系统架构设计师考前几页纸

1、决策支持系统【DSS】：

【决策支持系统（Decision Support System, DSS）】是一个由语言系统、知识系统和问题处理系统 3 个互相关联的部分组成的，基于计算机的系统。

DSS 应具有以下特征：

1.数据和模型是 DSS 的主要资源。

2.DSS 用来支援用户做决策而不是代替用户做决策。

3.DSS 主要用于解决半结构化及非结构化问题。

4.DSS 的作用在于提高决策的有效性而不是提高决策的效率。

2、专家系统【ES】：

【专家系统（Expert System，ES）】是一个智能计算机程序系统，其内部含有某个领域具有专家水平的大量知识与经验，能够利用人类专家的知识和解决问题的方法来处理该领域的问题。

知识库：存储求解实际问题的领域知识。

综合数据库：存储问题的状态描述、中间结果、求解过程的记录等信息。

推理机：实质是【规则解释器】。

知识获取：两方面功能——知识的编辑求精及知识自学习。

解释程序：面向用户服务的。

3、EAI 提供 4 个层次的服务（从高层到低层）：

EAI 提供 4 个层次的服务	功能
流程控制服务	解决人工参与的长期的工作流程控制问题
应用连接服务	应用连接适配器将应用接口连接至 EAI 平台
信息传递与转化服务	负责传递消息和转化消息
通讯服务	通过通讯中间件进行消息的路由

4、企业集成技术的架构层次（从高层到低层）：

企业集成技术的架构层次	说明
会聚集成	集成化运行
应用集成	语用互操作 模式：集成适配器、集成信使、集成面板和集成代理 4 种
数据集成	语义互通 模式：数据联邦、数据复制、基于接口的数据集成
网络集成	语法互联

5、智能制造体系架构中，系统层级是指与企业生产活动相关的组织结构的层级划分，包括设备层、单元层、车间层、企业层和协同层。

6、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。

9 个核心工作流：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。

7、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。

8、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】

9、构件的组装：

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造一个新的构件。

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。

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。

组装可能出现 3 种不兼容：参数不兼容、操作不兼容、操作不完备。

10、黑盒测试、白盒测试与灰盒测试：

白盒测试【结构测试】：关注内部结构与逻辑。

控制流分析、数据流分析、路径分析、程序变异【错误驱动测试】
【路径覆盖】 【逻辑覆盖】 修改条件/判定>条件组合>条件/判定覆盖>条件覆盖>判定覆盖>语句覆盖

黑盒测试【功能测试】：关注输入输出及功能。

等价类划分：不同等价类，揭示不同问题；有效等价类/无效等价类。
边界值分析：1≤x≤10，可取 x 的值为 0、1、10 和 11 作为测试数据。
错误推测：依靠测试人员的经验和直觉。
判定表：最适合描述在多个逻辑条件取值的组合所构成的复杂情况下，分别要执行哪些不同的动作。
因果图：根据输入条件与输出结果之间的因果关系来设计测试用例。

灰盒测试。灰盒测试介于黑盒与白盒测试之间。灰盒测试除了重视输出相对于输入的正确性，也看重其内部的程序逻辑。但是，它不可能像白盒测试那样详细和完整。它只是简单地靠一些象征性的现象或标志来判断其内部的运行情况，因此在内部结果出现错误，但输出结果正确的情况下可以采取灰盒测试方法。

