，速度快但易串扰，距离受限

★ DMA 的价值是“搬数据不占用 CPU”。看到“直接存储器访问”就是它。

82. 编译器处理流程

编译过程把高级语言源程序翻译成目标代码，分六个阶段。

阶段	做什么
词法分析	源程序 → 词法单元 token
语法分析	token → 语法树
语义分析	检查类型与语义合法性
中间代码生成	生成与机器无关的中间表示
代码优化	等价变换，提高效率
目标代码生成	生成机器码

★ “词、法、义、中、优、标”六步。链接、装载不属于编译器，属链接器/装载器。

83. AI 芯片架构

AI 芯片需同时支持训练与推理，主流技术架构分三类。

- GPU：并行算力强，训练主力
- FPGA：可编程、灵活、功耗低，适合推理与定制
- ASIC：专用定制，性能功耗最优但不可改，如 TPU、NPU

★ “GPU 通用并行、FPGA 可编程、ASIC 专用”——算力与灵活性的三角取舍。

84. 前趋图与调度

前趋图是有向无环图，用来表达进程之间的执行先后约束。

- 有向边 $P_i \rightarrow P_j$ 表示 P_j 开始前 P_i 必须完成
- 无前趋的节点可并行开始；无后继的节点是汇合点
- 图中最长路径决定最小完成时间（关键路径思想）

★ “前趋图就是有向无环图 DAG”。求最短完成时间找最长链，和项目管理的`关键路径`是同一个道理。

85. 概率统计与均匀分布

软考数学题以均匀分布、古典概率为主，通常可用面积法或对立法秒解。

- 均匀分布 $U(a, b)$ ：概率 = 满足条件的区间长度 / 总区间长度
- 两随机事件“相隔不超过 t”类问题：画正方形，算带状区域面积占比

- $P(X < 0.5)$ 在 $U(0, 1)$ 上 = 0.5

★ 均匀分布一律用“长度比 / 面积比”，不求积分。二维题画正方形，答案就是阴影面积除以总面积。

86. 建模概念与数学建模

数学建模是把现实问题抽象成数学结构的过程，包含建模、求解、检验三步。

- 求出最优解后还要做**灵敏度分析**，检验结果对参数变化的敏感程度
- 建模的目标是“**容易理解、容易编程实现**”，不是追求最精确
- 模型参数会随环境变化，所以建模不是一次性工作

★ “算完还要**验**”——求解之后必有灵敏度分析。题干问“还需要做什么”，答案就是它。

87. 通信原理基础

- **香农公式**: $C = W \cdot \log_2 (1 + S/N)$ ，靠**提高带宽或信噪比**提速
- **带时钟同步**的编码：曼彻斯特编码（每位中间跳变）
- 半双工通信双方至少需要 **2 条信道**（一收一发，不能同时）

★ 香农看“**带宽 + 信噪比**”。曼彻斯特 = “每位都跳一下”，所以自带时钟同步。

模块十二 网络与存储

88. OSI 七层与 TCP/IP 四层

OSI 七层	TCP/IP 四层	典型协议
应用层 / 表示层 / 会话层	应用层	HTTP、FTP、SMTP、DNS
传输层	传输层	TCP、UDP
网络层	网际层	IP、ICMP、ARP
数据链路层 / 物理层	网络接口层	以太网、PPP

★ TCP/IP 把 OSI 的上三层并成应用层、下两层并成网络接口层，中间两层不变。

89. 常用端口号

端口	服务	端口	服务
20 / 21	FTP 数据 / 控制	110	POP3
22	SSH	143	IMAP
23	Telnet	161	SNMP
25	SMTP	443	HTTPS
53	DNS	3306	MySQL
80	HTTP	6379	Redis

★ “HTTP 80、HTTPS 443、DNS 53、FTP 21”四个必记，其余按需。

90. TCP 与 UDP

维度	TCP	UDP
连接	面向连接（三次握手）	无连接
可靠性	可靠、有序、不丢不重	尽力而为
开销	大（20 字节头）	小（8 字节头）
适用	文件、网页、邮件	视频、语音、DNS、IoT

