

全国计算机技术与软件专业资格（水平）考试

系统架构设计师

文老师系统架构设计师课程内部辅导资料

案例分析总复习资料

淘宝搜索：**文老师系统架构设计师**



淘宝店铺购课



微信公众号



抖音号

或在浏览器输入下面链接，登录文老师官方淘宝店（包含系分、架构、系规、高项、中项、软设、网工、网规、数据库、评测、信安等全类目软考辅导）：

wenlaoshi.taobao.com

一	案例分析概述	6
1	案例分析答题卡	6
2	历年真题考点分析	6
3	解题技巧	8
二	历年案例精华考点汇总	9
1	软件架构设计	9
2	系统开发基础	15
3	数据库系统	20
4	嵌入式系统	25
5	Web 应用开发	29
三	第二版下篇架构设计考点	37
1	信息系统架构设计	37
1.1	信息系统架构基本概念	37
1.2	信息系统架构	37
1.3	信息系统架构设计方法	39
1.4	信息系统架构案例分析	41
2	层次式架构设计	45
2.1	层次式架构概述	45
2.2	表现层框架设计	45
2.3	中间层架构设计	46
2.4	数据访问层设计	48
2.5	数据架构规划与设计	49
2.6	物联网层次架构设计	50
2.7	层次式架构案例分析	51
3	云原生架构设计	53
3.1	云原生架构内涵	53
3.2	云原生架构相关技术	54
3.3	云原生架构案例分析	56
4	面向服务架构设计	59
4.1	SOA 概述和发展	59
4.2	SOA 的参考架构	60
4.3	SOA 主要协议和规范	62
4.4	SOA 设计标准和原则	63
4.5	SOA 的设计模式	64

4.6	SOA 构建和实施.....	65
5	嵌入式系统架构设计.....	66
5.1	嵌入式系统软件架构原理与特征.....	66
5.2	嵌入式系统软件架构设计方法.....	71
5.3	嵌入式系统软件架构案例分析.....	73
6	通信系统架构设计.....	77
6.1	通信系统网络架构.....	77
6.2	网络构建关键技术.....	83
6.3	网络构建和设计方法.....	84
6.4	通信网络构建案例分析.....	85
7	安全架构设计.....	87
7.1	安全架构概述.....	87
7.2	安全模型.....	88
7.3	系统安全体系架构规划框架.....	91
7.4	信息安全整体架构设计.....	92
7.5	网络安全体系架构设计.....	95
7.6	数据库系统的安全设计.....	97
7.7	系统架构的脆弱性分析.....	98
7.8	安全架构设计案例分析.....	101
8	大数据架构设计.....	102
8.1	大数据处理系统架构分析.....	102
8.2	Lambda 架构.....	103
8.3	Kappa 架构.....	105
8.4	Lambda 架构与 Kappa 架构的对比和设计选择.....	106
8.5	大数据架构设计案例分析.....	107
四	历年案例真题.....	111
	2022 年下半年试题 1.....	111
	2022 年下半年试题 2.....	113
	2022 年下半年试题 3.....	115
	2022 年下半年试题 4.....	117
	2022 年下半年试题 5.....	117
	2021 年下半年试题 1.....	119
	2021 年下半年试题 2.....	120
	2021 年下半年试题 3.....	122

2021 年下半年试题 4	123
2021 年下半年试题 5	124
2020 年下半年试题 1	125
2020 年下半年试题 2	127
2020 年下半年试题 3	128
2020 年下半年试题 4	129
2020 年下半年试题 5	130
2019 年下半年试题 1	132
2019 年下半年试题 2	134
2019 年下半年试题 3	135
2019 年下半年试题 4	136
2019 年下半年试题 5	138
2018 年下半年试题 1	139
2018 年下半年试题 2	140
2018 年下半年试题 3	142
2018 年下半年试题 5	143
2017 年下半年试题 1	145
2017 年下半年试题 2	146
2017 年下半年试题 3	147
2017 年下半年试题 4	150
2017 年下半年试题 5	151
2016 年下半年试题 1	153
2016 年下半年试题 2	154
2016 年下半年试题 3	155
2016 年下半年试题 4	156
2016 年下半年试题 5	157
2015 年下半年试题 1	159
2015 年下半年试题 2	160
2015 年下半年试题 3	163
2015 年下半年试题 4	165
2015 年下半年试题 5	166
2014 年下半年试题 1	167
2014 年下半年试题 2	168
2014 年下半年试题 3	169
2014 年下半年试题 4	171
2014 年下半年试题 5	172
2013 年下半年试题 1	174
2013 年下半年试题 2	175
2013 年下半年试题 3	176
2013 年下半年试题 4	179
2013 年下半年试题 5	181
五 案例真题答案解析	182
2022 年下半年试题 1	182
2022 年下半年试题 2	182
2022 年下半年试题 3	183

2022 年下半年试题 4	184
2022 年下半年试题 5	185
2021 年下半年试题 1	186
2021 年下半年试题 2	186
2021 年下半年试题 3	187
2021 年下半年试题 4	187
2021 年下半年试题 5	188
2020 年下半年试题 1	189
2020 年下半年试题 2	190
2020 年下半年试题 3	191
2020 年下半年试题 4	194
2020 年下半年试题 5	196
2019 年下半年试题 1	197
2019 年下半年试题 2	198
2019 年下半年试题 3	198
2019 年下半年试题 4	199
2019 年下半年试题 5	199
2018 年下半年试题 1	200
2018 年下半年试题 2	201
2018 年下半年试题 3	201
2018 年下半年试题 4	202
2018 年下半年试题 5	203
2017 年下半年试题 1	203
2017 年下半年试题 2	204
2017 年下半年试题 3	205
2017 年下半年试题 4	206
2017 年下半年试题 5	207
2016 年下半年试题 1	207
2016 年下半年试题 2	209
2016 年下半年试题 3	209
2016 年下半年试题 4	211
2016 年下半年试题 5	213
2015 年下半年试题 1	213
2015 年下半年试题 2	213
2015 年下半年试题 3	216
2015 年下半年试题 4	217
2015 年下半年试题 5	219
2014 年下半年试题 1	220
2014 年下半年试题 2	221
2014 年下半年试题 3	222
2014 年下半年试题 4	223
2014 年下半年试题 5	224
2013 年下半年试题 1	226
2013 年下半年试题 2	227
2013 年下半年试题 3	229

2013 年下半年试题 4	230
2013 年下半年试题 5	232

一 案例分析概述

1 案例分析答题卡

案例分析第一题必做，后面四个四选二，答题卡上选项如下，将选做的题目涂黑。

选答题说明				
试题二至试题五为选答题，请从中任选两题解答，并将标识框涂黑。若试题多选、少选或未选，则对题号最小的两道试题进行评分。				
例如：如果选答题三和试题五，应先将选答题一中的试题三标识框涂黑，并在选答题一解答栏中作答试题三；然后将选答题二中的试题五标识框涂黑，并在选答题二解答栏中作答试题五。				

选答题一： 试题二 试题三 试题四 试题五				
	☐	☒	☐	☐
选答题一	解 答 栏			
问题 1				

选答题二： 试题二 试题三 试题四 试题五				
	☐	☐	☐	☒
选答题二	解 答 栏			
问题 1				

2 历年真题考点分析

根据历年真题考点分析，我们将架构案例分析真题分为如下几个大类：

1、软件架构设计：每年会必考 1-2 题，并且是第 1 题必选题，必须掌握，主要涉及到质量属性、软件架构风格、软件架构评估、MVC 架构、面向服务的架构 SOA、ESB、J2EE 架构等。对于其他未考察的架构领域重点知识如 DSSA、ABSD 等，也必须重点掌握

2、系统开发基础：几乎每年必考 1 题，主要涉及到 UML 的图、关系的识别，尤其是类图、用例图、活动图、状态图；设计模式识别；数据流图、E-R 图等简单识别；信息安全相关技术；项目管理-进度管理-关键路径。

3、数据库系统：偶尔会考察一题，主要考察的是数据库的一些新技术的比较，如关系型数据库、内存数据库及 NoSQL 等，还会包括反规范化技术、主从复制、负载均衡等。

4、嵌入式系统：几乎每年必考一题，选做题，考察比较多的是嵌入式系统的实时性和可靠性以及容错等概念。大概率会考到一些嵌入式领域陌生技术，如果是完全没见过的技术，不选即可。

5、Web 应用开发：主要考察 Web 相关技术，一般结合架构进行考察。偶尔会考到新技术，遇到完全没听说过的技术，就不选。

此外，早年的考试中，偶尔考到一些完全陌生的架构和技术，没有看的必要，可忽略，因为陌生技术不会再考第二次，无法去归纳总结，也没有了解的必要。

具体历年真题考点归纳如下（红色加粗字体是试题一必做题，带删除线部分是完全陌生技术，没有关注的必要）：

时间	所属范围	考察知识点
2022 年 11 月 试题 1	软件架构	架构风格，质量属性
2022 年 11 月 试题 2	系统开发	结构化分析：数据流图、ER 图、数据字典
2022 年 11 月 试题 3	嵌入式	宇航装备架构、看图填空、故障分析
2022 年 11 月 试题 4	数据库	同步和异步、缓存分片、布隆过滤器
2022 年 11 月 试题 5	Web 应用	MQTT 协议、看图填空、云计算、边缘计算
2021 年 11 月 试题 1	软件架构	架构风格，质量属性
2021 年 11 月 试题 2	系统开发	用例图、顺序图填空、模型对比
2021 年 11 月 试题 3	软件架构	数据定义分布管理涵义、基于 FACE 的架构（题目不全）
2021 年 11 月 试题 4	数据库	反规范化设计方法、数据不一致、redis 同步
2021 年 11 月 试题 5	Web 应用	云平台智能家居，看图填空，TCP/UDP 区别
2020 年 11 月 试题 1	软件架构	架构风格，质量属性
2020 年 11 月 试题 2	数据库	逻辑设计、关系模式、主键、超类实体、派生属性
2020 年 11 月 试题 3	嵌入式	需求到架构映射、FACE 架构
2020 年 11 月 试题 4	数据库	内存数据库 redis，内存淘汰机制
2020 年 11 月 试题 5	Web 应用	非功能性需求，SSM 框架，数据访问机制
2019 年 11 月 试题 1	软件架构	架构风格，质量属性
2019 年 11 月 试题 2	系统开发	数据流图求实体、加工、补充数据流；系统流程图区别
2019 年 11 月 试题 3	嵌入式	信息物理系统三层结构概念、填空；三类安全威胁
2019 年 11 月 试题 4	数据库	数据库读写并发操作、key/value 方案探讨
2019 年 11 月 试题 5	Web 应用	非功能性需求、分布式架构图、SQL 注入攻击
2018 年 11 月 试题 1	软件架构	非功能性需求、C/S 架构
2018 年 11 月 试题 2	系统开发	数据流图、ER 图、实体和类、用例
2018 年 11 月 试题 3	嵌入式	简单任务和复杂任务、基本消息通信 BMTS
2018 年 11 月 试题 4	数据库	MemCache 和 Redis、数据可靠性和一致性
2018 年 11 月 试题 5	web 应用	SOA、ESB、信息安全、根据描述填图
2017 年 11 月 试题 1	软件架构	质量属性效用树、架构风险、敏感点、权衡点
2017 年 11 月 试题 2	软件架构	MVC、EJB、J2EE
2017 年 11 月 试题 3	嵌入式	机器人操作系统 ROS 和 RTOS、根据描述填流程图
2017 年 11 月 试题 4	数据库	ORM 和数据库程序在线访问、数据访问层、工厂设计模式

2017 年 11 月试题 5	web 应用	响应式 web 设计、高并发 web 架构、主从复制机制
2016 年 11 月试题 1	软件架构	质量属性、架构风格对比、根据描述填空
2016 年 11 月试题 2	系统开发	用例图参与者、用例关系、类图关系
2016 年 11 月试题 3	嵌入式	RTOS 特点、实时性分类、缺陷故障失效关系图
2016 年 11 月试题 4	web 应用	应用服务器、PHP 和 Java、J2EE 架构
2016 年 11 月试题 5	系统开发	Scrum 敏捷开发状态图、MVC 架构应用
2015 年 11 月试题 1	软件架构	质量属性效用树、架构风险、敏感点、权衡点
2015 年 11 月试题 2	系统开发	状态图、活动图特点，及根据描述填图
2015 年 11 月试题 3	嵌入式	可靠性子特性、硬件软件可靠性、恢复快和 N 版本程序设计
2015 年 11 月试题 4	数据库	关系数据库、文件系统、内存数据库、SQL 查询性能
2015 年 11 月试题 5	web 应用	数据持久层特点和其实现技术
2014 年 11 月试题 1	软件架构	MVC 架构、扩展接口模式
2014 年 11 月试题 2	系统开发	数据流图元素、找 DFD 错误、CRUD 矩阵
2014 年 11 月试题 3	嵌入式	构件获取和开发、构件标准、特点判断、构件接口
2014 年 11 月试题 4	软件架构	质量属性效用树、架构风险、敏感点、权衡点
2014 年 11 月试题 5	web 应用	负载均衡(DNS、反向代理)、主从复制、数据库分区、memcached
2013 年 11 月试题 1	软件架构	根据描述填表、ESB 定义和功能
2013 年 11 月试题 2	项目管理	项目管理计划、进度和成本计算
2013 年 11 月试题 3	嵌入式	故障模型影响分析 FMEA 内容、步骤等
2013 年 11 月试题 4	web 应用	MVC 架构、XML 界面设计
2013 年 11 月试题 5	信息安全	公钥体系、数字信封、数据库加密（解密 API 和透明加密）

3 解题技巧

（十分重要）案例分析学习建议：

1、先完整的听一遍文老师视频课程案例分析专题课程，里面详细讲解了大量典型案例真题以及解题技巧，该如何去分析题目，是非常重要的。

2、然后看本复习资料第二章案例精华考点内容，将相关考点都理解记忆，不理解的再去看视频对应部分知识点。

3、最后开始做本资料第四章历年案例真题，请一定按年份做题，并掐表练习，即每次抽 90 分钟出来，模拟真实考试，浏览五个题目，从中选出三个来做，90 分钟后停笔对答案，看看能得多少分。

这种做题方式非常有必要，切忌用零碎时间一题一题去做，没有意义，架构案例考试更多的是考察学员临场应变能力，如果选题，选自己能拿分的题才是最重要的。

关于第二版教材改版之后，下篇的内容，我们全部在案例专题里做了更新，更新在本资料第三章，属于第二版教材纯新增内容，有大量关于不同架构设计实战内容，涉及到不少新技术，是比较难理解

的，学员一定要先听这一块的文老师视频讲解，再来看，否则很难把握重点，同时，因为这一块实战性强，新技术多，文老师也难免会有遗漏错误之处，敬请理解，也欢迎大家一起讨论反馈。

其他解题技巧如下：

先看问题，再看题目描述，第一题必做，不用犹豫，先做完第一题。

后面四选二，快速浏览所有题目的问题，看看问题描述是否能看懂，再选择。

问题与题目描述相结合，尤其是遇到系统分析和设计问题，以及新技术问题，答案一般都在题目描述里，从题目描述抽象总结出问题答案。

以最简练的语言写出答案，一般题目会有最大字数要求，不能超过。

遇到新的知识点，不要慌，稳住心态。

列条目回答问题，把自己认为对的，都写上，多写不会扣分，少写一定会扣分。

分析题目问题的倾向性，顺势答题。

二 历年案例精华考点汇总

注意：这里只是结合历年真题考察情况对上午基础只是的一个精简汇总，真正还是要以上午综合精华知识点为准，将案例分析涉及到的知识点章节内容全部理解记忆，必要时可以将官方教材通读一遍。

1 软件架构设计

系统架构设计师案例分析必考题，每年会必考 1-2 题，并且是第 1 题必选题，必须掌握，主要涉及到质量属性、软件架构风格、软件架构评估、MVC 架构、面向服务的架构 SOA 等。

对于其他未考察的架构领域重点知识如 DSSA、ABSD 等，也必须重点掌握，最好将软件架构设计一章节通读一遍。

本题考察比较简单，知识点固定，一般可以拿到 20 分。

质量属性效用树、质量属性判断

1、性能：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。如响应时间、吞吐量。设计策略：优先级队列、增加计算资源、减少计算开销、引入并发机制、采用资源调度等。

2、可靠性：是软件系统在应用或系统错误面前，在意外或错误使用的情况下维持软件系统的功能特性的基本能力。如 MTTF、MTBF。设计策略：心跳、Ping/Echo、冗余、选举。

3、可用性：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。如故障间隔时间。设计策略：心跳、Ping/Echo、冗余、选举。

4、安全性：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。如保密性、完整性、不可抵赖性、可控性。设计策略：入侵检测、用户认证、用户授权、追踪审计。

5、可修改性：指能够快速的以较高的性能价格比对系统进行变更的能力。通常以某些具体的变更更为基准，通过考察这些变更的代价衡量。设计策略：接口-实现分类、抽象、信息隐藏。

6、功能性：是系统所能完成所期望的工作的能力。一项任务的完成需要系统中许多或大多数构件的相互协作。

7、可变性：指体系结构经扩充或变更而成为新体系结构的能力。这种新体系结构应该符合预先定义的规则，在某些具体方面不同于原有的体系结构。当要将某个体系结构作为一系列相关产品的基

础时，可变性是很重要的。

8、互操作性：作为系统组成部分的软件不是独立存在的，经常与其他系统或自身环境相互作用。为了支持互操作性，软件体系结构必须为外部可视的功能特性和数据结构提供精心设计的软件入口。程序和用其他编程语言编写的软件系统的交互作用就是互操作性的问题，也影响应用的软件体系结构。

必背概念

软件架构风格是指描述特定软件系统组织方式的惯用模式。组织方式描述了系统的组成构件和这些构件的组织方式，惯用模式则反映众多系统共有的结构和语义。

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。

风险点与非风险点不是以标准专业术语形式出现的，只是一个常规概念，即可能引起风险的因素，可称为风险点。某个做法如果有隐患，有可能导致一些问题，则为风险点；而如果某件事是可行的 可接受的，则为非风险点。

敏感点：是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：是影响多个质量属性的特性，是多个质量属性的敏感点。

架构风格对比

所有架构风格汇总如下，要能够看表理清每个架构风格的含义，有生疏的再去看上午精华知识点部分详细介绍。

架构风格名	常考关键字及实例	简介
数据流-批处理	传统编译器，每个阶段产生的结果	一个接一个，以整体为单位。
数据流-管道-过滤器	作为下一个阶段的输入，区别在于整体。	一个接一个，前一个输出是后一个输入。
调用/返回-主程序/子程序		显示调用，主程序直接调用子程序。
调用/返回-面向对象		对象是构件，通过对象调用封装的方法和属性。
调用/返回-层次结构		分层，每层最多影响其上下两层，有调用关系。
独立构件-进程通信		进程间独立的消息传递，同步异步。
独立构件-事件驱动（隐式调用）	事件触发推动动作，如程序语言的语法高亮、语法错误提示	不直接调用，通过事件驱动。
虚拟机-解释器	自定义流程，按流程执行，规则随时改变，灵活定义，业务灵活组合	解释自定义的规则，解释引擎、存储区、数据结构。
虚拟机-规则系统	机器人。	规则集、规则解释器、选择器和工作内存，用于 DSS 和人工智能、专家系统。
仓库-数据库	现代编译器的集成开发环境 IDE，	中央共享数据源，独立处理单元。
仓库-超文本	以数据为中心。	网状链接，多用于互联网。

仓库-黑板	又称为数据共享风格	语音识别、知识推理等问题复杂、解空间很大、求解过程不确定的这一类软件系统，黑板、知识源、控制。
闭环-过程控制	汽车巡航定速，空调温度调节，设定参数，并不断调整。	发出控制命令并接受反馈，循环往复达到平衡。
C2 风格	构件和连接件、顶部和底部	通过连接件绑定在一起按照一组规则运作的并行构件网络

下午案例分析常考比较的架构风格对比：

架构风格	主要特点	主要优点	主要缺点	适合领域
管道-过滤器	过滤器相对独立	功能模块复用；可维护性和可扩展性较强；具有并发性；模块独立性高	不适于交互性强的应用，对于存在关系的数据流必须进行协调	系统可划分清晰的模块；模块相对独立；有清晰的模块接口。
面向对象	力争实现问题空间和软件系统空间结构的一致性。	高度模块性；实现封装；代码共享灵活；易维护；可扩充性好	增加了对象之间的依赖关系	多种领域
事件驱动	系统由若干子系统构成且称为一个整体；系统有统一的目标；子系统有主从之分；每一个子系统有自己的事件收集和处理机制	适合描写系统组；容易实现并发处理和多个任务；可扩展性好；具有类层次结构；简化代码；	因为树型结构所以削弱了对系统计算的控制能力；各个对象的逻辑关系复杂	一个系统对外部的表现可以从它对事件的处理表征出来。
分层风格	各个层次的组件形成不同功能级别的虚拟机；多层相互协同工作，而且实现透明	支持系统设计过程中的逐级抽象；可扩展性好；支持软件复用	不同层次之间耦合度高的系统很难实现	适合功能层次的抽象和相互之间低耦合的系统
数据共享风格	采用两个常用构件中央数据单元和一些相对独立的组件集合	中央数据单元实现了数据的集中，以数据为中心	适合于特定领域	适合于专家系统等人工智能领域问题的求解
解释器风格	系统核心是虚拟机	可以用多种操作来解释一个句子，灵活应对自定义场景	适合于特定领域	适合于模式匹配系统与语言编译器
闭环控制风格	通过不断的测量被控对象，认识和掌握被控对象；将控制理论引入体系结构构建	将控制理论引入到计算机软件体系结构中	适合于特定领域	该系统中一定存在有目标的作用、信息处理闭环控制过程。

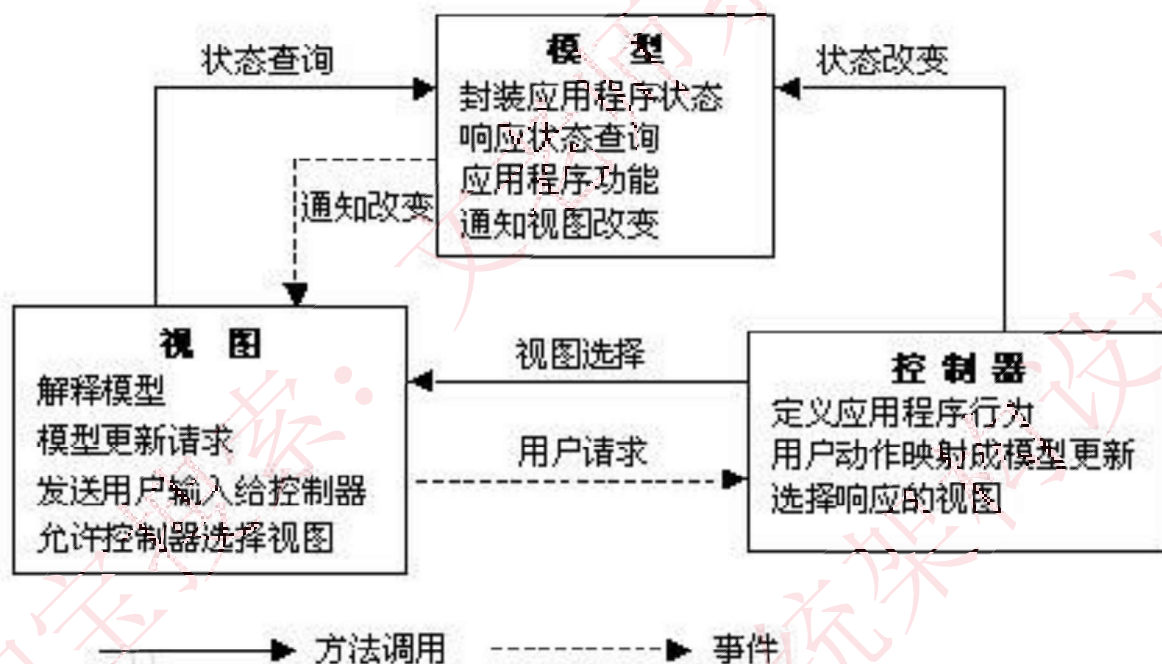
MVC 架构

MVC 强制性地把一个应用的输入、处理、输出流程按照视图、控制、模型的方式进行分离，形成了三个核心模块：控制器、模型、视图。

(1) 控制器 (Controller)：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。

(2) 模型 (Model)：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。模型表示业务数据和业务逻辑。

(3) 视图 (View)：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。是用户看到并与之交互的界面。视图向用户显示相关的数据，并能接收用户的输入数据，但是它并不进行任何实际的业务处理。

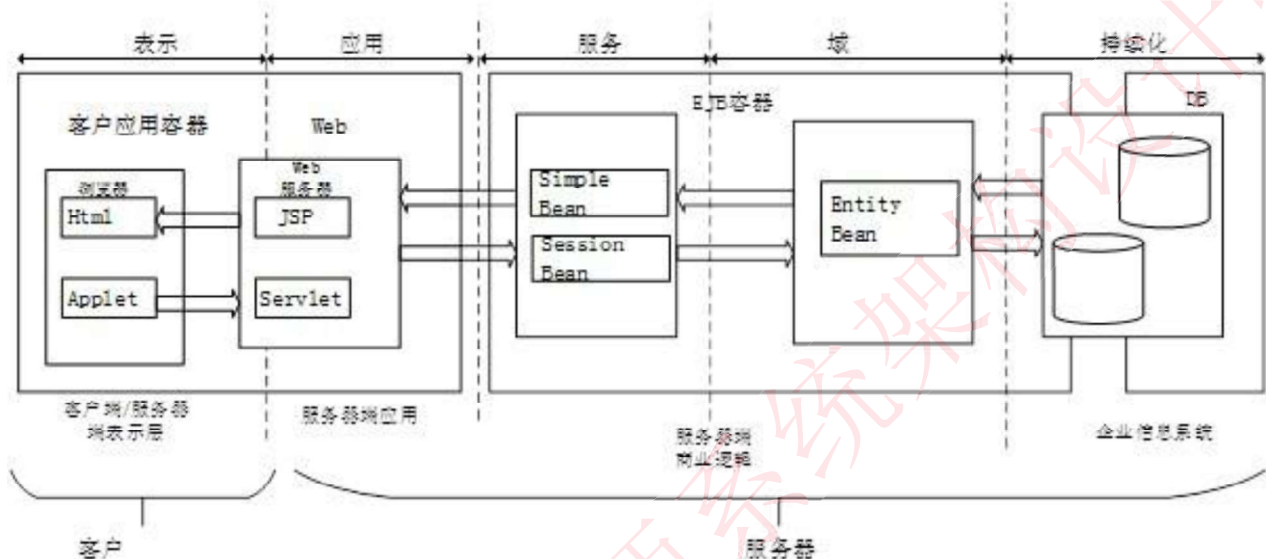


MVC 分层有助于管理复杂的应用程序，因为您可以在一个时间内专门关注一个方面。例如，您可以在不依赖业务逻辑的情况下专注于视图设计。同时也让应用程序的测试更加容易。

MVC 分层同时也简化了分组开发。不同的开发人员可同时开发视图、控制器逻辑和业务逻辑。

J2EE 架构

四层结构：



客户层组件: J2EE 应用程序可以是基于 web 方式的, 也可以是基于传统方式的. 静态的 HTML (标准通用标记语言下的一个应用) 页面和 Applets 是客户层组件

web 层组件: J2EE web 层组件可以是 JSP 页面或 Servlet。

业务层组件: 业务层代码的逻辑用来满足特定领域的业务逻辑处理。

EJB 是 JavaEE 服务器端组件模型, 设计目标与核心应用是部署分布式应用程序。简单来说就是把已经编写好的程序 (即: 类) 打包放在服务器上执行。凭借 java 跨平台的优势, 用 EJB 技术部署的分布式系统可以不限于特定的平台。

EJB 中有三种企业级的 bean: 会话 (session) beans, 实体 (entity) beans, 和消息驱动 (message-driven) beans. 会话 bean 表示与客户端程序的临时交互. 当客户端程序执行完后, 会话 bean 和相关数据就会消失. 相反, 实体 bean 表示数据库的表中一行永久的记录. 当客户端程序中止或服务器关闭时, 就会有潜在的服务保证实体 bean 的数据得以保存. 消息驱动 bean 结合了会话 bean 和 JMS 的消息监听器的特性, 允许一个业务层组件异步接收 JMS 消息。

信息系统层: 企业信息系统层处理企业信息系统软件包括企业基础建设系统例如企业资源计划 (ERP), 大型机事务处理, 数据库系统, 和其它的遗留信息系统. 例如, J2EE 应用组件可能为了数据库连接需要访问企业信息系统。

JSP+Servlet+JavaBean+DAO

JSP: 用于显示、收集数据的部分。作为 MVC 中的视图 V。

Servlet: 作为业务逻辑层, 用于处理复杂的业务逻辑, 如验证数据、实例化 JavaBean、调用 DAO 连接数据库等。作为 MVC 中的控制器 C。在其中会调用 Service 方法处理服务。

JavaBean: 用于数据的封装, 方便将查询结果在 servlet 与 jsp 页面之间进行传递等。

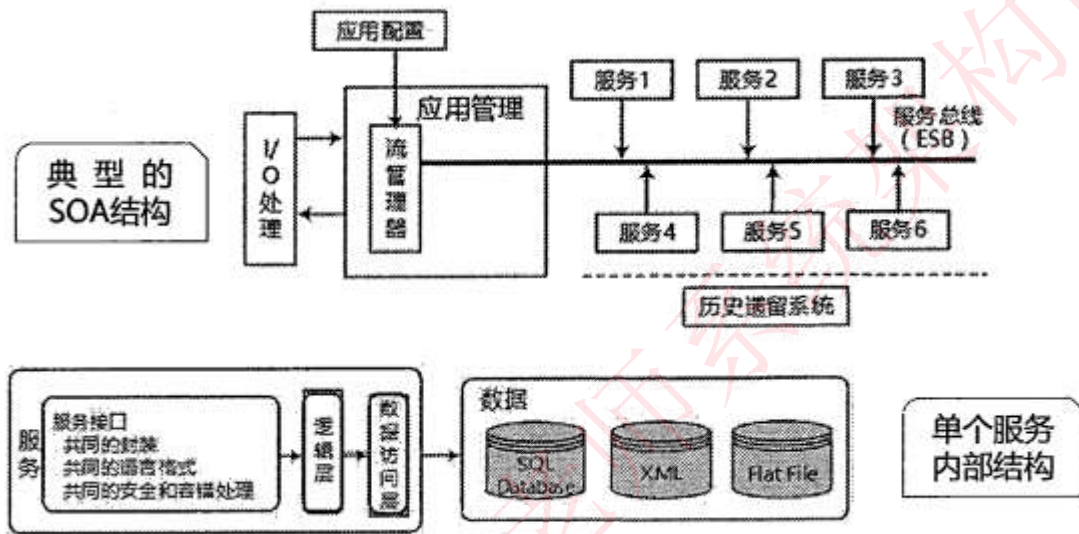
DAO: 用于连接数据库及进行数据库的操作如: 查询、删除、更改等。

DAO 与 JavaBean 合在一起为 MVC 中的模型 M。

基本流程: JSP 发一个数据到 servlet, servlet 收到后做下解析再根据数据调用相应的 service 去服务, service 如果有要调用数据库就通过 DAO 跟数据库交互, 使用 javaBean 完成封装, 返回结果给 servlet, servlet 再返回给 JSP。

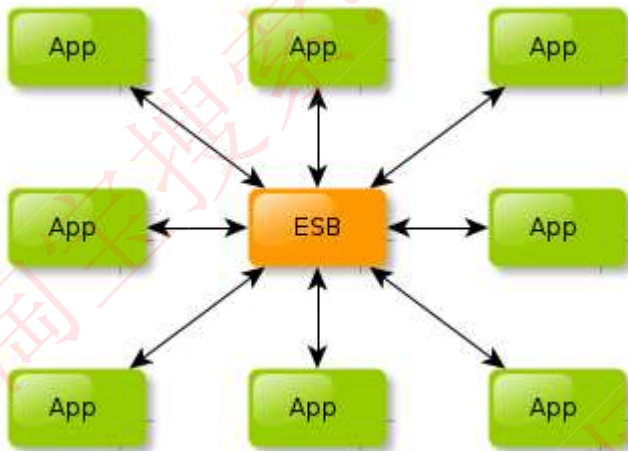
面向服务的架构 SOA

SOA 是一种设计理念，其中包含多个服务，服务之间通过相互依赖最终提供一系列完整的功能。各个服务通常以独立的形式部署运行，服务之间通过网络进行调用。



企业服务总线 ESB

简单来说是一根管道，用来连接各个服务节点。ESB 的存在是为了集成基于不同协议的不同服务，ESB 做了消息的转化、解释以及路由的工作，以此来让不同的服务互联互通。



ESB 特点：

- 1、SOA 的一种实现方式，ESB 在面向服务的架构中起到的是总线作用，将各种服务进行连接与整合；
- 2、描述服务的元数据和服务注册管理；
- 3、在服务请求者和提供者之间传递数据，以及对这些数据进行转换的能力，并支持由实践中总结出来的一些模式如同步模式、异步模式等；
- 4、发现、路由、匹配和选择的能力，以支持服务之间的动态交互，解耦服务请求者和提供者。高级一些的能力，包括对安全的支持、服务质量保证、可管理性和负载平衡等。

ESB 的主要功能：

- (1) 服务位置透明性；
- (2) 传输协议转换；
- (3) 消息格式转换；
- (4) 消息路由；
- (5) 消息增强；
- (6) 安全性；
- (7) 监控与管理。