11、软件测试阶段

单元测试：依据【详细设计】，模块测试，模块功能、性能、接口等。

集成测试：依据【概要设计】，模块间的接口。

系统测试：依据【需求文档】，包括功能测试、性能测试、验收测试、压力测试等。

确认测试：依据【需求文档】，验证软件与需求的一致性。内部确认测试、Alpha 测试、Beta 测试、验收测试。

12、其他测试：

其他测试	描述
AB 测试	多版本同时使用，利于收集各版本的用户反馈，评估出最好版本。故也算是一种【网页优化方法】。
Web 测试	Web 系统测试与其他系统测试测试内容基本相同，只是测试重点不同。 Web 代码测试包括：源代码规则分析、链接测试、框架测试、表格测试、图形测试等方面。
链接测试	链接测试可分为 3 个方面： 1、测试所有链接是否按指示的那样确实链接到了该链接的页面。 2、测试所链接的页面是否存在。 3、保证 Web 应用系统上没有孤立的页面。
表单测试	验证服务器是否能正确保存这些数据，后台运行的程序能否正确解释和使用这些信息。测试提交操作的完整性。
回归测试	测试软件变更之后，变更部分的正确性和对变更需求的符合性。

13、软件维护类型：

正确性维护【修 BUG】：识别和纠正软件错误/缺陷，测试不可能发现所有错误。

适应性维护【应变】：指使应用软件适应环境变化【外部环境、数据环境】而进行的修改。

完善性维护【新需求】：为扩充功能和改善性能而进行的修改。

预防性维护【针对未来】：为了适应未来的软硬件环境的变化，应主动增加预防性的新的功能，以使用系统适应各类变化而不被淘汰。经典实例：【专用】改【通用】。

14、架构的本质：

（1）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。

（2）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。

15、五大架构风格：

数据流风格：适合于分阶段做数据处理，交互性差，包括：批处理序列、管理过滤器。

调用/返回风格：一般系统都要用到，包括：主程序/子程序，面向对象，层次结构（分层越多，性能越差）。

独立构件风格：构件是独立的过程，连接件是消息传递。包括：进程通信，事件驱动系统（隐式调用）。

虚拟机风格：包括解释器与基于规则的系统，有自定义场景时使用该风格。

仓库风格（以数据为中心的风格）：以共享数据源为中心，其它构件围绕中心进行处理。包括：数据库系统、黑板系统（语言处理，信号处理），超文本系统。

16、闭环控制架构（过程控制）：定速巡航，空调温控。

17、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。

18、SOA：

粗粒度，松耦合，标准化。Webservice 与 ESB 是 SOA 的实现技术。其中：

【UDDI】是 Web 服务集成的一个体系框架，包含了服务描述与发现的标准规范。

【WSDL】服务描述语言，三个基本属性（服务做些什么/如何访问服务/服务位于何处。

【SOAP】基于 XML 的协议，在分布式环境中交换信息。4 个部分：SOAP 封装（消息内容，谁发的，谁接收）、SOAP 编码规则（数据类型实例）、SOAP RPC（远程过程调用和应答协定）、SOAP 绑定（使用底层协议交换信息）。

19、ESB：位置透明性、消息路由、服务注册命名、消息转换、多传输协议、日志与监控。

20、REST（Representational State Transfer，表述性状态转移）：

REST 是一种通常使用 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 大原则：所有事物抽象为资源、资源唯一标识、通过接口操作资源、操作不改变资源标识、操作无状态。

21、微服务：

特点：小，且专注于做一件事情；轻量级的通信机制；松耦合、独立部署。

优势：技术异构性、弹性、扩展、简化部署、与结构相匹配、可组合性、对可替代性的优化。

微服务与 SOA 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

22、MDA 的核心模型：计算无关模型（CIM），平台独立模型（PIM），平台相关模型（PSM），代码 Code。

23、ADL 的三个基本元素：

构件：计算或数据存储单元；

连接件：用于构件之间交互建模的体系结构构造块及其支配这些交互的规则；

架构配置：描述体系结构的构件与连接件的连接图。

24、DSSA：

基本活动：领域分析（建立领域模型），领域设计（获得 DSSA），领域实现（开发和组织可复用信息）。

DSSA 角色：领域专家（有经验的用户、分析、设计、实现人员，“给建议”），领域分析人员（有经验的分析师，完成领域模型），领域设计人员（有经验的设计师，完成 DSSA），领域实现人员（有经验的程序员完成代码编写）。

