

答案

第一章 系统工程与信息系统基础

1 软件开发方法 (☆☆)

(1) 结构化开发方法

用户至上，自顶向下，逐步分解（求解），严格区分工作阶段，每阶段有任务与成果，强调系统开发过程的整体性和全局性，系统开发过程工程化，文档资料标准化。

优点：

理论基础严密，它的指导思想是用户需求在系统建立之前就能被充分了解和理解。由此可见，结构化方法注重开发过程的整体性和全局性。

缺点：

开发周期长；文档、设计说明繁琐，工作效率低；

要求在开发之初全面认识系统的信息需求，充分预料各种可能发生的变化，但这并不十分现实；若用户参与系统开发的积极性没有充分调动，就会造成系统交接过程不平稳，使系统运行与维护管理难度加大。

阶段固化，不善变化，适用于需求明确的开发场景。

(2) 原型法开发方法

适用于需求不明确的发展，按功能分为水平原型（界面）、垂直原型（复杂算法）；按最终结果分为抛弃式原型、演化式原型。原型法的特点在于原型法对用户的需求是动态响应、逐步纳入的，系统分析、设计与实现都是随着对一个工作模型的不断修改而同时完成的，相互之间并无明显界限，也没有明确分工。系统开发计划就是一个反复修改的过程。适于用户需求开始时定义不清、管理决策方法结构化程度不高的系统开发，开发方法更易被用户接受；但如果用户配合不好，盲目修改，就会拖延开发过程。

(3) 面向对象方法

最早来源于仿真领域，其特点是系统的描述及信息模型的表示与客观实体相对应，符合人们的思维习惯，有利于系统开发过程中用户与开发人员的交流和沟通，缩短开发周期，提供系统开发的准确性和效率。具有更好的复用性，关键在于建立一个全面、合理、统一的模型，分析、设计、实现三个阶段界限不明确。

用面向对象方法开发软件，通常需要建立三种形式的模型：对象模型（描述系统数据结构）、动态模型（描述系统控制结构）、功能模型（描述系统功能）。

(4) 面向服务的方法

以粗粒度、松散耦合的系统功能为核心，强调系统功能的标准化和构件化，加强了系统的灵活性、可复用性和可演化性。

从概念上讲，SO 方法有三个主要的抽象级别：操作、服务、业务流程。

操作：代表单个逻辑工作单元（LUW）的事务。

服务：代表操作的逻辑分组。

业务流程：为实现特定业务目标而执行的一组长期运行的动作或活动。

2 信息系统的分类 (☆☆☆)

信息系统的分类	关键点
业务处理系统 【TPS】	早期最初级的信息系统【20世纪50-60年代】。 功能：数据输入、数据处理【批处理、OLTP】、数据库维护、文件报表产生。
管理信息系统 【MIS】	高度集成化的人机信息系统。 金字塔结构：分多个层级。
决策支持系统 【DSS】	由语言系统、知识系统和问题处理系统组成。 用于辅助决策、支持决策。
专家系统【ES】	知识 + 推理 = 专家系统。人工智能的一个重要分支。
办公自动化系统 【OAS】	由计算机设备、办公设备、数据通信及网络设备、软件系统组成。
企业资源计划 【ERP】	打通供应链，集成，整合。

3 政府信息化与电子政务 (☆☆)

电子政务主要有3类角色：政府（Government）、企（事）业单位（Business）及公民（Citizen）。如果有第4类就是公务员（Employee）。

类型	应用
G2G	基础信息的采集、处理和利用，如：人口信息、地理信息。及政府间：计划管理、财务管理、通信系统、各级政府决策支持。
G2B	政府给企业单位颁发【各种营业执照、许可证、合格证、质量认证】。
B2G	企业向政府缴税； 企业向政府供应各种商品和服务【含竞/投标】； 企业向政府提建议，申诉。
G2C	社区公安和水、火、天灾等与公共安全有关的信息； 户口、各种证件和牌照的管理。
C2G	个人应向政府缴纳的各种税款和费用； 个人向政府反馈民意【 征求群众意见 】； 报警服务（盗贼、医疗、急救、火警等）。
G2E	政府内部管理系统。

4 企业信息化与电子商务 (☆☆)

4.1 信息化概念

信息化是指在国家宏观信息政策指导下，通过信息技术开发、信息产业的发展、信息人才的配置，最大限度地利用信息资源以满足全社会的信息需求，从而加速社会各个领域的共同发展以推进信息社会的过程。

信息化的主体是全体社会成员（政府、企业、团体和个人），时域是一个长期过程，空域是经济和社会的一切领域，手段是先进社会生产工具。

4.2 企业信息化涉及三类创新

【技术创新】在生产工艺设计、产品设计中使用计算机辅助设计系统，并通过互联网及时了解 and 掌握创新的技术信息，加快从技术向生产的转化。还有，生产技术与信息技术相结合，能够大幅度地提高技术水平和产品的竞争力。

【管理创新】按照市场发展的要求，要对企业现有的管理流程重新整合，从作为管理核心的财务、资金管理，转向技术、物资、人力资源的管理，并延伸到企业技术创新、工艺设计、产品设计、生产制造过程的管理，进而扩展到客户关系管理、供应链的管理乃至发展到电子商务。

【制度创新】那些不适应企业信息化的管理体制、管理机制和管理制度必须得到创新。

4.3 信息化需求的3个层次

组织对信息化的需求是【组织信息化的原动力】。

一是战略需求。组织信息化的目标是提升组织的竞争能力、为组织的可持续发展提供一个支持环境。从某种意义上来说，信息化对组织不仅仅是服务的手段和实现现有战略的辅助工具；信息化可以把组织战略提升到一个新的水平，为组织带来新的发展契机。特别是对于企业，信息化战略是企业竞争的基础。

二是运作需求。组织信息化的运作需求是组织信息化需求非常重要且关键的一环，它包含三方面的内容：一是实现信息化战略目标的需要；二是运作策略的需要。三是人才培养的需要。

三是技术需求。由于系统开发时间过长等问题在信息技术层面对系统的完善、升级、集成和整合提出了需求。也有的组织，原来基本上没有大型的信息系统项目，有的也只是一些单机应用，这样的组织的信息化需求，一般是从头开发新的系统。

4.4 企业信息化方法

业务流程重构方法：“彻底的、根本性的”重新设计流程。

核心业务应用方法：围绕核心业务推动信息化。

信息系统建设方法：建设信息系统作为企业信息化的重点和关键。

主题数据库方法：建立面向企业的核心业务的数据库，消除“信息孤岛”。

资源管理方法：切入点是为企业资源管理提供强大的能力。如：ERP、SCM。

人力资本投资方法：人力资本理论【注意不是人力资源管理】把一部分企业的优秀员工看作是一种资本，能够取得投资收益。

4.5 信息系统战略规划（ISSP）（★★★★）

信息系统战略规划（Information System Strategic Planning, ISSP）是从企业战略出发，构建企业基本的信息架构，对企业内、外信息资源进行统一规划、管理与应用，利用信息控制企业行为，辅助企业进行决策，帮助企业实现战略目标。

ISSP 方法经历了三个主要阶段，各个阶段所使用的方法也不一样。

4.5.1 第一个阶段

主要以数据处理为核心，围绕职能部门需求的信息系统规划，主要的方法包括：

(1) 企业系统规划法（BSP）--CU 矩阵：自上而下识别系统目标，自下而上设计信息系统，对组织机构的变动具有适应性。

(2) 关键成功因素法（CSF）：找出实现目标的关键信息集合，从而确定开发优先次序。

(3) 战略集合转化法（SST）：把战略目标看成“信息集合”，把战略目标转变成信息系统的战略目标。

(4) 其它方法包括：投资回收法、征费法、零线预算法、阶石法。

4.5.2 第二个阶段

主要以企业内部管理信息系统为核心，围绕企业整体需求进行的信息系统规划，主要的方法包括战略数据规划法（SDP）、信息工程法（IE）和战略栅格法（SG）。

4.5.3 第三个阶段

在综合考虑企业内外环境的情况下，以集成为核心，围绕企业战略需求进行的信息系统规划，主要的方法包括价值链分析法（VCA）和战略一致性模型（SAM）。

4.6 企业资源计划（ERP）

ERP 是将企业所有资源（企业三大流：物流、资金链、信息流）进行集成整合，全面一体化管理的管理信息系统。

包括三方面：生产控制（计划、制造）、物流管理（分销、采购、库存管理）和财务管理（会计核算、财务管理）。这三个系统本身就是一个集成体，它们相互之间有相应的接口，能够很好地整合在一起。

4.7 客户管理 CRM

CRM（Customer Relationship Management）理念：将客户看作资产；客户关怀是中心，目的是与客户建立长期和有效的业务关系，最大限度地增加利润；核心是客户价值管理，提高客户忠诚度和留存率。

CRM 的主要模块：销售自动化；营销自动化；客户服务与支持；商业智能。

CRM 的价值：提高工作效率，节省开支；提高客户满意度；提高客户的忠诚度。

4.8 商业智能 BI

(1) 过程

需求分析→数据仓库建模→数据抽取→建立 BI 分析报表→用户培训和数据模拟测试→系统改进和完善

- (2) 相关技术：数据仓库+数据挖掘+OLAP
- (3) 用途：决策分析【分析历史数据预判未来】
- (4) 数据仓库

数据库	数据仓库【特点】
面向应用：应用组织数据	面向主题：主题组织数据
零散的：一个应用对应一个数据库	集成的：整个企业对应一个数据仓库
CRUD：增删改查是常态	相对稳定的（非易失的）： 查询为主、基本无修改与删除
解决当下应用问题	反映历史变化（时变的）： 各个阶段信息都有，并可预测未来趋势

- (5) 数据挖掘：分类：

关联分析：挖掘出隐藏在数据间的相互关系。

序列模式分析：侧重点是分析数据间的前后关系（因果关系）。

分类分析：为每一个记录赋予一个标记再按标记分类。

聚类分析：分类分析法的逆过程。

- (6) 数据湖

概念：数据湖是一个存储企业的各种各样原始数据的大型仓库，其中的数据可供存取、处理、分析及传输。

特点：数据湖从企业的多个数据源获取原始数据，并且针对不同的目的，同一份原始数据还可能有多种满足特定内部模型格式的数据副本。因此，数据湖中被处理的数据可能是任意类型的信息，从结构化数据到完全非结构化数据。

区别：数据仓库仅支持分析处理，数据湖既支持分析处理，也支持事务处理。

4.9 企业应用集成

4.9.1 企业集成分类

按集成点分：

	集成点	效果	解题关键点
界面集成	界面	统一入口，产生“整体”感觉	“整体”感觉 最小代价实现一体化操作
数据集成	数据	不同来源的数据逻辑或物理上“集中”	其他集成方法的基础
控制集成	应用逻辑	调用其他系统已有方法，达到集成效果	
业务流程集成（过程集成）	应用逻辑	跨企业，或优化流程而非直接调用	企业之间的信息共享能力
门户集成		将内部系统对接到互联网上	发布到互联网上

按传输方式分：

	特点
消息集成	数据量小 ，交互频繁，立即地， 异步
共享数据库	交互频繁，立即地， 同步
文件传输	数据量大 ，交互频度小，即时性要求低（月末，年末）

4.9.2 企业门户

企业信息门户（EIP，Enterprise Information Portal）：使员工/合作伙伴/客户/供应商都能够访问企业内部网络和因特网存储的各种自己所需的信息。【统一访问入口】

企业知识门户（EKP，Enterprise Knowledge Portal）：企业网站的基础上增加知识性内容。【企业知识库】

企业应用门户（EAP，Enterprise Application Portal）：以商业流程和企业应用为核心，把商业流程中功能不同的应用模块通过门户技术集成在一起。【企业信息系统的网上集成界面】

垂直门户：为某一特定的行业服务的，传送的信息只属于人们感兴趣的领域。

第二章 软件工程

1 软件过程模型（★★★★★）

（1）原型模型

典型的原型开发方法模型。适用于需求不明确的场景，可以帮助用户明确需求。可以分为【抛弃型原型】与【演化型原型】

原型模型两个阶段：

- 1、原型开发阶段；
- 2、目标软件开发阶段。

（2）瀑布模型

瀑布模型是将软件生存周期中的各个活动规定为依线性顺序连接的若干阶段的模型，包括需求分析、设计、编码、运行与维护。

瀑布模型的特点是严格区分阶段，每个阶段因果关系紧密相连，只适合需求明确的项目。

缺点：软件需求完整性、正确性难确定；

严格串行化，很长时间才能看到结果；

瀑布模型要求每个阶段一次性完全解决该阶段工作，这不现实。

(3) 增量模型

融合了瀑布模型的基本成分和原型实现的迭代特征，可以有多个可用版本的发布，核心功能往往最先完成，在此基础上，每轮迭代会有新的增量发布，核心功能可以得到充分测试。强调每一个增量均发布一个可操作的产品。