2 系统开发基础

选做题，几乎每年必考 1 题，但是不会涉及大范围的系统分析与设计原理，而是偏向于软件设计的范围，考察 UML 的图、关系的识别；设计模式识别；数据流图、E-R 图等简单识别；信息安全相关技术；项目管理-进度管理-关键路径。

本题考察比较简单，一般可以拿到 20 分。

结构化

结构化特点：自顶向下，逐步分解，面向数据。

三大模型：功能模型（数据流图）、行为模型（状态转换图）、数据模型（E-R 图）以及数据字典（数据元素、数据结构、数据流、数据存储、加工逻辑、外部实体）。

数据字典：数据字典是在 DFD 的基础上，对 DFD 中出现的所有命名元素都加以定义，使得每个图形元素的名字都有一个确切的解释。DFD 和数据字典等工具相配合，就可以从图形和文字两个方面对系统的逻辑模型进行完整的描述。数据字典中一般有 6 类条目，分别是数据元素、数据结构、数据流、数据存储、加工逻辑和外部实体。不同类型的条目有不同的属性需要描述。

用例建模



用例图：静态图，展现了一组用例、参与者以及它们之间的关系。用例图中的参与者是人、硬件或其他系统可以扮演的角色；用例是参与者完成的一系列操作。

主要考察参与者和用例的识别、用例之间的关系（包含 include、扩展 extend、泛化），例如：

登记外借信息用例包含用户登录用例，因为每次如果要登记外借信息，必然要先进行用户登录。而查询书籍信息的扩展是修改书籍信息，是因为每次查询书籍信息后，发现有错误才会修改，否则不修改，不是必要的操作。因此，区分用例间的关系是包含还是扩展，关键在于是不是必须操作。

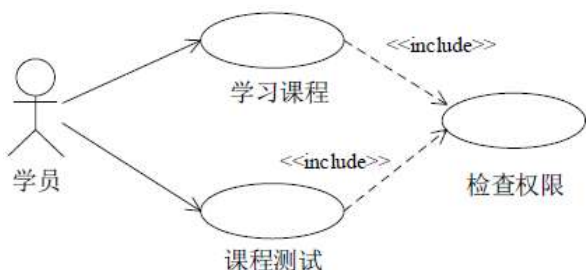


图 11-11 包含关系的例子

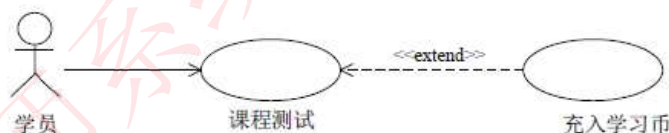
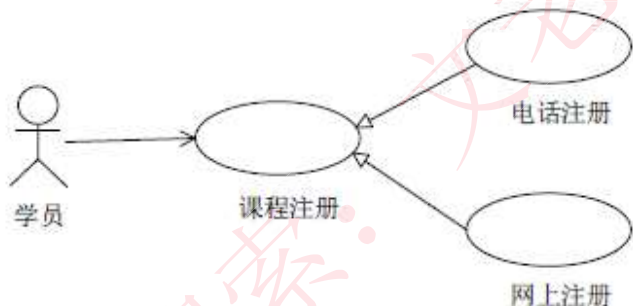


图 11-12 扩展关系的例子



类图：静态图，为系统的静态设计视图，展现一组对象、接口、协作和它们之间的关系。

UML 关系：依赖、关联、泛化、实现、组合、聚合。

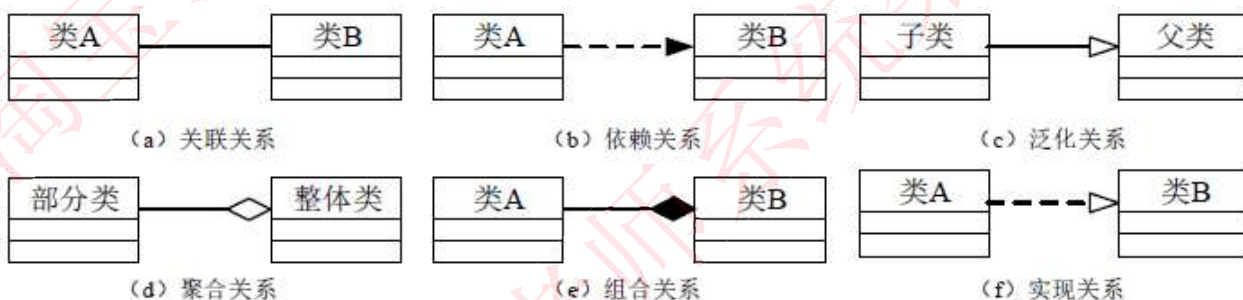


图 11-14 类之间的关系表示

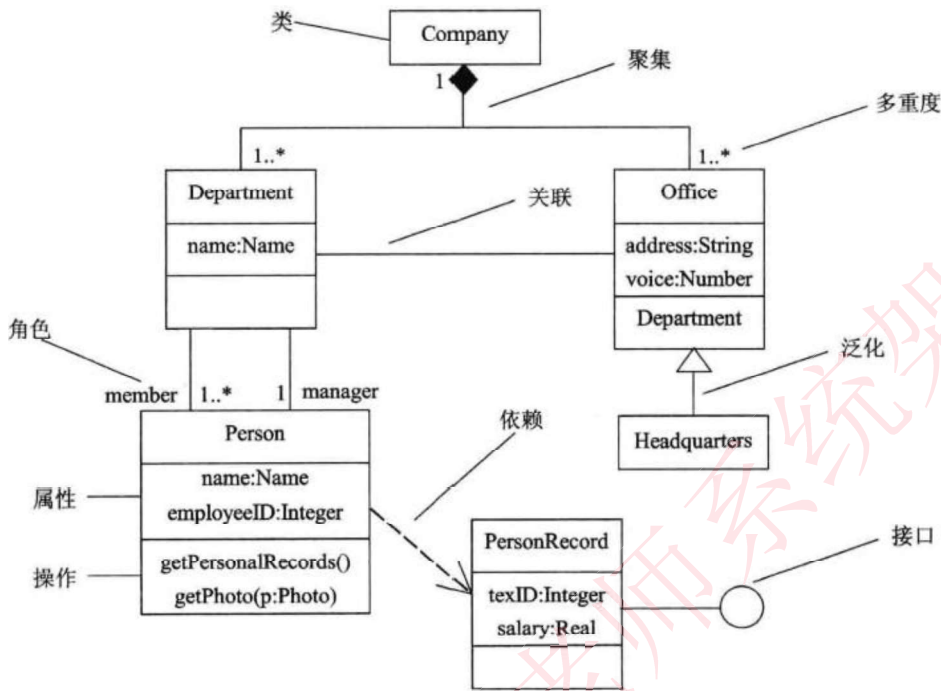
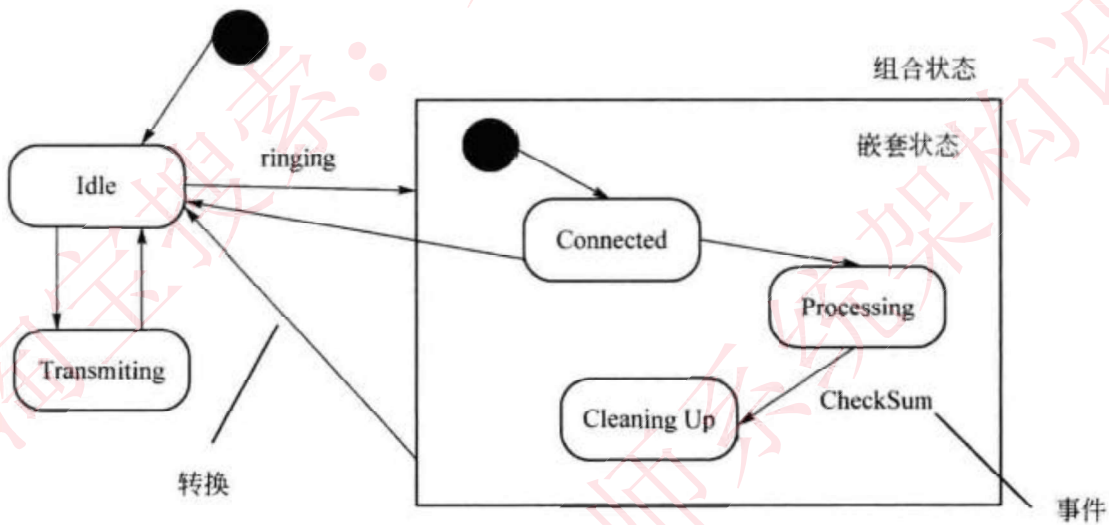
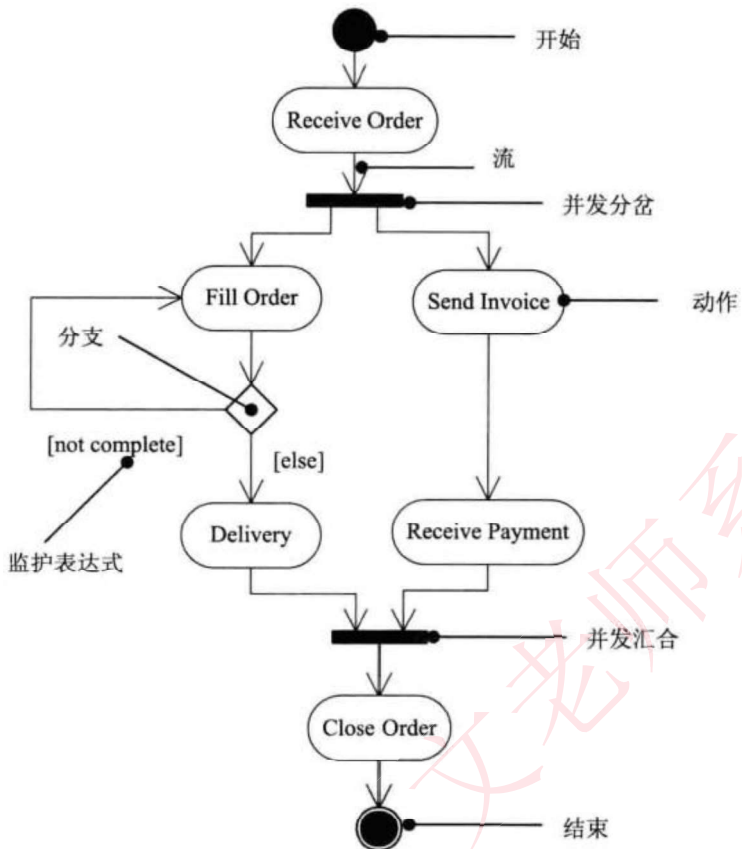


图 10-11 UML 类图

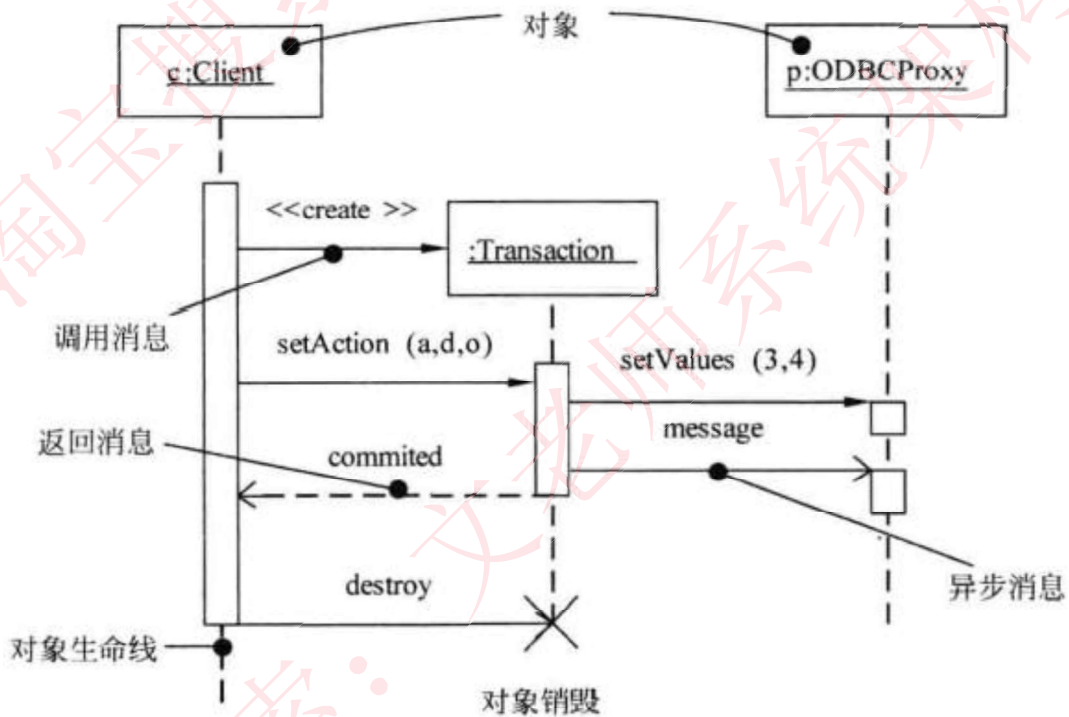
状态图：动态图，展现了一个状态机，描述单个对象在多个用例中的行为，包括简单状态和组合状态。转换可以通过事件触发器触发，事件触发后相应的监护条件会进行检查。



活动图：动态图，是一种特殊的状态图，展现了在系统内从一个活动到另一个活动的流程。用来描述操作的行为，也用于描述用例和对象内部的工作过程。



序列图：即顺序图，动态图，是场景的图形化表示，描述了以时间顺序组织的对象之间的交互活动。

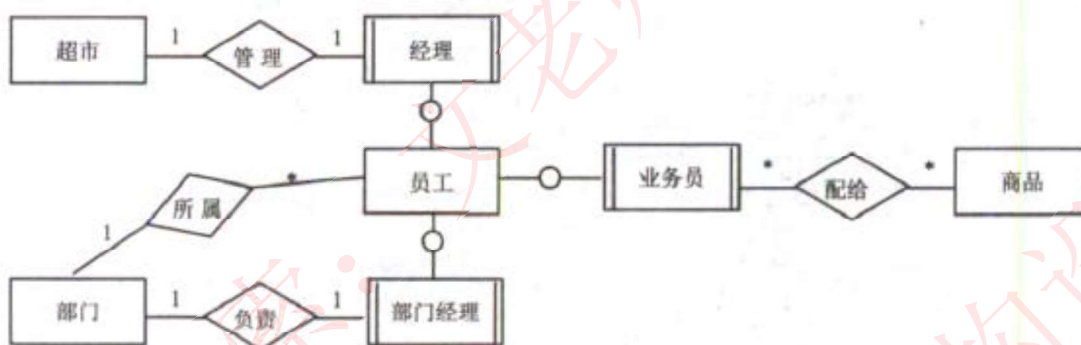


数据流图

组成元素：

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	□

E-R 图



进度管理

PERT（项目评估与评审技术）图是一种图形化的网络模型，描述一个项目中任务和任务之间的关系，每个节点表示一个任务，通常包括任务编号、名称、开始和结束时间、持续时间和松弛时间。

Gantt 图是一种简单的水平条形图，它以一个日历为基准描述项目任务，横坐标表示时间，纵坐标表示任务，图中的水平线段表示对一个任务的进度安排，线段的起点和终点对应在横坐标上的时间分别表示该任务的开始时间和结束时间，线段的长度表示完成该任务所需的时间。

PERT 图主要描述不同任务之间的依赖关系；Gantt 图主要描述不同任务之间的重叠关系。

关键路径：是项目的最短工期，但却是从开始到结束时间最长的路径。进度网络图中可能有多条关键路径，因为活动会变化，因此关键路径也在不断变化中。

关键活动：关键路径上的活动，最早开始时间=最晚开始时间。

通常，每个节点的活动会有如下几个时间：

- （1）最早开始时间（ES），某项活动能够开始的最早时间。
 - （2）最早结束时间（EF），某项活动能够完成的最早时间。EF=ES+工期
 - （3）最迟结束时间（LF）。为了使项目按时完成，某项活动必须完成的最迟时间。
 - （4）最迟开始时间（LS）。为了使项目按时完成，某项活动必须开始的最迟时间。LS=LF-工期
- 顺推：最高开始 ES=所有前置活动最早完成 EF 的最大值；最早完成 EF=最早开始 ES+持续时间。
- 逆推：最晚完成 LF=所有后续活动最晚开始 LS 的最小值；最晚开始 LS=最晚完成 LF-持续事件。



总浮动时间：在不延误项目完工时间且不违反进度制约因素的前提下，活动可以从最早开始时间推迟或拖延的时间量，就是该活动的进度灵活性。正常情况下，关键活动的总浮动时间为零。

总浮动时间=最迟开始LS-最早开始ES 或 最迟完成LF-最早完成EF 或 关键路径-非关键路径时长。

自由浮动时间：是指在不延误任何紧后活动的最早开始时间且不违反进度制约因素的前提下，活动可以从最早开始时间推迟或拖延的时间量。

自由浮动时间=紧后活动最早开始时间的最小值-本活动的最早完成时间。

掌握关键路径的计算，会依据表格和题目描述画图，然后求出关键路径。

3 数据库系统

在架构里考察并不算频繁，偶尔会考察一题，考察的范围与系分也完全不同，主要考察的是数据库的一些新技术的比较，如关系型数据库、内存数据库及 NoSQL 等，很少涉及到规范化，但也要掌握。

ORM 技术

ORM，即 Object-Relationl Mapping，它在关系型数据库和对象之间作一个映射，这样，我们在具体的操作数据库的时候，就不需要再去和复杂的 SQL 语句打交道，只要像平时操作对象一样操作即可。

面向对象编程把所有实体看成对象(object)，关系型数据库则是采用实体之间的关系(relation)连接数据。很早就有人提出，关系也可以用对象表达，这样的话，就能使用面向对象编程，来操作关系型数据库。

ORM 把数据库映射成对象。如：

数据库的表 (table) --> 类 (class)

记录 (record, 行数据) --> 对象 (object)

字段 (field) --> 对象的属性 (attribute)

ORM 优点：

- 1、使用 ORM 可以大大降低学习和开发成本。
- 2、程序员不用再写 SQL 来进行数据库操作。
- 3、减少程序的代码量。
- 4、降低由于 SQL 代码质量差而带来的影响。

ORM 缺点

- 1、不太容易处理复杂查询语句。
- 2、性能较直接用 SQL 差。

数据库分类比较

关系型数据库：关系数据库，是建立在关系模型基础上的数据库，借助集合代数等数学概念和方法来处理数据库中的数据。现实世界中的各种实体以及实体之间的各种联系均用关系模型来表示。简单说，关系型数据库是由多张能互相联接的二维行列表格组成的数据库。

NoSQL：泛指非关系型的数据库。随着互联网的兴起，传统的关系数据库在应付超大规模和高并发的纯动态网站已经显得力不从心，暴露了很多难以克服的问题，而非关系型的数据库则由于其本身的特点得到了非常迅速的发展。NoSQL 数据库的产生就是为了解决大规模数据集合多重数据种类带来的挑战，尤其是大数据应用难题，包括超大规模数据的存储。

内存数据库：将数据库整体存储在内存中，提高性能。

特征	关系数据库模式	NoSQL 模式
并发支持	支持并发、效率低	并发性能高
存储与查询	关系表方式存储、SQL 查询	海量数据存储、查询效率高
扩展方式	向上扩展	向外扩展
索引方式	B 树、哈希等	键值索引
应用领域	面向通用领域	特定应用领域
数据一致性	实时一致性	弱一致性
数据类型	结构化数据	非结构化
事物	高事务性	弱事务性
水平扩展	弱	强
数据容量	有限数据	海量数据

	主要数据模型	读写性能	存储容量	可靠性
内存数据库	Key-Value 模式 (键-值对模式)	内存直接读写， 性能相对较高	基于内存存储， 存储容量受限	恢复机制复杂， 可靠性较低
关系数据库	关系模式	外存读写，性能 相对较低	基于磁盘存储， 存储容量大	内建恢复机制， 可靠性较高

	设计难度	数据冗余程度	数据架构	应用扩展性
关系型数据库	针对特定应用系统设计，难度较大	遵守数据库范式，数据冗余较小	以数据库为中心组织，管理数据	数据库独立于应用系统，数据库系统接口标准化，易于在不同应用之间共享数据
文件系统	针对特定应用系统设计，难度较小	可能在多个文件中复制相同的数据属性，数据冗余较大	以应用为中心管理数据	符合特定应用系统要求的文件数据很难在不同的应用系统之间共享

并发控制

丢失更新：事务 1 对数据 A 进行了修改并写回，事务 2 也对 A 进行了修改并写回，此时事务 2 写回的数据会覆盖事务 1 写回的数据，就丢失了事务 1 对 A 的更新。即**对数据 A 的更新会被覆盖**。

不可重复读：事务 2 读 A，而后事务 1 对数据 A 进行了修改并写回，此时若事务 2 再读 A，发现数据不对。即**一个事务重复读 A 两次，会发现数据 A 有误**。

读脏数据：事务 1 对数据 A 进行了修改后，事务 2 读数据 A，而后事务 1 回滚，数据 A 恢复了原来的值，那么事务 2 对数据 A 做的事是无效的，读到了脏数据。

◆ 丢失更新

T1	T2
①读A=10 ② ③A=A-5写回 ④	读A=10 A=A-8 写回

◆ 不可重复读

T1	T2
①读A=20 读B=30 求和=50 ② ③读A=70 读B=30 求和=100 (验算不对)	读A=20 A←A+50 写A=70

◆ 读“脏”数据

T1	T2
①读A=20 A←A+50 写回70 ② ③ROLLBACK A恢复为20	读A=70

封锁协议

X 锁是排它锁（写锁）。若事务 T 对数据对象 A 加上 X 锁，则只允许 T 读取和修改 A，其他事务都不能再对 A 加任何类型的锁，直到 T 释放 A 上的锁。

S 锁是共享锁（读锁）。若事务 T 对数据对象 A 加上 S 锁，则只允许 T 读取 A，但不能修改 A，其他事务只能再对 A 加 S 锁（也即能读不能修改），直到 T 释放 A 上的 S 锁。

加锁遵循一个基本原则：如果该事务只读数据，就只加读锁；如果该事务要写数据，就加写锁。

不规范化带来的四大问题

设有一个关系模式 R (SNAME, CNAME, TNAME TADDRESS)，其属性分别表示学生姓名、选修的课程名、任课教师姓名和任课教师地址。仔细分析一下，就会发现这个模式存在下列存储异常的问题：

(1) 数据冗余：数据被重复存储，如某门课程有 100 个学生选修，那么在 R 的关系中就要出现 100 个元组，这门课程的任课教师姓名和地址也随之重复出现 100 次。

(2) 修改异常：修改导致数据不一致，如由于上述冗余问题，当需要修改这个教师的地址时，就要修改 100 个元组中的地址值，否则就会出现地址值不一致的现象。

(3) 插入异常：插入时异常，如不知道听课学生名单，这个教师的任课情况和家庭地址就无法进入数据库；否则就要在学生姓名处插入空值。

(4) 删除异常：删除了不该删除的数据，如当只有一条记录时，要删除这个学生选课信息，会将课程名、教师名和教师地址都给删除了。

反规范化技术

规范化设计后，数据库设计者希望牺牲部分规范化来提高性能，这种从规范化设计的回退方法称为反规范化技术。

采用反规范化技术的益处：降低连接操作的需求、降低外码和索引的数目，还可能减少表的数目，能够提高查询效率。

可能带来的问题：数据的重复存储，浪费了磁盘空间；可能出现数据的完整性问题，为了保障数据的一致性，增加了数据维护的复杂性，会降低修改速度。

(1) 增加冗余列：在多个表中保留相同的列，通过增加数据冗余减少或避免查询时的连接操作。

(2) 增加派生列：在表中增加可以由本表或其它表中数据计算生成的列，减少查询时的连接操作并避免计算或使用集合函数。

(3) 重新组表：如果许多用户需要查看两个表连接出来的结果数据，则把这两个表重新组成一个表来减少连接而提高性能。

(4) 水平分割表：根据一列或多列数据的值，把数据放到多个独立的表中，主要用于表数据规模很大、表中数据相对独立或数据需要存放到多个介质上时使用。

(5) 垂直分割表：对表进行分割，将主键与部分列放到一个表中，主键与其它列放到另一个表中，在查询时减少 I/O 次数。

分布式数据库是由一组数据组成的，这组数据分布在计算机网络的不同计算机上，网络中的每个节点具有独立处理的能力（称为场地自治），它可以执行局部应用，同时，每个节点也能通过网络通信子系统执行全局应用。分布式数据库系统是在集中式数据库系统技术的基础上发展起来的，具有如下特点：

(1) 数据独立性。在分布式数据库系统中，数据独立性这一特性更加重要，并具有更多的内容。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, 这样, 可以提高系统的可靠性和可用性, 同时也能提高系统性能。

(4) 全局的一致性、可串行性和可恢复性。

分布式数据库的优点:

(1) 分布式数据库可以解决企业部门分散而数据需要相互联系的问题。

(2) 如果企业需要增加新的相对自主的部门来扩充机构, 则分布式数据库系统可以在对当前机构影响最小的情况下进行扩充。

(3) 分布式数据库可以满足均衡负载的需要。

(4) 当企业已存在几个数据库系统, 而且实现全局应用的必要性增加时, 就可以由这些数据库自下而上构成分布式数据库系统。

(5) 相等规模的分布式数据库系统在出现故障的概率上不会比集中式数据库系统低, 但由于其故障的影响仅限于局部数据应用, 因此, 就整个系统来说, 它的可靠性是比较高的。

数据分片将数据库整体逻辑结构分解为合适的逻辑单位(片段), 然后由分布模式来定义片段及其副本在各场地的物理分布, 其主要目的是提高访问的局部性, 有利于按照用户的需求, 组织数据的分布和控制数据的冗余度。

(1) 水平分片。水平分片将一个全局关系中的元组分裂成多个子集, 每个子集为一个片段。分片条件由关系中的属性值表示。对于水平分片, 重构全局关系可通过关系的并操作实现。

(2) 垂直分片。垂直分片将一个全局关系按属性分裂成多个子集, 应满足不相交性(关键字除外)。对于垂直分片, 重构全局关系可通过连接运算实现。

(3) 导出分片。导出分片又称为导出水平分片, 即水平分片的条件不是本关系属性的条件, 而是其他关系属性的条件。

(4) 混合分片。混合分片是在分片中采用水平分片和垂直分片两种形式的混合。

分布透明性是指用户不必关心数据的逻辑分片, 不必关心数据存储的物理位置分配细节, 也不必关心局部场地上数据库的数据模型。

(1) 分片透明性是分布透明性的最高层次, 它是指用户或应用程序只对全局关系进行操作而不必考虑数据的分片。

(2) 位置透明性。位置透明性是指用户或应用程序应当了解分片情况, 但不必了解片段的存储场地。

(3) 局部数据模型透明性。局部数据模型透明性是指用户或应用程序应当了解分片及各片断存储的场地, 但不必了解局部场地上使用的是何种数据模型。

数据仓库集成是把多种来源的数据集中在一起, 建立数据仓库, 所有数据都驻留在单个数据库服务器上, 配置大型处理器和存储容量。数据仓库主要用于决策支持, 在数据处理过程中强调分析。其特点是:(1) 集成的数据。(2) 面向主题。(3) 数据相对稳定。(4) 包含历史信息。

数据仓库的结构通常包含四个层次:

- 1.数据源：是数据仓库系统的基础，是整个系统的数据源泉。
- 2.数据的存储与管理：是整个数据仓库系统的核心。
- 3.OLAP（联机分析处理）服务器：对分析需要的数据进行有效集成，按多维模型组织，以便进行多角度、多层次的分析，并发现趋势。
- 4.前端工具：主要包括各种报表工具、查询工具、数据分析工具、数据挖掘工具以及各种基于数据仓库或数据集市的应用开发工具。

4 嵌入式系统

几乎每年必考一题，选做题，考察比较的多的是嵌入式系统的实时性和可靠性以及容错等概念。大概率会考到一些嵌入式领域陌生技术，如果是完全没见过的技术，不选即可。

相关概念

系统可靠性是系统在规定的时间内及规定的环境条件下，完成规定功能的能力，也就是系统无故障运行的概率。

系统可用性是指在某个给定时间点上系统能够按照需求执行的概率。

可靠度就是系统在规定的条件下、规定的时间内不发生失效的概率。

失效率又称风险函数，也可以称为条件失效强度，是指运行至此刻系统未出现失效的情况下，单位时间系统出现失效的概率。

软件可靠性和硬件可靠性区别

复杂性：软件复杂性比硬件高，大部分失效来自于软件失效。

物理退化：硬件失效主要是物理退化所致，软件不存在物理退化。

唯一性：软件是唯一的，每个 COPY 版本都一样，而两个硬件不可能完全一样。

版本更新周期：硬件较慢，软件较快。

可靠性指标

平均无故障时间 $MTTF=1/\text{失效率}$ 。

平均故障修复时间 $MTTR=1/\text{修复率}$ 。

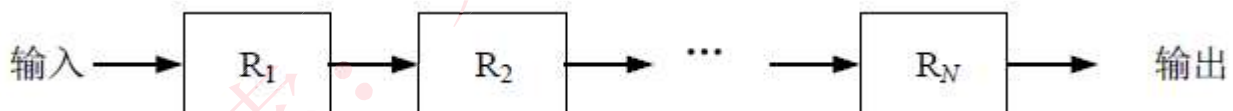
平均故障间隔时间 $MTBF=MTTF+MTTR$ 。

系统可用性= $MTTF/(MTTF+MTTR)*100\%$ 。

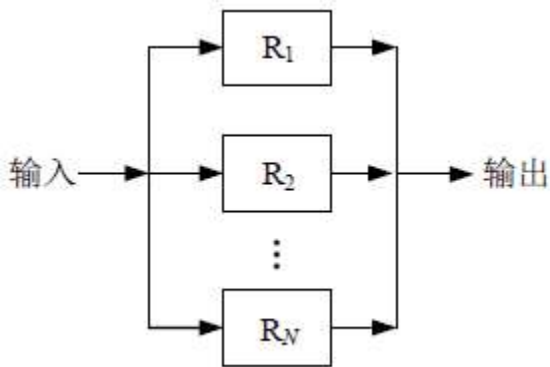
串并联系统可靠性计算

无论什么系统，都是由多个设备组成的，协同工作，而这多个设备的组合方式可以是串联、并联，也可以是混合模式，假设每个设备的可靠性为 $R_1, R_2, \dots, R_n$ ，则不同的系统的可靠性公式如下：

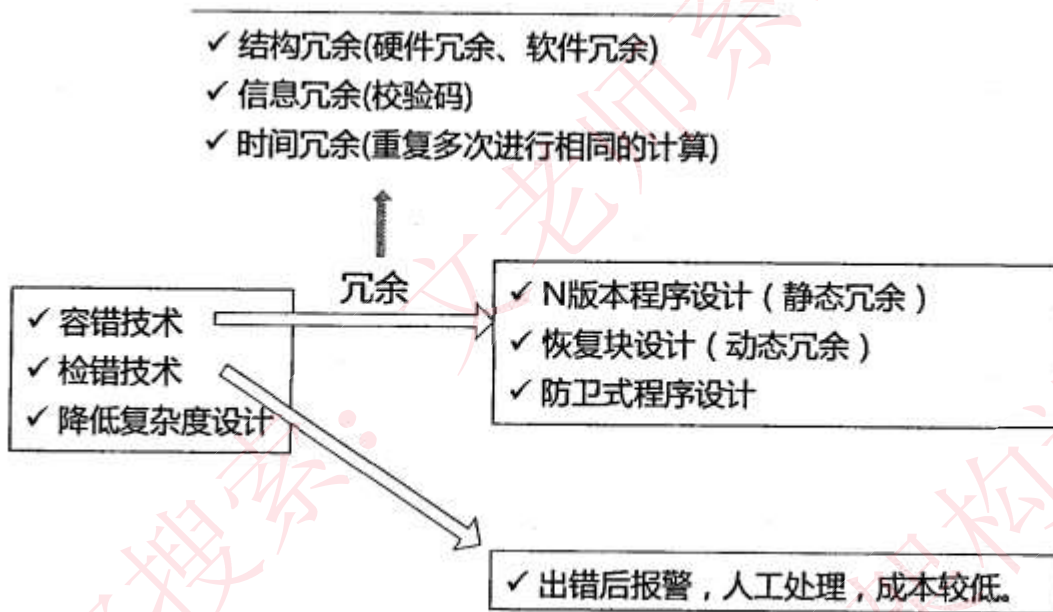
串联系统，一个设备不可靠，整个系统崩溃，整个系统可靠性 $R=R_1 * R_2 * \dots * R_n$ 。



并联系统，所有设备都不可靠，整个系统才崩溃，整个系统可靠性 $R=1-(1-R_1) * (1-R_2) * \dots * (1-R_n)$ 。



可靠性设计



常见的软件可靠性技术主要有容错设计、检错设计和降低复杂度设计等技术。

容错设计技术主要有恢复快设计、N 版本程序设计和冗余设计三种方法。

冗余技术

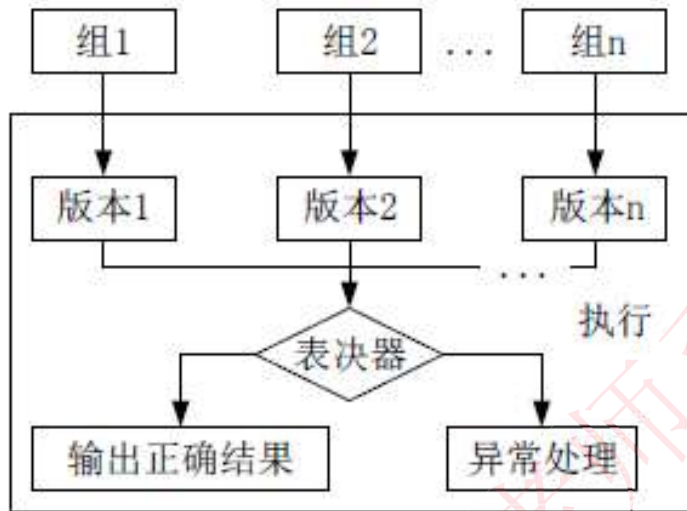
提高系统可靠性的技术可以分为避错（排错）技术和容错技术。避错是通过技术评审、系统测试和正确性证明等技术，在系统正式运行之前避免、发现和改正错误。