25、ABSD：

ABSD 方法是架构驱动，即强调由业务、质量和功能需求的组合驱动架构设计。

ABSD 方法有三个基础：功能的分解，通过选择架构风格来实现质量和业务需求，软件模板的使用。

ABSD 开发过程：

（1）架构需求（需求获取、生成类图、对类进行分组、打包成构件、需求评审）

（2）架构设计（提出架构模型、映射构件、分析构件相互作用，产生架构，设计评审）

（3）架构文档化：从使用者角度编写，分发给所有相关开发人员，保证开发者手中版本最新。

（4）架构复审：标识潜在的风险，及早发现架构设计中的缺陷和错误。

（5）架构实现（复审后的文档化架构，分析与设计，构件实现，构件组装，系统测试）

（6）架构演化（需求变化归类，架构演化计划，构件变动，更新构件相互作用，构件组装与测试，技术评审，演化后的架构）

26、基于软件系统的生命周期，可以将软件系统的质量属性分为开发期质量属性和运行期质量属性 2 个部分。

开发期质量属性主要指在软件开发阶段所关注的质量属性，主要包含 6 个方面：

易理解性、可扩展性、可重用性、可测试性、可维护性、可移植性。

运行期质量属性主要指在软件运行阶段所关注的质量属性，主要包含 7 个方面：

性能、安全性、可伸缩性、互操作性、可靠性、可用性、鲁棒性。

27、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 6 个方面进行描述：刺激源、刺激、制品、环境、响应、响应度量。

28、架构评审四大质量属性：

（1）性能：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：尽可能少的出错与尽快的恢复。代表参数（故障间隔时间，故障修复时间），设计策略（冗余、心跳线）。

（3）安全性：破坏机密性、完整性、不可否认性及可控性等特性。设计策略（追踪审计）

（4）可修改性：新增功能多少人·月能完成，设计策略（信息隐藏）

29、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：一般以某种做法，“是可以实现的”、“是可以接受的”方式进行描述。

敏感点：指为了实现某种特定的质量属性，一个或多个构件所具有的特性。

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。

30、基本场景的评估方法：ATAM，SAAM，CBAM。

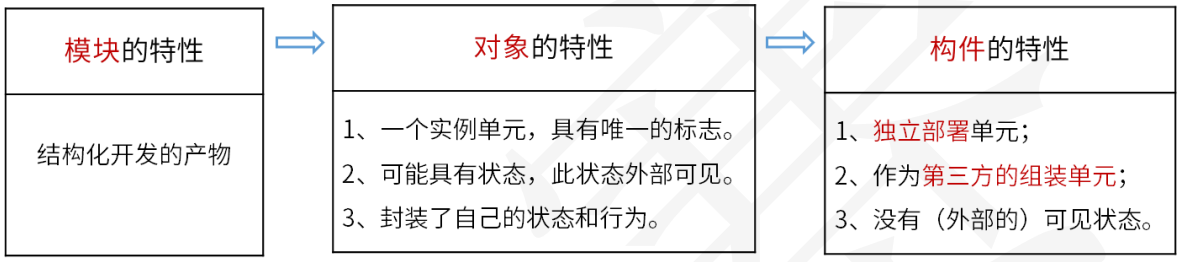
31、SAAM：最初用于分析架构可修改性，后扩展到其他质量属性。

SAAM 五个步骤：即场景开发、体系结构描述、单个场景评估、场景交互和总体评估。

32、ATAM 四大阶段：场景和需求收集、结构视图场景实现、属性模型构造和分析、折中。

ATAM：在 SAAM 的基础上发展起来的，主要针对性能、实用性、安全性和可修改性，在系统开发之前，对这些质量属性进行评价和折中。

33、构件、对象、模块的对比：



34、Bean 的分类：

（1）会话 Bean：描述了与客户端的一个短暂的会话。

（2）实体 Bean：持久化数据，O/R 映射。

（3）消息驱动 Bean：会话 Bean+JMS，客户把消息发送给 JMS 目的地，然后，JMS 提供者和 EJB 容器协作，把消息发送给消息驱动 Bean。支持异步消息。