(4) 螺旋模型

以快速原型为基础+瀑布模型，典型特点是引入了风险分析。它是由制定计划、风险分析、实施工程、客户评估这一循环组成的，它最初从概念项目开始第一个螺旋。

(5) V 模型

强调测试贯穿项目始终，而不是集中在测试阶段。是一种测试的开发模型。

(6) 喷泉模型

典型的面向对象的模型。特点是迭代、无间隙。会将软件开发划分为多个阶段，但各个阶段无明显界限，并且可以迭代交叉。

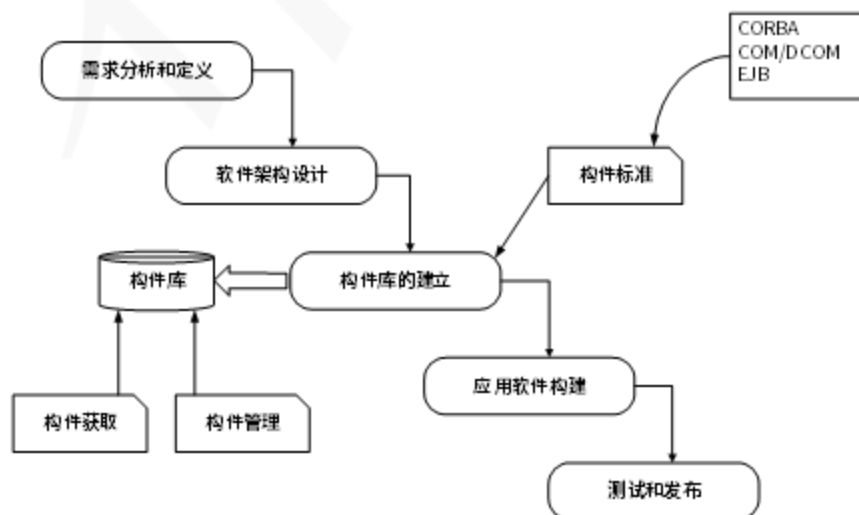
(7) 快速应用开发 RAD

概念：RAD 是瀑布模型的一个高速变种，适用比传统生命周期快得多的开发方法，它强调极短的开发周期，通常适用基于构件的开发方法获得快速开发。

过程：业务建模→数据建模→过程建模→应用生成→测试与交付

适用性：RAD 对模块化要求比较高，如果某项功能不能被模块化，则其构件就会出问题；如果高性能是一个指标，且必须通过调整结构使其适应系统构件才能获得，则 RAD 也有可能不能奏效；RAD 要求开发者和客户必须在很短的时间完成一系列的需求分析，任何一方配合不当都会导致失败；RAD 只能用于管理信息系统的开发，不适合技术风险很高的情况。

(8) 构件组装模型



【优点】易扩展、易重用、降低成本、安排任务更灵活。

【缺点】构件设计要求经验丰富的架构师、设计不好的构件难重用、强调重用可能牺牲其它指标（如性能）、第三方构件质量难控制。

（9）统一过程（在软考中 UP、RUP 都指统一过程）

典型特点是用例驱动、以架构为中心、迭代和增量。

统一过程把一个项目分为四个不同的阶段：

构思阶段（初始/初启阶段）：定义最终产品视图和业务模型、确定系统范围。

细化阶段（精化阶段）：设计及确定系统架构、制定工作计划及资源要求。

构造阶段：开发剩余构件和应用程序功能，把这些构件集成为产品，并进行详细测试。

移交阶段：确保软件对最终用户是可用的，进行β测试，制作产品发布版本。

9个核心 workflows：业务建模、需求、分析与设计、实现、测试、部署、配置与变更管理、项目管理、环境。

（10）敏捷开发

敏捷开发是一种以人为核心、迭代、循序渐进的开发方法，适用于小团队和小项目，具有小步快跑的思想。常见的敏捷开发方法有极限编程法、水晶法、并列争球法和自适应软件开发方法。

极限编程（XP）：一些对费用控制严格的公司中的使用，非常有效，近螺旋式的开发方法。四大价值观（沟通【加强面对面沟通】、简单【不过度设计】、反馈【及时反馈】、勇气【接受变更的勇气】），十二大最佳实践（简单设计、测试驱动、代码重构、结对编程、持续集成、现场客户、发行版本小型化、系统隐喻、代码集体所有制、规划策略、规范代码、40小时工作机制）。

水晶方法：提倡“机动性”的方法，拥有对不同类型项目非常有效的敏捷过程。

开放式源码：程序开发人员在地域上分布很广【其他方法强调集中办公】。

SCRUM：明确定义了可重复的方法过程。

特征驱动开发方法（FDD）：认为有效的软件开发需要3要素【人、过程、技术】。定义了6种关键的项目角色：项目经理、首席架构设计师、开发经理、主程序员、程序员和领域专家。

ASD 方法：其核心是三个非线性的、重叠的开发阶段：猜测、合作与学习。

动态系统开发方法（DSDM）：倡导以业务为核心。

敏捷宣言：

个体和交互胜过过程和工具；

可工作的软件胜过大量的文档；

客户合作胜过合同谈判；

响应变化胜过遵循计划。

2 基于构件的软件工程（CBSE）（☆☆☆）

CBSE 体现了“购买而不是重新构造”的哲学。

CBSE 的构件应该具备的特征：

- 1、可组装性：所有外部交互必须通过公开定义的接口进行。
- 2、可部署性：构件总是二进制形式的，能作为一个独立实体在平台上运行。
- 3、文档化：用户根据文档来判断构件是否满足需求。
- 4、独立性：可以在无其他特殊构件的情况下进行组装和部署。
- 5、标准化：符合某种标准化的构件模型。

【构件的组装】：

- 1、顺序组装：按顺序调用已经存在的构件，可以用两个已经存在的构件来创建一个新的构件。
- 2、层次组装：被调用构件的“提供”接口必须和调用构件的“请求”接口兼容。
- 3、叠加组装：多个构件合并形成新构件，新构件整合原构件的功能，对外提供新的接口。

3 需求工程 (☆☆)

3.1 需求工程阶段划分

软件需求是指用户对系统在功能、行为、性能、设计约束等方面的期望。

软件需求是指用户解决问题或达到目标所需的条件或能力，是系统或系统部件要满足合同、标准、规范或其他正式规定文档所需具有的条件或能力，以及反映这些条件或能力的文档说明。

需求开发包括：需求获取、需求分析、形成需求规格、需求确认与验证

需求管理包括：变更控制、版本控制、需求跟踪、需求状态跟踪

3.2 需求开发





3.2.1 需求获取

方法	特点
用户面谈	1对1-3, 有代表性的用户, 了解主观想法, 交互好。 成本高, 要有领域知识支撑。
联合需求计划 (JRP)	高度组织的群体会议, 各方参与, 了解想法, 消除分歧, 交互好, 成本高。
问卷调查	用户多, 无法一一访谈, 成本低。
现场观察	针对较为 复杂的流程 和操作。
原型化方法	通过简易系统方式 解决早期需求不确定问题。
头脑风暴法	一群人围绕新业务, 发散思维 , 不断产生新的观点。

3.2.2 需求分析

结构化需求分析 (SA) (☆☆)

结构化分析工具-数据流图 DFD：

元素	说明	图元
数据流	由一组固定成分的数据组成，表示数据的流向。每个数据流通常有一个合适的名词，反映数据流的含义	
加工	加工描述了输入数据流到输出数据流之间的变换，也就是输入数据流做了什么处理后变成了输出数据流	
数据存储 (文件)	用来表示暂时存储的数据，每个文件都有名字。流向文件的数据流表示写文件，流出的表示读文件	
外部实体	指存在于软件系统外的人员或组织	

面向对象需求分析 (★★★★★)

(1) 面向对象基本概念

对象：属性（数据）+方法（操作）+对象 ID

类（实体类/控制类/边界类）

实体类映射需求中的每个实体，实体类保存需要存储在永久存储体中的信息，例如，在线教育平台系统可以提取出学员类和课程类，它们都属于实体类。

控制类是用于控制用例工作的类，一般是由动宾结构的短语（“动词+名词”或“名词+动词”）转化来的名词，例如，用例“身份验证”可以对应于一个控制类“身份验证器”，它提供了与身份验证相关的所有操作。

边界类用于封装在用例内、外流动的信息或数据流。边界类位于系统与外界的交接处，包括所有窗体、报表、打印机和扫描仪等硬件的接口，以及与其他系统的接口。

继承与泛化：复用机制

封装：隐藏对象的属性和实现细节,仅对外公开接口。

多态：不同对象收到同样的消息产生不同的结果。

接口：一种特殊的类，它只有方法定义没有实现。

重载：一个类可以有多个同名而参数类型不同的方法。

消息和消息通信：消息是异步通信的。

(2) UML 概念

UML 包括两组公共分类，分别是类与对象（类表示概念，而对象表示具体的实体）、接口与实现（接口用来定义契约，而实现就是具体的内容）

结构事物：结构事物在模型中属于最静态的部分，代表概念上或物理上的元素。UML 有七种结构事物，分别是类、接口、协作、用例、活动类、构件和节点。类是描述具有相同属性、方法、关系和语义的对象的集合，一个类实现一个或多个接口；接口是指类或构件提供特定服务的一组操作的集合，接口描述了类或构件的对外的可见的动作；协作定义了交互的操作，是一些角色和其它事物一起工作，提供一些合作的动作，这些动作比事物的总和要大；用例是描述一系列的动作，产生有价值的结果。在模型中用例通常用来组织行为事物。用例是通过协作来实现的；活动类的对象有一个或多个进程或线程。活动类和类很相似，只是它的对象代表的事物的行为和其他事物是同时存在的；构件是物理上或可替换的系统部分，它实现了一个接口集合；节点是一个物理元素，它在运行时存在，代表一个可计算的资源，通常占用一些内存和具有处理能力。一个构件集合一般来说位于一个节点，但有可能从一个节点转到另一个节点。

行为事物：行为事物是 UML 模型中的动态部分，代表时间和空间上的动作。UML 有两种主要的行为事物。第一种是交互（内部活动），交互是由一组对象之间在特定上下文中，为达到特定目的而进行的一系列消息交换而组成的动作。交互中组成动作的对象的每个操作都要详细列出，包括消息、动作次序（消息产生的动作）、连接（对象之间的连接）；第二种是状态机，状态机由一系列对象的状态组成。

分组事物：分组事物是 UML 模型中组织的部分，可以把它们看成是个盒子，模型可以在其中进行分解。UML 只有一种分组事物，称为包。包是一种将有组织的元素分组的机制。与构件不同的是，包纯粹是一种概念上的事物，只存在于开发阶段，而构件可以存在于系统运行阶段。

注释事物：注释事物是 UML 模型的解释部分。

（3）UML 图关系

用例关系包括：包含关系、扩展关系、泛化关系。

包含关系：其中这个提取出来的公共用例称为抽象用例，而把原始用例称为基本用例或基础用例，当可以从两个或两个以上的用例中提取公共行为时，应该使用包含关系来表示它们。

扩展关系：如果一个用例明显地混合了两种或两种以上的不同场景，即根据情况可能发生多种分支，则可以将这个用例分为一个基本用例和一个或多个扩展用例，这样使描述可能更加清晰。

泛化关系：当多个用例共同拥有一种类似的结构和行为的时候，可以将它们的共性抽象成为父用例，其他的用例作为泛化关系中的子用例。在用例的泛化关系中，子用例是父用例的一种特殊形式，子用例继承了父用例所有的结构、行为和关系。