容错是指系统在运行过程中发生一定的硬件故障或软件错误时，仍能保持正常工作而不影响正确结果的一种性能或措施。容错技术主要是采用冗余方法来消除故障的影响。

冗余是指在正常系统运行所需的基础上加上一定数量的资源，包括信息、时间、硬件和软件。冗余是容错技术的基础，通过冗余资源的加入，可以使系统的可靠性得到较大的提高。主要的冗余技术有结构冗余（硬件冗余和软件冗余）、信息冗余、时间冗余和冗余附加 4 种。

软件容错的主要方法是提供足够的冗余信息和算法程序，使系统在实际运行时能够及时发现程序设计错误，采取补救措施，以提高系统可靠性，保证整个系统的正常运行。软件容错技术主要有 N 版本程序设计、恢复块方法和防卫式程序设计等。

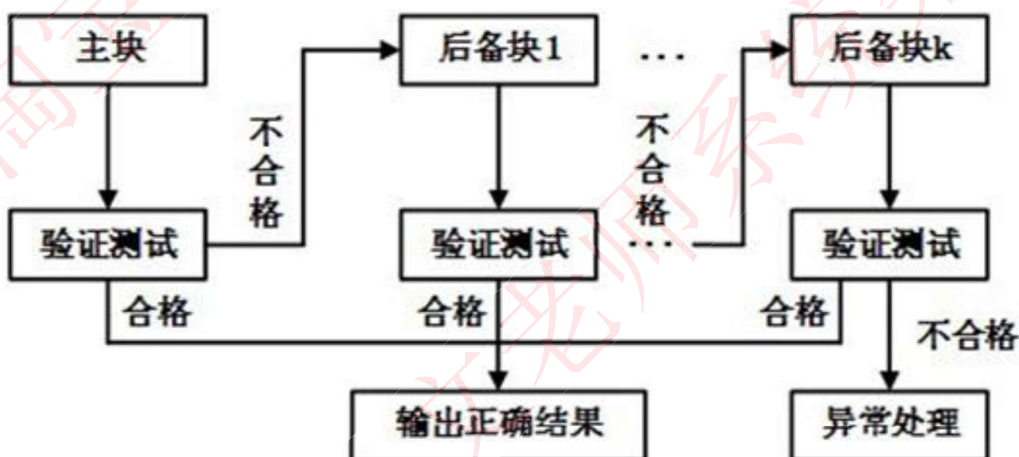
N 版本程序设计：是一种静态的故障屏蔽技术，其设计思想是用 N 个具有相同功能的程序同时执行一项计算，结果通过多数表决来选择。其中 N 个版本的程序必须由不同的人独立设计，使用不同的方法、设计语言、开发环境和工具来实现，目的是减少 N 个版本的程序在表决点上相关错误的概率。



与通常软件开发过程不同的是，N 版本程序设计增加了三个新的阶段：相异成分规范评审、相异性确认、背对背测试。

其他需要注意的问题：N 版本程序的同步、N 版本程序之间的通信、表决算法、一直比较问题、数据相异性。

恢复块设计（动态冗余）：动态冗余又称为主动冗余，它是通过故障检测、故障定位及故障恢复等手段达到容错的目的。其主要方式是多重模块待机储备，当系统检测到某工作模块出现错误时，就用一个备用的模块来替代它并重新运行。各备用模块在其待机时，可与主模块一样工作，也可以不工作。前者叫热备份系统（双重系统），后者叫冷备份系统（双工系统、双份系统）。



设计时应保证实现主块和后备块之间的独立性，避免相关错误的产生，使主块和备份块之间的共性错误降到最低程度。

二者比较

表 19-1 恢复块方法与N版本程序设计的比较

	恢复块方法	N 版本程序设计
硬件运行环境	单机	多机
错误检测方法	验证测试程序	表决
恢复策略	后向恢复	前向恢复
实时性	差	好

防卫式程序设计：是一种不采用任何传统的容错技术就能实现软件容错的方法，对于程序中存在的错误和不一致性，防卫式程序设计的基本思想是通过在程序中包含错误检查代码和错误恢复代码，使得一旦发生错误，程序就能撤销错误状态，恢复到一个已知的正确状态中去。其实现策略包括错误检测、破坏估计和错误恢复三个方面。

双机容错技术：是一种软硬件结合的容错应用方案。该方案是由两台服务器和一个外接共享磁盘阵列及相应的双机软件组成。

双机容错系统采用“心跳”方法保证主系统与备用系统的联系。所谓心跳，是指主从系统之间相互按照一定的时间间隔发送通信信号，表明各自系统当前的运行状态。一旦心跳信号表明主机系统发生故障，或者备用系统无法收到主系统的心跳信号，则系统的高可用性管理软件认为主系统发生故障，立即将系统资源转移到备用系统上，备用系统替代主系统工作，以保证系统正常运行和网络服务不间断。

工作模式：双机热备模式；双机互备模式；双机双工模式。

集群技术就是将多台计算机组织起来进行协同工作，它是提高系统可用性和可靠性的一种技术。在集群系统中，每台计算机均承担部分计算任务和容错任务，当其中一台计算机出现故障时，系统使用集群软件将这台计算机从系统中隔出离去，通过各计算机之间的负载转嫁机制完成新的负载分担，同时向系统管理人员发出警报。集群系统通过功能整合和故障过渡，实现了系统的高可用性和可靠性。

特点：可伸缩性、高可用性、可管理性、高性价比、高透明性。

分类：高性能计算集群、负载均衡集群、高可用性集群。

负载均衡是集群系统中的一项重要技术，可以提高集群系统的整体处理能力，也提高了系统的可靠性，最终目的是加快集群系统的响应速度，提高客户端访问的成功概率。集群的最大特征是多个节点的并行和共同工作，如何让所有节点承受的负荷平均，不出现局部过大负载或过轻负载的情况，是负载均衡的重要目的。

比较常用的负载均衡实现技术主要有以下几种：

(1) 基于特定软件的负载均衡。很多网络协议都支持重定向功能，例如，基于 HTTP 重定向服务，其主要原理是服务器使用 HTTP 重定向指令，将一个客户端重新定位到另一个位置。服务器返回一个重定向响应，而不是返回请求的对象。客户端确认新地址然后重发请求，从而达到负载均衡的目的。

(2) 基于 DNS 的负载均衡属于传输层负载均衡技术，其主要原理是在 DNS 服务器中为同一个主机名配置多个地址，在应答 DNS 查询时，DNS 服务器对每个查询将以 DNS 文件中主机记录的 IP 地址按顺序返回不同的解析结果，将客户端的访问引导到不同的节点上去，使得不同的客户端访问不同的节点，从而达到负载均衡的目的。

(3) 基于 NAT 的负载均衡。将一个外部 IP 地址映射为多个内部 IP 地址，对每次连接需求动态地转换为一个内部节点的地址，将外部连接请求引到转换得到地址的那个节点，从而达到负载均衡的目的。

(4) 反向代理负载均衡。将来自 Internet 上的连接请求以反向代理的方式动态地转发给内部网络上的多个节点进行处理，从而达到负载均衡的目的。

(5) 混合型负载均衡。

可维护性的评价指标：

(1) 易分析性。软件产品诊断软件中的缺陷或失效原因或识别待修改部分的能力。

(2) 易改变性。软件产品使指定的修改可以被实现的能力，实现包括编码、设计和文档的更改。如果软件由最终用户修改，那么易改变性可能会影响易操作性。

(3) 稳定性。软件产品避免由于软件修改而造成意外结果的能力。

(4) 易测试性。软件产品使已修改软件能被确认的能力。

(5) 维护性的依从性。软件产品遵循与维护性相关的标准或约定的能力。

软件维护的分类

(1) 改正性维护。为了识别和纠正软件错误、改正软件性能上的缺陷、排除实施中的误使用，应当进行的诊断和改正错误的过程就称为改正性维护。

(2) 适应性维护。在使用过程中，外部环境（新的硬、软件配置）、数据环境（数据库、数据格式、数据输入/输出方式、数据存储介质）可能发生变化。为使软件适应这种变化，而去修改软件的过程就称为适应性维护。

(3) 完善性维护。在软件的使用过程中，用户往往会对软件提出新的功能与性能要求。为了满足这些要求，需要修改或再开发软件，以扩充软件功能、增强软件性能、改进加工效率、提高软件的可维护性。这种情况下进行的维护活动称为完善性维护。

(4) 预防性维护。这是指预先提高软件的可维护性、可靠性等，为以后进一步改进软件打下良好基础。通常，预防性维护可定义为“把今天的方法学用于昨天的系统以满足明天的需要”。也就是说，采用先进的软件工程方法对需要维护的软件或软件中的某一部分（重新）进行设计、编码和测试。

5 Web 应用开发

主要考察 Web 相关技术，尤其是新技术的应用，涵盖知识点章节：系统设计-Web 应用设计。

Web 技术一般不会重复考察，每年都有新技术，需要依据答题技巧灵活应变，遇到完全没听说过的技术，就不选。

Web 应用技术分类

从架构来看：MVC，MVP，MVVM，REST，Webservice，微服务。

从缓存来看：MemCache，Redis，Squid。

从并发分流来看：集群（负载均衡）、CDN。

从数据库来看：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。

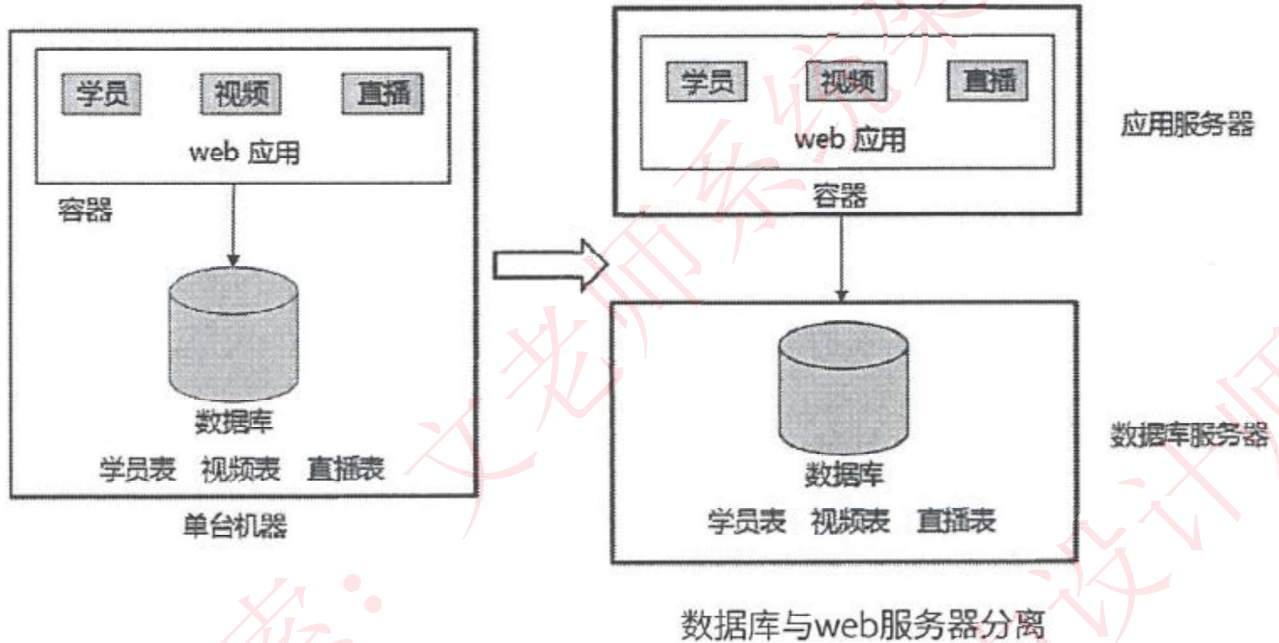
从分布存储来看: Hadoop, FastDFS, 区块链。

从数据编码看: XML, JSON。

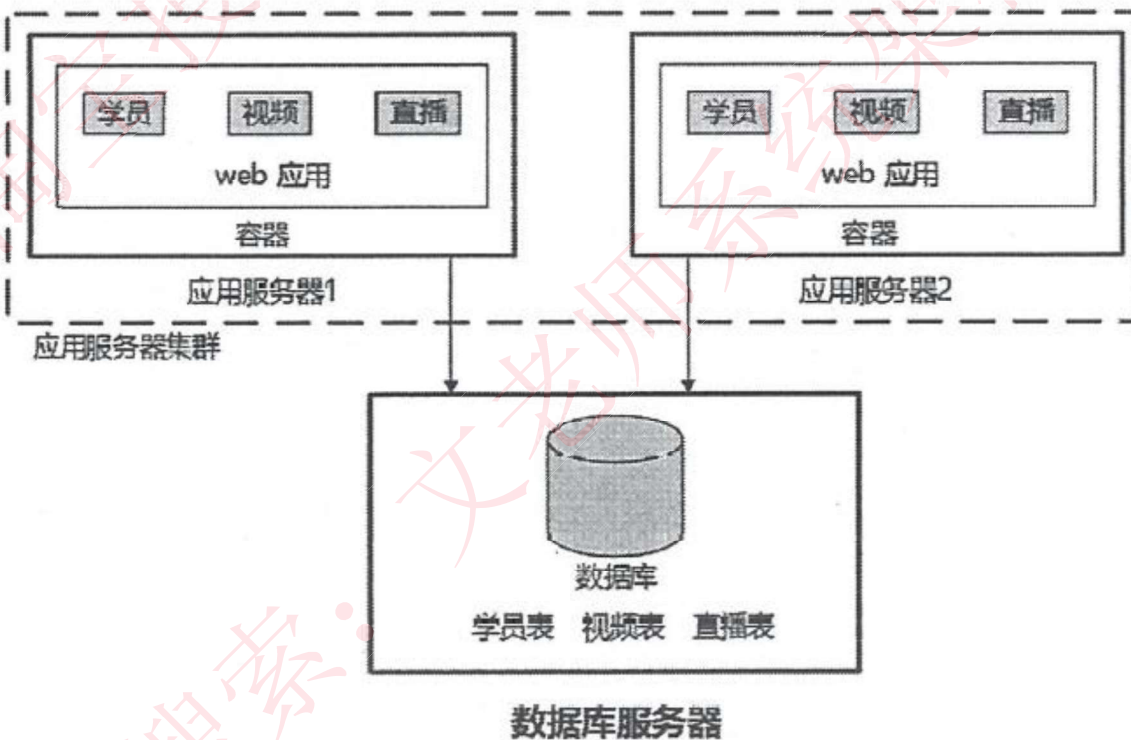
从 Web 应用服务器来看: Apache, WebSphere, Weblogic, Tomcat, JBOSS, IIS。

其它: 静态化, 有状态与无状态, 响应式 Web 设计。

单台机器到数据库与 Web 服务器分离



应用服务器集群

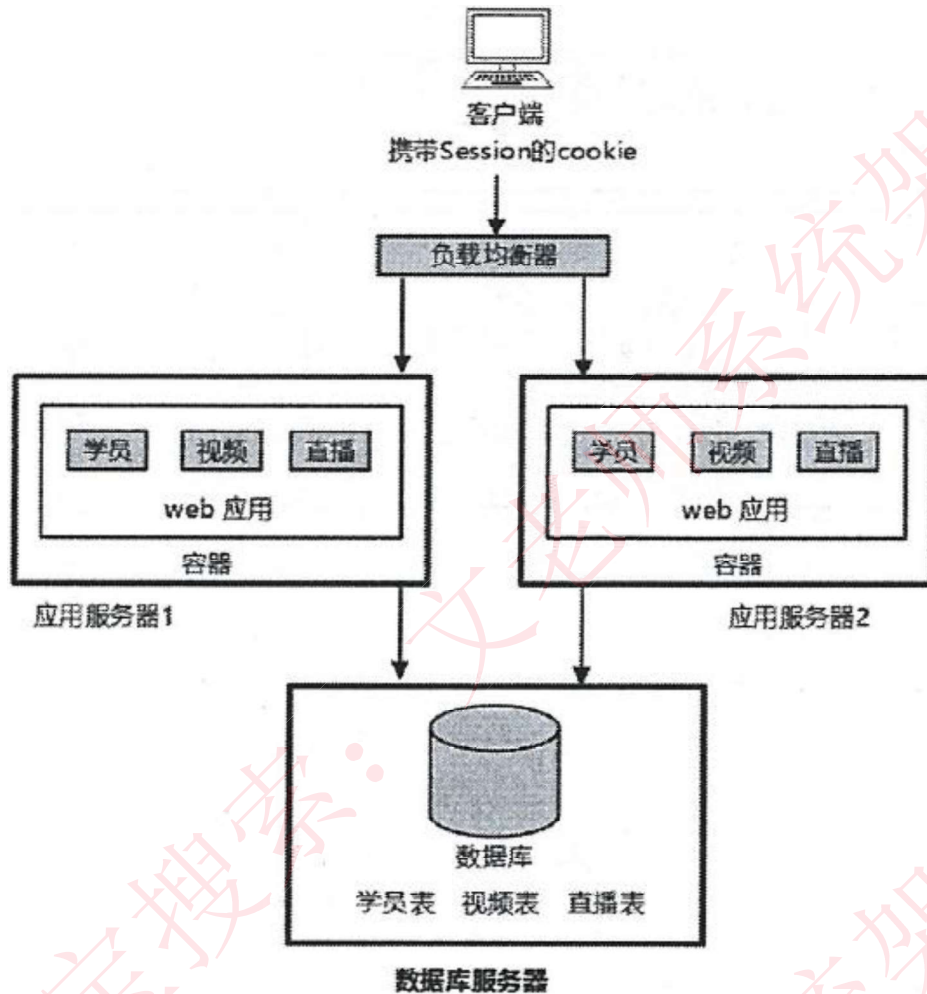


系统演变到这里, 将会出现问题:

用户的请求由谁来转发到具体的应用服务器；

用户如果每次访问到的服务器不一样，那么如何维护 session 的一致性。

负载均衡技术



应用层负载均衡技术

1、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。

2、反向代理服务器。在用户的请求到这反向代理服务器时（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。

传输层负载均衡技术

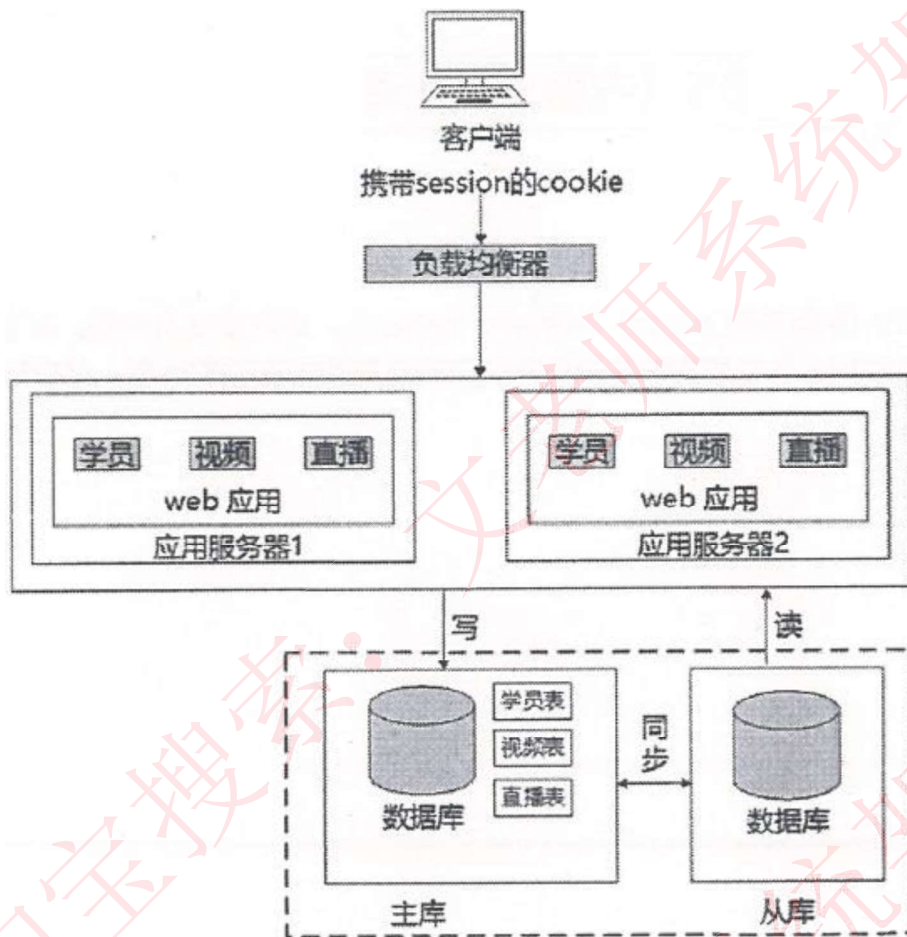
1、DNS 域名解析负载均衡。DNS 域名解析负载均衡就是在用户请求 DNS 服务器，获取域名对应的 IP 地址时，DNS 服务器直接给出负载均衡后的服务器 IP。

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。

2、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。

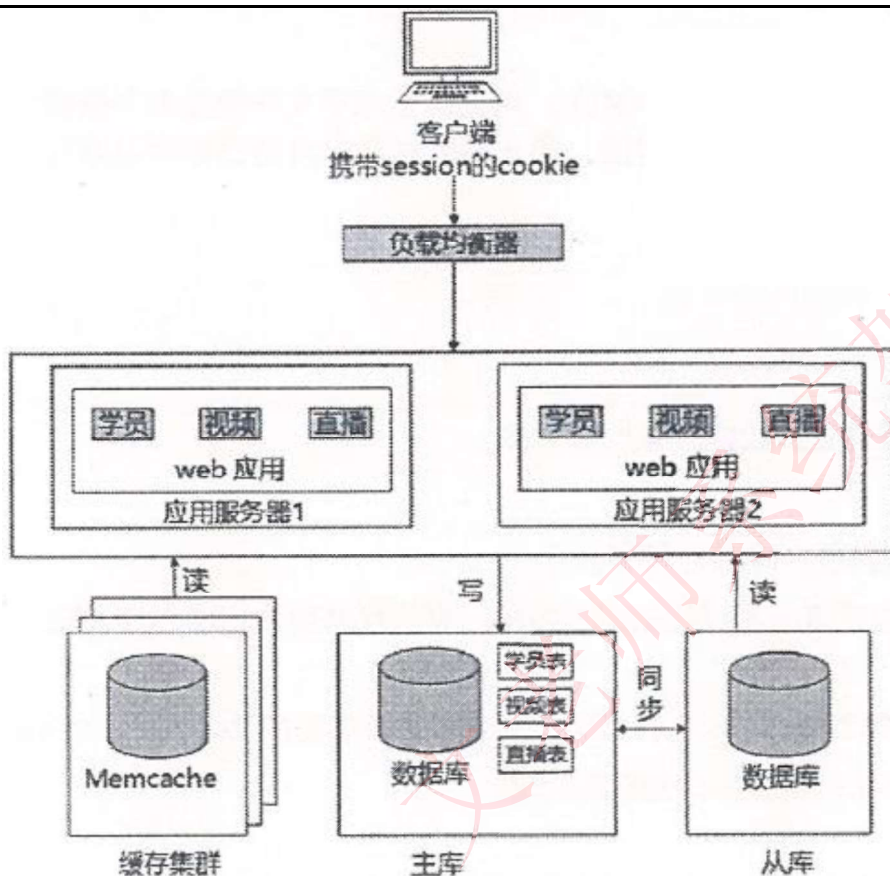
特点：技术较为成熟，一般在网关位置，可以通过硬件实现。像四层交换机一般就采用了这种技术。

数据库读写分离化



用缓存缓解库读取压力

MemCached 是一个自由开源的，高性能，分布式内存对象缓存系统。简洁的 key-value 存储系统。通过缓存数据库查询结果，减少数据库访问次数，以提高动态 Web 应用的速度、提高可扩展性。



有状态和无状态

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

CDN

CDN 的全称是 Content Delivery Network，即内容分发网络。

CDN 是构建在网络之上的内容分发网络，依靠部署在各地的边缘服务器，通过中心平台的负载均衡、内容分发、调度等功能模块，使用户就近获取所需内容，降低网络拥塞，提高用户访问响应速度和命中率。CDN 的关键技术主要有内容存储和分发技术。

CDN 的基本原理是广泛采用各种缓存服务器，将这些缓存服务器分布到用户访问相对集中的地区或网络中，在用户访问网站时，利用全局负载技术将用户的访问指向距离最近的工作正常的缓存服务器上，由缓存服务器直接响应用户请求。

REST（表述性状态传递）

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

- (1) 网络上的所有事物都被抽象为资源。
- (2) 每个资源对应一个唯一的资源标识。

- (3) 通过通用的连接件接口对资源进行操作。
- (4) 对资源的各种操作不会改变资源标识。
- (5) 所有的操作都是无状态的。

微服务

微服务架构建议将大型复杂的单体架构应用划分为一组微小的服务，每个微服务根据其负责的具体业务职责提炼为单一的业务功能；每个服务可以很容易地部署并发布到生产环境里隔离和独立的进程内部，它可以很容易地扩展和变更；对于一个具体的服务来说可以采用任何适用的语言和工具来快速实现；服务之间基于基础设施互相协同工作。

微服务的优势：

- (1) 解决了复杂性问题。它把庞大的单一模块应用分解为一系列的服务，同时保持总体功能不变。
- (2) 让每个服务能够独立开发，开发者能够自由选择可行的技术，让服务来决定 API 约定。
- (3) 每个微服务都能独立配置，开发者不必协调对于本地服务配置上的变化，这种变化一旦测试完成就被配置了。
- (4) 让每个服务都可以独立调整，你可以给每个服务配置正好满足容量和可用性限制的实例数。

微服务架构带来的挑战：

- (1) 并非所有的系统都能转成微服务。例如一些数据库层的底层操作是不推荐服务化的。
- (2) 部署较以往架构更加复杂：系统由众多微服务搭建，每个微服务需要单独部署，从而增加部署的复杂度，容器技术能够解决这一问题。
- (3) 性能问题：由于微服务注重独立性，互相通信时只能通过标准接口，可能产生延迟或调用出错。例如一个服务需要访问另一个服务的数据，只能通过服务间接口来进行数据传输，如果是频繁访问，则可能带来较大的延迟。
- (4) 数据一致性问题：作为分布式部署的微服务，在保持数据一致性方面需要比传统架构更加困难。

微服务与 SOA 比较

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

缓存技术

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, 它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。

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。

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。

Redis 与 Memcache 的差异

1、Redis 和 Memcache 都是将数据存放在内存中, 都是内存数据库。他们都支持 key-value 数据类型。同时 Memcache 还可用于缓存其他东西, 例如图片、视频等等, Redis 还支持 list、set、hash 等数据结构的存储。

2、Redis 支持数据的持久化, 可以将内存中的数据保持在磁盘中, 重启的时候可以再次加载进行使用, Memcache 挂掉之后, 数据就没了。

3、灾难恢复—Memcache 挂掉后, 数据不可恢复; Redis 数据丢失后可以恢复。

4、在 Redis 中, 并不是所有的数据都一直存储在内存中的。这是和 Memcache 相比一个最大的区别。当物理内存用完时, Redis 可以将一些很久没用到的 value 交换到磁盘。

5、Redis 在很多方面支持数据库的特性, 可以这样说他就是一个数据库系统, 而 Memcache 只是简单地 K/V 缓存。

所以在选择方面如果有持久方面的需求或对数据类型和处理有要求的应该选择 Redis。

如果简单的 key/value 存储应该选择 Memcache。

XML 和 JSON

扩展标记语言 (Extensible Markup Language, XML), 用于标记电子文件使其具有结构性的标记语言, 可以用来标记数据、定义数据类型, 是一种允许用户对自己的标记语言进行定义的源语言。

XML 的优点

格式统一, 符合标准;

容易与其他系统进行远程交互, 数据共享比较方便。

XML 的缺点

XML 文件庞大, 文件格式复杂, 传输占带宽;

服务器端和客户端都需要花费大量代码来解析 XML, 导致服务器端和客户端代码变得异常复杂且不易维护;

客户端不同浏览器之间解析 XML 的方式不一致, 需要重复编写很多代码;

服务器端和客户端解析 XML 花费较多的资源和时间。

JSON(JavaScript Object Notation) 一种轻量级的数据交换格式, 具有良好的可读和便于快速编写的特性。可在不同平台之间进行数据交换。

JSON 的优点

数据格式比较简单, 易于读写, 格式都是压缩的, 占用带宽小;

易于解析, 客户端 JavaScript 可以简单的通过 eval() 进行 JSON 数据的读取;

支持多种语言, 包括 ActionScript, C, C#, ColdFusion, Java, JavaScript, Perl, PHP, Python, Ruby 等服务器端语言, 便于服务器端的解析;

因为 JSON 格式能直接为服务器端代码使用, 大大简化了服务器端和客户端的代码开发量, 且完成任务不变, 并且易于维护。

JSON 的缺点

没有 XML 格式这么推广的深入人心和喜用广泛, 没有 XML 那么通用性。

Web 应用服务器

WEB 应用服务器可以理解为两层意思:

- (1) WEB 服务器: 其职能较为单一, 就是把浏览器发过来的 Request 位请求, 返回 Html 页面。
- (2) 应用服务器: 进行业务逻辑的处理。

Apache: Web 服务器, 市场占有率达 60% 左右。它可以运行在几乎所有的 Unix、Windows、Linux 系统平台上。

IIS 早期 Web 服务器, 目前小规模站点仍有应用。

Tomcat: 开源、运行 servlet 和 JSPWeb 应用程序的基于 Java 的 Web 应用软件容器。

JBoss: JBoss 是基于 J2EE 的开放源代码的应用服务器, 一般与 Tomcat 或 Jetty 绑定使用。

WebSphere: 一种功能完善、开放的 Web 应用程序服务, 它是基于 Java 的应用环境, 用于建立、部署和管理 Internet 和 IntranetWeb 应用程序。

Weblogic: BEA Weblogic Server 是一种多功能、基于标准的 web 应用服务器, 为企业构建自己的应用提供了坚实的基础。

Jetty: Jetty 是一个开源的 servlet 容器, 是基于 Java 的 web 容器。

其他

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。

方法与策略

- (1) 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。
- (2) 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。

三 第二版下篇架构设计考点

1 信息系统架构设计

1.1 信息系统架构基本概念

信息系统架构（ISA）是指对某一特定内容里的信息进行统筹、规划、设计、安排等一系列有机处理的活动。

为了更好地理解信息系统架构的定义，特作如下说明：

- (1) 架构是对系统的抽象，它通过描述元素、元素的外部可见属性及元素之间的关系来反映这种抽象。因此，仅与内部具体实现有关的细节是不属于架构的，即定义强调元素的“外部可见”属性。
- (2) 架构由多个结构组成，结构是从功能角度来描述元素之间的关系的，具体的结构传达了架构某方面的信息，但是个别结构一般不能代表大型信息系统架构。
- (3) 任何软件都存在架构，但不一定有对该架构的具体表述文档。即架构可以独立于架构的描述而存在。如文档已过时，则该文档不能反映架构。
- (4) 元素及其行为的集合构成架构的内容。体现系统由哪些元素组成，这些元素各有哪些功能（外部可见），以及这些元素间如何连接与互动。即在两个方面进行抽象：在静态方面，关注系统的大粒度（宏观）总体结构（如分层）；在动态方面，关注系统内关键行为的共同特征。
- (5) 架构具有“基础”性：它通常涉及解决各类关键重复问题的通用方案（复用性），以及系统设计中影响深远（架构敏感）的各项重要决策（一旦贯彻，更改的代价昂贵）。
- (6) 架构隐含有“决策”，即架构是由架构设计师根据关键的功能和非功能性需求（质量属性及项目相关的约束）进行设计与决策的结果。

1.2 信息系统架构

信息系统架构的定义，架构风格与之前介绍的软件架构一样，不再赘述。

信息系统架构可分为物理结构与逻辑结构两种，物理结构是指不考虑系统各部分的实际工作与功能结构，只抽象地考察其硬件系统的空间分布情况。逻辑结构是指信息系统各种功能子系统的综合体。

物理结构一般分为集中式与分布式两大类。

在信息系统开发中，强调各子系统之间的协调一致性和整体性。要达到这个目的，就必须在构造信息系统时注意对各种子系统进行统一规划，并对各子系统进行综合。

1) 横向综合：将同一管理层次的各种职能综合在一起，例如，将运行控制层的人事和工资子系统综合在一起，使基层业务处理一体化。

2) 纵向综合：把某种职能的各个管理层次的业务组织在一起，这种综合沟通了上下级之间的联系，如工厂的会计系统和公司的会计系统综合在一起，它们都有共同之处，能形成一体化的处理过程。

3) 纵横综合：主要是从信息模型和处理模型两个方面来进行综合，做到信息集中共享，程序尽量模块化，注意提取通用部分，建立系统公用数据库和统一的信息处理系统。

信息系统架构指的是在全面考虑企业的战略、业务、组织、管理和技术的基础上，着重研究企业信息系统的组成成分及成分之间的关系，建立起多维度分层次的、集成的开放式体系结构，并为企业提供具有一定柔性的信息系统及灵活有效的实现方法。