35、负载均衡技术：

（1）应用层负载均衡：http 重定向、反向代理服务器；（2）传输层负载均衡：DNS 域名解析负载均衡、基于 NAT 的负载均衡；（3）硬件负载均衡：F5；（4）软件负载均衡：LVS、Nginx、HAproxy。

36、有状态和无状态：

（1）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。

（2）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。

37、Redis：

Redis 与 MemCache 能力比较

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	支持（Redis5.0 及以前版本不支持）
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

Redis 分布式存储方案

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。

38、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。

39、反规范化的技术手段以及优缺点

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

反规范化的优点：连接操作少，检索快、统计快；需要查的表减少，检索容易。

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 3、物化视图
更新和插入代码更难写	无解

40、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。

其中物化视图：将视图的内容物理存储起来，其数据随原始表变化，同步更新。

41、分表和分区：

	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、都降低了数据库频繁 I/O 的压力值	
差异	逻辑上还是一张表	逻辑上已是多张表。

其中分区的优点：

（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。

42、关系型数据库和 NoSQL 对比

对比维度	关系数据库	NoSQL
应用领域	面向通用领域	特定应用领域
数据容量	有限数据	海量数据
数据类型	结构化数据【二维表】	非结构化数据
并发支持	支持并发、但性能低	高并发
事务支持	高事务性	弱事务性
扩展方式	向上扩展	向外扩展

43、软件复用是一种系统化的软件开发过程，通过识别、分析、分类、获取和修改软件实体，以便在不同软件开发过程中重复使用它们。

软件开发过程中重复使用相同或相似软件元素的过程。

软件架构复用类型	机会复用	开发过程中，只要发现有可复用资产，就对其进行复用
	系统复用	开发之前，要进行规划，以决定哪些需要复用

软件架构复用的基本过程：复用的基本过程主要包括 3 个阶段：首先构造/获取可复用的软件资产，其次管理这些资产，最后针对特定的需求，从这些资产中选择可复用的部分，以开发满足需求的应用系统。

44、WPDRRC 模型：

WPDRRC 信息安全模型是我国“八六三”信息安全专家组提出的适合中国国情的信息系统安全保障体系建设模型。

WPDRRC 模型包括 6 个环节和 3 大要素。

6 个环节包括：预警、保护、检测、响应、恢复和反击。模型蕴涵的网络安全能力主要是预警能力、保护能力、检测能力、响应能力、恢复能力和反击能力。

3 大要素包括人员、策略和技术。

45、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），因此交易对手无需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。

（3）共识算法（博弈论）/全民记账

一般有：POW（工作量证明）、PoS（权益证明）、DPoS（股份授权证明机制）

比特币采用了 POW（工作量证明）：

争夺记账权=挖矿

计算出来的账单节点哈希值前 13 个字符为 0，则符合规则，得到记账权。

有一个节点计算出结果，则广播消息告知其它节点，其它节点更新数据。

46、国产密码算法：

算法名称	算法特性描述	备注
SM1	对称加密，分组长度和密钥长度都为 128 比特	广泛应用于电子政务、电子商务及国民经济的各个应用领域
SM2	非对称加密，用于公钥加密算法、密钥交换协议、数字签名算法	国家标准推荐使用素数域 256 位椭圆曲线

SM3	杂凑算法，杂凑值长度为 256 比特	适用于商用密码应用中的数字签名和验证
SM4	对称加密，分组长度和密钥长度都为 128 比特	适用于无线局域网产品
SM9	标识密码算法	不需要申请数字证书，适用于互联网应用的各种新兴应用的安全保障

47、局域网是指在有限地理范围内将若干计算机通过传输介质互联成的计算机组（即通信网络），通过网络软件实现计算机之间的文件管理、应用软件共享、打印机共享、工作组内的日程安排、电子邮件和传真通信服务等功能。局域网是封闭型的。