类图/对象图关系：

依赖关系：一个事物发生变化影响另一个事物。

泛化关系：特殊/一般关系

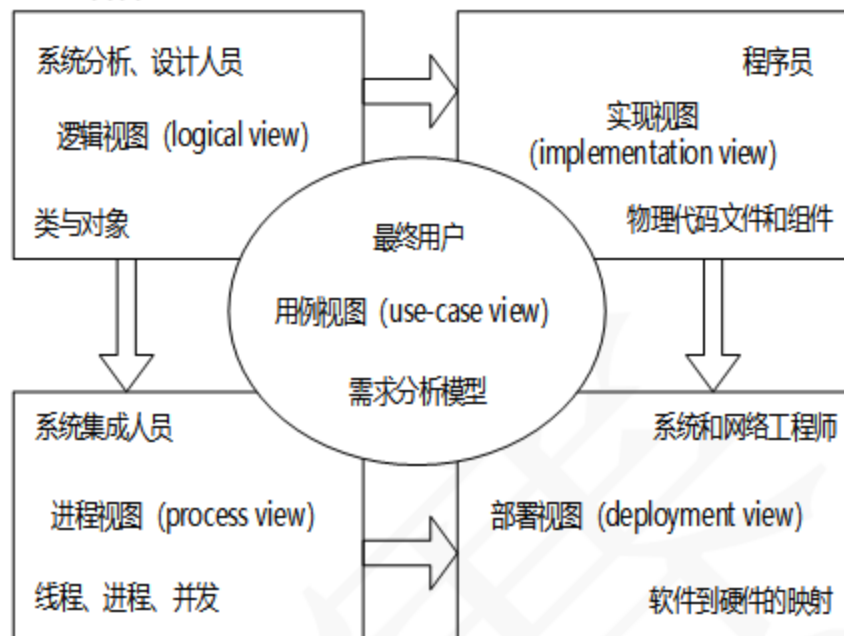
关联关系：描述了一组链，链是对象之间的连接。

聚合关系：整体与部分生命周期不同。

组合关系：整体与部分生命周期相同。

实现关系：接口与类之间的关系

(4) “4+1” 视图 (☆☆)



UML 采用4+1视图来描述软件和软件开发过程：

逻辑视图：以问题域的语汇组成的类和对象集合。

进程视图：可执行线程和进程作为活动类的建模，它是逻辑视图的一次执行实例，描绘了所设计的并发与同步结构。

实现视图：对组成基于系统的物理代码的文件和组件进行建模。

部署视图：把构件部署到一组物理的、可计算的节点上，表示软件到硬件的映射及分布结构。

用例视图：最基本的需求分析模型。

4 系统设计

4.1 界面设计 (☆☆)

用户界面设计是指用户与系统之间架起一座桥梁，主要内容包括：定义界面形式、定义基本的交互控制形成、定义图形和符号、定义通用的功能键和组合键的含义及其操作内容、定义帮助策略等。

黄金三法则：置于用户控制之下、减少用户的记忆负担、保持界面的一致性。

4.2 结构化设计 (☆☆)

(1) 概要设计【外部设计】：功能需求分配给软件模块，确定每个模块的功能和调用关系，形成模块结构图。

(2) 详细设计【内部设计】：为每个具体任务选择适当的技术手段和处理方法。

(3) 结构化设计原则：

模块独立性原则（高内聚、低耦合）；

保持模块的大小适中；

多扇入，少扇出；

深度和宽度均不宜过高。

(4) 模块四要素：

输入和输出：模块的输入来源和输出去向都是同一个调用者，即一个模块从调用者那儿取得输入，进行加工后再把输出返回调用者。

处理功能：指模块把输入转换成输出所做的工作。

内部数据：指仅供该模块本身引用的数据。

程序代码：指用来实现模块功能的程序。

(5) 模块独立性的度量

1. 聚合：衡量模块内部各元素结合的紧密程度

内聚类型	描述
功能内聚	完成一个 单一功能 ，各个部分协同工作，缺一不可。
顺序内聚	处理元素相关，而且 必须顺序执行 。
通信内聚	所有处理元素集中在一个数据结构的区域上。
过程内聚	处理元素相关，而且必须按特定的次序执行。
时间内聚 (瞬时内聚)	所包含的任务必须在 同一时间间隔内 执行。
逻辑内聚	完成逻辑上相关的一组任务。
偶然内聚 (巧合内聚)	完成一组没有关系或松散关系的任务。

2. 耦合：度量不同模块间互相依赖的程度

耦合类型	描述
非直接耦合	两个模块之间没有直接关系，它们之间的联系完全是通过主模块的控制和调用来实现的。
数据耦合	一组模块借助参数表传递 简单数据 。
标记耦合	一组模块通过参数表传递记录信息（ 数据结构 ）。
控制耦合	模块之间传递的信息中包含用于 控制模块内部逻辑 的信息。
外部耦合	一组模块都访问同一 全局简单变量 ，而且不是通过参数表传递该全局变量的信息。
公共耦合	多个模块都访问同一个公共数据环境。
内容耦合	一个模块直接访问另一个模块的内部数据；一个模块不通过正常入口转到另一个模块

的内部；两个模块有一部分程序代码重叠；一个模块有多个入口。

4.3 面向对象设计 (★★★★★)

设计原则：

单一职责原则：设计目的单一的类。

开放-封闭原则：对扩展开放，对修改封闭。

李氏 (Liskov) 替换原则：子类可以替换父类。

依赖倒置原则：要依赖于抽象，而不是具体实现；针对接口编程，不要针对实现编程。

接口隔离原则：使用多个专门的接口比使用单一的总接口要好。

组合重用原则：要尽量使用组合，而不是继承关系达到重用目的。

迪米特 (Demeter) 原则 (最少知识法则)：一个对象应当对其他对象有尽可能少的了解。

第三章 软件架构设计

1 软件架构的概念 (★)

架构设计就是需求分配，即将满足需求的职责分配到组件上。

软件架构风格是描述某一特定应用领域中系统组织方式的惯用模式。架构风格定义一个系统家族，即一个体系结构定义一个词汇表和一组约束。词汇表中包含一些构件和连接件类型，而这组约束指出系统是如何将这些构件和连接件组合起来的。

软件架构为软件系统提供了一个结构、行为和属性的高级抽象，由构成系统的元素的描述、这些元素的相互作用、指导元素集成的模式以及这些模式的约束组成。

架构的本质：

软件架构为软件系统提供了一个结构、行为和属性的高级抽象。

软件架构风格是特定应用领域的惯用模式，架构定义一个词汇表和一组约束。

架构的作用：

软件架构是项目干系人进行交流的手段，明确了对系统实现的约束条件，决定了开发和维护组织的组织结构，制约着系统的质量属性。

软件架构使推理和控制的更改更加简单，有助于循序渐进的原型设计，可以作为培训的基础。

软件架构是可传递和可复用的模型，通过研究软件架构可能预测软件的质量。

软件架构 = 软件体系结构

2 软件架构风格 (★★★★★)

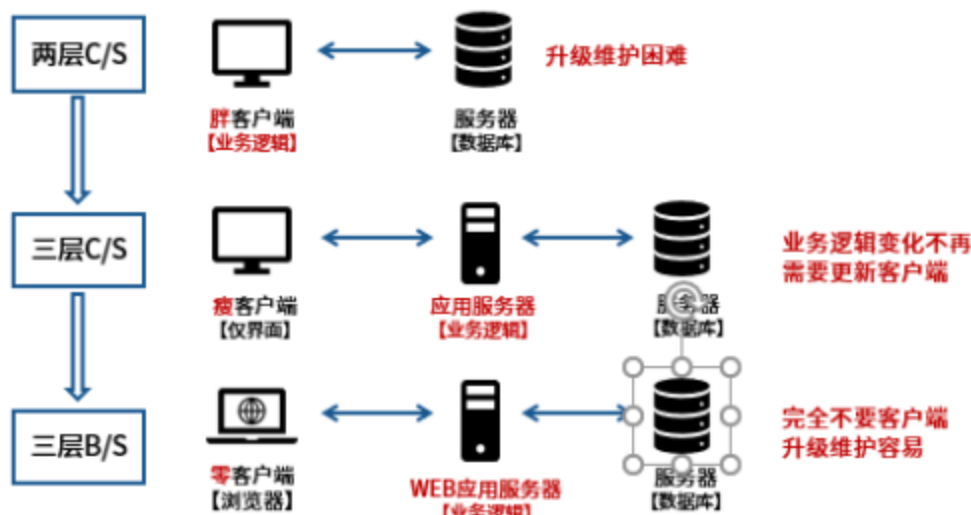
架构风格定义了用于描述系统的术语表和一组指导构建系统的规则

五大架构风格	子风格
数据流风格 【Data Flow】	批处理【Batch Sequential】、管道-过滤器【Pipes and Filters】
调用/返回风 【Call/Return】	主程序/子程序【Main Program and Subroutine】 面向对象【Object-oriented】、分层架构【Layered System】

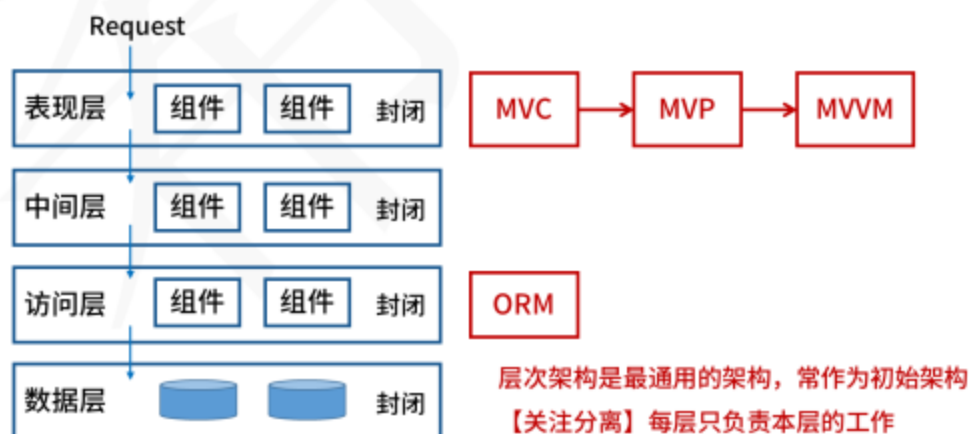
独立构件风格 【Independent Components】	进程通信【Communicating Processes】、 事件驱动系统（隐式调用）【Event system】
虚拟机风格 【Virtual Machine】	解释器【interpreter】、规则系统【Rule-based System】
以数据为中心 【Data-centered】	数据库系统【Database System】、黑板系统【Blackboard System】、 超文本系统【Hypertext System】

3 典型架构应用

3.1 层次架构



层次架构是最通用的架构，常作为初始架构，【关注分离】每层只负责本层的工作。



(1) MVC

Model（模型）是应用程序中用于处理应用程序数据逻辑的部分。通常模型对象负责在数据库中存取数据。

View（视图）是应用程序中处理数据显示的部分。通常视图是依据模型数据创建的。

Controller（控制器）是应用程序中处理用户交互的部分。通常控制器负责从视图读取数据，控制用户输入，并向模型发送数据。

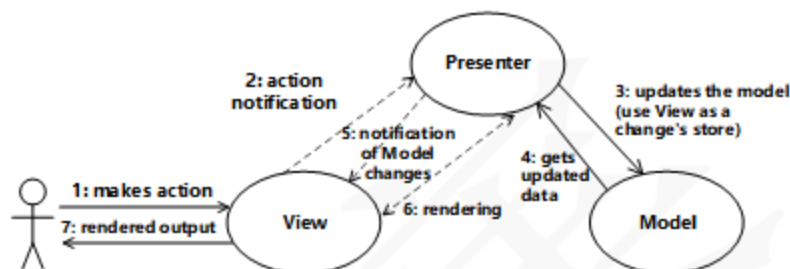
J2EE 体系结构中：

视图（View）：JSP

控制（Controller）：Servlet

模型（Model）：Entity Bean、Session Bean

(2) MVP



MVP 与 MVC 关系：MVP 是 MVC 的变种。

MVP 的优点：

模型与视图完全分离，我们可以修改视图而不影响模型。

可以更高效地使用模型，因为所有的交互都发生在一个地方——Presenter 内部。

我们可以将一个 Presenter 用于多个视图，而不需要改变 Presenter 的逻辑。这个特性非常的有用，因为视图的变化总是比模型的变化频繁。

如果我们把逻辑放在 Presenter 中，那么我们就可以脱离用户接口来测试这些逻辑（单元测试）。

3.2 富互联网应用（RIA）

RIA 结合了 C/S 架构反应速度快、交互性强的优点，以及 B/S 架构传播范围广及容易传播的特性。

RIA 简化并改进了 B/S 架构的用户交互。

数据能够被缓存在客户端，从而可以实现一个比基于 HTML 的响应速度更快且数据往返于服务器的次数更少的用户界面。

优点：反应速度快、易于传播、交互性强。

3.3 REST

REST 含义：REST（Representational State Transfer，表述性状态转移）是一种只使用 HTTP 和 XML 进行基于 Web 通信的技术，可以降低开发的复杂性，提高系统的可伸缩性。

REST 的5个原则：

网络上的所有事物都被抽象为资源。

每个资源对应一个唯一的资源标识。

通过通用的连接件接口对资源进行操作。

对资源的各种操作不会改变资源标识。

所有的操作都是无状态的。

3.4 微服务-混合风格

(1) 什么是微服务

微服务顾名思义，就是很小的服务，所以它属于面向服务架构的一种。

(2) 微服务的优势

优点	解读
【复杂应用解耦】	小服务（且专注于做一件事情） 化整为零，易于小团队开发
【独立】	独立开发 独立测试及独立部署（简单部署） 独立运行（每个服务运行在其独立进程中）
【技术选型灵活】	支持异构（如：每个服务使用不同数据库）
【容错】	故障被隔离在单个服务中，通过重试、平稳退化等机制实现应用层容错
【松耦合，易扩展】	可根据需求独立扩展