★ TCP 端口号的作用 = 标识主机中的应用进程，不是标识主机（那是 IP 的事）。

91. 层次化网络设计与冗余

局域网通常按三层模型设计：核心层、汇聚层、接入层。

- 核心层：高速转发，不做复杂策略
- 汇聚层：策略实施、路由聚合、接入层汇合点
- 接入层：终端接入，端口密度高，成本敏感
- 冗余设计目的是消除单点故障，不是提升带宽

★ 核心层“只转发不讲策略”。二层交换机工作在数据链路层。

92. QoS: IntServ 与 DiffServ

模型	机制	特点
IntServ 集成服务	资源预留（RSVP）	每流预留，精细但扩展性差
DiffServ 区分服务	IPv4 头 DS 码点标记	按类聚合，扩展性好

★ DiffServ 用 IPv4 头的 DS 字段打标记，内部路由器按码点值转发。

93. SDN 与 5G 网络切片

SDN 把控制平面与数据平面分离；5G 网络切片用虚拟化把一张物理网切成多张逻辑独立子网。

- SDN 三部分：应用层、控制层（控制器）、数据层（转发设备）
 - 网络切片的关键是**逻辑独立**：一个切片故障不影响其他切片
 - SDN 架构中**不包含**“业务层”“管理层”这类杜撰层级
- ★ SDN = “**控数分离**”三件套：应用、控制、数据。多一个“业务层”就是错项。

94. 网络存储：NAS 与 SAN

方式	特点
DAS 直连存储	直接挂本机，简单但共享差
NAS 网络附加存储	文件级 共享，走以太网
SAN 存储区域网络	块级 访问，专用高速网，性能最高

★ NAS 给的是**文件**，SAN 给的是**块**。数据库这种块级访问选 SAN。

模块十三 新技术与人工智能

95. Transformer 架构与自注意力机制

Transformer 对 NLP 的关键贡献是引入自注意力机制（Self-Attention），处理每个位置时同时关注所有其他位置。

- 自注意力允许模型同时关注序列中所有其他位置的信息
 - 解决 RNN/LSTM 的长距离依赖和无法并行计算问题；完全不用 RNN 和 CNN
 - 核心公式： $Attention(Q, K, V) = \text{softmax}(QK^T / \sqrt{d_k})V$
 - 代表模型：BERT（编码器）、GPT（解码器）
- ★ Transformer = “自注意力 + 并行计算”，把 RNN 的“逐字读”换成“一眼看全”。

96. 知识表示方法

知识表示是把领域知识转成计算机可处理形式的方法，常见形式有五种，本体是其中之一。

- 产生式规则：IF-THEN，专家系统最典型
- 语义网络：节点 + 有向边
- 框架表示：槽与填充值
- 谓词逻辑 / 本体：OWL、RDF Schema 描述

★ “规则、网络、框架、逻辑、本体”五种。冒牌货是“知识图谱”——它是应用，不是表示方法。

97. 知识图谱中的本体 (Ontology)

本体是知识图谱的“骨架/模式层”，定义领域中有哪些概念、概念间什么关系、相关约束。

- 本体不负责：存储数据（图数据库）、执行查询（SPARQL）、可视化（前端）
- 三元组：（实体，关系，实体）即（主语，谓语，宾语）
- 描述语言：OWL、RDF Schema

★ 本体 = “字典 + 语法规则”，它不管数据，只管结构。

98. 边缘计算与雾计算

边缘计算把计算下沉到靠近数据源一侧，与云协同构成端-边-云三层。

- 三个特征：极低延迟（毫秒级）、离线自治（断网仍工作）、减少云端带宽
- 边缘计算在设备端或网关侧；雾计算在边缘到云之间
- 常见误读：取代云计算（错，是协同）、等于 CDN（错，CDN 只分发不计算）