信息系统常用四种架构模型

1. 单机应用模式：是最简单的软件结构，是指运行在一台物理机器上的独立应用程序。单机系统本身也可以很复杂。

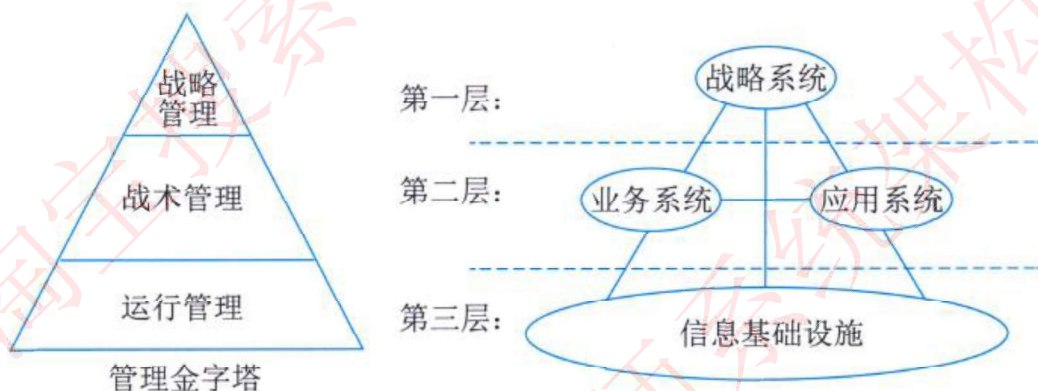
2. 客户机/服务器模式：即两层、三层 C/S、B/S 模式、MVC 模式等。

3. 面向服务架构（SOA）模式。

4. 企业数据交换总线：不同的企业应用之间进行信息交换的公共通道

企业信息系统的总体框架

要在企业中建立一个有效集成的 ISA，必须考虑企业中的四个方面：战略系统、业务系统、应用系统和信息基础设施。



1. 战略系统是指企业中与战略制定、高层决策有关的管理活动和计算机辅助系统。

在 ISA 中战略系统由两个部分组成，其一是为以计算机为基础的高层决策支持系统，其二是企业的战略规划体系。

在 ISA 中设立战略系统有两重含义：一是它表示信息系统对企业高层管理者的决策支持能力；二是它表示企业战略规划对信息系统建设的影响和要求。

2. 业务系统是指企业中完成一定业务功能的各部分（物质、能量、信息和人）组成的系统。

作用：对企业现有业务系统、业务过程和业务活动进行建模，并在企业战略的指导下，采用业务流程重组（BPR）的原理和方法进行业务过程优化重组。

3. 应用系统即应用软件系统，指信息系统中的应用软件部分。如 TPS、MIS、DSS 等。

包含两个基本组成部分：内部功能实现部分和外部界面部分。

4. 企业信息基础设施（EII）是指根据企业当前业务和可预见的发展趋势，及对信息采集、处理、存储和流通的要求，构筑由信息设备、通信网络、数据库、系统软件和支持性软件等组成的环境。这里可以将企业信息基础设施分成三部分：技术基础设施、信息资源设施和管理基础设施。

- 技术基础设施由计算机、网络、系统软件、支持性软件、数据交换协议等组成；
- 信息资源设施由数据与信息本身、数据交换的形式与标准、信息处理方法等组成；
- 管理基础设施指企业中信息系统部门的组织结构、信息资源设施管理人员的分工、企业信息基础设施的管理方法与规章制度等。

1.3 信息系统架构设计方法

TOGAF 是一种开放式企业架构框架标准，它为标准、方法论和企业架构专业人员之间的沟通提供一致性保障

该框架旨在通过以下四个目标帮助企业组织和解决所有关键业务需求：确保从关键利益相关方到团队成员的所有用户都使用相同的语言、避免被“锁定”到企业架构的专有解决方案、节省时间和金钱更有效地利用资源、实现可观的投资回报。

TOGAF 框架核心思想：模块化架构、内容框架、扩展指南、架构风格。

TOGAF 的关键是架构开发方法（ADM），为开发企业架构所需要执行各个步骤以及它们之间的关系进行详细的定义。

ADM 方法是由一组按照架构领域的架构开发顺序而排列成一个环的多个阶段所构成。

将 ADM 全生命周期十个阶段主要活动：

ADM 阶段	ADM 阶段内的活动
准备阶段	为实施成功的企业架构项目做好准备，包括定义组织机构、特定的架构框架、架构原则和工具
需求管理	完成需求的识别、保管和交付，相关联的 ADM 阶段则按优先级顺序对需求进行处理 TOGAF 项目的每个阶段，都是建立在业务需求之上并且需要对需求进行确认
阶段 A：架构愿景	设置 TOGAF 项目的范围、约束和期望。创建架构愿景，包括： ● 定义利益相关者； ● 确认业务上下文环境； ● 创建架构工作说明书； ● 取得上级批准
阶段 B：业务架构 阶段 C：信息系统架构（应用 & 数据） 阶段 D：技术架构	从业务、信息系统和技术三个层面进行架构开发，在每一个层面分别完成以下活动： ● 开发基线架构描述； ● 开发目标架构描述； ● 执行差距分析

ADM 阶段	ADM 阶段内的活动
阶段 E：机会和解决方案	进行初步实施规划，并确认在前面阶段中确定的各种构建块的交付物形式； ● 确定主要实施项目； ● 对项目分组并纳入过渡架构； ● 决定途径（制造/购买/重用、外包、商用、开源）； ● 评估优先顺序； ● 识别相依性
阶段 F：迁移规划	对阶段 E 确定的项目进行绩效分析和风险评估，制订一个详细的实施和迁移计划
阶段 G：实施治理	定义实施项目的架构限制； ● 提供实施项目的架构监督； ● 发布实施项目的架构合同； ● 监测实施项目以确保符合架构要求
阶段 H：架构变更管理	提供持续监测和变更管理的流程，以确保架构可以响应企业的需求并且将架构对于业务的价值最大化

ADM 三个级别的迭代：基于 ADM 整体的迭代、多个开发阶段间的迭代、在一个阶段内部的迭代。

信息化总体架构方法

信息化是指培育、发展以智能化工具为代表的新的生产力并使之造福于社会的历史过程。

实现信息化就要构筑和完善 6 个要素（开发利用信息资源，建设国家信息网络，推进信息技术应用，发展信息技术和产业，培育信息化人才，制定和完善信息化政策）的国家信息化体系。

完整的信息化内涵包括四方面内容：信息网络体系、信息产业基础、社会运行环境、效用积累过程。

信息化建设指品牌利用现代信息技术来支撑品牌管理的手段和过程。信息化建设包括了企业规模，企业在电话通信、网站、电子商务方面的投入情况，在客户资源管理、质量管理体系方面的建设成就等。

信息化主要体现以下 6 种特征：易用性；健壮性；平台化、灵活性、扩展性；安全性；门户化、整合性；移动性。

信息化架构一般有两种模式，一种是数据导向架构，一种是流程导向架构。对于数据导向架构重点是在数据中心，BI 商业智能等建设中使用较多，关注数据模型和数据质量；对于流程导向架构，SOA 本身就是关键方法和技术，关注端到端流程整合，以及架构对流程变化的适应度。两种架构并没有严格的边界，而是相互配合和补充。

到数据导向架构研究的是数据对象和数据对象之间的关系，这个是首要的内容。在这个完成后仍然要开始考虑数据的产生、变更、废弃等数据生命周期，这些自然涉及的数据管理的相关流程。

流程导向架构关注的是流程，架构本身的目的是为了端到端流程整合服务。因此研究切入点会是价值链分析，流程分析和分解，业务组件划分。

信息系统的生命周期可以分为系统规划、系统分析、系统设计、系统实施、系统运行和维护等五个阶段。

1. 系统规划阶段：任务是对组织的环境、目标及现行系统的状况进行初步调查，根据组织目标和发展战略确定信息系统的发展战略，对建设新系统的需求做出分析和预测，同时考虑建设新系统所受的各种约束，研究建设新系统的必要性和可能性。根据需要与可能，给出制建系统的备选方案。

输出：可行性研究报告、系统设计任务书。

2. 系统分析阶段：任务是根据系统设计任务书所确定的范围，对现行系统进行详细调查，描述现行系统的业务流程，指出现行系统的局限性和不足之处，确定新系统的基本目标和逻辑功能要求，即提出新系统的逻辑模型。系统分析阶段又称为逻辑设计阶段。这个阶段是整个系统建设的关键阶段，也是信息系统建设与一般工程项目的重要区别所在。

输出：系统说明书。

3. 系统设计阶段：系统分析阶段的任务是回答系统“做什么”的问题，而系统设计阶段要回答的问题是“怎么做”。该阶段的任务是根据系统说明书中规定的功能要求，具体设计实现逻辑模型的技术方案，也就是设计新系统的物理模型。这个阶段又称为物理设计阶段，可分为总体设计（概要设计）和详细设计两个子阶段。

输出：系统设计说明书（概要设计、详细设计说明书）。

4. 系统实施阶段：是将设计的系统付诸实施的阶段。这一阶段的任务包括计算机等设备的购置、安装和调试、程序的编写和调试、人员培训、数据文件转换、系统调试与转换等。这个阶段的特点是几个互相联系、互相制约的任务同时展开，必须精心安排、合理组织。系统实施是按实施计划分阶段完成的，每个阶段应写出实施进展报告。系统测试之后写出系统测试分析报告。

输出：实施进展报告、系统测试分析报告。

5. 系统运行和维护阶段：系统投入运行后，需要经常进行维护和评价，记录系统运行的情况，根据一定的规则对系统进行必要的修改，评价系统的工作质量和经济效益。

1.4 信息系统架构案例分析

价值驱动的体系结构

价值模型核心的特征可以简化为三种基本形式：

(1) 价值期望值：表示对某一特定功能的需求，包括内容(功能)、满意度(质量)和不同级别质量的实用性。

(2) 反作用力：系统部署实际环境中，实现某种价值期望值的难度，通常期望越高难度越大，即反作用力。

(3) 变革催化剂：表示环境中导致价值期望值发生变化的某种事件，或者是导致不同结果的限制因素。

反作用力和变革催化剂称为限制因素，把这三个统称为价值驱动因素。

体系结构挑战是因为一个或多个限制因素使得满足一个或多个期望值变得更困难。

制定体系的体系结构策略始于：

(1) 识别合适的价值背景并对其进行优先化。

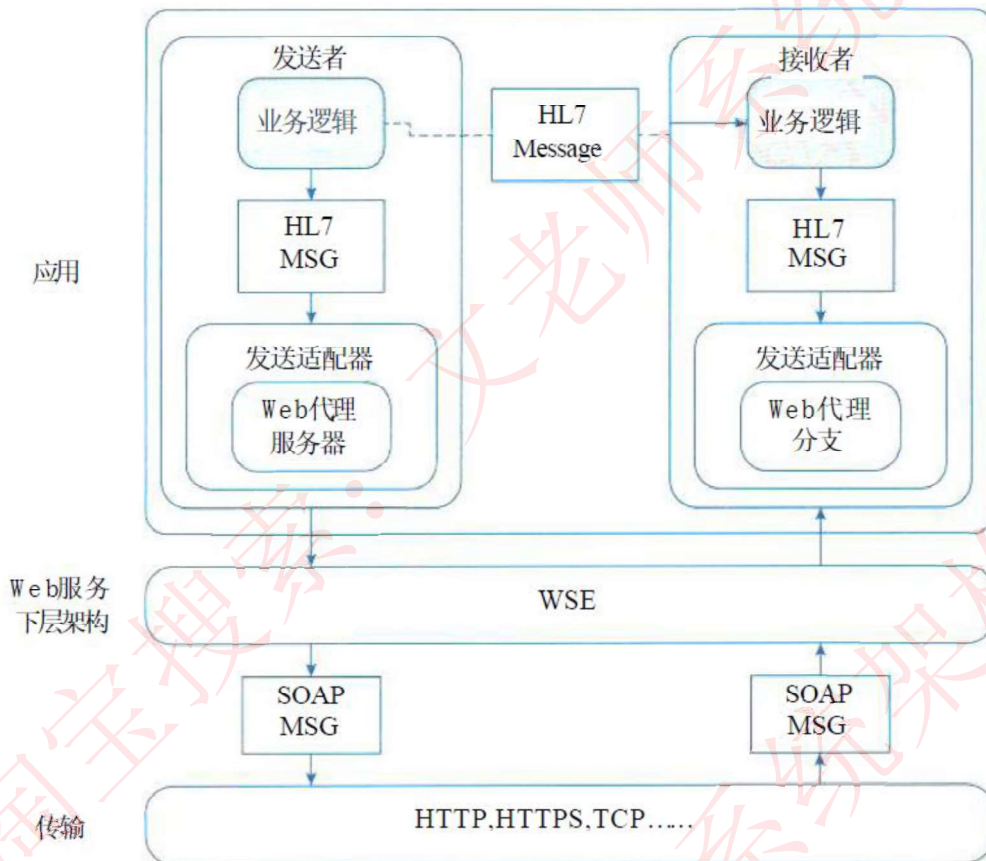
- (2)在每一背景中定义效用曲线和优先化期望值。
- (3)识别和分析每一背景中的反作用力和变革催化剂。
- (4)检测限制因素使满足期望值变难的领域。

优化体系结构需要权衡：重要性、程度、后果、隔离。

Web 服务在 HL7 上的应用

对于一个给定的卫生保健领域，HL73.0 版本说明书是基于参考信息模型的(RIM)。这是一种公共的模型框架，包括病例模型、信息模型、交互模型、消息模型和实现信息说明书。

HL7 应用参考体系结构：



能够抽象出 HL7 发送者/接收者内部的这两组功能：商业逻辑和 Web 服务适配器。商业逻辑的任务如下。

(1)发送端：创建一种具体 HL7 消息类型的 XML 描述，消息类型包含消息体、Transmission and Control Wrappers。将消息传送到 Web 服务适配器，适配器负责传送到接收应用端。

(2)接收端：“找回”由 Web 服务适配器接收的 HL7 消息，同时从接收到的 XML 消息那里打开 Transmission Wrapper、Control Wrapper 和消息体；验证 HL7 消息是否满足用来交互的商业规则和约束；核实发送应用端是否需要一个应用层的确认信息(HL7 消息类型 MCI)——如果是那样的话，发送那个消息。

Web 服务适配器的功能主要是用来处理消息的分发和确认信息。因此，主要包括如下内容。

1) 发送端

(1) 读取接收到的 HL7 消息的 Transmission Wrapper, 以便决定如何到达 Web 服务基层结构上的发送容器(例如接收应用软件), 从而配置 SOAP。

(2) 基于 HL7 消息类型、应用配置和规则(如安全性)来准备一个 SOAP 消息, 包括作为一个 SOAP 消息体部分的 HL7XML 消息, 这个消息被发送到 Web 服务基层组织。

(3) 把 SOAP 消息传递到 Web 服务代理, 通过网络进行传输。

(4) 无论发送端什么时候请求, 都准备接收并存储来自接收端的相应信息或是应用层的确认消息。

2) 接收端

(1) 从 Web 服务站处接收 SOAP 消息。

(2) 验证接收到的 SOAP 消息满足应用配置和一些约束条件(如安全性)。

(3) 或者将这些接收到的消息在内存中以永久的形式保留。

(4) 有选择性地从 SOAP 消息里打开 HL7 XML 消息, 同时核对接收到的 HL7 消息是否与期望的 HL7 消息类型相符合。

(5) 验证是否任意通信层的确认信息都需要被执行, 在哪种情况下需要返回一个合适的消息发送到源消息发送端。

(6) 传递 HL7 消息给接收应用端。

以服务为中心的企业整合

某航空公司已经在几个主要的核心系统之间构建了用于信息集成的信息 Hub, 其他应用间也有不少点到点的集成。然而还存在如下困难:

(1) 因为大部分核心应用构建在主机之上, 所以 Information Hub 是基于主机技术开发, 很难被开放系统使用。

(2) Information Hub 对 Event 支持不强, 被集成的系统间的事件以点到点流转为主, 被集成系统间耦合性强。

(3) 牵扯到多个系统间的业务协作以硬编码为主, 将业务活动自动化的成本高, 周期长, 被开发的业务活动模块重用性差。

为了解决这些企业集成中的问题, 该公司决定以 Ramp Control 系统为例探索一条以服务为中心的企业集成道路。

在航空业中, Ramp Coordination 是指飞机从降落到起飞过程中所需要进行的各种业务活动的协调过程。需要协调的业务活动有: 检查机位环境是否安全, 以及卸货、装货和补充燃料是否方便和安全等。

三种类型航班: short turn around 航班是降落后不久就起飞的航班、Arrival Only 航班指降落后需要隔夜才起飞的, Departure Only 航班是指每天一早第一班飞机。

每种细分的航班类型的 Ramp Coordination 的流程都是略有不同。如此多的流程之间共享着一个业务活动的集合, 如此多种类型的流程都是这些业务活动的不同组装方式。以服务为中心的企业集成中流程服务就是通过这些流程间共享的业务活动抽象为可重用的服务, 并通过流程服务提供的流程编排的能力将它们组成各种大同小异的流程类型, 来降低流程集成成本, 加快流程集成开发效率的。

以服务为中心的企业集成，通过服务建模过程发现这些可重用的服务，并通过流程模型将这些服务组装在一起。

Ramp Coordination 相关的服务模型和 Ramp Coordination 流程相关的有两个业务组件：

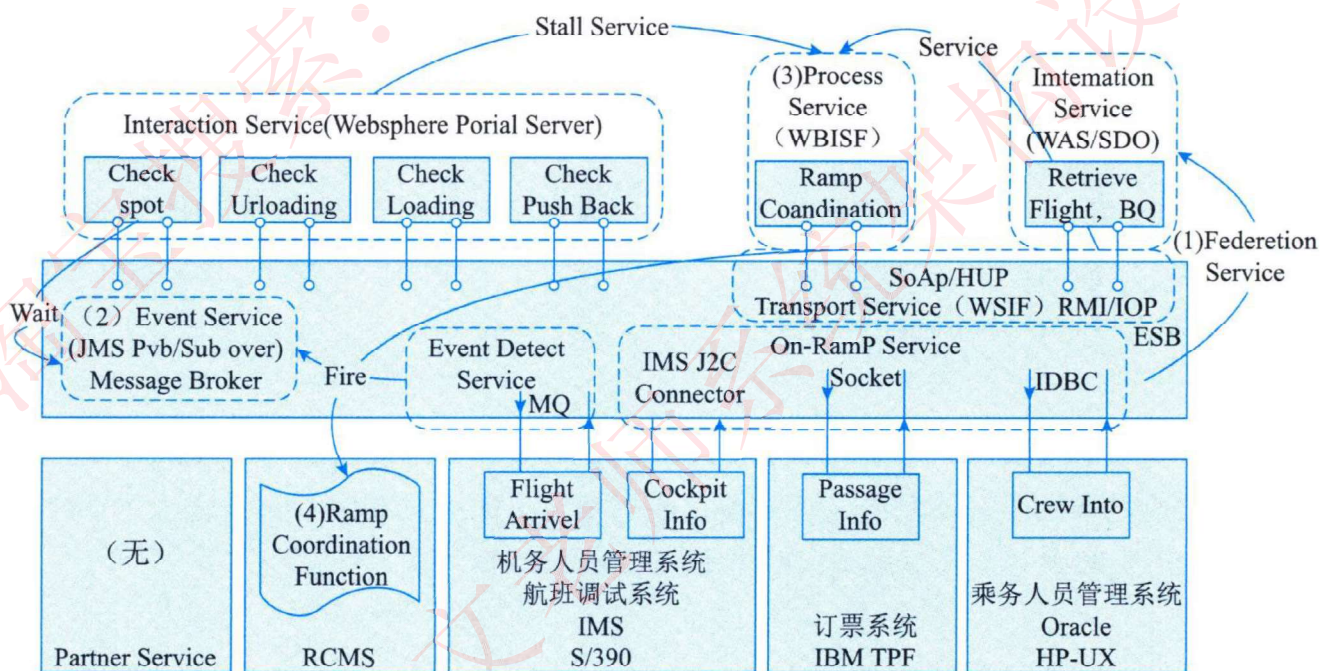
- ①Ramp Control 负责 Ramp Control 相关各种业务活动的组件；
- ②Flight Management 负责航班相关信息的管理，包括航班日程，乘客信息等。

这两个业务组件分别输出如下服务。

- (1)Retrieve Flight B0: 由Flight Management 输出，主要用于提取和航班相关的数据信息。
- (2)Ramp Coordination: 由Ramp Control 输出，主要用于 Ramp Coordination 流程的编排。
- (3)Check Spot: 由Ramp Control 输出，用于检测机位安全信息。
- (4)Check Unloading: 由Ramp Control 输出，用于检查卸货状况。
- (5)Check Loading: 由Ramp Control 输出，用于检查装货状况。
- (6)Check Push Back: 由Ramp Control 输出，用于检查关门动作。

目前，Ramp Coordination 流程需要 4 种类型的外围应用交互。

- (1)从乘务人员管理系统提取航班乘务员的信息。
- (2)从订票系统中提取乘客信息。
- (3)从机务人员管理系统中提取机务人员信息。
- (4)接收来自航班调度系统的航班到达事件。



主要架构元素如下：

(1)信息服务。Federation Service 是 Ramp Coordination 流程中需要从已有系统中提取 4 类信息，在 Service 建模阶段这 4 类信息被聚合为 Flight B0(Business Object)，集成了 Crew Info、Cockpit Info 和 Passage Info 等信息。

(2) 企业服务总线中的事件服务。Event Service 是在检查机务环境安全(Check Spot) 前, Ramp Coordinator 需要被通知航班已经到达。这个业务事件由航班调度系统激发, Flight Arrival 是典型事件发现服务(Event Detect Service), 它通过 MQ 将事件传递给 Message Broker, 通过 JMS 的 Pub/Sub, 这个事件被分发给 Check Spot。

(3) 流程服务。

(4) 企业服务总线中的传输服务。RCMS 是即将新建系统, 用于提供包括 Ramp Coordination 在内的 Ramp Control 的功能。

2 层次式架构设计

2.1 层次式架构概述

软件体系结构贯穿于软件研发的整个生命周期内, 具有重要的影响, 表现为三个方面: 利益相关人员之间的交流、系统设计的前期决策、可传递的系统级抽象。

层次式体系结构设计是将系统组成一个层次结构, 每一层为上层服务, 并作为下层客户。在一些层次系统中, 除了一些精心挑选的输出函数外, 内部的层接口只对相邻的层可见。连接件通过决定层间如何交互的协议来定义, 拓扑约束包括对相邻层间交互的约束。

软件层次式体系结构是最通用的架构, 也被叫作 N 层架构模式。大部分的应用会分成表现层(或称为展示层)、中间层(或称为业务层)、数据访问层(或称为持久层)和数据层。

2.2 表现层框架设计

MVC、MVP、MVVM 架构, 在之前已经讲解过, 这里不再赘述。

使用 XML 设计表现层。

UIP 提供了一个扩展的框架, 用于简化用户界面与商业逻辑代码的分离的方法, 可以用它来写复杂的用户界面导航和工作流处理, 并且它能够复用在不同的场景、并可以随着应用的增加而进行扩展。

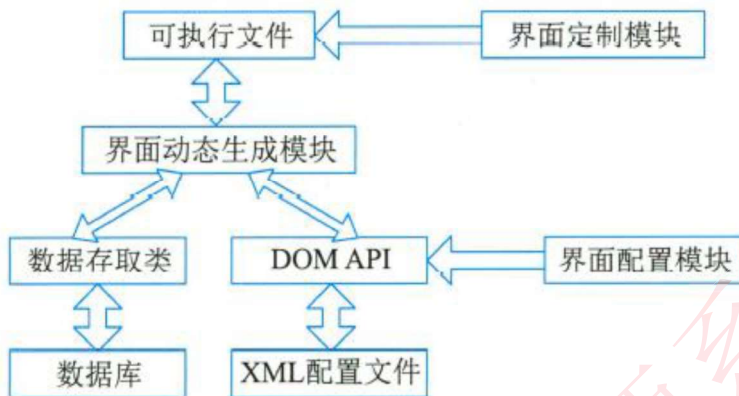
使用 UIP 框架的应用程序把表现层分为了以下几层。

- User Interface Components: 这个组件就是原来的表现层, 用户看到的和进行交互都是这个组件, 它负责获取用户的数据并且返回结果。

- User Interface Process Components: 这个组件用于协调用户界面的各部分, 使其配合后台的活动, 例如导航和工作流控制, 以及状态和视图的管理。用户看不到这一组件, 但是这些组件为 User Interface Components 提供了重要的支持功能。

UIP 的组件主要负责的功能是: 管理经过 User Interface Components 的信息流; 管理 UIP 中各个事件之间的事务; 修改用户过程的流程以响应异常; 将概念上的用户交互流程从实现或者涉及的设备上分离出来; 保持内部的事务关联状态, 通常是持有一个或者多个的与用户交互的事务实体。因此, 这些组件也能从 UI 组件收集数据, 执行服务器的成组的升级或是跟踪 UIP 中的任务过程的管理。

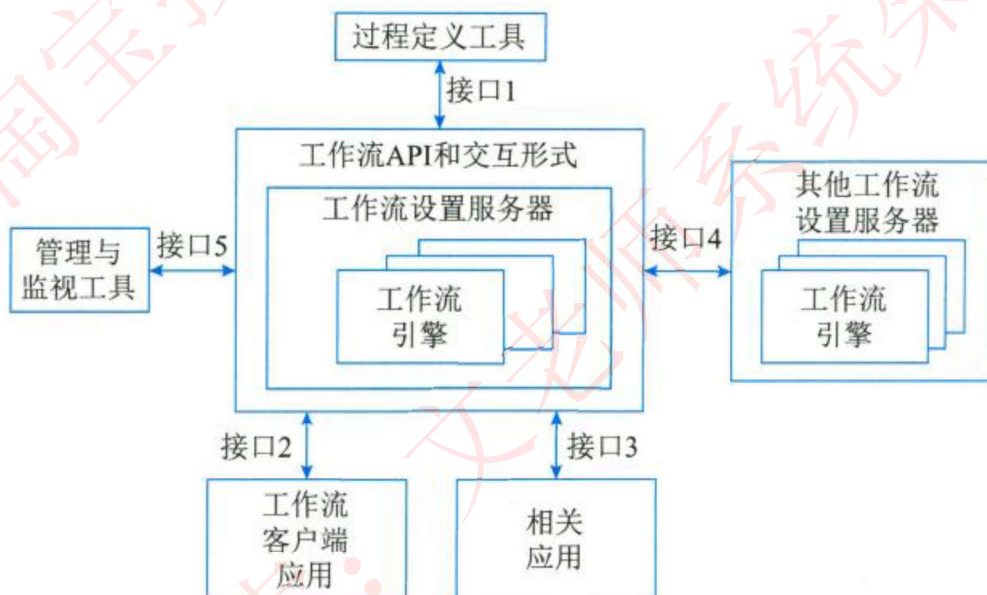
表现层动态生成设计：基于 XML 的界面管理技术可实现灵活的界面配置、界面动态生成和界面定制。其思路是用 XML 生成配置文件及界面所需的元数据，按不同需求生成界面元素及软件界面。基于 XML 界面管理技术，包括界面配置（静态）、界面动态生成和界面定制（动态）三部分。



2.3 中间层架构设计

组件设计：业务逻辑组件分为接口和实现类两个部分。接口用于定义业务逻辑组件，定义业务逻辑组件必须实现的方法是整个系统运行的核心。增加业务逻辑组件的接口，是为了提供更好的解耦，控制器无须与具体的业务逻辑组件耦合，而是面向接口编程。

workflow设计： workflow定义为：业务流程的全部或部分自动化，在此过程中，文档、信息或任务按照一定的过程规则流转，实现组织成员间的协调工作以达到业务的整体目标。 workflow参考模型见图：其包含 6 个基本模块，分别是 workflow 执行服务、 workflow 引擎、流程定义工具、客户端应用、 调用应用和管理监控工具。



(1) interface 1: 过程定义导入/导出接口。这个接口的特点是：转换格式和 A P I 调用，从而支持过程定义信息间的互相转换。

(2) interface 2: 客户端应用程序接口。通过这个接口工作流机可以与任务表处理器交互, 代表用户资源来组织任务。然后由任务表处理器负责, 从任务表中选择、推进任务项。由任务表处理器或者终端用户来控制应用工具的活动。

(3) interface 3: 应用程序调用接口。允许工作流机直接激活一个应用工具, 来执行一个活动。典型的是调用以后台服务为主的应用程序, 没有用户接口。当执行活动要用到的工具, 需要与终端用户交互, 通常是使用客户端应用程序接口来调用那个工具, 这样可以为用户安排任务时间表提供更多的灵活性。

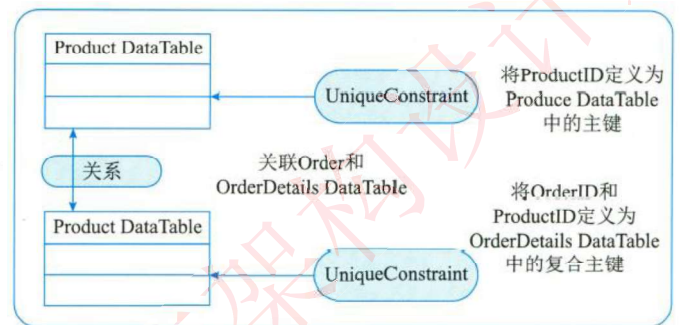
(4) interface 4: 工作流机协作接口。其目标是定义相关标准, 以使不同开发商的工作流系统产品相互间能够进行无缝的任务项传递。

(5) interface 5: 管理和监视接口。提供的功能包括用户管理、角色管理、审查管理、资源控制、过程管理和过程状态处理器等。

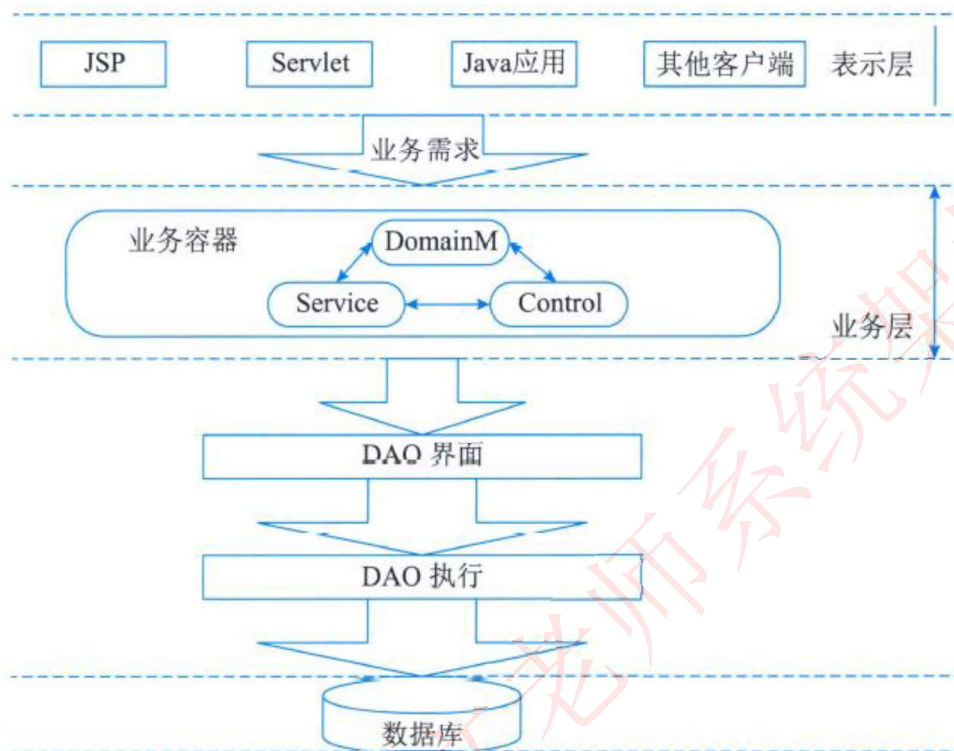
实体设计: 业务逻辑层实体提供对业务数据及相关功能(在某些设计中)的状态编程访问。业务逻辑层实体可以使用具有复杂架构的数据来构建, 这种数据通常来自数据库中的多个相关表。

在应用程序中表示业务逻辑层实体的方法有很多(从以数据为中心的模型到更加面向对象的表示法), 如 XML、通用 DataSet、有类型的 DataSet 等。

```
<?xml version="1.0"?>
<Product xmlns="urn:aUniqueNamespace">
  <ProductID>1</ProductID>
  <ProductName>Chai</ProductName>
  <QuantityPerUnit>10 boxes x 20 bags</QuantityPerUnit>
  <UnitPrice>18.00</UnitPrice>
  <UnitsInStock>39</UnitsInStock>
  <UnitsOnOrder>0</UnitsOnOrder>
  <ReorderLevel>10</ReorderLevel>
</Product>
```



业务框架位于系统架构的中间层, 是实现系统功能的核心组件。采用容器的形式, 便于系统功能的开发、代码重用和管理。下图便是在吸收了 SOA 思想之后的一个三层体系结构的简图。



业务层采用业务容器的方式存在于整个系统当中，采用此方式可以大大降低业务层和相邻各层的耦合，表示层代码只需要将业务参数传递给业务容器，而不需要业务层多余的干预。如此一来，可以有效地防止业务层代码渗透到表示层。