(3) 微服务面临的挑战

分布式环境下的数据一致性【更复杂】

测试的复杂性【服务间依赖测试】

运维的复杂性

(4) 微服务与 SOA 的对比

微服务	SOA
能拆分的就拆分	是整体的，服务能放一起的都放一起
纵向业务划分	是水平分多层
由单一组织负责	按层级划分不同部门的组织负责
细粒度	粗粒度
两句话可以解释明白	几百字只相当于 SOA 的目录
独立的子公司	类似大公司里面划分了一些业务单元（BU）
组件小	存在较复杂的组件
业务逻辑存在于每一个服务中	业务逻辑横跨多个业务领域
使用轻量级的通信方式，如 HTTP	企业服务总线（ESB）充当了服务之间通信的角色
微服务架构实现	SOA 实现
团队级，自底向上开展实施	企业级，自顶向下开展实施

一个系统被拆分成多个服务，粒度细	服务由多个子系统组成，粒度大
无集中式总线，松散的服务架构	企业服务总线，集中式的服务架构
集成方式简单（HTTP/REST/JSON）	集成方式复杂（ESB/WS/SOAP）
服务能独立部署	单块架构系统，相互依赖，部署复杂

3.5 云原生架构风格

（1）云计算基本概念：

云计算是集合了大量计算设备和资源，对用户屏蔽底层差异的分布式处理架构，其用户与提供实际服务的计算资源是相分离的。

云计算优点：超大规模、虚拟化、高可靠性、高可伸缩性、按需服务、成本低【前期投入低、综合使用成本也低】。

（2）分类

按服务类型分类：

SaaS【软件即服务】	基于多租户技术实现，直接提供 应用程序
PaaS【平台即服务】	虚拟中间件服务器 、运行环境和操作系统
IaaS【基础设施即服务】	包括 服务器 、存储和网络等服务

按部署方式分类：

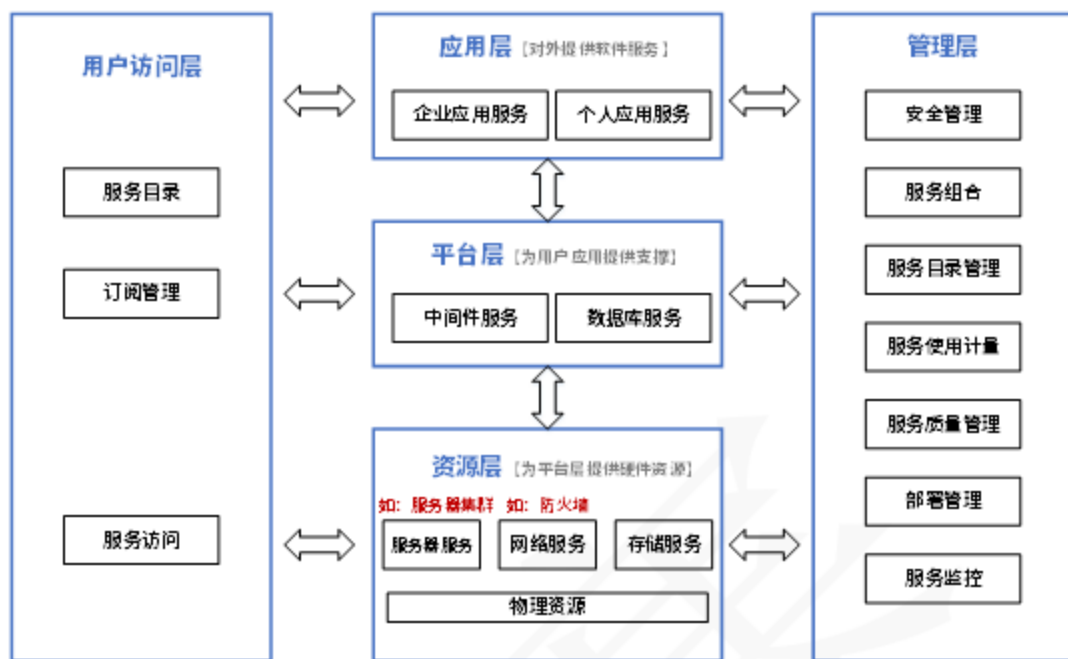
公有云：面向互联网用户需求，通过开放网络提供云计算服务

私有云：面向企业内部提供云计算服务

混合云：兼顾以上两种情况的云计算服务

（3）云计算架构

【云原生】是基于分布部署和统一运管的分布式云，以容器、微服务、DevOps 等技术为基础建立的一套云技术产品体系。



【管理层】提供对所有层次云计算服务的管理功能。

【用户访问层】方便用户使用云计算服务所需的各种支撑服务，针对每个层次的云计算服务都需要提供相应的访问接口。

【应用层】提供软件服务，如：财务管理，客户关系管理，商业智能。

【平台层】为用户提供对资源层服务的封装，使用户可以构建自己的应用。

【资源层】提供虚拟化的资源，从而隐藏物理资源的复杂性。如：服务器，存储。

3.6 边缘计算

边缘计算是指在靠近物或数据源头的一侧，采用网络、计算、存储、应用核心能力为一体的开放平台，就近提供最近端服务。

边缘计算的本质：计算处理职能的本地化。

4 特定领域软件架构 (DSSA) (★★★★)

定义：特定领域软件架构以一个特定问题领域为对象，形成由领域参考模型、参考需求、参考架构等组成的开发基础架构，支持一个特定领域中多个应用的生成。



垂直域：某一个狭小领域或者说某个行业的共性抽象。

水平域：多个行业可通用的一些共性的抽象。

5 基于架构的软件开发方法 (★★★★)

(1) 基于架构的软件设计 (ABSD)

ABSD 能很好的支持软件重用。

ABSD 方法是架构驱动，即强调由业务【商业】、质量和功能需求的组合驱动架构设计。

ABSD 方法有三个基础。第一个基础是功能的分解。在功能分解中，ABSD 方法使用已有的基于模块的内聚和耦合技术；第二个基础是通过选择架构风格来实现质量和业务需求；第三个基础是软件模板的使用。软件模板利用了一些软件系统的结构。

视角与视图：从不同的视角来检查，所以会有不同的视图。

用例用来捕获功能需求、特定场景【刺激、环境、响应】用来捕获质量需求。

6 架构评估

6.1 架构设计重点关注非功能设计（质量属性） (★★★★)

(1) 性能

性能 (performance) 是指系统的响应能力，即要经过多长时间才能对某个事件做出响应，或者在某段时间内系统所能处理的事件的个数。例如：a.同时支持1000并发；b.响应时间小于1s；c.显示分辨率达到4K。

代表参数：响应时间、吞吐量 设计策略：优先级队列、资源调度

(2) 可用性

可用性 (availability) 是系统能够正常运行的时间比例。经常用两次故障之间的时间长度或在出现故障时系统能够恢复正常的速度来表示。例如：a.主服务器故障，1分钟内切换至备用服务器；b.系统故障，1小时内修复；c.系统支持7×24小时工作。

代表参数：故障间隔时间 设计策略：冗余、心跳线

(3) 安全性

安全性 (security) 是指系统在向合法用户提供服务的同时能够阻止非授权用户使用的企图或拒绝服务的能力。安全性又可划分为机密性【信息不泄露给未授权的用户】、完整性【防止信息被篡改】、不可否认性【不可抵赖】及可控性【对信息的传播及内容具有控制的能力】等特性。例如：a.可抵御 SQL 注入攻击；b.对计算机的操作都有完整记录；c.用户信息数据库授权必须保证99.9%可用。

设计策略：追踪审计

(4) 可修改性

可修改性 (modifiability) 是指能够快速地对系统性能价格比进行变更的能力。通常以某些具体的变更为准，通过考察这些变更的代价衡量可修改性。（可扩展性与之相近）例如：a.更改系统报表模块，必须在2人周内完成；b.对 Web 界面风格进行修改，修改必须在4人月内完成。

主要策略：信息隐藏（二义性：良好的封装能够做到信息隐藏，一般归于可修改性策略；信息隐藏也能够体现在安全性当中）

(5) 易用性

易用性关注的是对用户来说完成某个期望任务的容易程度和系统所提供的用户支持的种类。例如：
a.界面友好；b.新用户学习使用系统时间不超过2小时。

(6) 可测试性

软件可测试性是指通过测试揭示软件缺陷的容易程度。例如：a.提供远程调试接口，支持远程调试。

6.2 软件架构评估方法

风险点：系统架构风险是指架构设计中潜在的、存在问题的架构决策所带来的隐患。

非风险点：是指不会带来隐患，一般以“XXX 要求是可以实现【或接受】的”方式表达。

敏感点：敏感点是一个或多个构件（和/或构件之间的关系）的特性，它能影响系统的某个质量属性。

权衡点：影响多个质量属性的特性，是多个质量属性的敏感点。

场景：场景是从风险承担者的角度与系统交互的简短描述。场景可从六个方面进行描述：刺激源、刺激、制品、环境、响应、响应度量。



刺激源 (Source)：这是某个生成该刺激的实体（人、计算机系统或者任何其他刺激器）。

刺激 (Stimulus)：该刺激是当刺激到达系统时需要考虑的条件。

环境 (Environment)：该刺激在某些条件内发生。当激励发生时，系统可能处于过载、运行或者其他情况。

制品 (Artifact)：某个制品被激励。这可能是整个系统，也可能是系统的一部分。

响应 (Response)：该响应是在激励到达后所采取的行动。

响应度量 (Measurement)：当响应发生时，应当能够以某种方式对其进行度量，以对需求进行测试。

7 产品线

7.1 特点

核心资源、产品集合、过程驱动、特定领域、技术支持、以架构为中心。

7.2 建立方式

	演化方式	革命方式
基于现有产品	基于现有产品架构设计产品线的架构，经演化现有构件，开发产品线构件	核心资源的开发基于现有产品集的需求和可预测的、将来需求的超集

全新产品线	产品线核心资源随产品新成员的需求而演化	开发满足所有预期产品线成员的需求的核心资源
-------	---------------------	-----------------------

将现有产品演化为产品线

用软件产品线替代现有产品集

全新软件产品线的演化

全新软件产品线的开发

7.3 成功实施产品线主要取决因素

对该领域具备长期和深厚的经验

一个用于构建产品的好的核心资源库

好的产品线架构

好的管理（软件资源、人员组织、过程）支持

8 大型网站系统架构演化

8.1 维度

维度	涉及技术内容
从架构来看	MVC, MVP, MWM, REST, Webservice, 微服务
从并发分流来看	集群（负载均衡）、CDN
从缓存来看	MemCache, Redis, Squid
从数据来看	主从库（主从复制），内存数据库，反规范化技术，NoSQL，分区（分表）技术，视图与物化视图
从持久化来看	Hibernate, Mybatis
从分布存储来看	Hadoop, FastDFS, 区块链
从数据编码来看	XML, JSON
从Web应用服务器来看	Apache, WebSphere, WebLogic, Tomcat, JBOSS, IIS
从安全性来看	SQL 注入攻击
其它	静态化，有状态与无状态，响应式 Web 设计，中台

8.2 缓存

（1）MemCache：MemCache 是一个高性能的分布式的内存对象缓存系统，用于动态 Web 应用以减轻数据库负载。MemCache 通过在内存里维护一个统一的巨大的 hash 表，能够用来存储各种格式的数据，包括图像、视频、文件以及数据库检索的结果等。

（2）Redis：Redis 是一个开源的使用 ANSI C 语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value 数据库，并提供多种语言的 API。

（3）Squid：Squid 是一个高性能的代理缓存服务器，Squid 支持 FTP、gopher、HTTPS 和 HTTP 协议。和一般的代理缓存软件不同，Squid 用一个单独的、非模块化的、I/O 驱动的进程来处理所有的客户端请求。

(4) Redis 和 MemCache 对比：

工作	MemCache	Redis
数据类型	简单 key/value 结构	丰富的数据结构
持久性	不支持	支持
分布式存储	客户端哈希分片/一致性哈希	多种方式，主从、Sentinel、Cluster 等
多线程支持	支持	不支持 (Redis6.0 开始支持)
内存管理	私有内存池/内存池	无
事务支持	不支持	有限支持
数据容灾	不支持，不能做数据恢复	支持，可以在灾难发生时，恢复数据