★ 边缘 = “最靠近设备”，雾 = “边缘到云之间”。两者都是把计算往前挪，位置不同。

99. 企业应用集成 EAI 四层

EAI 把异构系统连起来，分四个层次，从下往上越来越靠近业务。

层次	解决什么
数据集成	数据的共享与同步
连接集成（消息）	系统之间的通信通道
流程集成	跨系统的业务流程编排
门户集成	统一的用户界面入口

★ “数、接、流、门”——数据、连接、流程、门户，从底层到用户。

100. 多模态 AI —— 模态对齐

融合多种信息模态（视觉、听觉、文本等）时，使不同模态数据在时空和语义上相互关联的技术是模态对齐。

- 模态对齐：使不同模态数据在时空和语义上对应
- 特征提取：从原始数据提取有意义的特征表示
- 数据融合：将多源数据合并为统一表示
- 跨模态转换：将一种模态信息转换为另一种

★ “对齐” = 把不同模态的数据对齐到同一坐标系（图里的狗和文字“狗”对应起来）。

101. AI 安全 —— 数据投毒

数据投毒指在训练数据中掺入恶意样本，使模型学到错误规律，在推理时表现异常。

- 攻击发生在**训练阶段**，不是推理阶段
- 与对抗样本的区别：投毒改训练数据，对抗样本改输入
- 防护：数据溯源、异常检测、模型鲁棒性训练

★ 投毒 = “喂错料”，对抗样本 = “骗一次”。一个污染训练，一个只在推理时捣乱。

模块十四 专业英语

102. 专业英语题型与答题策略

专业英语固定 5 道填空，每题独立计分，考查架构领域英文原文理解。

步骤	动作
1	先读选项，判断词性与语义方向
2	回原文找空前空后的 搭配线索 （介词、冠词、并列结构）
3	五空独立判断 ，一个生词不影响其余四空
4	生僻词大概率是干扰项，拿不准选最常见中性词

★ 目标“保 3 争 4”。五空独立，为 1 个生词丢掉另外 4 个是最亏的。

103. 高频英文架构术语

英文	中文	英文	中文
architecture	架构	component	构件
connector	连接件	constraint	约束
quality attribute	质量属性	tradeoff	权衡
sensitivity point	敏感点	risk	风险点
stakeholder	干系人	scenario	场景
modifiability	可修改性	interoperability	互操作性
decomposition	分解	encapsulation	封装
reconstruction	重建	obfuscation	代码混淆

★ 题干出现 sensitivity point 想敏感点、tradeoff 想权衡点，这两个词直接决定选项。

104. 英文原文常见句型

- “The objective of ____ is to determine what parts of the application should be reused” → **reengineering**
- “____ is an interpretive, interactive, and iterative process by which we examine the existing system” → **reverse engineering**
- “A system’s architecture is a representation of a system in which there is a mapping of functionality onto components” → 架构定义
- “Micro-service advocates dividing a single application into a suite of small services” → 微服务定义

★ 这几句是原题复现，认得句型就等于认出答案，不必逐词翻译。

模块十五 数学与运筹

105. 最小生成树与最短路径

最小生成树是连通加权无向图中包含所有顶点、边权之和最小的树，n 个顶点恰好 n-1 条边。

算法	思路	适合
Prim	从一点出发，每次选与已选顶点集相连的最小权边	稠密图
Kruskal	所有边按权值排序，每次选最小权边且不形成环	稀疏图

★ Prim = “Point”从点出发；Kruskal = “按边排序”。生成树要连通全部顶点，最短路径只问两点之间。

106. 线性规划与工期计算

线性规划先列约束再写目标函数，最优解在可行域顶点上；工期问题用“工作量 = 效率 × 时间”。

- A 效率 = 1/20，B 效率 = 1/30，合做效率 = 1/20 + 1/30 = 1/12
- A 单独做 15 天完成 15/20 = 3/4，剩余 1/4，合做天数 = (1/4) ÷ (1/12) = 3 天
- 关键路径是最长路径，决定总工期；总时差为 0 的活动在关键路径上