在业务容器中，业务逻辑是按照 Domain Model—Service—Control 思想来实现的。

(1)Domain Model 是领域层业务对象，它仅仅包含业务相关的属性。

(2)Service 是业务过程实现的组成部分，是应用程序的不同功能单元，通过在这些服务之间定义良好的接口和契约联系起来。接

(3)Control 服务控制器，是服务之间的纽带，不同服务之间的切换就是通过它来实现的。

2.4 数据访问层设计

5 种数据访问模式

1. 在线访问：会占用一个数据库连接，读取数据，每个数据库操作都会通过这个连接不断地与后台的数据源进行交互。

2. DataAccess Object：是标准 J2EE 设计模式之一，开发人员常常用这种模式将底层数据访问操作与高层业务逻辑分离开。

3. Data Transfer Object：是经典 EJB 设计模式之一。DTO 本身是这样一组对象或是数据的容器，它需要跨不同的进程或是网络的边界来传输数据。这类对象本身应该不包含具体的业务逻辑，并且通常这些对象内部只能进行一些诸如内部一致性检查和基本验证之类的方法，而且这些方法最好不要再调用其他的对象行为。

4. 离线数据模式是以数据为中心, 数据从数据源获取之后, 将按照某种预定义的结构(这种结构可以是 SDO 中的 Data 图表结构, 也同样可以是 ADO.NET 中的关系结构)存放在系统中, 成为应用的中心。离线, 对数据的各种操作独立于各种与后台数据源之间的连接或是事务。

5. 对象/关系映射(Object/Relation Mapping, O/R Mapping): 大多数应用中的数据都是依据关系模型存储在关系型数据库中; 而很多应用程序中的数据在开发或是运行时则是以对象的形式组织起来的。那么, 对象/关系映射就提供了这样一种工具或是平台, 能够帮助将应用程序中的数据转换成关系型数据库中的记录; 或是将关系数据库中的记录转换成应用程序中代码便于操作的对象。

工厂模式在数据库访问层的应用

首先定义一个操纵数据库的接口 `DataAccess`, 然后根据数据库的不同, 由类工厂决定实例化哪个类。

因为 `DataAccess` 的具体实现类有一些共同的方法, 所以先从 `DataAccess` 实现一个抽象的 `AbstractDataAccess` 类, 包含一些公用方法。然后, 分别为 SQL Server、Oracle 和 OleDb 数据库编写三个数据访问的具体实现类。

现在已经完成了所要的功能, 下面需要创建一个 `Factory` 类, 来实现自动数据库切换的管理。这个类很简单, 主要的功能就是根据数据库类型, 返回适当的数据库操纵类。

事务处理设计

JavaBean 中使用 JDBC 方式进行事务处理: 在 JDBC 中, 打开一个连接对象 `Connection` 时, 默认是 `auto-commit` 模式, 每个 SQL 语句都被当作一个事务, 即每次执行一个语句, 都会自动地得到事务确认。为了能将多个 SQL 语句组合成一个事务, 要将 `auto-commit` 模式屏蔽掉。在 `auto-commit` 模式屏蔽掉之后, 如果不调用 `commit()` 方法, SQL 语句不会得到事务确认。在最近一次 `commit()` 方法调用之后的所有 SQL 会在方法 `commit()` 调用时得到确认。

连接对象管理设计: 通过资源池解决资源频繁分配、释放所造成的问题。

建立连接池的第一步, 就是要建立一个静态的连接池。所谓静态, 是指池中的连接是在系统初始化时就分配好的, 并且不能够随意关闭。Java 中给我们提供了很多容器类, 可以方便地用来构建连接池, 如 `Vector`、`Stack` 等。在系统初始化时, 根据配置创建连接并放置在连接池中, 以后所使用的连接都是从该连接池中获取的, 这样就可以避免连接随意建立、关闭造成的开销。

有了这个连接池, 下面就可以提供一套自定义的分配、释放策略。当客户请求数据库连接时, 首先看连接池中是否有未分配出去的连接。如果存在空闲连接则把连接分配给客户, 并标记该连接为已分配。若连接池中沒有空闲连接, 就在已经分配出去的连接中, 寻找一个合适的连接给客户, 此时该连接在多个客户间复用。

当客户释放数据库连接时, 可以根据该连接是否被复用, 进行不同的处理。如果连接没有使用者, 就放入到连接池中, 而不是被关闭。

2.5 数据架构规划与设计

数据库设计与 XML 设计融合

XML 文档分为两类：一类是以数据为中心的文档，这种文档在结构上是规则的，在内容上是同构的，具有较少的混合内容和嵌套层次，人们只关心文档中的数据而并不关心数据元素的存放顺序，这种文档简称为数据文档，它常用来存储和传输 Web 数据。另一类是以文档为中心的文档，这种文档的结构不规则，内容比较零散，具有较多的混合内容，并且元素之间的顺序是有关的，这种文档常用来在网页上发布描述性信息、产品性能介绍和 E-mail 信息等。

经提出的 XML 文档的存储方式有两种：基于文件的存储方式和数据库存储方式。

(1) 基于文件的存储方式。基于文件的存储方式是指将 XML 文档按其原始文本形式存储，主要存储技术包括操作系统文件库、通用文档管理系统和传统数据库的列。这种存储方式需维护某种类型的附加索引，以建立文件之间的层次结构。基于文件的存储方式的特点：无法获取 XML 文档中的结构化数据；通过附加索引可以定位具有某些关键字的 XML 文档，一旦关键字不确定，将很难定位；查询时，只能以原始文档的形式返回，即不能获取文档内部信息；文件管理存在容量大、管理难的缺点。

(2) 数据库存储方式。数据库在数据管理方面具有管理方便、存储占用空间小、检索速度快、修改效率高和安全性好等优点。一种比较自然的想法是采用数据库对 XML 文档进行存取和操作，这样可以利用相对成熟的数据库技术处理 XML 文档内部的数据。数据库存储方式的特点：能够管理结构化和半结构化数据；具有管理和控制整个文档集合本身的能力；可以对文档内部的数据进行操作；具有数据库技术的特性，如多用户、并发控制和一致性约束等；管理方便，易于操作。

2.6 物联网层次架构设计

物联网可以分为三个层次，底层是用来感知数据的感知层，即利用传感器、二维码、RFID 等设备随时随地获取物体的信息。第二层是数据传输处理的网络层，即通过各种传感网络与互联网的融合，将对象当前的信息实时准确地传递出去。第三层则是与行业需求结合的应用层，即通过智能计算、云计算等将对象进行智能化控制。

感知层用于识别物体、采集信息。感知层包括二维码标签和识读者、RFID 标签和读写器、摄像头、GPS、传感器、M 2 M 终端、传感器网关等，主要功能是识别对象、采集信息，与人体结构中皮肤和五官的作用类似。感知层解决的是人类世界和物理世界的获取数据问题。

网络层用于传递信息和处理信息。网络层包括通信网与互联网的融合网络、网络管理中心、信息中心和智能处理中心等。网络层将感知层获取的信息进行传递和处理，类似于人体结构中的神经中枢和大脑。网络层解决的是传输和预处理感知层所获得数据的问题。

应用层实现广泛智能化。应用层是物联网与行业专业技术的深度融合，结合行业需求实现行业智能化，这类似于人们的社会分工。

物联网应用层利用经过分析处理的感知数据，为用户提供丰富的特定服务。应用层解决的是信息处理和人机交互的问题。

2.7 层次式架构案例分析

电子商务网站(网上商店 PetShop)

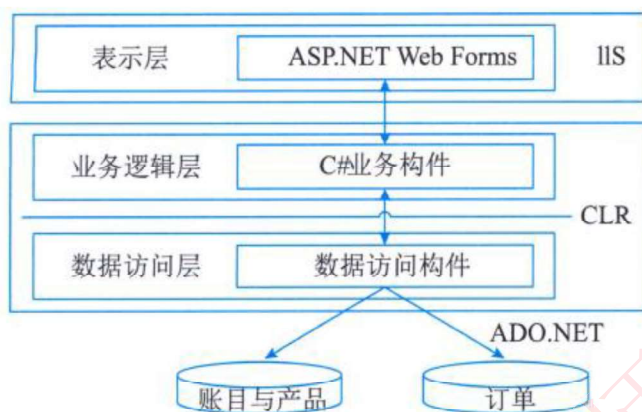


图 13-15 Net 中标准的 BS 分层式结构

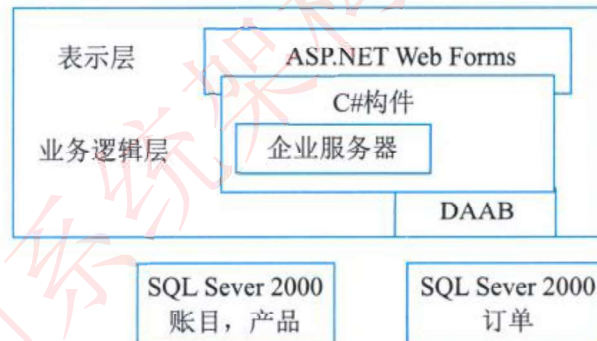


图 13-16 PetShop 2.0 的体系架构

从图 13-16 中可以看到, 并没有明显的数据库访问层设计。这样的设计虽然提高了数据库访问的性能, 但也同时导致了业务逻辑层与数据库访问的职责混乱。

PetShop 3.0 纠正了此前层次不明的问题, 将数据库访问逻辑作为单独的一层独立出来。

PetShop 4.0 基本上延续了 3.0 的结构, 但在性能上作了一定的改进, 引入了缓存和异步处理机制, 同时又充分利用了 ASP.Net 2.0 的新功能 MemberShip。

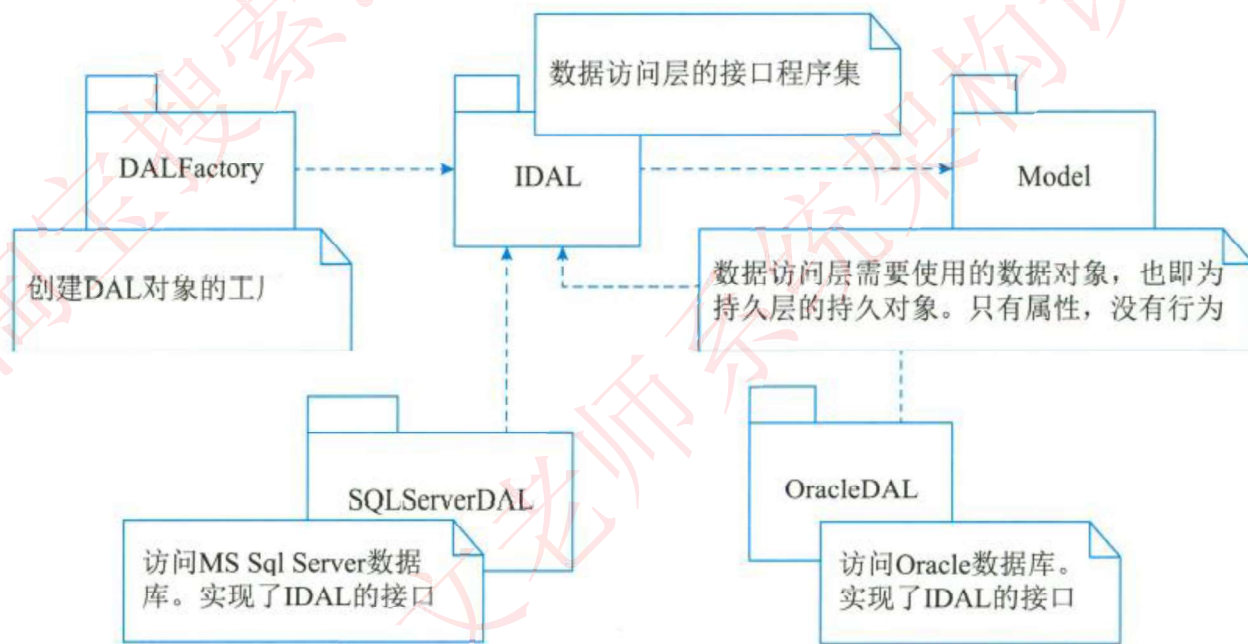


图 13-19 数据访问层的模块结构图

可以看到, 在数据访问层中, 完全采用了“面向接口编程”思想。抽象出来的 IDAL 模块, 脱离了与具体数据库的依赖, 从而使得整个数据访问层有利于数据库迁移。DALFactory 模块专门管理 DAL 对象的创建, 便于业务逻辑层访问。SQLServerDAL 和 OracleDAL 模块均实现 IDAL 模块的接口, 其中包含的逻辑就是对数据库的 Select、Insert、Update 和 Delete 操作。因为数据库类型的不同, 对数据库的操作也有所不同, 代码也会因此有所区别。

此外，抽象出来的 I D A L 模块，除了解除了向下的依赖之外，对于其上的业务逻辑层同样仅存在弱依赖关系，如图 13-20 所示。

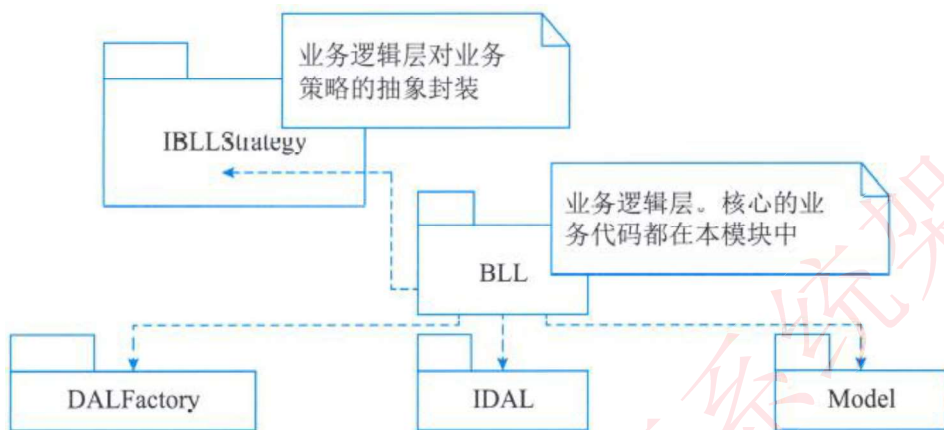


图 13-20 业务逻辑层的模块结构图

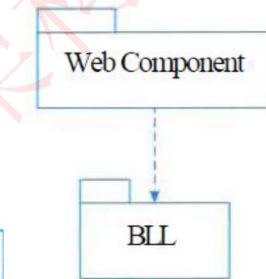
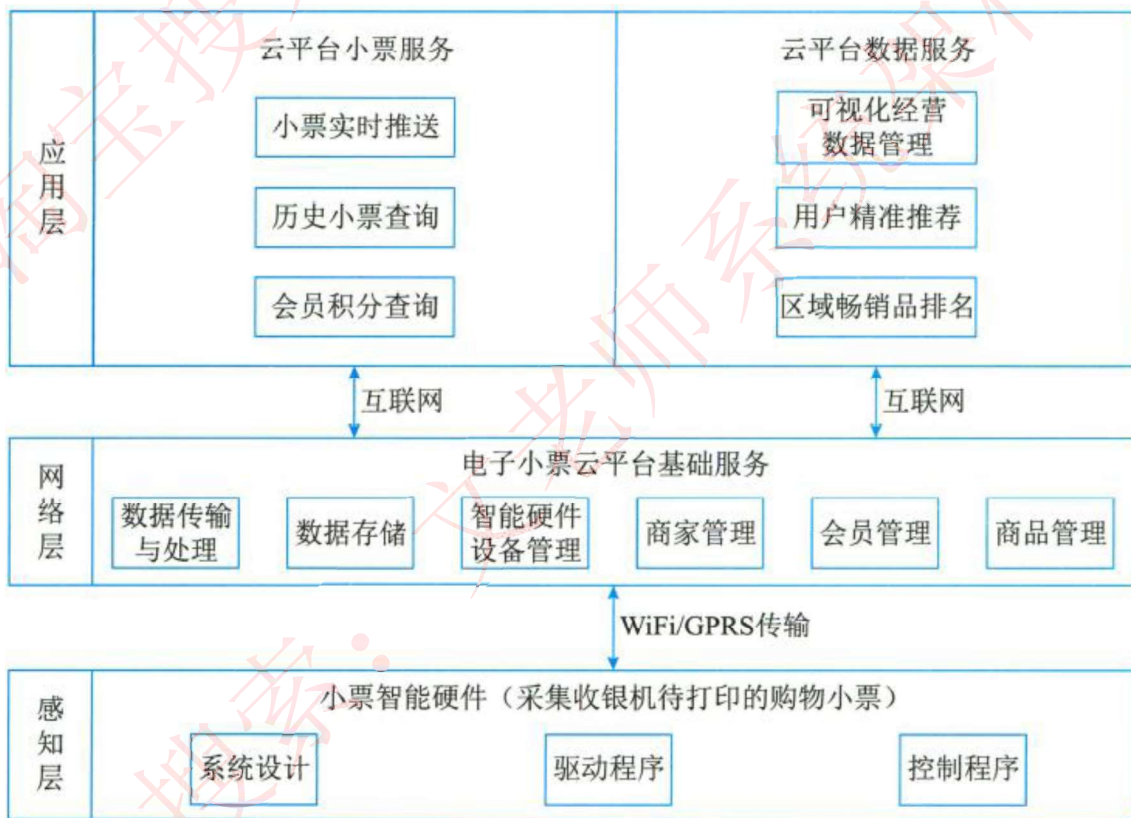


图13-21 表示层的模块结构

图 13-20 中，B L L 是业务逻辑层的核心模块，它包含了整个系统的核心业务。在业务逻辑层中，不能直接访问数据库，而必须通过数据访问层。注意，图 13-20 中对数据访问业务的调用，是通过接口模块 I D A L 来完成的。既然与具体的数据访问逻辑无关，则层与层之间的关系就是松散耦合的。如果此时需要修改数据访问层的具体实现，只要不涉及 I D A L 的接口定义，那么业务逻辑层就不会受到任何影响。毕竟，具体实现的 SQLServerDAL 和 OracalDAL 根本就与业务逻辑层没有半点关系。

基于物联网架构的电子小票服务系统

采用感知层、网络层和应用层的 3 层物联网体系架构模型：



3 云原生架构设计

3.1 云原生架构内涵

云原生架构是基于云原生技术的一组架构原则和设计模式的集合，旨在将云应用中的非业务代码部分进行最大化的剥离，从而让云设施接管应用中原有的大量非功能特性(如弹性、韧性、安全、可观测性、灰度等)，使业务不再有非功能性业务中断困扰的同时，具备轻量、敏捷、高度自动化的特点。

云原生的代码通常包括三部分：业务代码、三方软件、处理非功能特性的代码。从业务代码中剥离大量非功能性特性(不会是所有，比如易用性还不能剥离)到 IaaS 和 PaaS 中，从而减少业务代码开发人员的关注范围，通过云厂商的专业性提升应用的非功能性能力。

具备云原生架构的应用可以最大程度利用云服务和提升软件交付能力，进一步加快软件开发。其特点包括：代码结构发生巨大变化、非功能性特性大量委托、高度自动化的软件交付。

云原生架构原则

服务化原则：拆分为微服务架构、小服务架构，分别迭代。

弹性原则：系统的部署规模可以随着业务量的变化而自动伸缩，无须根据事先的容量规划准备固定的硬件和软件资源。

可观测原则：通过日志、链路跟踪和度量等手段，使得一次点击背后的多次服务调用的耗时、返回值和参数都清晰可见。

韧性原则：当软件所依赖的软硬件组件出现各种异常时，软件表现出来的抵御能力。

所有过程自动化原则：一方面标准化企业内部的软件交付过程，另一方面在标准化的基础上进行自动化，通过配置数据自描述和面向终态的交付过程，让自动化工具理解交付目标和环境差异，实现整个软件交付和运维的自动化。

零信任原则：默认情况下不应该信任网络内部和外部的任何人/设备/系统，需要基于认证和授权重构访问控制的信任基础，以身份为中心。

架构持续演进原则：云原生架构本身也必须是一个具备持续演进能力的架构。

主要架构模式

1. 服务化架构模式：典型模式是微服务和小服务模式。通过服务化架构，把代码模块关系和部署关系进行分离，每个接口可以部署不同数量的实例，单独扩缩容，从而使得整体的部署更经济。

2. Mesh 化架构模式：把中间件框架(如 RPC、缓存、异步消息等)从业务进程中分离，让中间件 SDK 与业务代码进一步解耦，从而使得中间件升级对业务进程没有影响，甚至迁移到另外一个平台的中间件也对业务透明。分离后在业务进程中只保留很“薄”的 Client 部分，Client 通常很少变化，只负责与 Mesh 进程通信，原来需要在 SDK 中处理的流量控制、安全等逻辑由 Mesh 进程完成。

3. Serverless 模式：将“部署”这个动作从运维中“收走”，使开发者不用关心应用运行地点、操作系统、网络配置、CPU 性能等，从架构抽象上看，当业务流量到来/业务事件发生时，云会启动或调度一个已启动的业务进程进行处理，处理完成后云会自动会关闭/调度业务进程，等待下一次触发，也就是把应用的整个运行都委托给云。

4. 存储计算分离模式：。在云环境中，推荐把各类暂态数据(如 session)、结构化和非结构化持久数据都采用云服务来保存，从而实现存储计算分离。

5. 分布式事务模式：大颗粒度的业务需要访问多个微服务，必然带来分布式事务问题，否则数据就会出现不一致。架构师需要根据不同的场景选择合适的分布式事务模式。

6. 可观测架构：可观测架构包括 Logging、Tracing、Metrics 三个方面，其中 Logging 提供多个级别的详细信息跟踪，由应用开发者主动提供；Tracing 提供一个请求从前端到后端的完整调用链路跟踪，对于分布式场景尤其有用；Metrics 则提供对系统量化的多维度度量。

7. 事件驱动架构：本质上是一种应用/组件间的集成架构模式。可用于服务解耦、增强服务韧性、数据变化通知等场景中。

典型的云原生架构反模式：庞大的单体应用、单体应用“硬拆”为微服务、缺乏自动化能力的微服务。

3.2 云原生架构相关技术

容器技术

容器作为标准化软件单元，它将应用及其所有依赖项打包，使应用不再受环境限制，在不同计算环境间快速、可靠地运行。

通过容器技术，企业可以充分发挥云计算弹性优势，降低运维成本。一般而言，借助容器技术，企业可以通过部署密度提升和弹性降低 50% 的计算成本。

Kubernetes 已经成为容器编排的事实标准，被广泛用于自动部署，扩展和管理容器化应用。Kubernetes 提供了分布式应用管理的核心能力，包括：资源调度、应用部署与管理、自动修复、服务发现与负载均衡、弹性伸缩、声明式 API、可扩展性架构、可移植性。

云原生微服务

微服务模式将后端单体应用拆分为松耦合的多个子应用，每个子应用负责一组子功能。这些子应用称为“微服务”，多个“微服务”共同形成了一个物理独立但逻辑完整的分布式微服务体系。这些微服务相对独立，通过解耦研发、测试与部署流程，提高整体迭代效率。

微服务设计约束：

1) 微服务个体约束：功能在业务域划分上应是相互独立的，低耦合、单一职责。

2) 微服务与微服务之间的横向关系：主要从微服务的可发现性和可交互性处理服务间的横向关系，一般需要服务注册中心。

3) 微服务与数据层之间的纵向约束：在微服务领域，提供数据存储隔离原则，即数据是微服务的私有资产，对于该数据的访问都必须通过当前微服务提供的 API 来访问。

4) 全局视角下的微服务分布式约束：故障发现时效性和根因精确性始终是开发运维人员的核心诉求。

主要微服务技术：

Apache Dubbo 作为源自阿里巴巴的一款开源高性能 R P C 框架，特性包括基于透明接口的 RPC、智能负载均衡、自动服务注册和发现、可扩展性高、运行时流量路由与可视化的服务治理。

Spring Cloud 作为开发者的主要微服务选择之一，为开发者提供了分布式系统需要的配置管理、服务发现、断路器、智能路由、微代理、控制总线、一次性 Token、全局锁、决策竞选、分布式会话与集群状态管理等能力和开发工具。

Eclipse MicroProfile 作为 Java 微服务开发的基础编程模型，它致力于定义企业 Java 微服务规范，MicroProfile 提供指标、API 文档、运行状况检查、容错与分布式跟踪等能力，使用它创建的云原生微服务可以自由地部署在任何地方，包括服务网格架构。

Tars 是腾讯将其内部使用的微服务框架，包含一整套开发框架与管理平台，兼顾多语言、易用性、高性能与服务治理，理念是让开发更聚焦业务逻辑，让运维更高效。

SOFAStack 是由蚂蚁金服开源的一套用于快速构建金融级分布式架构的中间件，也是在金融场景里的最佳实践。

DAPR(分布式应用运行时)是微软新推出的一种可移植的、无服务器的、事件驱动的运行时代，它使开发人员可以轻松构建弹性，无状态和有状态微服务，这些服务运行在云和边缘上，并包含多种语言和开发框架。

无服务器技术(Serverless) 因为屏蔽了服务器的各种运维复杂度，让开发人员可以将更多精力用于业务逻辑设计与实现，而逐渐成为云原生主流技术之一。Serverless 计算包含以下特征：

(1)全托管的计算服务，客户只需要编写代码构建应用，无需关注同质化的、负担繁重的基于服务器等基础设施的开发、运维、安全、高可用等工作；

(2)通用性，结合云 BaaS API 的能力，能够支撑云上所有重要类型的应用；

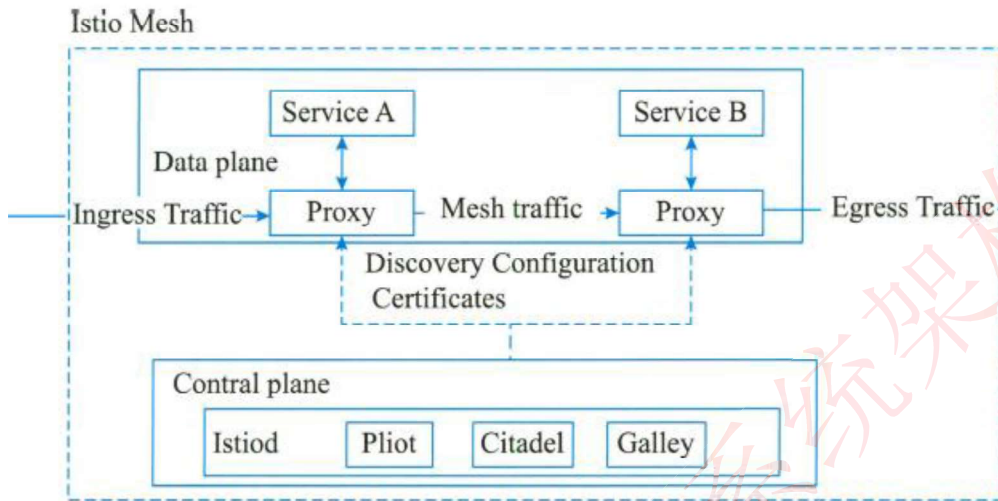
(3)自动弹性伸缩，让用户无需为资源使用提前进行容量规划；

(4)按量计费，让企业使用成本得到有效降低，无需为闲置资源付费。

函数计算(FaaS)是 Serverless 中最具代表性的产品形态。通过把应用逻辑拆分多个函数，每个函数都通过事件驱动的方式触发执行。

无服务器技术关注点：计算资源弹性调度、负载均衡和流控、安全性。

服务网格(ServiceMesh) 是分布式应用在微服务软件架构之上发展起来的新技术，旨在将那些微服务间的连接、安全、流量控制和可观测等通用功能下沉为平台基础设施，实现应用与平台基础设施的解耦。这个解耦意味着开发者无需关注微服务相关治理问题而聚焦于业务逻辑本身，提升应用开发效率并加速业务探索和创新。



在这张架构图中，服务 A 调用服务 B 的所有请求，都被其下的服务代理截获，代理服务 A 完成到服务 B 的服务发现、熔断、限流等策略，而这些策略的总控是在控制平面 (ControlPlane) 上配置。

3.3 云原生架构案例分析

某旅行公司云原生改造

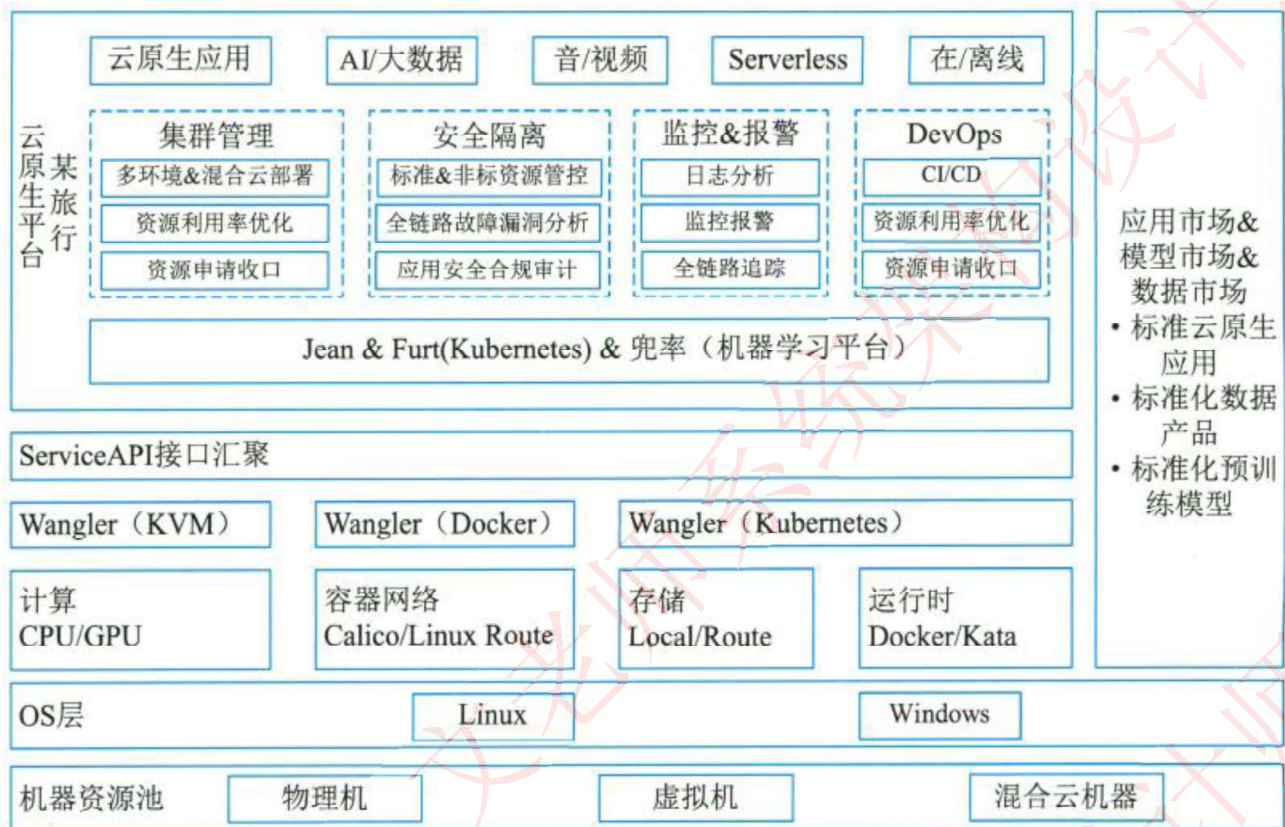
面临两个问题：公司主体由两个公司合并，技术体系不同，需要整合为一体；节假日高并发流量。

改造第一阶段，某旅行技术团队为了提升集群资源利用率，降低资源使用成本。利用云原生思维重构部分技术体系，将多套旧有系统合并、收拢到一套以云原生应用为核心的私有云平台上，同时将 IDC、物理网络、虚拟网络、计算资源、存储资源等通过 IaaS、PaaS 等，实现虚拟化封装、切割、再投产的自动化流程。

随着服务器集群规模的扩大，部分机器开始频繁出现故障。此时，保障服务稳定性成了第二阶段改造的首要任务。

第二阶段基于公有云、私有云和离线专属云集群等新型动态计算环境，某旅行公司的技术团队帮助业务构建和运行具有弹性的云原生应用，促进业务团队开始使用声明式 API，同时通过不可变基础设施、服务网格和容器服务，来构建容错性好、易于管理和观察的应用系统，并结合平台可靠的自动化恢复、弹性计算来完成整个服务稳定性的提升。

第三阶段通过基础组件、服务的云原生改造、服务依赖梳理和定义等方式，使应用不再需要考虑底层资源、机房、运行时间和供应商等因素。此外，还利用标准的云原生应用模型，实现了服务的跨地域、跨云自动化灾备、自动部署，并向云原生场景下的 DevOps 演进。架构图如下：



云原生技术助力某汽车公司数字化转型实践

战略性构建容器云平台。通过平台实现对某云行 App、二手车、在线支付、优惠券等核心互联网应用承载。以多租户的形式提供弹性计算、数据持久化、应用发布等面向敏捷业务服务，并实现高水平资源隔离。标准化交付部署，快速实现业务扩展，满足弹性要求。利用平台健康检查、智能日志分析和监控告警等手段及时洞察风险，保障云平台和业务应用稳定运行。

数字混合云交付。采用私有云+公有云的混合交付模式，按照服务的敏态/稳态特性和管控要求划分部署，灵活调度公有云资源来满足临时突发或短期高 T P S 业务支撑的需求。利用 PaaS 平台标准化的环境和架构能力，实现私有云和公有云一致交付体验。