(5) Redis 集群切片方式

集群切片方式	核心特点
客户端分片	在客户端通过 key 的 hash 值对应到不同的服务器。
中间件实现分片	在应用软件和 Redis 中间，例如：Twemproxy、Codis 等，由中间件实现服务到后台 Redis 节点的路由分派。
客户端服务端协作分片	RedisCluster 模式，客户端可采用一致性哈希，服务端提供错误节点的重定向服务 slot 上。不同的 slot 对应到不同服务器。

(6) Redis 分布存储方案

分布式存储方案	核心特点
主从 (Master/Slave) 模式	一主多从，故障时手动切换。
哨兵 (Sentinel) 模式	有哨兵的一主多从，主节点故障自动选择新的主节点。
集群 (Cluster) 模式	分节点对等集群，分 slots，不同 slots 的信息存储到不同节点。

(7) Redis 分片方案

分片方案	分片方式	说明
范围分片	按数据范围值来做分片	例：按用户编号分片，0-999999 映射到实例 A；1000000-1999999 映射到实例 B。
哈希分片	通过对 key 进行 hash 运算分片	可以把数据分配到不同实例，这类似于取余操作，余数相同的，放在一个实例上。
一致性哈希分片	哈希分片的改进	可以有效解决重新分配节点带来的无法命中问题。

(8) Redis 持久化

RDB：传统数据库中快照的思想。指定时间间隔将数据进行快照存储。

AOF：传统数据库中日志的思想，把每条改变数据集的命令追加到 AOF 文件末尾，这样出问题了，可以重新执行 AOF 文件中的命令来重建数据集。

对比维度	RDB 持久化	AOF 持久化
------	---------	---------

备份量	重量级的 全量备份 ，保存整个数据库	轻量级 增量备份 ，一次只保存一个修改命令
保存间隔时间	保存 间隔时间长	保存间隔时间短，默认 1 秒
还原速度	数据还原速度快	数据还原速度慢
阻塞情况	save 会阻塞，但 bgsave 或者自动不会阻塞	无论是平时还是 AOF 重写，都不会阻塞
数据体积	同等数据体积： 小	同等数据体积： 大
安全性	数据安全性： 低 ，容易丢数据	数据安全性： 高 ，根据策略决定

(9) 淘汰机制

淘汰作用范围	机制名	策略
不淘汰	noeviction	禁止驱逐数据，内存不足以容纳新入数据时，新写入操作就会报错。系统默认的一种淘汰策略。
设置了过期时间的键空间	volatile-random	随机移除某个 key
	volatile-lru	优先移除最近未使用的 key
	volatile-ttl	ttl 值小的 key 优先移除
全键空间	allkeys-random	随机移除某个 key
	allkeys-lru	优先移除最近未使用的 key

8.3 服务集群

(1) 应用层负载均衡

http 重定向。HTTP 重定向就是应用层的请求转发。用户的请求其实已经到了 HTTP 重定向负载均衡服务器，服务器根据算法要求用户重定向，用户收到重定向请求后，再次请求真正的集群。

特点：实现简单，但性能较差。

反向代理服务器。在用户的请求到达反向代理服务器时（已经到达网站机房），由反向代理服务器根据算法转发到具体的服务器。常用的 apache, nginx 都可以充当反向代理服务器。

特点：部署简单，但代理服务器可能成为性能的瓶颈。

(2) 传输层负载均衡

DNS 域名解析负载均衡。DNS 域名解析负载均衡就是在用户请求 DNS 服务器，获取域名对应的 IP 地址时，DNS 服务器直接给出负载均衡后的服务器 IP。

特点：效率比 HTTP 重定向高，减少维护负载均衡服务器成本。但一个应用服务器故障，不能及时通知 DNS，而且 DNS 负载均衡的控制权在域名服务商那里，网站无法做更多的改善和更强大的管理。

基于 NAT 的负载均衡。基于 NAT 的负载均衡将一个外部 IP 地址映射为多个 IP 地址，对每次连接请求动态地转换为一个内部节点的地址。

特点：技术较为成熟，一般在网关位置，可以通过硬件实现。像四层交换机一般就采用了这种技术。

(3) 硬件负载均衡：F5

(4) 软件负载均衡：LVS、Nginx、HAproxy

(5) 算法分类

静态算法（不考虑动态负载）：

(1) 轮转算法：轮流将服务请求（任务）调度给不同的节点（即：服务器）。

(2) 加权轮转算法：考虑不同节点处理能力的差异。

(3) 源地址哈希散列算法：根据请求的源 IP 地址，作为散列键从静态分配的散列表找出对应的节点。

(4) 目标地址哈希散列算法：根据请求目标 IP 做散列找出对应节点。

(5) 随机算法：随机分配，简单，但不可控。

动态算法（考虑动态负载）

(1) 最小连接数算法：新请求分配给当前活动请求数量最少的节点，每个节点处理能力相同的情况下。

(2) 加权最小连接数算法：考虑节点处理能力不同，按最小连接数分配。

(3) 加权百分比算法：考虑了节点的利用率、硬盘速率、进程个数等，使用利用率来表现剩余处理能力。

Session 有状态和无状态问题

无状态服务（stateless service）对单次请求的处理，不依赖其他请求，也就是说，处理一次请求所需的全部信息，要么都包含在这个请求里，要么可以从外部获取到（比如说数据库），服务器本身不存储任何信息。

有状态服务（stateful service）则相反，它会在自身保存一些数据，先后的请求是有关联的。

8.4 数据库读写分离

主从数据库结构特点：

一般：一主多从，也可以多主多从。

主库做写操作，从库做读操作。

主从复制步骤：

主库（Master）更新数据完成前，将操作写 binlog 日志文件。

从库（Slave）打开 I/O 线程与主库连接，做 binlog dump process，并将事件写入中继日志。

从库执行中继日志事件，保持与主库一致

8.5 响应式 Web 设计

(1) 概念

响应式 WEB 设计是一种网络页面设计布局，其理念是：集中创建页面的图片排版大小，可以智能地根据用户行为以及使用的设备环境进行相对应的布局。

(2) 方法与策略

采用流式布局和弹性化设计：使用相对单位，设定百分比而非具体值的方式设置页面元素的大小。

响应式图片：不仅要同比的缩放图片，还要在小设备上降低图片自身的分辨率。

8.6 中台

概念：中台是一套结合互联网技术和行业特性，将企业核心能力以共享服务形式沉淀，形成“大中台、小前台”的组织和业务机制，供企业快速低成本地进行业务创新的企业架构。中台又可以进一步细分，比如业务中台，数据中台，XX 中台。本质上，都是对企业通用能力在不同层面的沉淀，并对外能力开放。

业务中台：提供重用服务，例如学员中心、课程中心之类的开箱即用可重用能力。

数据中台：提供数据整合分析能力，帮助企业从数据中学习改进，调整方向。

技术中台：提供技术重用组件能力，帮助解决基础技术平台的复用。如：中间件，分布式存储，AI，负载均衡等基础设施。

数据中台必备的4个核心能力

- 1、数据汇聚整合能力
- 2、数据提纯加工能力
- 3、数据服务可视化
- 4、价值变现方面

第四章 信息安全技术基础知识

1 信息加解密技术

1.1 对称加密

对称加密（又称为私人密钥加密/共享密钥加密）：加密与解密使用同一密钥。

特点：加密强度不高，但效率高；密钥分发困难。

（大量明文为了保证加密效率一般使用对称加密）

常见对称密钥加密算法：

DES：替换+移位、56位密钥、64位数据块、速度快、密钥易产生

3DES（三重 DES）：两个56位的密钥 K1、K2

加密：K1加密->K2解密->K1加密

解密：K1解密->K2加密->K1解密 RC-5

IDEA：128位密钥、64位数据块、比 DES 的加密性好、对计算机功能要求相对低，PGP。

RC-5算法：RSA 数据安全公司的很多产品都使用了 RC-5。

AES 算法：高级加密标准，又称 Rijndael 加密法，是美国政府采用的一种区块加密标准

1.2 非对称加密

概念：

非对称加密（又称为公开密钥加密）：密钥必须成对使用（公钥加密，相应的私钥解密）。

特点：加密速度慢，但强度高。秘钥分发容易。

常见非对称密钥加密算法

RSA：2048位（或1024位）密钥、计算量极大、难破解

ECC-椭圆曲线算法

Elgamal：安全性依赖于计算有限域上离散对数这一难题

2 数字签名技术 (☆☆)

(1) 信息摘要：单向散列函数【不可逆】、固定长度的散列值。

摘要用途：确保信息【完整性】，防篡改。

常用的消息摘要算法有 MD5, SHA 等，市场上广泛使用的 MD5, SHA 算法的散列值分别为128和160位，由于 SHA 通常采用的密钥长度较长，因此安全性高于 MD5。

(2) 数字签名的过程：发送者使用自己的私钥对摘要签名，接收者利用发送者的公钥对接收到的摘要进行验证

3 安全架构概述

(1) 被动攻击：收集信息为主，破坏保密性

攻击类型	攻击名称	描述
被动攻击	窃听（网络监听）	用各种可能的合法或非法的手段窃取系统中的信息资源和敏感信息。
	业务流分析	通过对系统进行长期监听，利用统计分析方法对诸如通信频度、通信的信息流向、通信总量的变化等参数进行研究，从而发现有价值的信息和规律。
	非法登录	有些资料将这种方式归为被动攻击方式。

(2) 主动攻击：主动攻击的类别主要有：中断（破坏可用性），篡改（破坏完整性），伪造（破坏真实性）。

攻击类型	攻击名称	描述
主动攻击	假冒身份	非法用户冒充成为合法用户，特权小的用户冒充成为特权大的用户。
	抵赖	否认自己曾经发布过的某条消息、伪造一份对方来信等。
	旁路控制 【旁路攻击】	密码学中是指绕过对加密算法的繁琐分析，利用密码算法的硬件实现的运算中泄露的信息。如执行时间、功耗、电磁辐射等，结合统计理论快

		速的破解密码系统。
	重放攻击	所截获的某次合法的通信数据拷贝，出于非法的目的而被重新发送。加【时间戳】能识别并应对重放攻击。
	拒绝服务 (DOS)	破坏服务的【可用性】，对信息或其他资源的合法访问被无条件的阻止。
	XSS 跨站脚本攻击	通过利用网页【开发时留下的漏洞】，通过巧妙的方法注入恶意指令代码到网页。
	CSRF 跨站请求伪造攻击	攻击者通过一些技术手段欺骗用户的浏览器与访问一个自己曾经认证过的网站并执行一些操作（如转账或购买商品等）。
	缓冲区溢出攻击	利用【缓冲区溢出漏洞】所进行的攻击。在各种操作系统、应用软件中广泛存在。
	SQL 注入攻击	攻击者把 SQL 命令插入到 Web 表单,欺骗服务器执行恶意的 SQL 命令。 SQL 注入攻击的方式: 【恶意拼接查询】、【利用注释执行非法命令】、【传入非法参数】、【添加额外条件】。 抵御 SQL 攻击的方式包括: 【使用正则表达式】、【使用参数化的过滤性语句】、【检查用户输入的合法性】、【用户相关数据加密处理】、【存储过程来执行查询】、【使用专业的漏洞扫描工具】。

4 区块链技术

(1) 【区块链】 ≠ 比特币，比特币底层采用了区块链技术。比特币交易在我国定性为【非法运用】。

(2) 区块链的特点：

去中心化：由于使用分布式核算和存储，不存在中心化的硬件或管理机构，任意节点的权利和义务都是均等的，系统中的数据块由整个系统中具有维护功能的节点来共同维护。

开放性：系统是开放的，如：区块链上的【交易信息是公开的】，不过【账户身份信息是高度加密的】。

自治性：区块链采用基于协商一致的规范和协议（比如一套公开透明的算法）使得整个系统中的所有节点能够在去信任的环境自由安全的交换数据，使得对“人”的信任改成了对机器的信任，任何人为的干预不起作用。

安全性（信息不可篡改）：数据在多个节点存储了多份，篡改数据得改掉51%节点的数据，这太难。同时，还有其它安全机制，如：比特币的每笔交易，都由付款人用私钥签名，证明确实是他同意向某人付款，其它人无法伪造。

匿名性（去信任）：由于节点之间的交换遵循固定的算法，其数据交互是无需信任的（区块链中的程序规则会自行判断活动是否有效），因此交易对手无须通过公开身份的方式让对方自己产生信任，对信用的累积非常有帮助。

第五章 项目管理

1 进度网络图-关键路径法（PERT）

（1）总时差（松弛时间）：

在不延误总工期的前提下，该活动的机动时间。活动的总时差等于该活动最迟完成时间与最早完成时间之差，或该活动最迟开始时间与最早开始时间之差。

（2）自由时差：

在不影响紧后活动的最早开始时间前提下，该活动的机动时间。

对于有紧后活动的活动，其自由时差等于所有紧后活动最早开始时间减本活动最早完成时间所得之差的最小值。

对于没有紧后活动的活动，也就是以网络计划终点节点为完成节点的活动，其自由时差等于计划工期与本活动最早完成时间之差。

对于网络计划中以终点节点为完成节点的活动，其自由时差与总时差相等。此外，由于活动的自由时差是其总时差的构成部分，所以，当活动的总时差为零时，其自由时差必然为零，可不必进行专门计算。