★ 把总工作量设为 1，效率就是“每天做几分之一”，相加即可。关键路径看**最长**不是最短。

107. 动态规划与指派问题

动态规划适用于具有**最优子结构**与**重叠子问题**的多阶段决策问题。

- 典型题型：资源分配、背包、最长公共子序列、矩阵连乘、最短路
- 指派问题（n 人 n 事求最小总工时）用小规模枚举或匈牙利法

- 与贪心的区别：贪心只看当前最优，动态规划考虑全局

★ “**最优子结构** + **重叠子问题**”两个条件齐全才是动态规划。考生常错在把它和分治法混为一谈。



附录 高频易混淆对比速查表

考前 10 分钟只看这一页

对比项	区分
易用性 vs 可用性	好用（体验）vs 能用（运行时间比例）
可靠性 vs 可用性	坏了还能不能用 vs 一年能用多少时间
敏感点 vs 权衡点	影响一个属性 vs 影响多个属性且有正有负
风险点 vs 非风险点	可能出问题 vs 确认没问题
包含 vs 扩展	必须执行 vs 可选执行
SAAM vs ATAM	可修改性/最早 vs 多属性/权衡
ATAM vs CBAM	评架构质量 vs 加成本收益算 ROI
AHP vs SAAM	决策方法（Saaty）vs 架构评估（Kazman）
胖客户 vs 瘦客户	界面 + 逻辑 vs 只有界面
串联 vs 并联	一个坏全坏/更不可靠 vs 1 减连乘/更可靠
硬实时 vs 软实时	超时 = 灾难 vs 超时 = 降级
机会复用 vs 系统复用	偶然发现 vs 提前规划
DAC vs MAC	自主决定 vs 强制等级
对称 vs 非对称加密	同一密钥快 vs 公私钥慢但解决分发
数字签名 vs 数字信封	私钥签名保完整性 vs 对称加密正文保机密
认证 vs 授权 vs 审计	你是谁 vs 你能做什么 vs 你做了什么
边缘计算 vs 雾计算	最边缘/设备端 vs 边缘到云之间
数据投毒 vs 对抗样本	污染训练数据 vs 只在推理时改输入
顺序图 vs 通信图	强调时间顺序 vs 强调链接关系
专利 vs 著作权	需申请/20 或 10 年 vs 自动产生/终生加 50 年
瀑布 vs 螺旋	文档驱动 vs 风险驱动
CP vs AP	牺牲可用性（ZooKeeper）vs 牺牲一致性（Cassandra）

编制说明：本册体例参考题灯教育《2026 上半年 架构师 必背指南》（71 条 / 12 模块），内容全部依据自有题库重新编写：`quiz/quiz.db` 中 41 套真题（含 2013 - 2026 年综合知识 16 套）、官方解析与本机错题本，不含任何第三方押题材料，也未转录他人原文。

覆盖度：本册覆盖综合知识 106 个考点中的 102 个，按题量计 99.0%（1139 / 1150 题）；剩余 1% 为零散单题点（软件开发环境、系统维护、预测决策等），不值得单列条目。

排序依据：按考点在 16 套综合知识真题中的出现题数降序——质量属性 75 题 / 15 套居首，其后为专业英语 65 题、架构风格 63 题、软件测试 46 题、架构评估 46 题。

编排逻辑：按「架构主线 → 支撑技术 → 系统通识 → 专项」排列。模块一至四解决“架构怎么设计、怎么评估”；模块五至七是软件工程与设计基础；模块八至十二是系统层面通识（测试、数据库、安全、OS、网络）；模块十三至十五是新技术、英语与数学。**同一考点不拆两处**，避免来回翻。

使用建议：第一遍读定义与表格（约 35 分钟）；第二遍只看 ★ 记忆技巧（约 12 分钟）；考前 10 分钟只看附录易混淆表。**本册是回忆材料，不是理解材料**——概念首次接触请先看各分册正文。