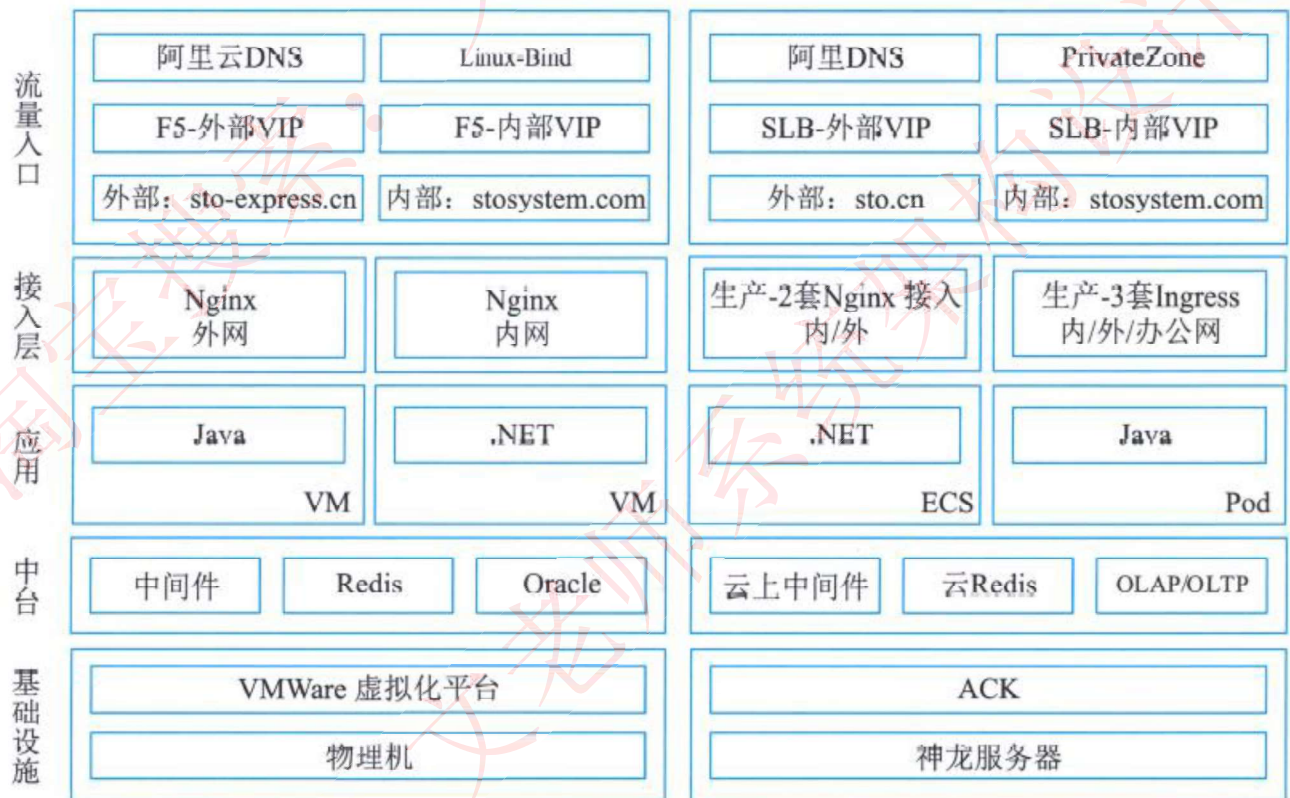
深度融合微服务治理体系，实现架构的革新和能力的沉淀，逐步形成支撑数字化应用的业务中台。通过领域设计、系统设计等关键步骤，对原来庞大的某云体系应用进行微服务拆分，形成能量、社群、用户、车辆、订单等多共享业务服务，同步制定了设计与开发规范、实施路径和配套设施，形成一整套基于微服务的分布式应用架构规划、设计方法论。



某快递公司核心业务系统云原生改造

某快递公司核心业务系统原架构基于 Vmware+Oracle 数据库进行搭建。随着搬迁上阿里云，架构全面转型为基于 Kubernetes 的云原生架构体系。

通过引入云原生数据库、应用容器化、微服务改造技术，得到了如下的上云架构：



基础设施，全部计算资源取自阿里云的神龙裸金属服务器。

流量接入，阿里云提供两套流量接入，一套是面向公网请求，另外一套是服务内部调用。

基于 Kubernetes 打造的云原生 PaaS 平台优势明显突出。

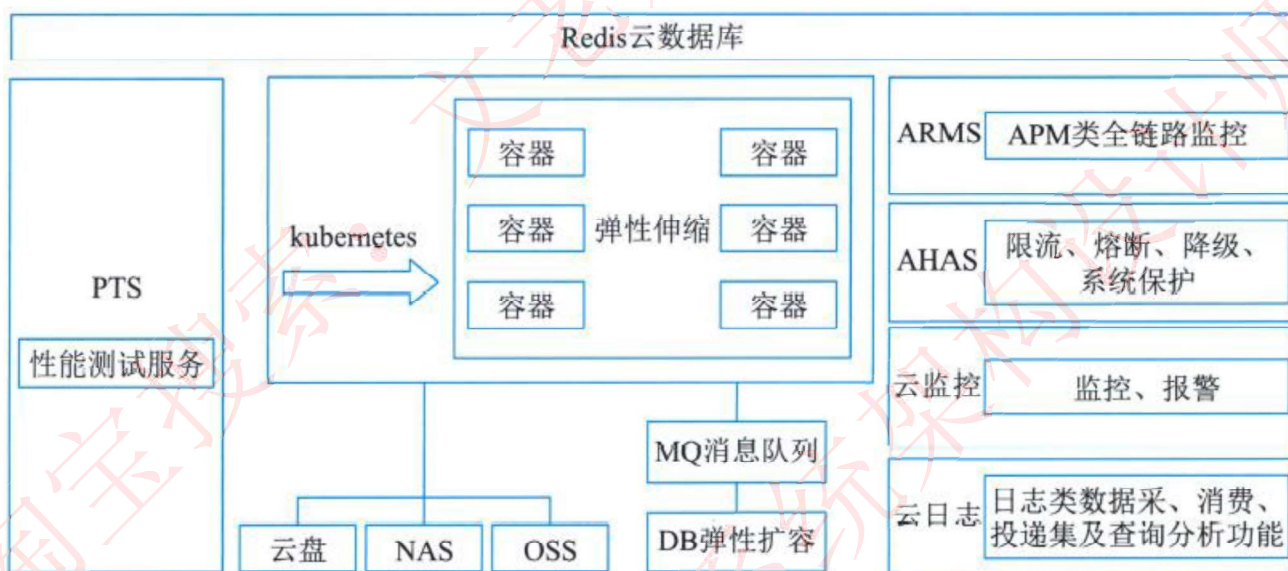
每个应用都在 Kubernetes 上面创建单独的一个 Namespace，应用和应用之间实现资源隔离。

线上 Kubernetes 集群采用阿里云托管版容器服务，免去了运维 Master 结点的工作，只需要制定 Worker 结点上线及下线流程即可。

某电商业务云原生改造

方案的关键点是：

- 通过容器化部署，利用阿里云容器服务的快速弹性应对大促时的资源快速扩容。
- 提前接入链路追踪产品，用于对分布式环境下复杂的服务调用进行跟踪，对异常服务进行定位，帮助客户在测试和生产中快速定位问题并修复，降低对业务的影响。
- 使用阿里云性能测试服务(PTS)进行压测，利用秒级流量拉起、真实地理位置流量等功能，以最真实的互联网流量进行压测，确保业务上线后的稳定运营。
- 采集压测数据，解析系统强弱依赖关系、关键瓶颈点，对关键业务接口、关键第三方调用、数据库慢调用、系统整体负载等进行限流保护。
- 配合阿里云服务团队，在大促前进行 ECS/RDS/安全等产品扩容、链路梳理、缓存/连接池预热、监控大屏制作、后端资源保障演练等，帮助大促平稳进行。



4 面向服务架构设计

4.1 SOA 概述和发展

在面向服务的体系结构(SOA)中，服务的概念有了延伸，泛指系统对外提供的功能集。

从应用的角度定义，可以认为 SOA 是一种应用框架，它着眼于日常的业务应用，并将它们划分为单独的业务功能和流程，即所谓的服务。SOA 使用户可以构建、部署和整合这些服务，且无需依赖应用程序及其运行平台，从而提高业务流程的灵活性。

从软件的基本原理定义，可以认为 SOA 是一个组件模型，它将应用程序的不同功能单元(称为服务)通过这些服务之间定义良好的接口和契约联系起来。接口是采用中立的方式进行定义的，它应该独立于实现服务的硬件平台、操作系统和编程语言。

业务流程是指为了实现某种业务目的行为所进行的流程或一系列动作。

BPEL：面向 Web 服务的业务流程执行语言，是一种使用 Web 服务定义和执行业务流程的语言。使用 BPEL，用户可以通过组合、编排和协调 Web 服务自上而下地实现面向服务的体系结构。BPEL 目前用于整合现有的 Web Services，将现有的 Web Services 按照要求的业务流程整理成为一个新的 Web Services，在这个基础上，形成一个从外界看来和单个 Service 一样的 Service。

SOA 的微服务化发展

SOA 架构向更细粒度、更通用化程度发展，就成了所谓的微服务了。SOA 与微服务的区别在于如下几个方面：

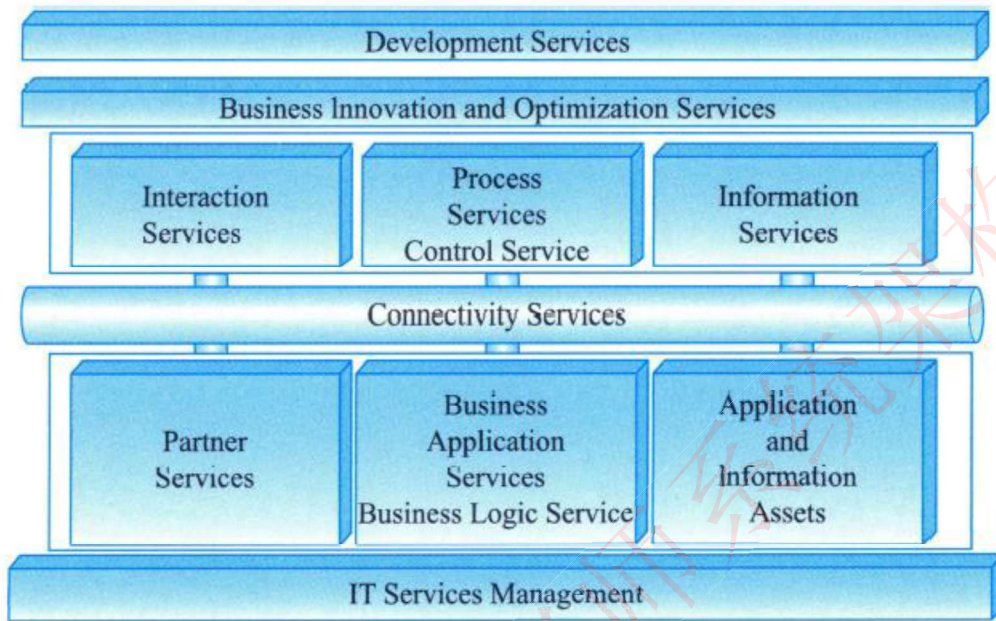
- (1)微服务相比于 SOA 更加精细，微服务更多地以独立的进程的方式存在，互相之间并无影响；
- (2)微服务提供的接口方式更加通用化，例如 HTTP RESTful 方式，各种终端都可以调用，无关语言、平台限制；
- (3)微服务更倾向于分布式去中心化的部署方式，在互联网业务场景下更适合。

SOA 架构是一个面向服务的架构，可将其视为组件模型，其将系统整体拆分为多个独立的功能模块，模块之间通过调用接口进行交互，有效整合了应用系统的各项业务功能，系统各个模块之间是松耦合的。SOA 架构以企业服务总线链接各个子系统，是集中式的技术架构，应用服务间相互依赖导致部署复杂，应用间交互使用远程通信，降低了响应速度。

微服务架构是 SOA 架构的进一步优化，去除了 ESB 企业服务总线，是一个真正意义上去中心化的分布式架构。其降低了微服务之间的耦合程度，不同的微服务采用不同的数据库技术，服务独立，数据源唯一，应用极易扩展和维护，同时降低了系统复杂性。

4.2 SOA 的参考架构

典型的以服务为中心的企业集成架构如下图所示，采用“关注点分离”的方法规划企业集成中的各种架构元素，同时从服务视角规划每种架构元素提供的服务，以及服务如何被组合在一起完成某种类型的集成。可划分为六大类：



(1) 业务逻辑服务：包括用于实现业务逻辑的服务和执行业务逻辑的能力，其中包括业务应用服务、业务伙伴服务以及应用和信息资产。

(2) 控制服务：包括实现人、流程和信息集成的服务，以及执行这些集成逻辑的能力。

(3) 连接服务：通过提供企业服务总线提供分布在各种架构元素中服务间的连接性。

(4) 业务创新和优化服务：用于监控业务系统运行时服务的业务性能，并通过及时了解到的业务性能和变化，采取措施适应变化的市场。

(5) 开发服务：贯彻整个软件开发生命周期的开发平台，从需求分析，到建模、设计、开发、测试和维护等全面的工具支持。

(6) IT 服务管理：支持业务系统运行的各种基础设施管理能力或服务，如安全服务、目录服务、系统管理和资源虚拟化。

1. 连接服务——企业服务总线：ESB 的基本特征和能力包括：描述服务的元数据和服务注册管理；在服务请求者和提供者之间传递数据，以及对这些数据进行转换的能力，并支持由实践中总结出来的一些模式如同步模式、异步模式等；发现、路由、匹配和选择的能力，以支持服务之间的动态交互，解耦服务请求者和提供者。高级一些的能力，包括对安全的支持、服务质量保证、可管理性和负载均衡等。

2. 业务逻辑服务

1) 整合已有应用——应用和信息访问服务：实现对已有应用和信息的集成，主要有两类访问服务：可接入服务、事件发现服务。

2) 整合新开发的应用——业务应用服务：实现新应用集成，主要有三类业务应用服务：组件服务（可重用）、核心服务（运行时）、接口服务。

3) 整合客户和业务伙伴 (B2C/B2B)——伙伴服务：提供与企业外部的 B 2 B 的集成能力，包括：社区服务、文档服务、协议服务。

3. 控制服务

1) 数据整合——信息服务：提供集成数据的能力，目前主要包括如下集中信息服务：联邦服务（不同类型数据聚合）、复制服务（远程数据本地访问）、转换服务（格式转换）、搜索服务。

2) 流程整合——流程服务：完成业务流程集成，包括：编排服务（预定义流程顺序）、事务服务（保证 ACID）、人工服务（人工活动集成到流程中）。

3) 用户访问整合——交互服务：实现用户访问集成，包括：交付服务（运行时交互框架）、体验服务、资源服务（运行时交互组件的管理）。

4. 开发服务：开发环境和工具中为不同开发者的角色提供的功能被称为开发服务。根据开发过程中开发者角色和职责的不同，有如下 4 类服务：建模服务、设计服务、实现服务、测试服务。

5. 业务创新和优化：以业务绩效管理 (BPM) 技术为核心提供业务事件发布、收集和关键业务指标监控能力。包括以下服务：

(1) 公共事件框架服务：通过一个公共事件框架提供 IT 和业务事件的激发、存储和分类等。

(2) 采集服务通过基于策略的过滤和相关性分析检测感兴趣的服务。

(3) 监控服务：通过事件与监控上下文间的映射，计算和管理业务流程的关键性能指标。

6. IT 服务管理：为业务流程和服务提供安全、高效和健康的运行环境，包括：安全和目录服务、系统管理和虚拟化服务。

4.3 SOA 主要协议和规范

Web 服务最基本的协议包括 UDDI、WSDL 和 SOAP，通过它们，可以提供直接而又简单的 Web Service 支持，如图所示。



UDDI（统一描述、发现和集成协议）计划是一个广泛的、开放的行业计划，它使得商业实体能够彼此发现；定义它们怎样在 Internet 上互相作用，并在一个全球的注册体系架构中共享信息。

WSDL (Web 服务描述语言)，是一个用来描述 Web 服务和说明如何与 Web 服务通信的 XML 语言。可描述三个基本属性：服务做什么、如何访问服务、服务位于何处。

SOAP 是在分散或分布式的环境中交换信息的简单的协议，是一个基于 XML 的协议。它包括 4 个部分：SOAP 封装，定义了一个描述消息中的内容是什么，是谁发送的，谁应当接收并处理它以及如何处理它们的框架；SOAP 编码规则，用于表示应用程序需要使用的数据类型的实例；SOAP RPC 表示是远程过程调用和应答的协定；SOAP 绑定是使用底层协议交换信息。

REST（表述性状态转移）的设计不只是一是要适用于互联网环境，而是一个普遍的设计理念，目的是为了不同的软件或者应用程序在任何网络环境下都可以进行信息的互相传递。包括四个要素：

资源：REST 是以资源为中心构建，资源可以是一个订单，也可以是一幅图片。将互联网中一切暴露给客户端的事物都可以看作是一种资源，对资源相关数据和表述进行组合，借助 U R I（统一资源标识符）标识 Web 上的资源。

表述：用表述描述资源在 Web 中某一个时间的状态。

状态转移：分为应用状态（用户请求会话信息快照）和资源状态（时间资源请求表述的快照）。

超链接：在页面中嵌入链接和其他资源建立联系。

4.4 SOA 设计标准和原则

SOA 设计的要求

文档标准化：XML 文档，Web 描述语言。

通信协议标准：用消息进行通信，消息使用 XML Schema 来定义。

应用程序统一登记与集成：通过扮演目录列表角色的登记处来维护，UDDI 是标准。

服务质量（QoS）：每项 SOA 服务都有一个与之相关的服务质量。QoS 的一些关键元素有安全需求（例如认证和授权）、可靠通信以及谁能调用服务的策略。其服务和标准包括：

可靠性：“仅且仅仅传送一次”“最多传送一次”“重复消息过滤”和“保证消息传送”等特性消息的发送和确认。

安全性：主要包括认证交换、消息完整性和消息保密。

策略：服务提供者有时候会要求服务消费者与某种策略通信。

控制：在 SOA 中，进程是使用一组离散的服务创建的。BPEL4WS 或者 WSBPEL 是用来控制这些服务的语言。

管理：让系统管理员管理所有，运行在多种环境下的服务的管理系统。

SOA 的设计原则

(1) 无状态。以避免服务请求者依赖于服务提供者的状态。

(2) 单一实例。避免功能冗余。

(3) 明确定义的接口。使用者依赖服务规约调用服务，所以服务定义必须长时间稳定，一旦公布，不能随意更改；服务的定义应尽可能明确，减少使用者的不适当使用；不要让使用者看到服务内部的私有数据。

(4) 自包含和模块化。服务封装了那些在业务上稳定、重复出现的活动和组件，实现服务的功能实体是完全独立自主的，独立进行部署、版本控制、自我管理和恢复。

(5) 粗粒度。服务数量不应该太大，依靠消息交互而不是远程过程调用(RPC)，通常消息量比较大，但是服务之间的交互频度较低。

(6) 服务之间的松耦合性。服务使用者看到的是服务的接口，其位置、实现技术和当前状态等对使用者是不可见的，服务私有数据对服务使用者是不可见的。

(7) 重用能力。服务应该是可以重用的。

(8) 互操作性、兼容和策略声明。为了确保服务规约的全面和明确，策略成为一个越来越重要的方面。这可以是技术相关的内容，例如一个服务对安全性方面的要求；也可以是跟业务有关的语义方面的内容，例如需要满足的费用或者服务级别方面的要求，这些策略对于服务在交互时是非常重要的。

4.5 SOA 的设计模式

服务注册表模式，支持如下 SOA 治理功能：

(1) 服务注册：应用开发者，也叫服务提供者，向注册表公布他们的功能。他们公布服务合同，包括服务身份、位置、方法、绑定、配置、方案和策略等描述性属性。

(2) 服务位置：也就是服务应用开发者，帮助他们查询注册服务，寻找符合自身要求的服务。注册表让服务的消费者检索服务合同。对谁可以访问注册表，以及什么服务属性通过注册表暴露的控制

(3) 服务绑定：服务的消费者利用检索到的服务合同来开发代码，开发的代码将与注册的服务绑定、调用注册的服务以及它们实现互动。开发者常常利用集成的开发环境自动将新开发的服务与不同的新协议、方案和程序间通信所需的其他接口绑在一起。

企业服务总线模式，由中间件技术实现的支持面向服务架构的基础软件平台，支持异构环境中的服务以基于消息和事件驱动模式的交互，并且具有适当的服务质量和可管理性。

一个典型的在 ESB 环境中组件之间的交互过程是：首先由服务请求者触发一次交互过程，产生一个服务请求消息，并将该消息按照 ESB 的要求标准化，然后标准化的消息被发送给服务总线。ESB 根据请求消息中的服务名或者接口名进行目的组件查找，将消息转发至目的组件，并最终将处理结果逆向返回给服务请求者。这种交互过程不再是点对点的直接交互模式，而是由事件驱动的消息交互模式。

ESB 的核心功能如下。

- (1) 提供位置透明性的消息路由和寻址服务。
- (2) 提供服务注册和命名的管理功能。
- (3) 支持多种消息传递范型(如请求/响应、发布/订阅等)。
- (4) 支持多种可以广泛使用的传输协议。
- (5) 支持多种数据格式及其相互转换。
- (6) 提供日志和监控功能。

微服务模式，不再强调传统 SOA 架构里面比较重的 ESB 企业服务总线，同时 SOA 的思想进入到单个业务系统内部实现真正的组件化。

微服务模式特点：复杂应用解耦、独立、技术选型灵活、容错、松耦合易扩展。

常见的微服务设计模式：

1) 聚合器微服务：聚合器调用多个微服务实现系统应用程序所需功能，具体有两种形式，一种是将检索到的数据信息进行处理并直接展示；另一种是对获取到的数据信息增加业务逻辑处理后，再进一步发布成一个新的微服务作为一个更高层次的组合微服务，相当于从服务消费者转换成服务提供者。

2)链式微服务：客户端或服务在收到请求后，会返回一个经过合并处理的响应，服务之间形成一条调用链。

3)数据共享微服务：当服务之间存在强耦合关系时，可能存在多个微服务共享缓存与数据库存储的现象。

4)异步消息传递微服务：消息队列将消息写入一个消息队列中，实现业务逻辑以异步方式运行，从而加快系统响应速度。对于一些不必要以同步方式运行的业务逻辑，可以使用消息队列代替 REST 实现请求、响应，加快服务调用的响应速度。

微服务架构的问题与挑战：微服务架构分布式特点带来的复杂性；微服务架构的分区数据库体系，不同服务拥有不同数据库；增加了测试的复杂性；。在大规模应用部署中，在监控、管理、分发及扩容等方面，微服务也存在着巨大挑战。

4.6 SOA 构建和实施

SOA 构建注意问题

(1) 原有系统架构中的集成需求：当 SOA 架构师遇到一个十分复杂的企业系统时，首先考虑的应该是如何重用已有的投资而不是替换遗留系统。集成类型包括：应用程序集成的需求，终端用户界面集成的需求，流程集成的需求以及已有系统信息集成的需求。

(2) 服务粒度的控制以及无状态服务的设计：SOA 系统中服务的构建有两点需要特别注意的地方：首先是对于服务粒度的控制，另外就是对于无状态服务的设计。

SOA 系统架构中的具体服务应该都是独立的、自包含的请求，在实现这些服务的时候不需要前一个请求的状态，也就是说服务不应该依赖于其他服务的上下文和状态，即 SOA 架构中的服务应该是无状态的服务。当某一个服务需要依赖时，最好把它定义成具体的业务流程(BPEL)。

SOA 的实施过程

从以下三个方面选择 SOA 最佳的解决方案：尽量选择能进行全局规划的方案、选择时充分考虑企业自身的需求、从平台实施等技术方面进行考察。

业务流程分析过程：

1. 建立服务模型

1)自顶向下分解法：从业务着手进行分析，选择端到端的业务流程进行逐层分解至业务活动，并对其间涉及的业务活动和业务对象进行变化分析。

2)业务目标分析法：通过关键性能指标分析来验证已有服务候选者以及发现遗漏的服务候选者，这也可以称为“目标服务建模”。

3)自底向上分析法：利用已有资产来实现服务。

2. 建立业务流程

1)建立业务对象：业务对象是对数据进行检索和处理的组件，是简单的真实世界的软件抽象。业务对象的分类包括：实体、过程、事件。

2)建立服务接口。

3) 建立业务流程：流程是指定的活动顺序，包含明确确定的用于提供业务值的输入和输出。

5 嵌入式系统架构设计

5.1 嵌入式系统软件架构原理与特征

嵌入式系统的典型架构可概括为两种模式，即层次化模式架构和递归模式架构。

1. 层次化模式架构：位于高层的抽象概念与低层的更加具体的概念之间存在着依赖关系。

层次化模式架构主要设计思想是：

(1) 当一个系统存在高层次的抽象，这些抽象的表现形式是一个个的抽象概念，而这些抽象概念需要具体的低层概念进行实现时，就可采用层次化模式。

(2) 分层模式结构只包含了一个主要的元素(域包)和它的接口，以及用来说明模式结构的约束条件。

(3) 层次化模式可以分为两种：封闭型和开放性。封闭型的特征是：一层中的对象只能调用同一层或下一个底层的对象提供的方法。而开放型一层中的对象可以调用同一层或低于该层的任意一层的对象提供的方法。

2. 递归模式架构。递归模式解决的问题是：需要将一个非常复杂的系统进行分解，并且还要确保分解过程是可扩展的，即只要有必要，该分解过程就可以持续下去。

在创建这种模式的实例时，通常使用两种相反的工作流程。

● 自顶向下：自顶向下的工作流从系统层级开始并标识结构对象，这些对象提供实现协作的服务。在实时系统和嵌入式系统中，大多数情况下是基于某个标准方法，将系统分成一个个子系统。当开发人员逐步降低抽象层级，向下推进时，容易确保开发者的工作没有偏离用例中所规定的需求。

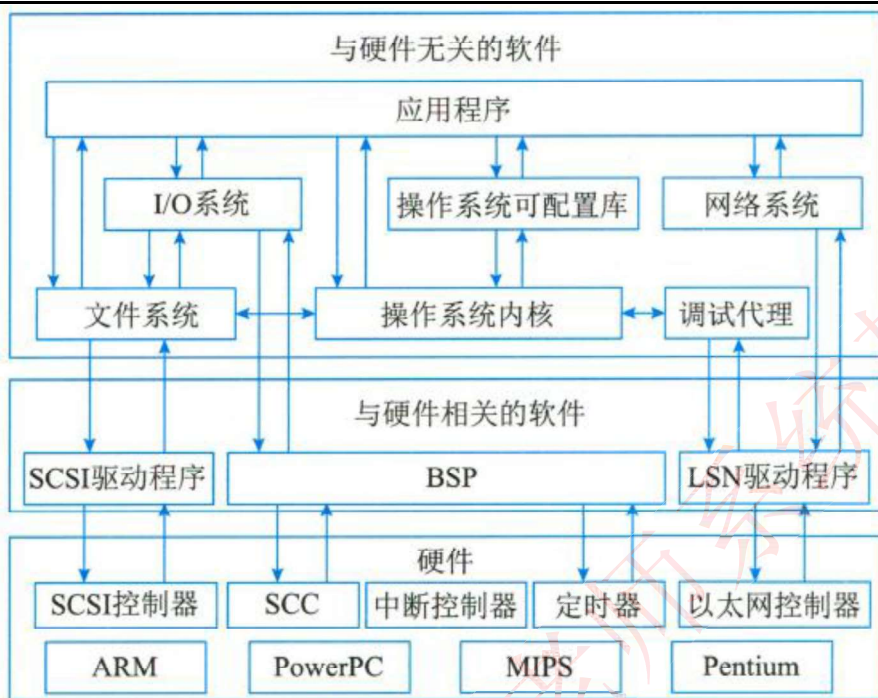
● 自底向上：自底向上专注于域的构造——首先确定域中的关键类和关系。这种方法之所以可行是因为：开发者以往有丰富的开发经验，并能将其他领域所获得的知识映射到当前开发所在的域中。通过这种方法，最终开发者会到达子系统级的抽象。

嵌入式操作系统(EOS)是指用于嵌入式系统的操作系统。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。

嵌入式操作系统特点：可裁剪性、可移植性、强实时性、强紧凑性、高质量代码、强定制性、标准接口、强稳定性、弱交互性、强确定性、操作简洁方便、较强的硬件适应性、可固化性。

从嵌入式操作系统体系架构看，主要存在 4 种结构：整体结构、层次结构、客户/服务器结构和面向对象结构。

整体结构也称为模块结构或无序结构，它是基于结构化程序设计的一种软件设计方法，其架构如下图：



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（即单体内核）和微内核。

任务管理是嵌入式操作系统最基本功能之一，这里的任务(task)是指嵌入式操作系统调度的最小单位，类似于一般操作系统进程或线程的概念。

在实时系统的任务调度中，存在大量的实时调度方法，大致可以概述为主要三种划分，即离线和在线调度（在运行前还是运行时获得调度信息）、抢占和非抢占调度（能否打断正在运行的任务）、静态和动态调度（在设计时还是运行时确定任务优先级）等。

典型的强实时调度算法：

(1)最早截止时间优先(Earliest Deadline First, EDF) 算法。根据任务的开始截止时间来确定任务的优先级。截止时间愈早，其优先级愈高。

(2)最低松弛度优先(Least Laxity First, LLF) 算法。根据任务紧急(或松弛)的程度，来确定任务的优先级。任务的紧急程度愈高，该任务被赋予的优先级就愈高，以使之优先执行。松弛度(又叫富裕度)即进程的富裕时间，或者可以理解为自由时差。

(3)单调速率调度算法(Rate Monotonic Scheduling, RMS)。是一种静态优先级调度算法，是经典的周期性任务调度算法。RMS的基本思路是任务的优先级与它的周期表现为单调函数的关系，任务的周期越短，优先级越高；任务的周期越长，优先级越低。

操作系统任务之间一般存在以下关系：相互独立、竞争、同步、通信。要实现多任务间的协同工作，操作系统必须提供任务间的通信手段。嵌入式操作系统一般都会提供多任务间通信的方法，常用的通信方式包括：

- 共享内存：数据的简单共享。多任务访问同一地址空间。
- 信号量：基本的互斥和同步。最优，主要手段，任务间最快速通信。

● 消息队列: 同一 CPU 内多任务间消息传递。类似于缓冲区的对象, 像一个管道接收发送者消息等待接收者读取。

● Socket 和远程调用: 任务间透明的网络通信, 不同计算机之间通信。

● Signals (信号): 用于异常处理。通知进程发生了异步事件, 是对中断机制的一种模拟。

与传统数据库相比, 嵌入式数据库系统有以下几个主要特点: 嵌入式、实时性、移动性、伸缩性。

嵌入式数据库按存储位置的不同可分为三类: 基于内存方式、基于文件方式、基于网络方式。

(1) 基于内存的数据库系统(MMDB) 是实时系统和数据库系统的有机结合。内存数据库是支持实时事务的最佳技术, 其本质特征是以其“主拷贝”或“工作版本”常驻内存, 即活动事务只与实时内存数据库的内存拷贝打交道。典型产品是 eXtremeDB 嵌入式数据库。eXtremeDB 主要特点:

- 最小化支持持久数据所必需的资源: 实质上就是将内存资源减到最小。
- 保持极小的必要堆空间: 在某些配置上 eXtremeDB 只需要不到 1KB 的堆空间。
- 维持极小的代码体积。
- 通过紧密的集成持久存储和宿主应用程序语言消除额外的代码层。
- 提供对动态数据结构的本地支持: 例如变长字符串、链表和树。

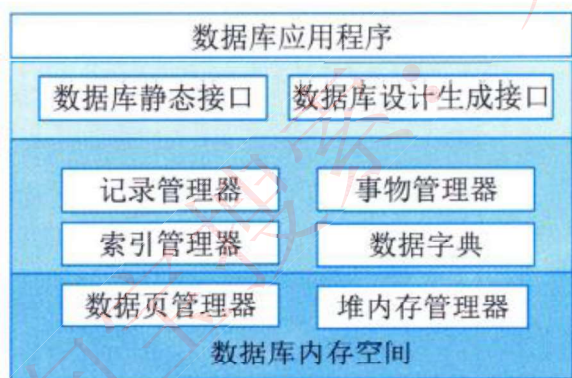


图 16-17 eXtremeDB 嵌入式数据库的架构

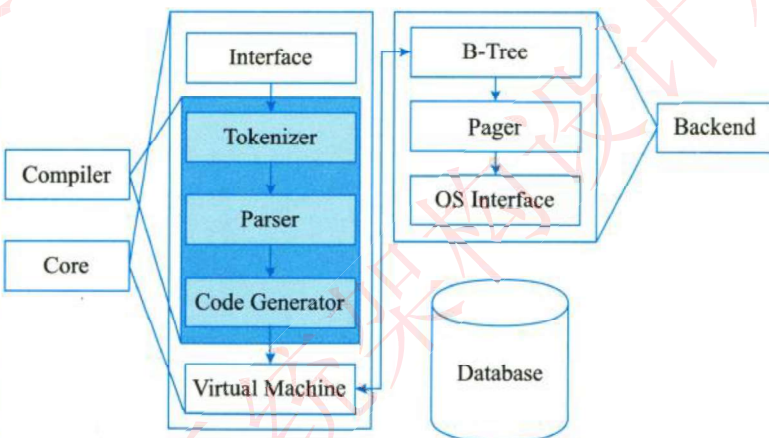


图 16-18 SQLite 嵌入式数据库系统架构

(2) 基于文件的数据库(FDB) 系统就是以文件方式存储数据库数据, 即数据按照一定格式储存在磁盘中。使用时由应用程序通过相应的驱动程序甚至直接对数据文件进行读写。典型产品是 SQLite, 它由公共接口、编译器系统、虚拟机和后端四个子系统组成。SQLite 主要特点:

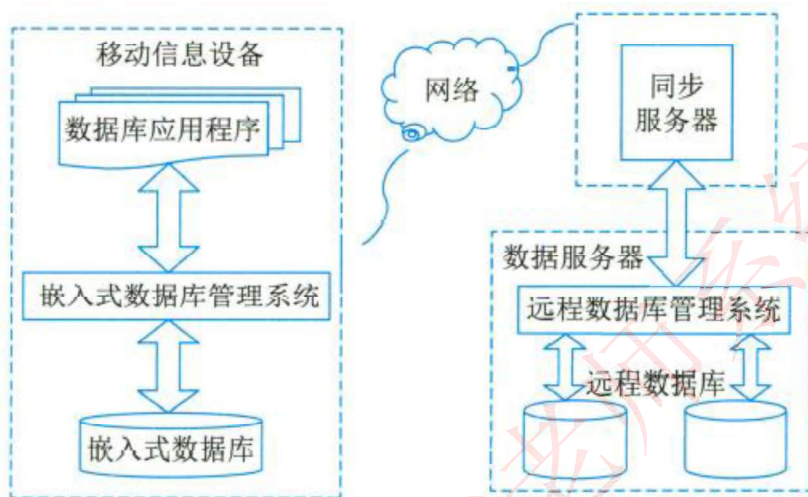
● SQLite 是一个开源的、内嵌式的关系型数据库。

● SQLite 数据库服务器就在你的数据库应用程序中, 其好处是不需要网络配置和管理, 也不需要通过网络设置数据源访问数据库服务器。

● SQLite 数据库的服务器和客户端运行在同一个进程中。这样可以减少网络访问的消耗, 简化数据库管理, 使你的程序部署起来更容易。

● SQLite 在处理数据类型时与其他的数据库不同。区别在于它所支持的类型以及这些类型是如何存储、比较、强化(enforce) 和指派(assign)。

(3) 基于网络的数据库(NDB) 系统是基于手机 4G/5G 的移动通信基础之上的数据库系统，在逻辑上可以把嵌入式设备看作远程服务器的一个客户端。实际上，嵌入式网络数据库是把功能强大的远程数据库映射到本地数据库，使嵌入式设备访问远程数据库就像访问本地数据库一样方便。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。其架构如下



数据库服务器架构：数据库客户端通常通过数据库驱动程序如 J D B C 、 O D B C 等访问数据库服务器，数据库服务器再操作数据库文件。数据库服务是一种客户端服务器模式，客户端和服务端是完全两个独立的进程。它们可以分别位于在不同的计算机甚至网络中。客户端和服务端通过 TCP/IP 进行通信。这种模式将数据与应用程序分离，便于对数据访问的控制和管理。

嵌入式数据库架构：嵌入式数据库不需要数据库驱动程序，直接将数据库的库文件链接到应用程序中。应用程序通过 API 访问数据库，而不是 TCP/IP。因此，嵌入式数据库的部署是与应用程序在一起的。

数据库服务器和嵌入式数据库对比如下：

(1) 数据库服务器通常允许非开发人员对数据库进行操作，而在嵌入式数据中通常只允许应用程序对其进行访问和控制。

(2) 数据库服务器将数据与程序分离，便于对数据库访问的控制。而嵌入式数据库则将数据的访问控制完全交给应用程序，由应用程序来进行控制。

(3) 数据库服务器需要独立的安装、部署和管理，而嵌入式数据通常和应用程序一起发布，不需要单独地部署一个数据库服务器，具有程序携带性的特点。

嵌入式数据库有其自身的特殊需要，它应具备的功能包括以下 4 点：

- 足够高效的数据存储机制；
- 数据安全控制(锁机制)；
- 实时事务管理机制；
- 数据库恢复机制(历史数据存储)。

这样，一般嵌入式数据库可划分成数据库运行处理、数据库存取、数据管理、数据库维护和数据库定义等功能。

在嵌入式环境下开发实时数据库系统需要特别解决以下几个设计问题：

(1) 存储空间管理模块。嵌入式实时数据库系统由于采用了内存缓冲的技术，必然要涉及嵌入式操作系统的内存管理。系统运行时，由该模块通过实时 O/S 向系统申请内存缓冲区，作为共享的内存数据区使用。之后，将历史数据库中的初始化数据调入内存区对这些空白内存进行初始化。

(2) 数据安全性、完整性控制模块。

(3) 事务并发控制模块。在实时数据库中的封锁粒度通常选择一张关系表为一个单位。

(4) 实时数据转储模块。该模块实现的功能是将实时数据存储为历史数据，通常由该模块先将历史数据保存在内存缓冲区中，缓冲区满时才一次性写入磁盘；读历史数据时，先从缓冲区内取数据，取不到数据时再进行文件的读写。

(5) 运行日志管理模块。日志文件可以用来进行事务故障恢复和系统故障恢复。

嵌入式中间件是在嵌入式系统中处于嵌入式应用和操作系统之间层次的中间软件，其主要作用是对嵌入式应用屏蔽底层操作系统的异构性，常见功能有网络通信、内存管理和数据处理等。

中间件=平台+通信。中间件的共性特点：通用性、异构性、分布性、协议规范性、接口标准化。具体到嵌入式中间件而言，它还应提供对下列环境的支持：

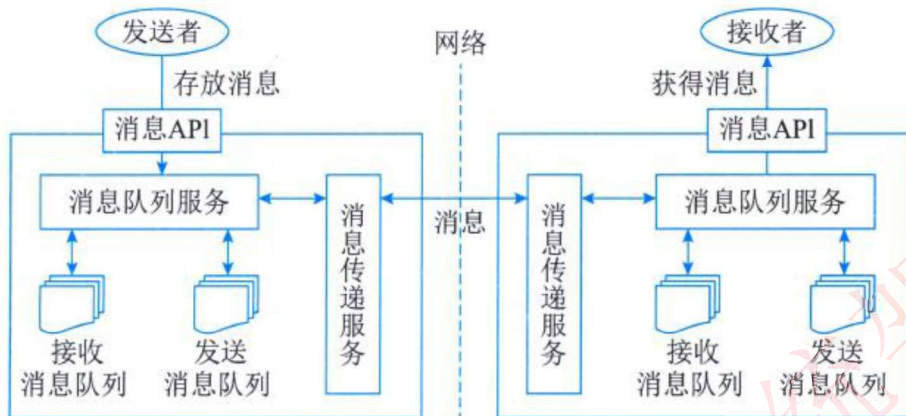
- 网络化：支持移动、无线环境下的分布应用，适应多种设备特性以及不断变化的网络环境；
- 支持流媒体应用，适应不断变化的访问流量和宽带约束；
- QoS 质量品质：在分布式嵌入式实时环境下，适应强 QoS 的分布应用的软硬件约束；
- 适应性：能够适应未来确定的应用要求。

通用中间件大致存在以下几类：

- 企业服务总线中间件：ESB 是一种开放的、基于标准的分布式同步/异步信息传递中间件。
- 事务处理监控器：为发生在对象间的事务处理提供监控功能，以保证操作成功。
- 分布式计算环境：指创建运行在不同平台上的分布式应用程序所需的一组技术服务。
- 远程过程调用：指客户机向服务器发送关于运行某程序的请求时所需的标准。
- 对象请求代理：为用户提供与其他分布式网络环境中对象通信的接口。
- 数据库访问中间件：支持用户访问各种操作系统或应用程序中的数据库。
- 消息传递：电子邮件系统是该类中间件的其中之一。
- 基于 XML 的中间件：XML 允许开发人员为实现 Internet 中交换结构化信息而创建文档。

消息中间件的一般架构：消息中间件是消息传输过程中保存消息的一种容器。架构如下图。包括点对点 and 发布/订阅两种模式。

两个基本特点：采用异步处理模式、应用程序和应用程序调用关系为松耦合关系。



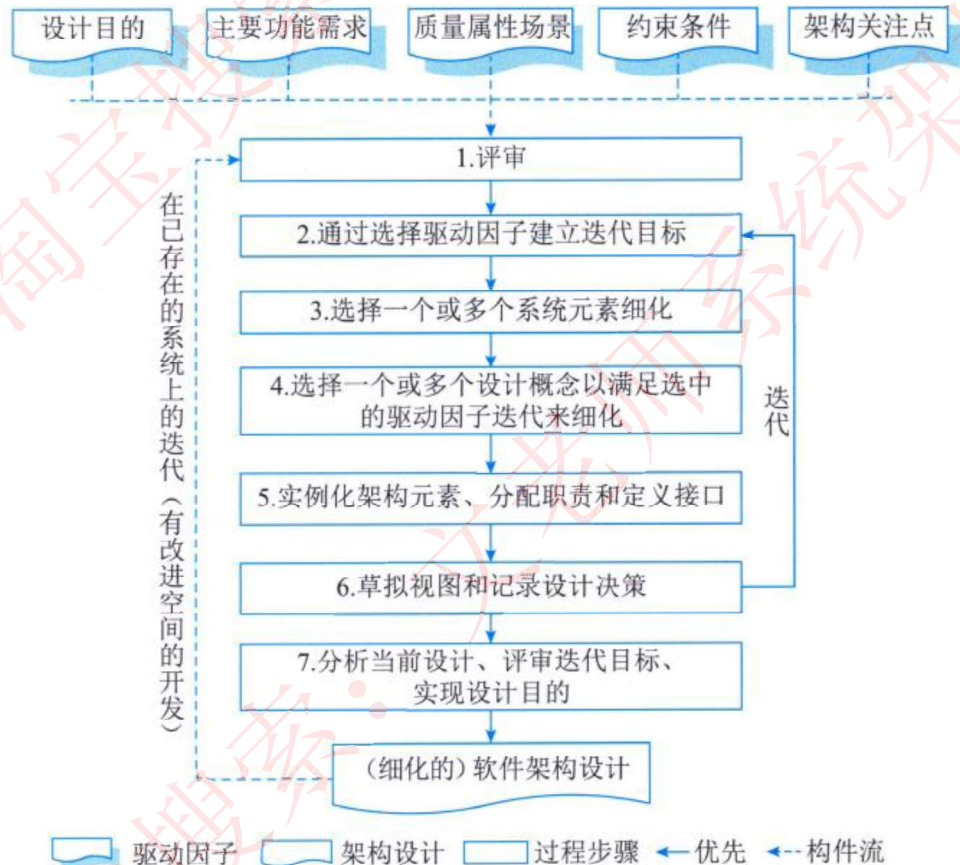
嵌入式系统软件开发环境、分布式中间件等内容在基础知识-嵌入式章节已有详解，不再赘述。

5.2 嵌入式系统软件架构设计方法

在嵌入式系统中，其设计通常采用了自顶向下的设计方法，**基于架构的软件设计 (ABSD)** 可适应于嵌入式系统的软件设计方法。ABSD 已经在系统架构设计章节有详解，不再赘述。

属性驱动的软件设计 (ADD) 是把一组质量属性场景作为输入，利用对质量属性实现与架构设计之间的关系的了解(如体系结构风格、质量战术等)对软件架构进行设计的一种方法。

采用 ADD 方法进行软件开发时，需要经历评审、选择驱动因子、选择系统元素、选择设计概念、实体化元素和定义接口、草拟视图和分析评价等七个阶段，如图所示：



步骤一：评审输入

首先需要确保设计流程的输入是可用且正确的。其次，确认设计目的是否符合设计的类型，要确保设计过程中其他的属性驱动因子也是可用的。最后，如果是设计一个已有的系统，还需要分析已经存在的架构设计的输入存在是否合理。

步骤二：通过选择驱动因子(架构)建立迭代目标

根据使用的开发模型去选择设计的回合。一个设计回合需要在一系列的设计迭代中进行，每一个迭代着重完成一个目标，特别是满足驱动因子的目标。

步骤三：选择一个或者多个系统元素来细化

系统元素可以是指一个软件模块，或者是指包含了多个元素或子模块的整个软件系统。本步骤主要是指选取可满足驱动因子需要的一个或者多个架构结构，这些结构是由具有内在关联的元素组成的。

步骤四：选择一个或者多个设计概念来细化

本步骤是从常用的架构设计模式中选取一种或多种设计概念，对选中的驱动因子进行细化。

步骤五：实例化架构元素、分配职责和定义接口

选择好了一个或者多个设计概念后，就要求做另一个设计决策了，包括所选择的实例化元素的设计概念。比如，如果选择分层，就需要决定分多少层。

步骤六：草拟视图和记录设计决策

本步骤就是将上述活动的结果用文字或图的方式记录或绘制出来，以供后续迭代使用。

步骤七：分析当前设计、评审迭代目标、实现设计目的

到本步骤，应该说已经创建好了部分设计，可以得到这个迭代设计建立的目标。在这个确定的目标前提下，可以得到项目利益相关者的认同，避免否定，导致返工。

实时系统设计方法（DARTS）主要是将实时系统分解为多个并发任务，并定义这些任务之间的接口。提供了一些分解规则和一套处理并发任务的设计步骤，还提供了一套把实时系统建造成并发任务的标准和定义并发任务间接口的指南。

RTSAD（实时结构化分析和设计方法）是 DARTS 方法的起源，是对传统结构化分析和设计方法的补充扩展，专门用于开发实时系统。

实时结构化分析(RTSA) 主要对传统的结构化分析方法扩充了行为建模部分，它通过状态转换图(STD) 刻画系统的行为特征，并利用控制转换与数据流图集成在一起。

实时结构化设计(RTSD) 是利用内聚和耦合原则进行程序设计的一个方法，它通过事务和变换两种策略将 RTSA 的分析结构 DFD/CFD 转换为程序结构图。

任务结构化标准可以为设计人员将实时系统分解为并发任务的时候提供帮助。这些标准是从设计并发系统所积累的经验中得到的启发。确定任务过程中主要考虑的问题是系统内部功能的并发特性。

信息隐藏作为封装数据存储的标准来使用。信息隐藏模块用于信息数据存储和状态转换表的内容和表示。

RTSAD 设计方法使用任务架构图来显示系统分解为并发任务的过程，以及采用消息、事件和信息隐藏模块形式的任务间接口。

DARTS 方法由以下 5 部分组成：

1) 用实时结构化分析方法(RTSA) 开发系统规范：本阶段要开发系统环境图(SCD) 和状态转换图(STD)。

2) 将系统划分为多个并发任务：任务结构化标准应用于数据流/控制流图层次集事件合中的叶子节点上。初步任务架构图(TAD) 可以显示使用任务结构化标准确定的任务。

3) 定义任务间接口：通过分析在上一阶段确认的任务间的数据/控制流接口可以定义任务间的接口。任务间的数据流被映射为松耦合的或紧密耦合的消息接口。事件流被映射为事件信号。数据存储被映射为信息隐藏模块。这个阶段应该完成时间约束分析。

4) 设计每个任务：每个任务都代表了一个顺序程序的执行。在这个阶段要定义各个模块的功能以及与其他模块之间的接口。此外，还要设计各个模块的内部结构。

5) 设计过程的成果：RTSA 规范、任务结构规范（定义每个并发任务功能及接口）、任务分解（定义每个任务分解为模块的过程以及模块的功能接口详细设计）。

DARTS 开发方法的主要优势：

- 强调把系统分解成并发的任务，并提供了确认这些任务的标准。强调并发在并发实时系统的设计中非常重要；

- 提供了详细的定义任务间接口的指南；

- 强调了用任务架构图(STD) 的重要性，这在实时系统的设计中也非常重要；

- 提供了从 RTSA 规格到实时设计的转换。

DARTS 开发方法的不足之处：

- 用结构化的设计方法把任务创建成了程序模块，而并非完全用 IHM 来封装数据存储。

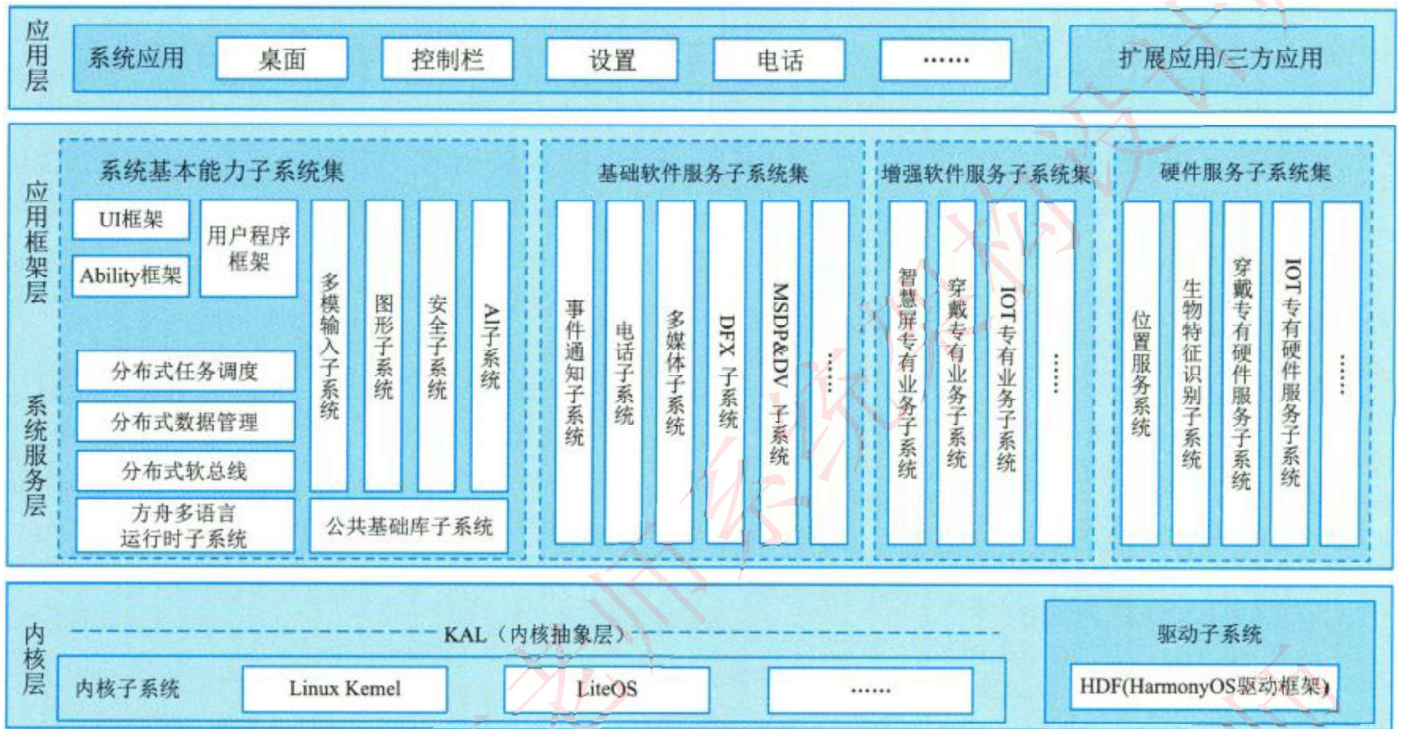
- 如果 RTSA 阶段的工作没有做好，创建任务就非常困难。

5.3 嵌入式系统软件架构案例分析

鸿蒙操作系统架构案例分析

鸿蒙操作系统(HarmonyOS)是一款“面向未来”、面向全场景(移动办公、运动健康、社交通信、媒体娱乐等)的分布式操作系统。在传统的单设备系统能力的基础上， HarmonyOS 提出了基于同一套系统能力、适配多种终端形态的分布式理念，能够支持多种终端设备的能力。

鸿蒙(HarmonyOS) 整体采用分层的层次化设计，从下向上依次为：内核层、系统服务层、框架层和应用层。系统功能按照“系统”→“子系统”→“功能/模块”逐级展开，在多设备部署场景下，支持根据实际需求裁剪某些非必要的子系统或功能/模块，



1) 内核层：主要由内核子系统和驱动子系统组成。

内核子系统：HarmonyOS 采用多内核设计，支持针对不同资源受限设备选用适合的 OS 内核。内核抽象层通过屏蔽多内核差异，对上层提供基础的内核能力，包括进程/线程管理、内存管理、文件系统、网络管理和外设管理等。

驱动子系统：HarmonyOS 驱动框架是 HarmonyOS 硬件生态开放的基础，提供统一外设访问能力和驱动开发、管理框架。

2) 系统服务层：是 HarmonyOS 的核心能力集合，通过框架层对应用程序提供服务。该层包含 4 个部分：

系统基本能力子系统集：为分布式应用在 HarmonyOS 多设备上的运行、调度、迁移等操作提供了基础能力。

基础软件服务子系统集：为 HarmonyOS 提供公共的、通用的软件服务。

增强软件服务子系统集：为 HarmonyOS 提供针对不同设备的、差异化的能力增强型软件服务。

硬件服务子系统集：为 HarmonyOS 提供硬件服务。

3) 框架层：为 HarmonyOS 的应用程序提供了 Java/C/C++/JS 等多语言的用户程序框架和 Ability 框架，以及各种软硬件服务对外开放的多语言框架 API；同时为采用 HarmonyOS 的设备提供了 C/C++/JS 等多语言的框架 API，不同设备支持的 API 与系统的组件化裁剪程度相关。

4) 应用层：包括系统应用和第三方非系统应用。HarmonyOS 的应用由一个或多个 FA (Feature Ability) 或 PA (Particle Ability) 组成。其中，FA 有 UI 界面，提供与用户交互的能力；而 PA 无 UI 界面，提供后台运行任务的能力以及统一的数据访问抽象。

鸿蒙操作系统架构具有 4 个技术特性：

1) 分布式架构首次用于终端 OS，实现跨终端无缝协同体验。

2) 确定时延引擎和高性能 IPC 技术实现系统天生流畅。

3) 基于微内核架构重塑终端设备可信安全。

4) 通过统一 IDE 支撑一次开发，多端部署，实现跨终端生态共享。

在 HarmonyOS 架构中，重点关注于分布式架构所带来的优势，主要体现在分布式软总线、分布式设备虚拟化、分布式数据管理和分布式任务调度等四个方面。

分布式软总线是多种终端设备的统一基座，为设备之间的互联互通提供了统一的分布式通信能力，能够快速发现并连接设备，高效地分发任务和传输数据；

分布式设备虚拟化平台可以实现不同设备的资源融合、设备管理、数据处理，多种设备共同形成一个超级虚拟终端。针对不同类型的任务，为用户匹配并选择能力合适的执行硬件，让业务连续地不同设备间流转，充分发挥不同设备的资源优势；

分布式数据管理基于分布式软总线的能力，实现应用程序数据和用户数据的分布式管理。用户数据不再与单一物理设备绑定，业务逻辑与数据存储分离，应用跨设备运行时数据无缝衔接，为打造一致、流畅的用户体验创造了基础条件；

分布式任务调度构建统一的分布式服务管理(发现、同步、注册、调用)机制，支持对跨设备的应用进行远程启动、远程调用、远程连接以及迁移等操作，能够根据不同设备的能力、位置、业务运行状态、资源使用情况，以及用户的习惯和意图，选择合适的设备运行分布式任务。

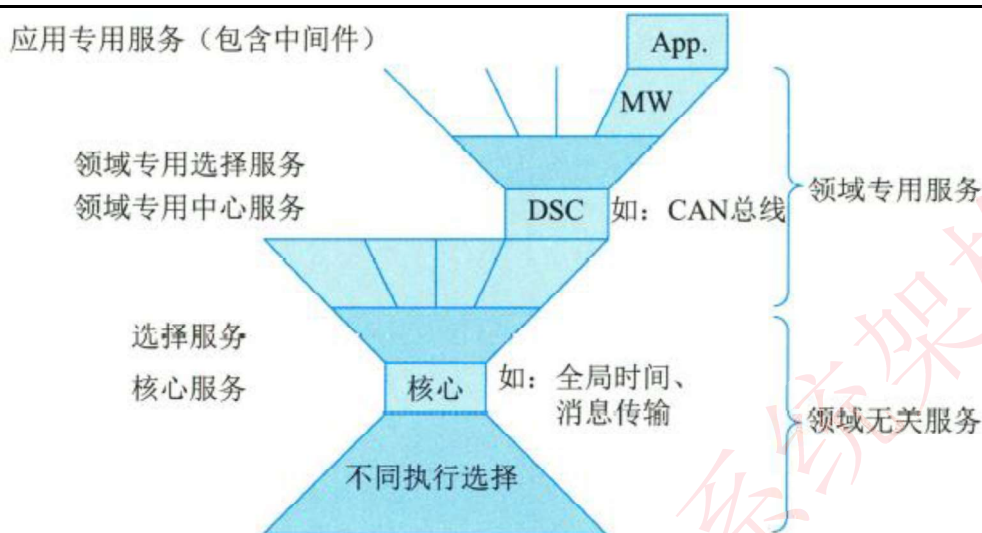
HarmonyOS 架构的系统安全性主要体现在搭载 HarmonyOS 的分布式终端上，可以保证“正确的人，通过正确的设备，正确地使用数据”。这里通过“分布式多端协同身份认证”来保证“正确的人”，通过“在分布式终端上构筑可信运行环境”来保证“正确的设备”，通过“分布式数据在跨终端流动的过程中，对数据进行分类分级管理”来保证“正确地使用数据”。

HarmonyOS 架构提供了基于硬件的可信执行环境来保护用户的个人敏感数据的存储和处理，确保数据不泄露。

面向安全攸关系统的跨领域 GENESYS 系统架构案例分析

GENESYS 是一种跨领域的通用嵌入式架构平台，主要解决了嵌入式系统面临的三方面挑战：复杂性管理的挑战(采用消息交换方式提高软硬件抽象级别)、系统健壮性的挑战(设计故障的隔离框架)、能量有效使用的挑战(采用综合化资源管理方法)。

GENESYS 整个架构包括两类构成系统：即构件和基础平台。基础平台提供了一种“腰”型核心服务和大量用于实现系统构件的可选择服务的最小集合。



GENESYS 架构主要提供了三组服务，即领域无关服务、领域专用服务和应用专用服务(包括中间件)。领域无关服务又分为核心服务和选择服务；领域专用服务又分为领域专用中心服务(DSC) 和领域专用选择服务(DSO)。

(1)核心服务。是强制性的，是 GENESYS 架构实例的一部分。核心服务应包含那些可构造较高级服务或者为了维持该结构性质而不可缺少的服务，它是系统服务中的最小集。

(2)选择服务。是在核心服务之上构造的。它是一种需要时可以扩展的开放式的集合。

(3)领域专用服务。是由领域特有的选择服务子集加上待开发的领域特征的特定服务组合。

GENESYS 架构的重要思想是分离计算与通信，将计算构件和通信设施作为独立构件进行设计。GENESYS 的通信设施构件是基于消息传输的风格。构件中的基本交往机制是多播单向消息的交换。消息在发送时刻发出，在某个稍后的时刻达到在接收者那里。每一个消息有专门标识的发送者和若干个接受者。

GENESYS 架构将构件归为四类：硬件构件和软件构件、系统构件和应用构件。硬件构件的功能使用硬件被预先确定，因此不能修改。在软件构件中，将加载在软件构件上的软件称为作业。将作业分配给适当的可以执行该作业的硬件单元就创建了新的构件。系统构件是提供某些架构服务的构件。系统构件可以广泛重用在许多不同的应用场景中，应用设计者只考虑应用构件的开发。

基于 GENESYS 架构的四类基于消息的构件接口：

链接接口(LIF):LIF 提供了构件与构件之间基于消息的操作服务，它是构件的综合接口。

局部接口(LI):LI 是构件连接到外部环境(如 I/O、其他系统)的接口，它建立了构件和局部环境之间的连接关系。

技术无关接口(TII):TII 是指用于系统运行需要的配置或管理资源的接口，它属于非功能属性范畴。

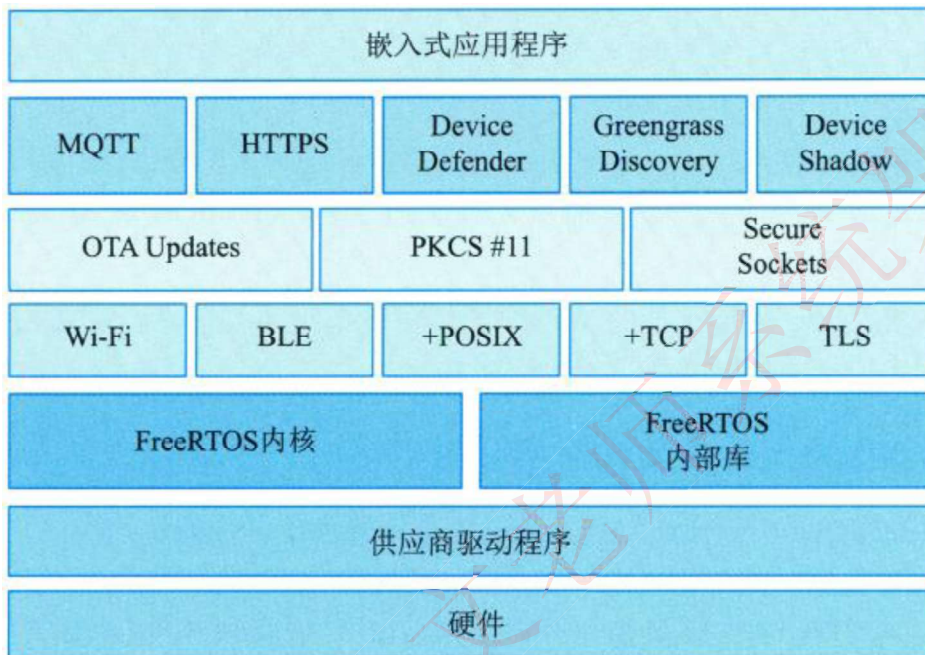
技术相关的接口(TDI):TDI 是指用于查看构件内部、观察构件的内部变量的接口，如构件诊断。

GENESYS 定义了三级的集成，即芯片级(Chip Level)、设备级(Device Level)和系统级(System Level)。芯片级的构件是 IP 核，IP 核间可通过 NoC(Network of Chip) 相互连接；设备级的构件是芯片，芯片间可以由内部通信芯片互相连接。一个设备可以是在互联网上的一个可寻址实体，也可以是一个 IP 地址；系统级的构件是设备，它们可以由有线或无线通信服务互相连接。

物联网操作系统软件架构案例分析

在物联网应用中三项关键，分别是感知层、网络传输层和应用层。

FreeRTOS 是一款开源的物联网操作系统，主要由 BSP 驱动、内核和组件等组成。



物联网操作系统的主要特征：内核尺寸伸缩性以及整体架构的可扩展性、内核的实时性、高可靠性、低功耗。

6 通信系统架构设计

6.1 通信系统网络架构

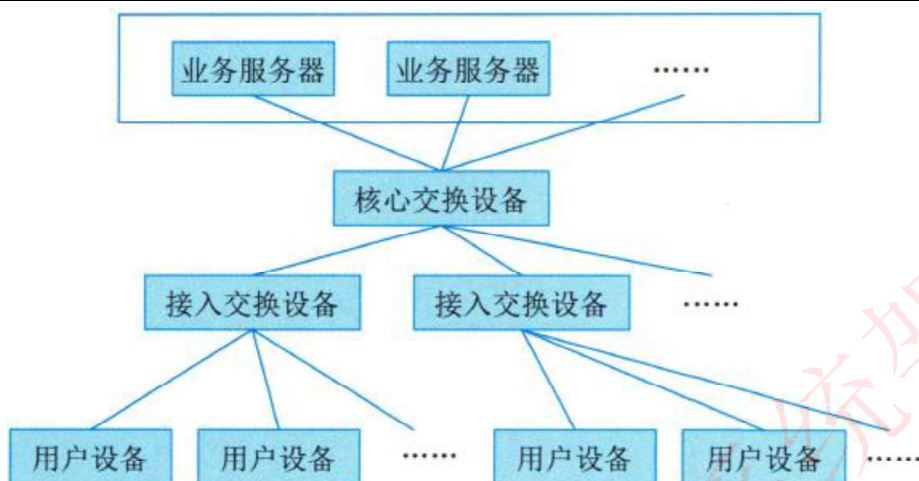
当今，通信网络从大的方面主要包括局域网、广域网、移动通信网等网络形式。不同的网络会采用不同的技术进行网络构建。

局域网，即计算机局部区域网络，是一种为单一机构所拥有的专用计算机网络。其特点是：覆盖地理范围小，通常限定在相对独立的范围内。低误码率，可靠性高；通常为单一部门或单位所有；支持多种传输介质支持实时应用。就网络拓扑而言，有总线型、环型、星型、树型等型式。从传输介质来说，包含有线局域网和无线局域网。

局域网已从早期只提供二层交换功能的简单网络发展到如今不仅提供二层交换功能，还提供三层路由功能的复杂网络。局域网，现代通常用在园区网络的构建中，某种意义上，局域网也称为园区网。

以下给出局域网的几种典型架构风格。

1) 单核心架构：通常由一台核心二层或三层交换设备充当网络的核心设备，通过若干台接入交换设备将用户设备(如用户计算机、智能设备等)连接到网络中。



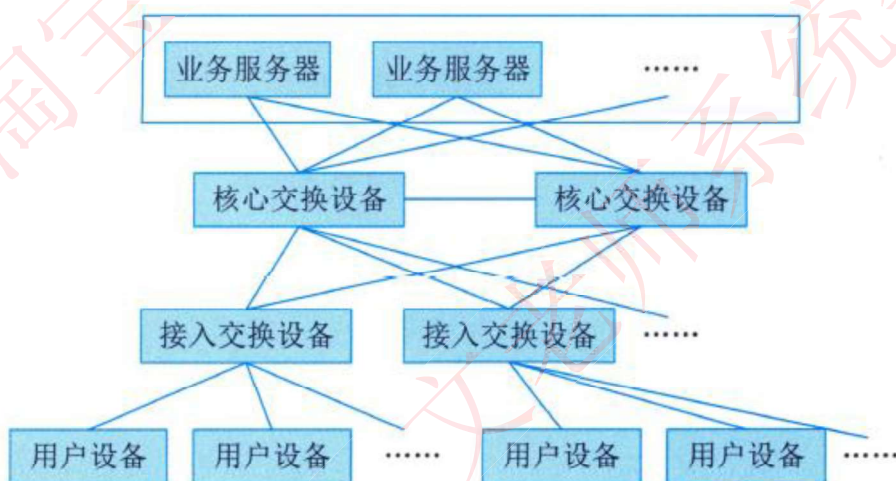
此类局域网可通过连接核心网交换设备与广域网之间的互连路由设备(边界路由器或防火墙)接入广域网，实现业务跨局域网的访问。单核心网具有如下特点：