2 软件质量管理（★★★★★）

2.1 软件质量控制与质量保证

（1）质量保证一般是每隔一定时间（例如，每个阶段末）进行的，主要通过系统的质量审计和过程分析来保证项目的质量。独特工具包括：质量审计和过程分析。

（2）质量控制是实时监控项目的具体结果，以判断它们是否符合相关质量标准，制定有效方案，以消除产生质量问题的原因。

（3）质量保证的主要目标

【事前预防】工作。

尽量在刚刚引入缺陷时即将其捕获，而不是让缺陷扩散到下一个阶段。

作用于【过程】而【不是最终产品】。

贯穿于【所有的活动之中】，而不是只集中于一点。

3 软件配置管理（★）

（1）产品配置是指一个产品在其生命周期各个阶段所产生的各种形式（机器可读或人工可读）和各种版本的文档、计算机程序、部件及数据的集合。

（2）关于配置项

基线配置项（可交付成果）：需求文档、设计文档、源代码、可执行代码测试用例、运行软件所需数据等

非基线配置项：各类计划（如项目管理计划，进度管理计划）、各类报告

软件配置管理核心内容包括【版本控制】和【变更控制】。

按软件过程活动将软件工具分为：

软件开发工具：需求分析工具、设计工具、编码与排错工具。

软件维护工具：版本控制工具（VSS、CVS、SCCS、SVN）、文档分析工具、开发信息库工具、逆向工程工具、再工程工具。

软件管理和软件支持工具：项目管理工具、配置管理工具、软件评价工具、软件开发工具的评价和选择。

第六章 计算机系统基础知识

1 存储系统

时间局部性：指程序中的某条指令一旦执行，不久以后该指令可能再次执行，典型原因是由于程序中存在大量的循环操作。

空间局部性：指一旦程序访问了某个存储单元，不久以后，其附近的存储单元也将被访问，即程序在一段时间内所访问的地址可能集中在一定的范围内，其典型情况是程序顺序执行。

工作集理论：工作集是进程运行时被频繁访问的页面集合。

2 操作系统概述

2.1 特殊的操作系统

分类	特点
批处理操作系统	单道批：一次一个作业入内存，作业由程序、数据、作业说明书组成 多道批：一次多个作业入内存，特点：多道、宏观上并行微观上串行
分时操作系统	采用时间片轮转的方式为多个用户提供服务，每个用户感觉独占系统 特点：多路性、独立性、交互性和及时性
实时操作系统	实时控制系统和实时信息系统

	交互能力要求不高，可靠性要求高（规定时间内响应并处理）
网络操作系统	方便有效共享网络资源，提供服务软件和有关协议的集合 主要的网络操作系统有：Unix、Linux 和 Windows Server 系统
分布式操作系统	任意两台计算机可以通过通信交换信息 是网络操作系统的更高级形式，具有透明性、可靠性和高性能等特性
微机操作系统	Windows：Microsoft 开发的图形用户界面、多任务、多线程操作系统 Linux：免费使用和自由传播的类 Unix 操作系统，多用户、多任务、多线程和多 CPU 的操作系统
嵌入式操作系统	运行在智能芯片环境中 特点：微型化、可定制（针对硬件变化配置）、实时性、可靠性、易移植性（HAL 和 BSP 支持）

2.2 存储管理 (☆☆)

(1) 页式存储：将程序与内存均划分为同样大小的块，以页为单位将程序调入内存。

优点：利用率高，碎片小，分配及管理简单

缺点：增加了系统开销；可能产生抖动现象

(2) 段式存储：按用户作业中的自然段来划分逻辑空间，然后调入内存，段的长度可以不一样。

优点：多道程序共享内存，各段程序修改互不影响

缺点：内存利用率低，内存碎片浪费大

(3) 段页式存储：段式与页式的综合体。先分段，再分页。1个程序有若干个段，每个段中可以有若干页，每个页的大小相同，但每个段的大小不同。

优点：空间浪费小、存储共享容易、存储保护容易、能动态连接

缺点：由于管理软件的增加，复杂性和开销也随之增加，需要的硬件以及占用的内容也有所增加，使得执行速度大大下降

2.3 磁盘管理

(1) 存取时间=寻道时间+等待时间，寻道时间是指磁头移动到磁道所需的时间；等待时间为等待读写的扇区转到磁头下方所用的时间。

(2) 读取磁盘数据的时间应包括以下三个部分：

找磁道的时间。

找块（扇区）的时间，即旋转延迟时间。

传输时间。

(3) 磁盘移臂调度算法：

先来先服务 FCFS（谁先申请先服务谁）；

最短寻道时间优先 SSTF（申请时判断与磁头当前位置的距离，谁短先服务谁）；

扫描算法 SCAN（电梯算法，双向扫描）；

循环扫描 CSCAN（单向扫描）。

3 文件系统（☆☆）

（1）文件（File）是具有符号名的、在逻辑上具有完整意义的一组相关信息项的集合，例如，一个源程序、一个目标程序、编译程序、一批待加工的数据和各种文档等都可以各自组成一个文件。

一个文件包括文件体和文件说明。

- 文件体是文件真实的内容；
- 文件说明是操作系统为了管理文件所用到的信息，包括文件名、文件内部标识、文件类型、文件存储地址、文件长度、访问权限、建立时间和访问时间等。

（2）文件的类型

按文件的性质和用途分类可将文件分为系统文件、库文件和用户文件。

按信息保存期限分类可将文件分为临时文件、档案文件和永久文件。

按文件的保护方式分类可将文件分为只读文件、读/写文件、可执行文件和不保护文件。

4 性能评估（☆☆☆）

方法	描述	特点
时钟频率法	以时钟频率高低衡量速度	仅考虑 CPU
指令执行速度法	表示机器运算速度的单位是 MIPS	仅考虑 CPU
等效指令速度法 (吉普森混合法)	通过各类指令在程序中所占的比例 (W_i) 进行计算得到的。	仅考虑 CPU， 综合考虑指令比例不同的问题。
数据处理速率法 (PDR)	PDR 值的方法来衡量机器性能，PDR 值越大，机器性能越好。	$PDR = L/R$ 仅考虑 CPU+存储
综合理论性能法 (CTP)	CTP 用 MTOPS 表示。CTP 的估算方法是，首先算出处理部件每个计算单元的有效计算率，再按不同字长加以调整，得出该计算单元的理论性能，所有组成该处理部件的计算单元的理论性能之和即为 CTP。	仅考虑 CPU+存储
基准程序法	把应用程序中用得最多、最频繁的那部分核心程序作为评估计算机系统性能的标准程序，称为 基准测试程序 (benchmark) 。	综合考虑多部分 ，基准程序法是目前一致承认的测试系统性能的较好方法。

【测试精确度排名】 真实的程序 > 核心程序 > 小型基准程序 > 合成基准程序

常见的 Web 服务器性能评测方法有基准性能测试、压力测试和可靠性测试。

系统监视：进行系统监视通常有3种方式：一是通过系统本身提供的命令，如 UNIX/Linux 系统中的 W、ps、last，Windows 中的 netstat 等；二是通过系统记录文件查阅系统在特定时间内的运行状态；三是集成命令、文件记录和可视化技术的监控工具，如 Windows 的 Perfmon 应用程序。

第七章 嵌入式系统

1 嵌入式系统概述

1.1 基本概念

(1) 嵌入式系统是以应用为中心、以计算机技术为基础，并将可配置与可裁剪的软、硬件集成于一体的专用计算机系统，需要满足应用对功能、可靠性、成本、体积和功耗等方面的严格要求。

(2) 从计算机角度看，嵌入式系统是指嵌入各种设备及应用产品内部的计算机系统。它主要完成信号控制的功能，体积小、结构紧凑，可作为一个部件埋藏于所控制的装置中。

(3) 一般嵌入式系统由嵌入式处理器、相关支撑硬件、嵌入式操作系统、支撑软件以及应用软件组成。

嵌入式系统初始化过程：片级初始化→板级初始化→系统级初始化

1.2 嵌入式系统组成部件

嵌入式微处理器（MCU）、存储器（RAM/ROM）、内（外）总线逻辑、定时/计数器、看门狗电路（定时器溢出则中断，系统复位处理）、I/O 接口（串口、网络、USB、JTAG 接口--用来进行 CPU 调试的常用接口）、外部设备（UART、LED 等）、其他部件

2 嵌入式硬件

2.1 嵌入式微处理器分类

通常嵌入式处理器的选择还要根据使用场景不同选择不同类型的处理器，从处理器分类看，大致可分为 MPU、MCU、DSP、GPU、SoC

(1) 微处理器（Micro Processor Unit, MPU）：将微处理器装配在专门设计的电路板上，只保留与嵌入式应用有关的母板功能。微处理器一般以某一种微处理内核为核心，每一种衍生产品的处理器内核都是一样的，不同的是存储器和外设的配置和封装。

(2) 微控制器（Micro Control Unit, MCU）：又称单片机。与 MPU 相比 MCU 的最大优点在于单片化，体积大大减小，从而使功耗和成本下降，可靠性提高。

(3) 信号处理器（Digital Signal Processor, DSP）：DSP 处理器对系统结构和指令进行了特殊设计（通常，DSP 采用一种哈佛结构），使其适合于执行 DSP 算法，编译效率高，指令执行速度也高。

(4) 图形处理器（Graphics Processing Unit, GPU）：

GPU 是图形处理单元的缩写，是一种可执行染3D 图形等图像的半导体芯片（处理器）。

GPU 可用于个人电脑、工作站、游戏机和一些移动设备上做图像和图形相关运算工作的微处理器。

它可减少对 CPU 的依赖，并进行部分原本 CPU 的工作，尤其是在3D 图形处理中，GPU 采用了核心技术（如：硬件 T&L、纹理压缩等）保证了快速3D 渲染能力。

GPU 目前已广泛应用于各行各业，GPU 中集成了同时运行在 GHz 的频率上的成千上万个 core，可以高速处理图像数据。最新的 GPU 峰值性能可高达100 TFlops 以上。

(5) 片上系统（System on Chip, SoC）：

追求产品系统最大包容的集成器件。

它是一个产品，是一个有专用目标的集成电路，其中包含完整系统并有嵌入软件的全部内容。

同时它又是一种技术，用以实现从确定系统功能开始，到软/硬件划分，并完成设计的整个过程。成功实现了软硬件的无缝结合，直接在微处理器片内嵌入操作系统的代码模块。减小了系统的体积和功耗、提高了可靠性和设计生产效率。

2.2 AI 芯片

人工智能（Artificial Intelligence, AI）芯片的定义：从广义上讲只要能够运行人工智能算法的芯片都叫作 AI 芯片。但是通常意义上的 AI 芯片指的是针对人工智能算法做了特殊加速设计的芯片，现阶段，这些人工智能算法一般以深度学习算法为主，也可以包括其它机器学习算法。

人工智能芯片四大类（按技术架构分类）：

GPU

FPGA（现场可编程门阵列）

ASIC（专用集成电路）

类脑芯片

AI 芯片的关键特征：

新型的计算范式：AI 计算既不脱离传统计算，也具有新的计算特质

训练和推断：AI 系统通常涉及训练和推断过程

大数据处理能力：满足高效能机器学习的数据处理要求

数据精度：降低精度的设计

可重构的能力：针对特定领域而不针对特定应用的设计，可以通过重新配置，适应新的 AI 算法、架构和任务

开发工具：AI 芯片需要软件工具链的支持

2.3 嵌入式微处理器体系结构

体系结构分类	定义	特点	典型应用
冯·诺依曼结构	冯·诺依曼结构也称普林斯顿结构，是一种将程序指令存储器和数据存储器合并在一起的存储器结构。	指令与数据存储器合并在一起。 指令与数据都通过相同的数据总线传输。	一般用于 PC 处理器，如 I3、I5、I7 处理器。 注：常规计算机属于冯·诺依曼结构

哈佛结构	哈佛结构是一种并行体系结构，它的主要特点是将程序和数据存储在不同的存储空间中，即程序存储器和数据存储器是两个独立的存储器，每个存储器独立编址、独立访问。	指令与数据分开存储，可以并行读取，有较高数据的吞吐率。 有4条总线：指令和数据的数据总线与地址总线。	一般用于嵌入式系统处理器。 注：DSP 属于哈佛结构
-------------	--	---	--------------------------------------