48、局域网专用性非常强，具有比较稳定和规范的拓扑结构。常见的局域网拓扑结构有星状结构、树状结构、总线结构和环形结构。

49、嵌入式数据库分类

按照数据库存储位置的不同而进行分类是目前广泛采用的分类方法，它可以划分为三类：

基于内存方式（Main Memory Database System，MMDB）

基于文件方式（File Database，FDB）

基于网络方式（Netware Database，NDB）

50、嵌入式网络数据库主要由三部分组成：客户端、通信协议和远程服务器。客户端主要负责提供接口给嵌入式程序，通信协议负责规范客户端与远程服务器之间的通信，还需要解决多客户端的并发问题，远程服务器负责维护服务器上的数据库数据。

51、大数据特点

5 个 V ：大规模（Volume）、高速度（Velocity）、多样化（Variety）、价值密度低（Value）、真实性（Veracity）

52、Kappa 架构与 Lambda 的对比

对比内容	Lambda 架构	Kappa 架构
复杂度与开发、维护成本	需要维护两套系统（引擎），复杂度高，开发、维护成本高	只需要维护一套系统（引擎），复杂度低，开发、维护成本低
计算开销	需要一直运行批处理和实时计算，计算开销大	必要时进行全量计算，计算开销相对较小
实时性	满足实时性	满足实时性
历史数据处理能力	批式全量处理，吞吐量大，历史数据处理能力强	流式全量处理，吞吐量相对较低，历史数据处理能力相对较弱
使用场景	直接支持批处理，更适合对历史数据分析查询的场景，期望尽快得到分析结果，批处理可以更直接高效地满足这些需求	不是 Lambda 的替代架构，而是简化，Kappa 放弃了对批处理的支持，更擅长业务本身为增量数据写入场景的分析需求

选择依据	根据两种架构对比分析，将业务需求、技术要求、系统复杂度、开发维护成本和历史数据处理能力作为选择考虑因素。 计算开销虽然存在一定差别，但是相差不是很大，所以不作为考虑因素

53、CPS 的建设路径：CPS 体系设计→单元级 CPS 建设→系统级 CPS 建设→SoS 级 CPS 建设。

54、人工智能是利用数字计算机或者数字计算机控制的机器模拟、延伸和扩展人的智能，感知环境、获取知识并使用知识获得最佳结果的理论、方法、技术及应用系统。

目标：了解智能的实质，并生产出一种新的能以人类智能相似的方式做出反应的智能机器。【图灵测试】

分类：弱人工智能和强人工智能【是否能真正实现推理、思考和解决问题】

关键技术：自然语言处理、计算机视觉、知识图谱、人机交互、虚拟现实/增强现实、机器学习。

55、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。

56、数字孪生体的三项核心技术：建模、仿真和基于数据融合的数字线程。

能够做到统领建模、仿真和数字线程的系统工程和 MBSE，则称为数字孪生体的顶层框架技术。

物联网是数字孪生体的底层伴生技术。

云计算、机器学习、大数据、区块链则成为数字孪生体的外围使能技术。

57、数字孪生生态系统

行业应用层	包括智能制造、智慧城市在内的多方面应用
共性应用层	包括描述、诊断、预测、决策四个方面
模型构建与仿真分析层	包括数据建模、数据仿真和控制
数据互动层	包括数据采集、数据传输和数据处理等内容
基础支撑层	由具体的设备组成

58、使用许可：

按照被许可使用权的排他性强弱不同，可以将使用许可分为以下三种：

①独占使用许可—仅 1 个授权对象可用，著作权人不可用。

②排他使用许可—仅 1 个授权对象和著作权人可用。

③普通使用许可—多个授权对象和著作权人可用。

59、数据安全治理的目标主要有三个：满足合规要求、管理数据安全风险、促进数据开发利用。这些目标旨在确保数据安全与业务发展的双向促进，同时保障组织在数据安全方面的合规性。

数据安全治理体系是组织达成数据安全治理目标需要具备的能力框架：