2.4 总线

总线的基本概念：总线是一组能为多个部件分时共享的信息传送线，用来连接多个部件并为之提供信息交换通路。

特点：

挂接在总线上的多个部件只能分时向总线发送数据，但可同时从总线接收数据。

通过总线复用方式可以减少总线中信号线的数量，以较少的信号线传输更多的信息。

总线分类：

- (1) 从功能上来对总线进行划分：数据总线、地址总线和控制总线
- (2) 从数据传输的方式划分为并行总线和串行总线

3 嵌入式操作系统

3.1 嵌入式操作系统的定义及特点

嵌入式操作系统（Embedded Operating System，EOS）是指用于嵌入式系统的操作系统。嵌入式操作系统是一种用途广泛的系统软件，负责嵌入式系统的全部软、硬件资源分配、任务调度、控制、协调并行活动等工作。通常包括与硬件相关的底层驱动软件、系统内核、设备驱动接口、通信协议、图形界面、标准化浏览器等。

根据系统对时间的敏感程度可将嵌入式系统划分为：

- (1) 嵌入式非实时系统
- (2) 嵌入式实时系统：能够在指定或者确定的时间内完成系统功能和外部或内部、同步或异步时间做出响应的系统。

嵌入式操作系统具有一般操作系统的功能，同时具有嵌入式软件的特点，主要有：

- (1) 微型化
- (2) 代码质量高
- (3) 专业化
- (4) 实时性强
- (5) 可裁减、可配置。

实时操作系统的核心特点是实时性强。

嵌入式实时操作系统实时性的评价指标

- ✓ 中断响应和延迟时间
- ✓ 任务切换时间
- ✓ 信号量混洗时间

3.2 嵌入式实时操作系统调度算法

抢占式优先级调度算法：根据任务的紧急程度确定该任务的优先级。大多数 RTOS 调度算法都是抢占方式（可剥夺方式）。

最早截止期调度算法（EDF 算法）：根据任务的截止时间头端来确定其优先级，对于时间期限最近的任务，分配最高的优先级。

最晚截止期调度算法：根据任务的截止时间末端来确定其优先级，对于时间期限最近的任务，分配最高的优先级。

3.3 操作系统内核架构

内核是操作系统的核心部分，它管理着系统的各种资源。内核可以看成连接应用程序和硬件的一座桥梁，是直接运行在硬件上的最基础的软件实体。

目前从内核架构来划分，可分为宏内核（Monolithic Kernel）和微内核（Micro Kernel）。

第八章 数据库系统

1 数据库概述

1.1 数据库模式

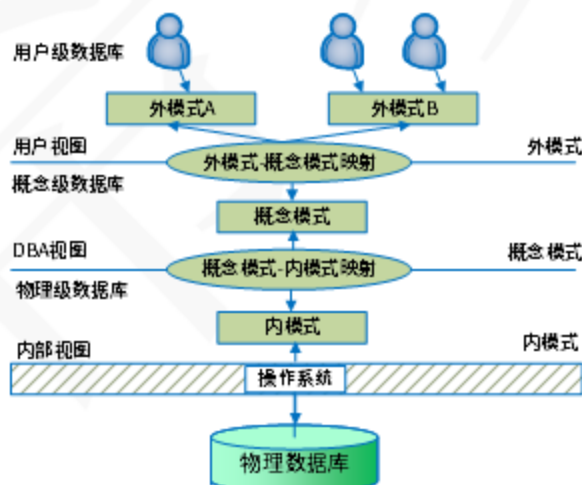
(1) 三级模式：外模式对应视图，模式（也称为概念模式）对应数据库表，内模式对应物理文件。

(2) 两层映像：外模式-模式映像，模式-内模式映像；两层映像可以保证数据库中的数据具有较高的逻辑独立性和物理独立性。

(3) 物理独立性：即数据库的内模式发生改变时，应用程序不需要改变。

(4) 逻辑独立性：即逻辑结构发生改变时，用户程序不需要改变。（逻辑独立性比物理独立性更难实现）

(5) 聚簇索引会影响内模式



1.2 分布式数据库

(1) 分布式数据库特点：

1.数据独立性。除了数据的逻辑独立性与物理独立性外，还有数据分布独立性（分布透明性）。

2.集中与自治共享结合的控制结构。各局部的 DBMS 可以独立地管理局部数据库，具有自治的功能。同时，系统又设有集中控制机制，协调各局部 DBMS 的工作，执行全局应用。

3.适当增加数据冗余度。在不同的场地存储同一数据的多个副本,可以提高系统的可靠性和可用性,同时也能提高系统性能。

(提高系统的可用性,即当系统中某个节点发生故障时,因为数据有其他副本在非故障场地上,对其他场地来说,数据仍然是可用的,从而保证数据的完备性。

4.全局的一致性、可串行性和可恢复性。

(2) 分布式透明性

分片透明性:分不分片,用户感受不到(不关心如何分片存储)。

(水平分片:按记录分;垂直分片:按字段分;混合分片)

位置透明性:数据存放在哪里,用户不用管(用户无需知道数据存放的物理位置)。

局部数据模型透明性(逻辑透明):用户或应用程序无需知道局部场地使用的是哪种数据模型。

两阶段提交协议 2PC:

2PC 事务提交的两个阶段:

表决阶段,目的是形成一个共同的决定。

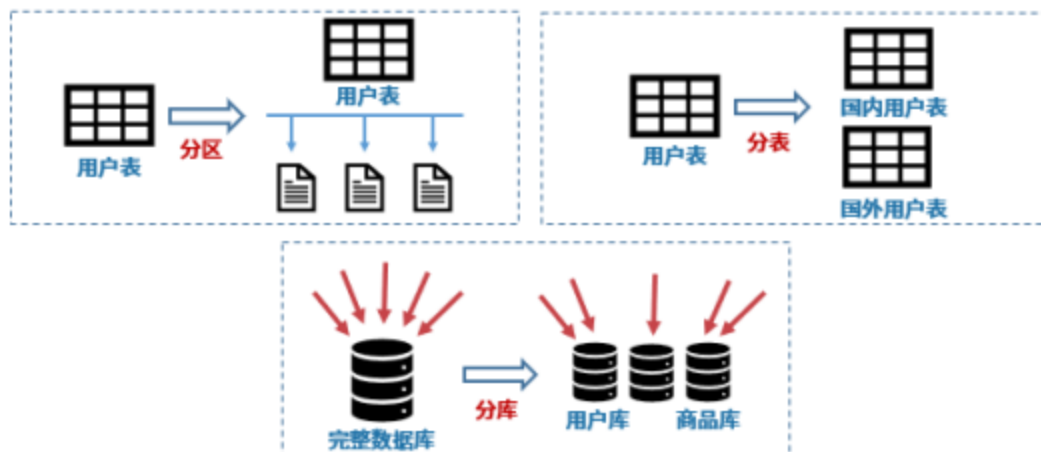
执行阶段,目的是实现这个协调者的决定。

两条全局提交规则:

只要有一个参与者撤销事务,协调者就必须做出全局撤销决定。

只有所有参与者都同意提交事务,协调者才能做出全局提交决定。

(3) 分库分区分表



	分区	分表
共性	1、都针对数据表 2、都使用了分布式存储 3、都提升了查询效率 4、都降低数据库的频繁 I/O 压力值	

差异	逻辑上还是一张表	逻辑上已是多张表
----	----------	----------

分区的常见方式

分区策略	分区方式	说明
范围分区 【RANGE】	按数据范围值来做分区	例：按用户编号分区，0-999999映射到分区 A；1000000-1999999映射到分区 B。
散列分区 【HASH】	通过对key进行hash运算分区	例：可以把数据分配到不同分区，这类似于取余操作，余数相同的，放在一个分区上。
列表分区 【LIST】	根据某字段的某个具体值进行分区	例：长沙用户分成一个区，北京用户一个区

分区的优点：

- 1、相对于单个文件系统或是硬盘，分区可以存储更多的数据。
- 2、数据管理比较方便，比如要清理或废弃某年的数据，就可以直接删除该日期的分区数据即可。
- 3、精准定位分区查询数据，不需要全表扫描查询，大大提高数据检索效率。
- 4、可跨多个分区磁盘查询，来提高查询的吞吐量。
- 5.在涉及聚合函数查询时，可以很容易进行数据的合并。

(5) 分布式数据库管理系统-组成：LDBMS、GDBMS、全局数据字典、通信管理（CM）

(6) 分布式数据库管理系统-结构

全局控制集中的 DDBMS

全局控制分散的 DDBMS

全局控制部分分散的 DDBMS

1.3 索引和视图

(1) 关系的3种类型

基本关系（通常又称为基本表或基表）：实际存在的表，实际存储数据的逻辑表示。

查询表：查询结果对应的表。

视图表：由基表或其他视图表导出的表，本身不独立存储，数据库只存放它的定义，常称为虚表。它是一个虚拟表（逻辑上的表），其内容由查询定义（仅保存 SQL 查询语句）。同真实的表一样，视图包含一系列带有名称的列和行数据。但是，视图并没有真正存储这些数据，而是通过查询原始表动态生成所需要的数据。

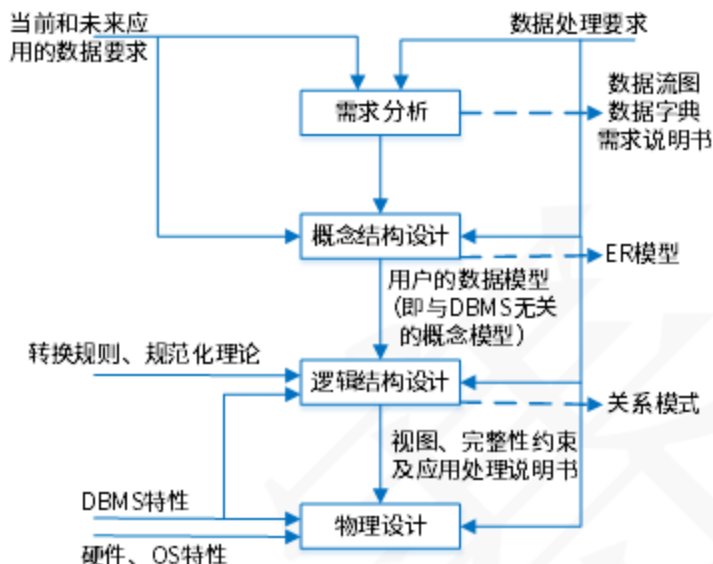
(2) 视图（View）并不在数据库中实际存在，而是一种虚拟表。

(3) 视图的优点：

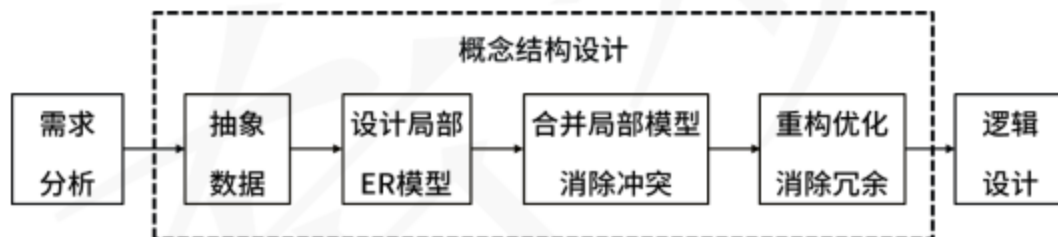
- 1、视图能简化用户的操作
- 2、视图机制可以使用户以不同的方式查询同一数据
- 3、视图对数据库重构提供了一定程度的逻辑独立性
- 4、视图可以对机密的数据提供安全保护

(4) 物化视图：它不是传统意义上虚拟视图，是实体化视图，其本身会存储数据。同时当原始表中的数据更新时，物化视图也会更新。

2 数据库设计过程



2.1 概念结构设计过程



(1) E-R 图集成的方法：

多个局部 E-R 图一次集成。

逐步集成，用累加的方式一次集成两个局部 E-R。

(2) 集成产生的冲突及解决办法：

属性冲突：包括属性域冲突和属性取值冲突。

命名冲突：包括同名异义和异名同义。

结构冲突：包括同一对象在不同应用中具有不同的抽象，以及同一实体在不同局部 E-R 图中所包含的属性个数和属性排列次序不完全相同。

2.2 逻辑结构设计

(1) 任务

1、E-R 图向关系模式的转换；

- 实体向关系模式的转换
- 联系向关系模式的转换

- 2、关系模式的规范化；
- 3、确定完整性约束（保证数据的正确性）；

实体完整性约束

参照完整性约束

用户自定义完整性约束

触发器

- 4、用户视图的确定（提高数据的安全性和独立性）。

- 根据数据流图确定处理过程使用的视图
- 根据用户类别确定不同用户使用的视图

- 5、应用程序设计

(2) 相关概念

目或度：关系模式中属性的个数。

候选码（候选键）

主码（主键）

主属性与非主属性：组成候选码的属性就是主属性，其它的就是非主属性。

外码（外键）

全码（ALL-Key）：关系模式的所有属性组是这个关系的候选码。

简单属性与复合属性、派生属性、多值属性

2.3 规范化理论

2.3.1 非规范化存在的问题

非规范化的关系模式，可能存在的问题包括：数据冗余、更新异常、插入异常、删除异常。

2.3.2 Armstrong 公理

对关系模式 $R \langle U, F \rangle$ 来说有以下的推理规则：

自反律（Reflexivity）：若 $Y \subseteq X \subseteq U$ ，则 $X \rightarrow Y$ 成立。

增广律（Augmentation）：若 $Z \subseteq U$ 且 $X \rightarrow Y$ ，则 $XZ \rightarrow YZ$ 成立。

传递律（Transitivity）：若 $X \rightarrow Y$ 且 $Y \rightarrow Z$ ，则 $X \rightarrow Z$ 成立。

根据 A1, A2, A3 这三条推理规则可以得到下面三条推理规则：

合并规则：由 $X \rightarrow Y$, $X \rightarrow Z$ ，有 $X \rightarrow YZ$ 。（A2, A3）

伪传递规则：由 $X \rightarrow Y$, $WY \rightarrow Z$ ，有 $XW \rightarrow Z$ 。（A2, A3）

分解规则：由 $X \rightarrow Y$ 及 $Z \subseteq Y$ ，有 $X \rightarrow Z$ 。（A1, A3）

2.3.3 范式

第一范式（1NF）：在关系模式 R 中，当且仅当所有域只包含原子值，即每个属性都是不可再分的数据项，则称关系模式 R 是第一范式。

第二范式（2NF）--消除非主属性对码的部分函数依赖：当且仅当关系模式 R 是第一范式（1NF），且每一个非主属性完全依赖候选键（没有不完全依赖）时，则称关系模式 R 是第二范式。

第三范式（3NF）--消除非主属性对码的传递函数依赖：当且仅当关系模式 R 是第二范式（2NF），且 R 中没有非主属性传递依赖于候选键时，则称关系模式 R 是第三范式。

BC 范式（BCNF）--根据定义判断：设 R 是一个关系模式，F 是它的依赖集，R 属于 BCNF 当且仅当其 F 中每个依赖的决定因素必定包含 R 的某个候选码。

（可以理解为在 3NF 基础上，消除主属性之间的传递函数依赖和部分函数依赖）

2.3.4 规范化过程-模式分解

保持函数依赖分解：设数据库模式 $\rho=\{R_1, R_2, \dots, R_k\}$ 是关系模式 R 的一个分解，F 是 R 上的函数依赖集， ρ 中每个模式 R_i 上的 FD 集是 F_i 。如果 $\{F_1, F_2, \dots, F_k\}$ 与 F 是等价的（即相互逻辑蕴涵），那么称分解 ρ 保持 FD。

无损分解：指将一个关系模式分解成若干个关系模式后，通过自然连接和投影等运算仍能还原到原来的关系模式。

2.3.5 反规范化

（1）反规范化手段

技术手段	说明
增加派生性冗余列	已有单价和数量列，增加“总价”列
增加冗余列	已有学号列，增加“姓名”列
重新组表	把拆分的表重新组表
分割表	把用户表做水平分割，长沙的用户存在长沙，上海的用户存在上海

（2）反规范化的优缺点

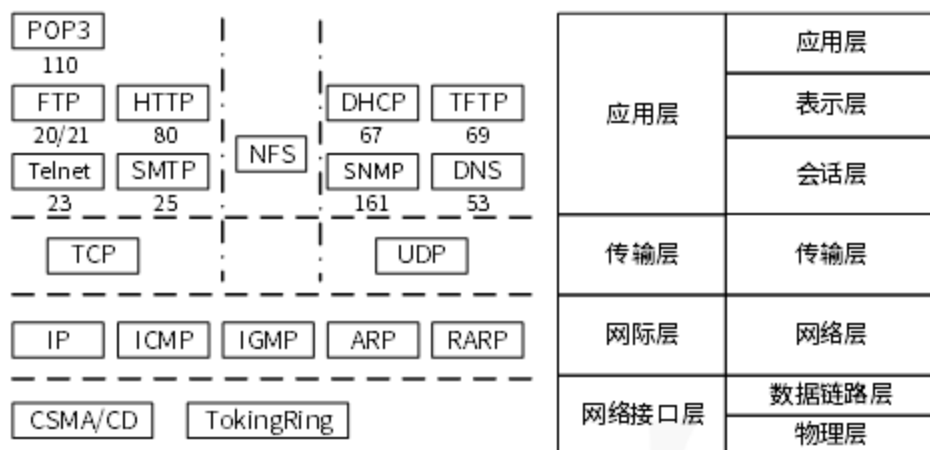
优点：连接操作少，检索快，统计快，需要查的表减少，检索容易。

缺点：

反规范化的缺点	解决方案
数据冗余，需要更大存储空间	无解
插入、更新、删除操作开销更大	无解
数据不一致 可能产生添加、修改、删除异常	1、触发器数据同步 2、应用程序数据同步
更新和插入代码更难写	无解

第九章 计算机网络

1 常见协议及功能



网际层是整个 TCP/IP 体系结构的关键部分，其功能是使主机可以把分组发往任何网络并使分组独立地传向目标。

POP3: 110端口，邮件收取

SMTP: 25端口，邮件发送

FTP: 20数据端口/21控制端口，文件传输协议

HTTP: 80端口，超文本传输协议，网页传输

DHCP: 67端口，IP 地址自动分配

SNMP: 161端口，简单网络管理协议

DNS: 53端口，域名解析协议，记录域名与 IP 的映射关系

TCP: 可靠的传输层协议，TCP 协议可以依据端口号将报文交付给上层的某一进程，可以对应用层进程进行寻址。

UDP: 不可靠的传输层协议

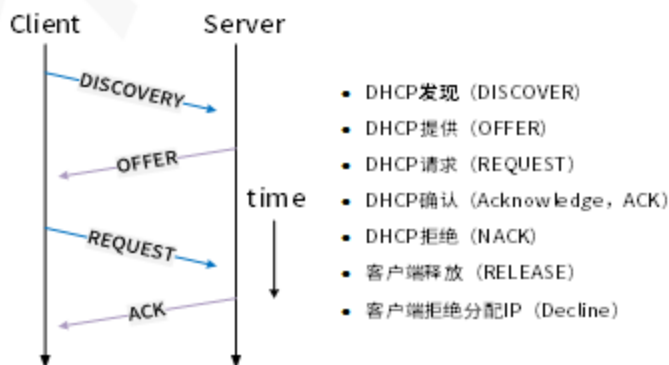
ICMP: 因特网控制协议，PING 命令来自该协议

IGMP: 组播协议

ARP: 地址解析协议，IP 地址转换为 MAC 地址

RARP: 反向地址解析协议，MAC 地址转 IP 地址

2 DHCP



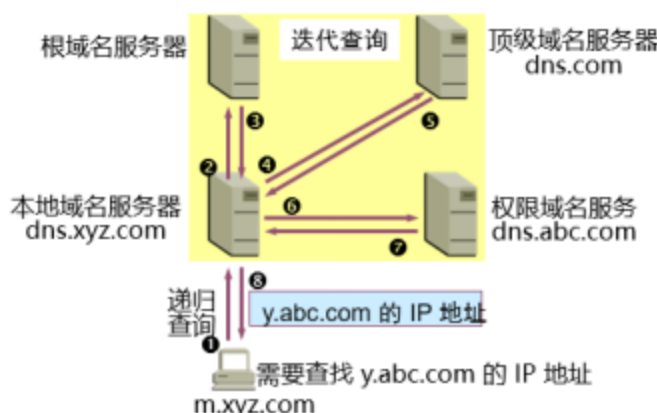
动态主机配置协议 (DHCP, Dynamic Host Configuration Protocol)

- (1) 客户机/服务器模型
- (2) 租约默认为8天
- (3) 当租约过半时，客户机需要向 DHCP 服务器申请续租；
- (4) 当租约超过87.5%时，如果仍然没有和当初提供 IP 的 DHCP 服务器联系上，则开始联系其他的 DHCP 服务器。

(5) 固定分配、动态分配和自动分配。

(6) 分配失败则 IP 显示为169.254.X.X (Windows) 和0.0.0.0 (Linux)

3 DNS 用法



(1) 查询方式

递归查询：服务器必须回答目标 IP 与域名的映射关系。

迭代查询：服务器收到一次迭代查询回复一次结果，这个结果不一定是目标 IP 与域名的映射关系，也可以是其它 DNS 服务器的地址。

(2) 查询过程

本机查询一般先查找本机 HOST 文件，没有相关映射时，查询域名服务器；

主机向本地域名服务器的查询一般采用的都是递归查询；

如果主机所询问的本地域名服务器不知道被查询域名的 IP 地址，那么本地域名服务器就以 DNS 客户的身份，向其他根域名服务器继续发出查询请求报文；

本地域名服务器向根域名服务器的查询通常是采用迭代查询。当根域名服务器收到本地域名服务器的迭代查询请求报文时，要么给出所要查询的 IP 地址，要么告诉本地域名服务器：“你下一步应当向哪一个域名服务器进行查询”。然后让本地域名服务器进行后续的查询。

根服务器或者流量较大的域名服务器都不使用递归查询，其原因也很简单，大量的递归查询会导致服务器过载；

在 Linux 系统中，DNS 配置文件 resolv.conf 的关键字主要有四个，分别是：

nameserver：定义 DNS 服务器的 IP 地址

domain：定义本地域名

search：定义域名的搜索列表

sortlist：对返回的域名进行排序

第十章 数学与经济管理

1 运筹方法 (☆☆)

线性规划（线性规划问题的数学模型通常由线性目标函数、线性约束条件、变量非负条件组成），特点如下：

（1）线性规划的可行解域是由一组线性约束条件形成的，从几何意义来说，就是由一些线性解面围割形成的区域，不一定是封闭的多边形或多面体。

（2）如果存在两个最优解，则连接这两点的线段内所有的点都是最优解，而线段两端延长线上可能会超出可行解区。

（3）增加一个约束条件时，要么缩小可行解域（新的约束条件分割了原来的可行解域），要么可行解域不变（新的约束条件与原来的可行解域不相交）。

（4）如果最优解在可行解域边界某个非顶点处达到，则随着等值域向某个方向移动，目标函数的值会增加或减少（与最优解矛盾）或没有变化（在此段边界上都达到最优解），从而仍会在可行解域的某个顶点处达到最优解。若最优解存在且唯一，则可以从可行解区顶点处比较目标函数值来求解。

2 数学建模 (☆)

2.1 概念

数学建模是一种数学的思考方法，是运用数学的语言和方法，通过抽象和简化，建立能近似刻画并解决实际问题的模型的一种强有力的数学手段。

2.2 建模过程

（1）模型准备：了解问题的实际背景，明确其实际意义，掌握对象的各种信息。用数学语言来描述问题。

（2）模型假设：根据实际对象的特征和建模的目的，对问题进行必要的简化，并用精确的语言提出一些恰当的假设。

（3）模型建立：在假设的基础上，利用适当的数学工具来刻画各变量之间的数学关系，建立相应的数学结构。只要能够把问题描述清楚，尽量使用简单的数学工具。

（4）模型求解：利用获取的数据资料，对模型的所有参数做出计算（估计）。

（5）模型分析：对所得的结果进行数学上的分析。

（6）模型检验：将模型分析结果与实际情形进行比较，以此来验证模型的准确性、合理性和适用性。如果模型与实际较吻合，则要对计算结果给出其实际含义，并进行解释。如果模型与实际吻合较差，则应该修改假设，再次重复建模过程。

（7）模型应用：应用方式因问题的性质和建模的目的而异。

2.3 数学建模方法和思路

（1）直接分析法：认识原理，直接构造出模型。

（2）类比法：根据类似问题模型构造新模型。

(3) 数据分析法：大量数据统计分析之后建模。

(4) 构想法：对将来可能发生的情况给出设想从而建模。

2.4 模型分析

(1) 模型的合理性分析：最佳、适中、满意等。

利用实际案例数据对模型进行检验。

可以请专家来分析模型是否合理。

利用计算机来模拟实际问题，再在计算机上检验该数学模型。

(2) 模型的误差分析：模型误差、观测误差、截断误差、舍入误差、过失误差、绝对误差、相对误差等。

(3) 参数的灵敏性分析：变量数据是否敏感，在最优方案不变的条件下这些变量允许变化的范围。