- (1)核心交换设备通常采用二层、三层及以上交换机；
- (2)接入交换设备采用二层交换机，仅实现二层数据链路转发；
- (3)核心交换设备和接入设备之间可采用 100M/GE/10GE 等以太网连接。

用单核心构建网络，其优点是：网络结构简单，可节省设备投资。需要使用局域网的部门接入较为方便，直接通过接入交换设备连接至核心交换设备空闲接口即可；

其不足是网络地理范围受限，要求使用局域网的部门分布较为紧凑；核心网交换设备存在单点故障，容易导致网络整体或局部失效；网络扩展能力有限；在局域网接入交换设备较多的情况下，对核心交换设备的端口密度要求高。

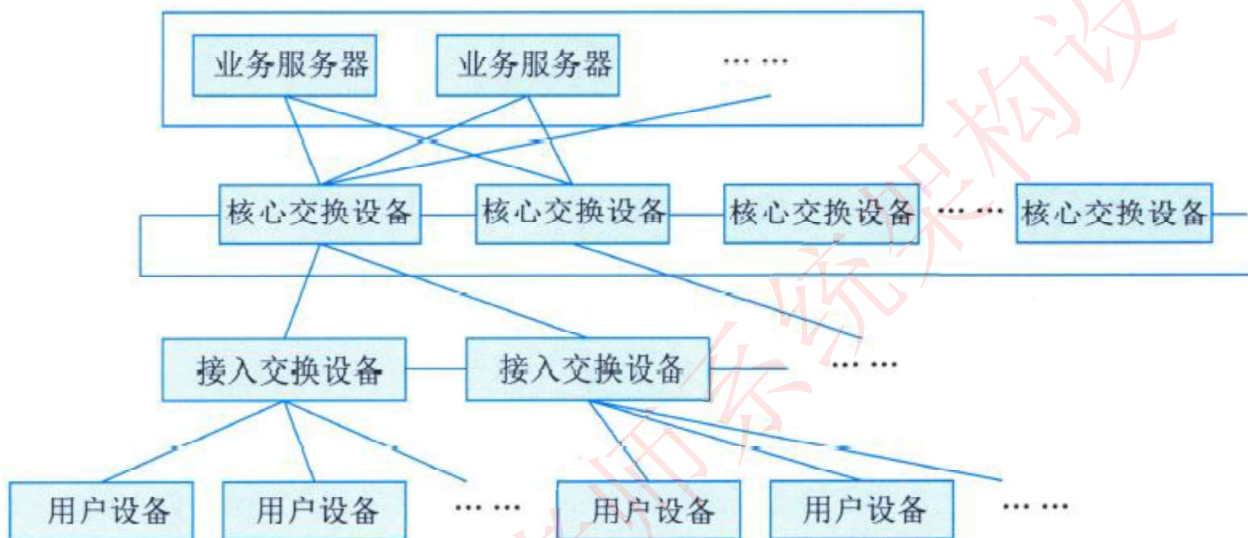
2)双核心架构：通常是指核心交换设备通常采用三层及以上交换机。核心交换设备和接入设备之间可采用 100M/GE/10GE 等以太网连接。



网络内划分 VLAN 时，各 VLAN 之间访问需通过两台核心交换设备来完成。网络中仅核心交换设备具备路由功能，接入设备仅提供二层转发功能。

核心交换设备之间互联，实现网关保护或负载均衡。需要使用局域网的部门接入较为方便，直接通过接入交换设备连接至核心交换设备空闲接口即可。

3) 环型架构: 是由多台核心交换设备连接成双 RPR 动态弹性分组环, 构建网络的核心。核心交换设备通常采用三层或以上交换机提供业务转发功能。



典型环型局域网网络内各 VLAN 之间通过 R P R 环实现互访。通过 R P R 组建大规模局域网时, 多环之间只能通过业务接口互通, 不能实现网络直接互通。环型局域网设备投资比单核心局域网的高。核心路由冗余设计实施难度较高, 且容易形成环路。

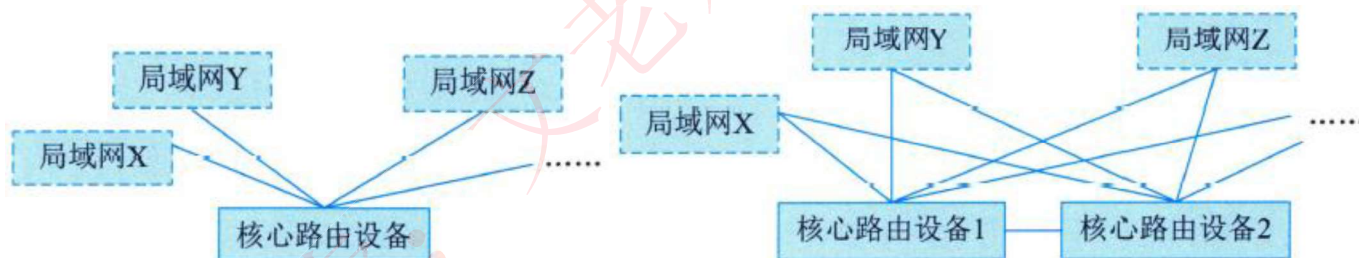
此网络通过与环上的交换设备互联的边界路由设备接入广域网。

4) 层次局域网架构(或多层局域网)由核心层交换设备、汇聚层交换设备和接入层交换设备, 以及用户设备等组成。

广域网属于多级网络, 通常由骨干网、分布网、接入网组成。在网络规模较小时, 可仅由骨干网和接入网组成。通常, 在大型网络构建中, 通过广域网将分布在各地域的局域网互连起来, 形成一个大的网络。以下给出不同形式的广域网构建模型以及各自的特点。

1) 单核心广域网: 通常由一台核心路由设备和各局域网组成。核心路由设备采用三层及以上交换机。网络内各局域网之间访问需要通过核心路由设备。

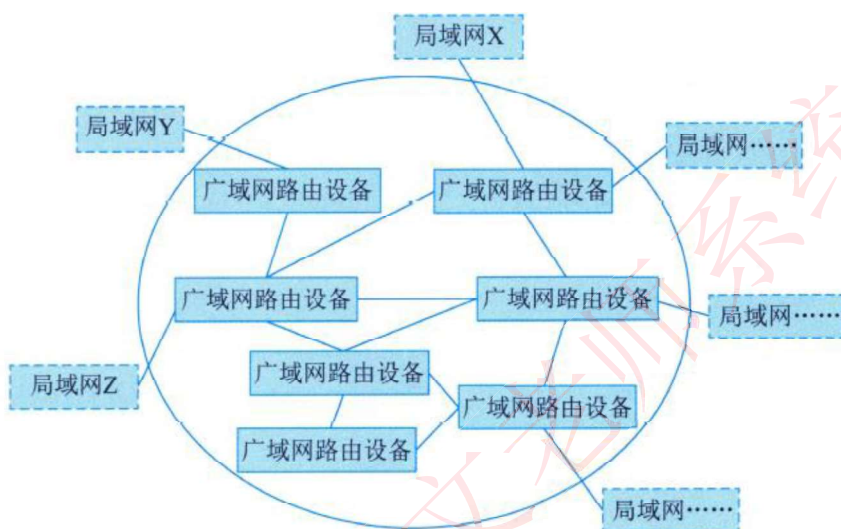
2) 双核心广域网: 通常由两台核心路由设备和各局域网组成。其主要特征是核心路由设备通常采用三层及以上交换机。核心路由设备之间实现网关保护或负载均衡。各局域网访问核心局域网, 以及它们相互访问可有多条路径选择, 可靠性更高, 路由层面可实现热切换, 提供业务连续性访问能力。



3) 环型广域网: 通常是采用三台以上核心路由器设备构成路由环路, 用以连接各局域网, 实现广域网业务互访。核心路由设备之间具备网关保护或负载均衡机制, 同时具备环路控制功能。各局域网

访问核心局域网，或互相访问，有多条路径可选择，可靠性更高，路由层面可实现无缝热切换，保证业务访问连续性。

4) 半冗余广域网：是由多台核心路由设备连接各局域网而形成的。其中，任意核心路由设备至少存在两条以上连接至其他路由设备的链路。如果任何两个核心路由设备之间均存在链接，则属于半冗余广域网特例，即全冗余广域网。



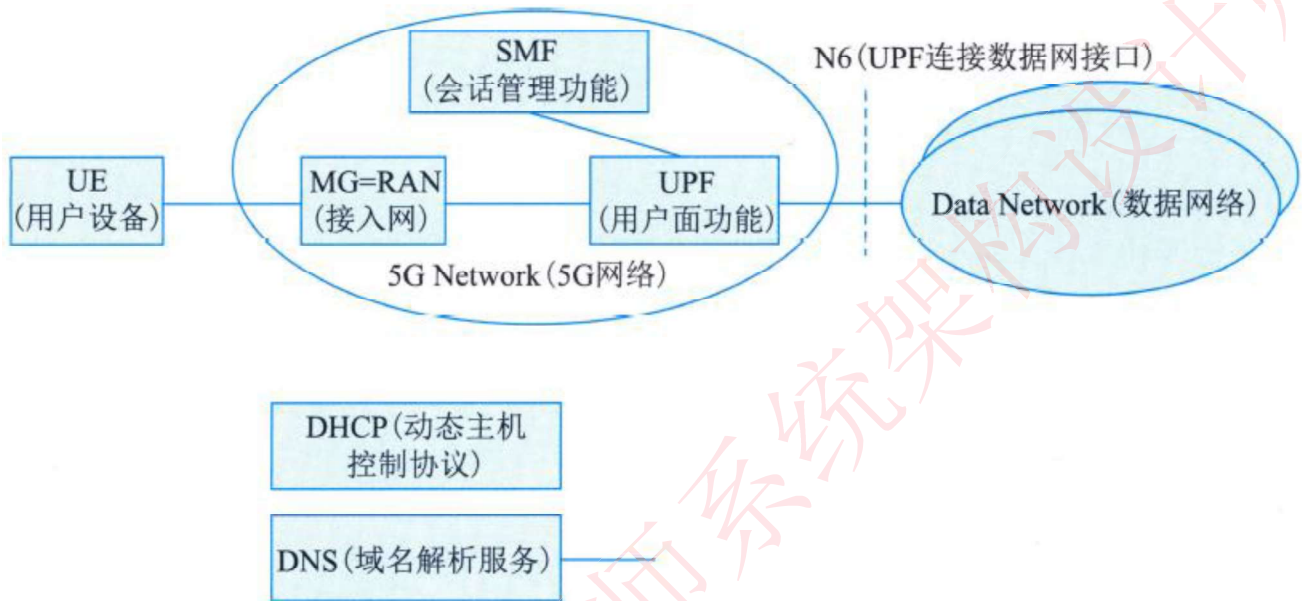
5) 对等子域广域网：是通过将广域网的路由设备划分成两个独立的子域，每个子域路由设备采用半冗余方式互连。两个子域之间通过一条或多条链路互连，对等子域中任何路由设备都可接入局域网网络。

对等子域广域网的主要特征是对等子域之间的互访是以对等子域之间互连链路为主。对等子域之间可做到路由汇总或明细路由条目匹配，路由控制灵活。通常，子域之间链路带宽应高于子域内链路带宽。

6) 层次子域广域网：是将大型广域网路由设备划分成多个较为独立的子域，每个子域内路由设备采用半冗余方式互连，多个子域之间存在层次关系，高层次子域连接多个低层次子域。层次子域中任何路由设备都可以接入局域网。

移动通信网网络架构

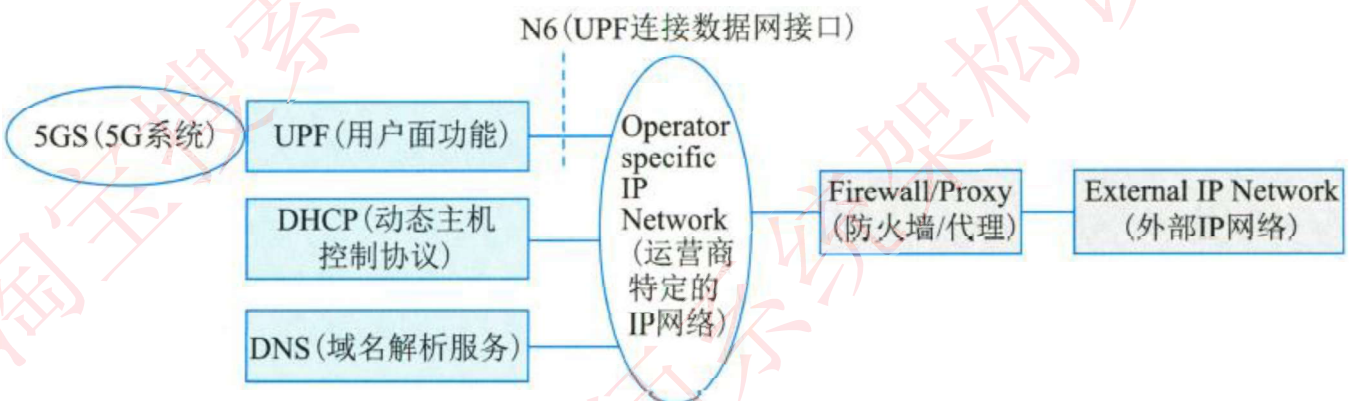
5GS (5G System) 在为移动终端用户 (UE) 提供服务时通常需要 DN (Data Network) 网络，各式各样的上网、语音、AR/VR、工业控制和无人驾驶等 5GS 中 UPF 网元作为 DN 的接入点。5GS 和 DN 之间通过 5GS 定义的 N6 接口互连。5GS 和 DN 之间是一种路由关系。



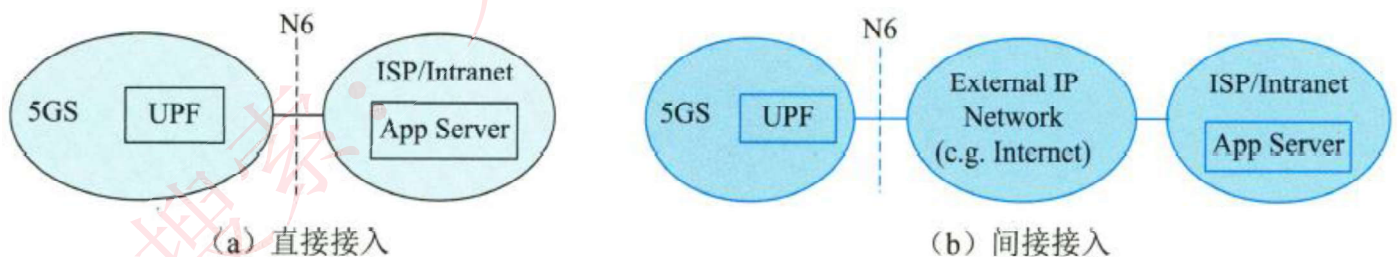
从 UE 通过 5GS 接入 DN 的方式来说，存在两种模式，即透明模式和非透明模式。

1) 透明模式：5GS 通过 UPF 的 N6 接口直接连至运营商特定的 IP 网络，然后通过防火墙或代理服务器连至 DN (即外部 IP 网络)，如 Internet 等。

在此模式下，5GS 至少为 UE 提供一个基本 ISP 服务。对于 5GS 而言，它只须提供基本的隧道 QoS 流服务即可。UE 访问某个 Intranet 网络时，UE 级别的配置仅在 UE 和 Intranet 网络之间独立完成，这对 5GS 而言是透明的。

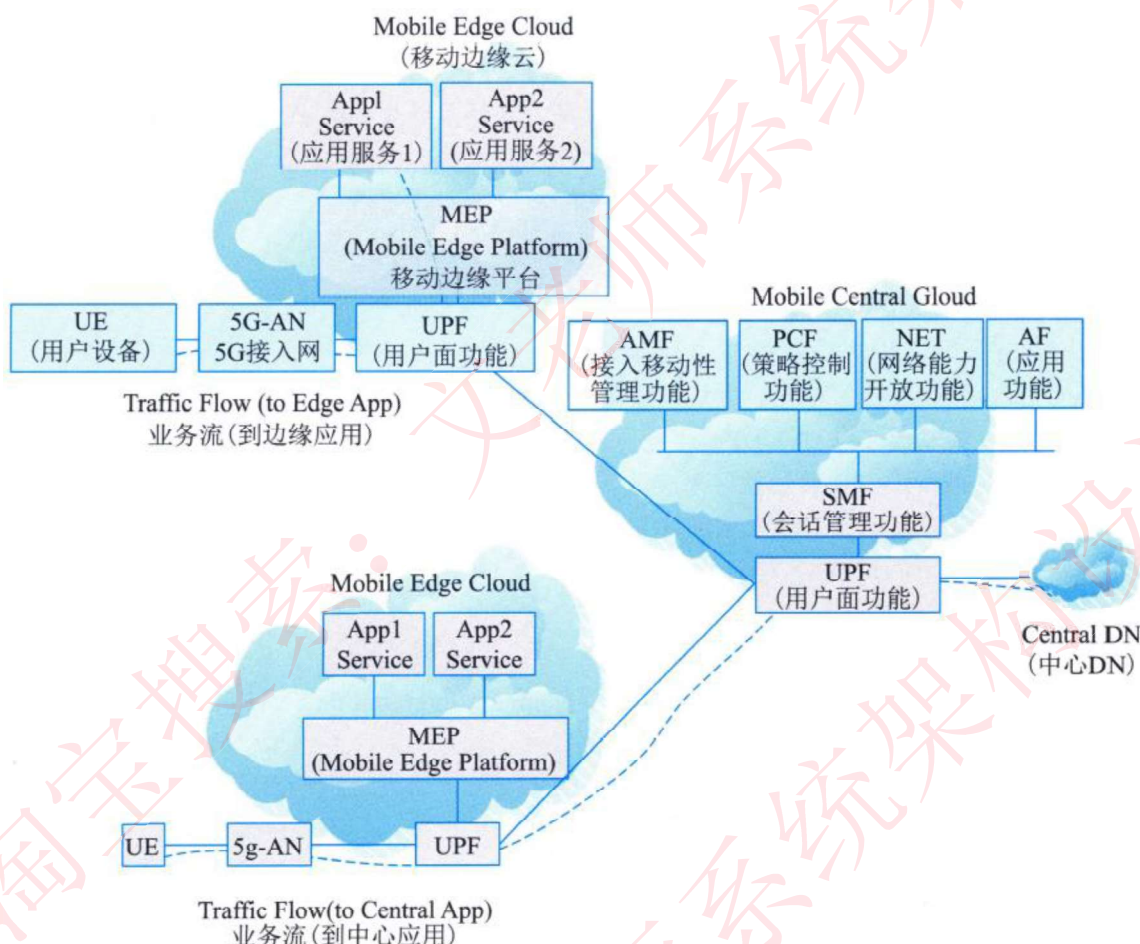


2) 非透明模式：5GS 可直接接入 Intranet/ISP，或通过其他 IP 网络 (如 Internet) 接入 Intranet/ISP。如 5GS 通过 Internet 方式接入 Intranet/ISP，通常需要在 UPF 和 Intranet/ISP 之间建立专用隧道来转发 UE 访问 Intranet/ISP 的业务。UE 被指派属于 Intranet/ISP 地址空间的 IP 地址。此地址用于 UE 业务在 UPF、Intranet/ISP 中转发。



5 G 网络的边缘计算(MEC) 架构如图所示，支持在靠近终端用户 UE 的移动网络边缘部署 5G UPF 网元，结合在移动网络边缘部署边缘计算平台 (MEP)，为垂直行业提供诸如以时间敏感、高带宽为特征的业务就近分流服务。

运营商自有应用或第三方应用 AF 通过 5GS 提供的能力开放功能网元 NEF，触发 5G 网络为边缘应用动态地生成本地分流策略，由 PCF 将这些策略配置给相关 S M F，S M F 根据终端用户位置信息或用户移动后发生的位置变化信息动态实现 UPF（即移动边缘云中部署的 UPF）在用户会话中插入或移除，以及对这些 UPF 分流规则的动态配置，达到用户访问所需业务的极佳效果。



存储网络架构：DAS、NAS、SAN，在计算机网络章节已有详解，不再赘述。

软件定义网络 (Software Defined Network, SDN)：核心思想是通过将网络设备的控制面与数据面进行分离，控制面集中化管控，同时对外提供开放的可编程接口，为网络应用创新提供极佳的能力开放平台；而数据面则通用化、轻量化，高效转发，以提升网络的整体运行效能。

具体来说，SDN 利用分层的思想，将网络分为控制层和数据层。控制层包括可编程控制器，具有网络控制逻辑的中心，掌握网络的全局信息，方便运营商或网络管理人员配置网络和部署新协议等。数据层包括哑交换机 (与传统的二层交换机不同，专指用于转发数据的设备)，仅提供简单的数据转发功能，可以快速处理匹配的数据包。两层之间采用开放的统一接口 (如 OpenFlow 等) 进行交互。通过此接口控制器向转发设备 (如交换机等) 下发统一标准的转发规则，转发设备仅需按照这些规则执行相应动作即可。



由下至上分为数据平面、控制平面和应用平面。

数据平面由网络转发设备(如通常由通用硬件构成)组成,网络转发设备之间通过由不同规则形成的 SDN 数据通路连接起来;控制平面包含了逻辑上为中心的 SDN 控制器,它掌握着网络全局信息,负责转发设备的各种转发规则的下发;应用平面包含各种基于 SDN 的网络应用,应用无须关心网络底层细节就可以编程、部署新应用。

以控制器为逻辑中心,南向接口负责与数据平面进行通信,北向接口负责与应用平面进行通信,东西向接口负责多控制器之间的通信。

6.2 网络构建关键技术

网络的高可用性是一个系统级的概念。对于一个网络来说,它由网络元素(或网络部件),按照一定的连接模型连接在一起而构成。因此,网络可用性包括网络部件、网络连接模型以及有关网络协议等方面的可靠性。

1)网络部件:是组成网络的基本要素,典型代表有各种交换机、路由器等网络设备。网络部件的高可用性是网络高可用性的关键。包括硬件结构和软件系统。硬件可用性通过冗余、热备等保证;软件可用性通过异常保护、数据冗余等保证。

2)网络连接模型:除了网络部件本身的高可用性外,网络物理拓扑连接形式也影响网络的可用性程度。这就涉及到串并联系统的可靠性计算。

3)网络协议及配置:高可用性离不开运行于网络中的路由、链路检测等协议,可以部署链路检测协议发现故障。

IPV4 和 IPV6 三种过渡技术、SDN 等不再赘述。

6.3 网络构建和设计方法

网络需求分析是网络构建及开发过程的起始环节，也是极其重要的阶段。在该阶段，可尽早明确客户使用网络的真实用途或痛点，以便为后续能够构建和设计出更贴近客户真实诉求的网络打下坚实基础。需求分析过程，主要围绕以下几个方面来开展：业务需求、用户需求、应用需求、计算机平台需求和网络需求。最终形成约束后续网络设计的网络需求规格说明书。

网络技术遴选及设计：网络遴选工作是通信系统设计中关键的一项工作，根据计划实施的网络建设要求，遴选工作通常分为局域网、广域网和路由协议的选择。

1. 局域网技术遴选：生成树协议（STP）、虚拟局域网、无线局域网、线路冗余设计、交换设备功能的合理使用、服务器冗余设计。

2. 广域网技术遴选：远程接入技术、广域网互联技术。

3. 地址规划模型：地址分配应遵循以下原则：

(1) 使用结构化网络层编址模型，即对地址进行层次化的规划。

(2) 通过中心授权机构管理地址，比如由组织的 IT 部门为网络层编址提供一个全局模型。根据网络的核心、汇聚、接入层次化结构，为组织的各个区域、分支机构等进行地址规划。

(3) 编址授权下发。即由地址授权管理中心，将编址授权给分支机构来进行地址规划。

(4) 为终端用户设备指派动态地址，即对于频繁变更位置、移动性角度的用户分配动态地址。

(5) 私有地址合理使用。使用私有地址的用户在访问外部网络，需要进行地址转换(NAT)。

4. 路由协议选择包括以下内容：

(1) 路由协议类型的选择：路由协议选择主要包括距离矢量协议和链路状态协议。

(2) 路由选择协议度量值的合理设置。

(3) 路由选择协议顺序的合理指定。

(4) 层次化与非层次化路由选择协议。

(5) 内部与外部路由选择协议。

(6) 分类与无类路由选择协议。

(7) 静态路由指定。

5. 层次化网络模型设计：核心层、汇聚层、接入层。

6. 网络高可用设计方法：提高网络可靠性、缩短网络恢复时间。

网络安全

1. 防火墙

2. VPN：虚拟专用网。

3. 访问控制技术：主体依据控制策略或权限对客体本身或其资源实施的不同授权访问。

4. 网络安全隔离：是在网络运行过程中将网络攻击隔离在可信网络之外，同时保证可信网络内信息不被外泄。

5. 网络安全协议：SSL、SET、HTTPS 等。

6. 网络安全审计：是对网络的脆弱性进行测试评估和分析，最大限度保障业务的安全正常运行的切行为和手段。

绿色网络设计原则

1) 标准化：在设计之初就应考虑解决方案标准化，整体架构标准化可大幅减少转换设备，从而大大降低了能耗。

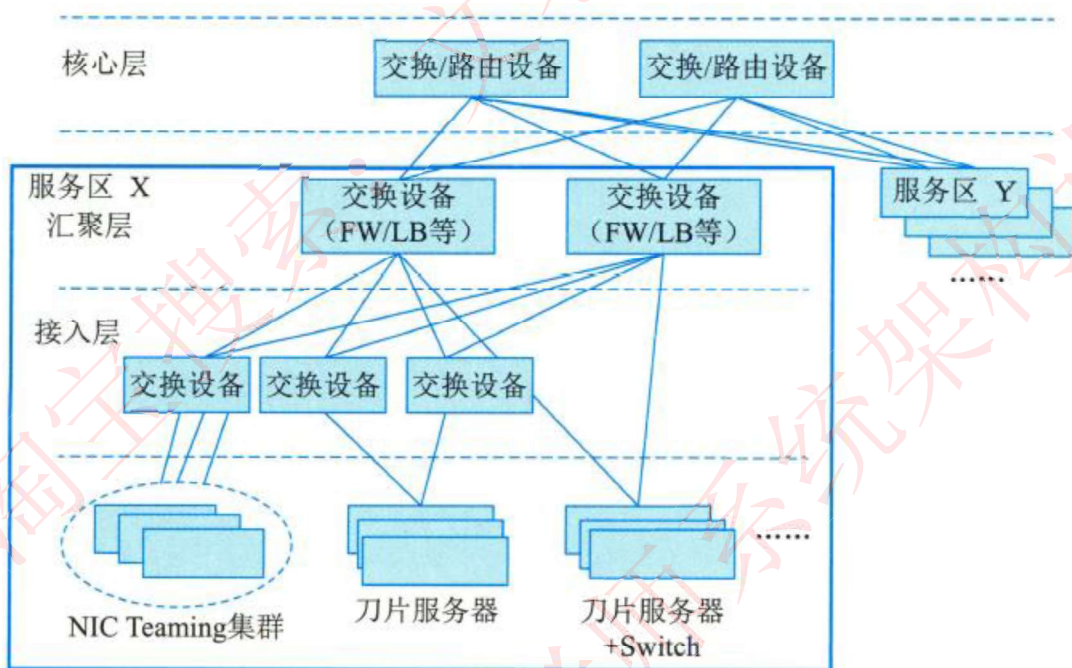
2) 集成化：可使得整个网络系统的通信设备数量尽可能降低，通过减少设备总量、降低设备使用所需资源(空间、机架、线缆、人力等)，来实现节能减排目标。

3) 虚拟化：是一种网络资源可以灵活调配、按需使用的重要途径。

4) 智能化：一方面可以降低人力投入从而降低 TCO，达到绿色环保目的；另一方面智能化方案可以通过智能处理直接降低资源的占用，实现绿色设计。

6.4 通信网络构建案例分析

高可用网络构建分析



1. 网络接入层高可用性设计。高可用接入层具有下述特征：

- (1) 使用冗余引擎和冗余电源获得系统级冗余，为关键用户群提供高可靠性；
 - (2) 与具备冗余系统的汇聚层采用双归属连接，获得默认网关冗余，支持在汇聚层的主备交换机间快速实现故障切换；
 - (3) 通过链路汇聚提供带宽利用率，同时降低复杂度；
 - (4) 通过配置 802.1x, 动态 ARP 检查及 IP 源地址保护等功能增加安全性，有效防止非法访问。
- 接入层到汇聚层有 4 种连接方式，对比如下：

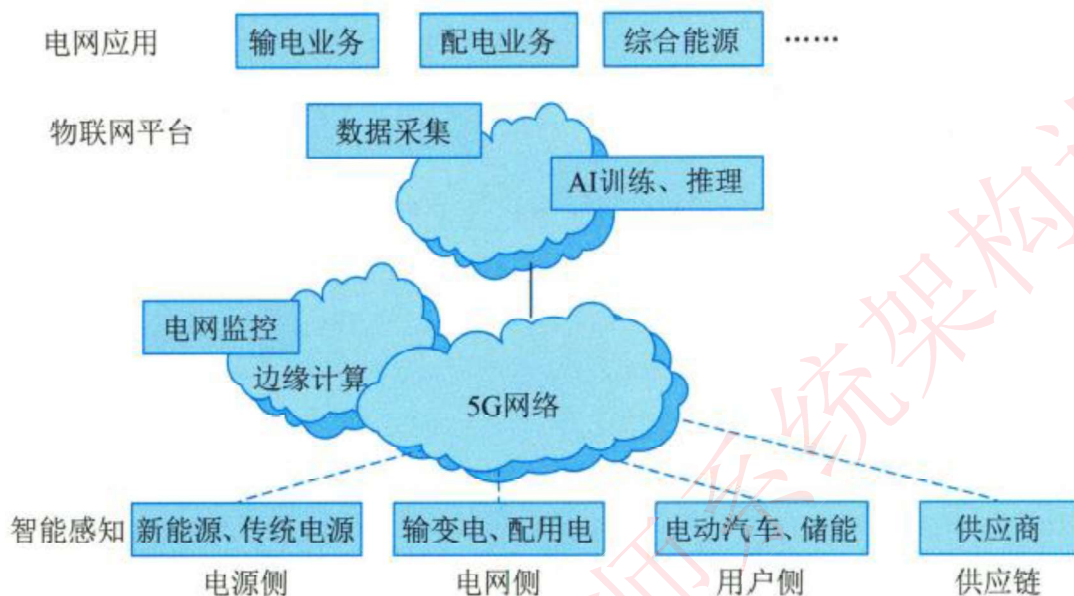
拓扑	优点	缺点
模型一 (倒 U 形)	(1) 无环路，不启用 STP，网络管理简单 (2) VLAN 可以跨汇聚层交换机，用户设备二层扩展灵活	汇聚交换机故障会造成其同侧接入交换机所连用户设备不可达，无法实现高可用接入
模型二 (U 形)	(1) 无环路，不启用 STP，网络管理简单 (2) 接入交换机与汇聚交换机之间有冗余链路保护	(1) VLAN 不能跨汇聚交换机，用户设备部署不灵活 (2) 接入交换机间链路故障时，VRRP 心跳报文无法传递，网络处于不稳定状态
模型三 (矩形)	(1) 接入交换机与汇聚交换机之间有冗余链路保护 (2) VLAN 可以跨汇聚层交换机	(1) 存在环路，启用 STP (2) 当接入交换机上行链路故障时，所有流量将从另一侧交换机上行，网络收敛比变小，网络易拥塞，降低了网络可用性
模型四 (三角形)	(1) 接入交换机与汇聚交换机之间有冗余链路、冗余路径保护 (2) VLAN 可以跨汇聚层交换机，用户设备部署灵活	存在环路，启用 STP，生成树计算较矩形拓扑的复杂

2. 网络汇聚层高可用设计。汇聚层到核心层间采用 OSPF 等动态路由协议实现路由层面高可用保障。典型连接方式有两种，组网模型一为三角形连接方式，从汇聚层到核心层具有全冗余链路和转发路径；组网模型二为矩形连接方式，从汇聚层到核心层为非全冗余链路，当主链路发生故障时，需要通过路由协议计算获得从汇聚到核心的其他路径。可见，组网模型一(即三角形连接方式)的故障收敛时间较小，不足的是，三角形连接方式要占用更多设备端口，建网成本较高。

3. 网络核心层高可用设计。从系统冗余性角度，应考虑部署双核心或多核心设备，以主备或负荷分担方式工作。

5G 网络应用

5G 网络在智能电网中的应用如图所示，通过 5G 网络将种类繁多数据巨大的设备，如电网智能感知设备(传统电源、新能源电源等)，电网中的输变电网设备、配电设备等，用户电表、电动汽车等连接到物联网(IoT)平台中，由 IoT 平台进行电网各个环节的数据采集和智能分析，从而为电网的高级应用(输电业务、配电业务、综合能源管理等业务部门)的科学决策提供有力的支撑。



7 安全架构设计

7.1 安全架构概述

对于信息系统来说，威胁可以是针对物理环境、通信链路、网络系统、操作系统、应用系统以及管理系统等方面。

物理安全威胁是指对系统所用设备的威胁，如自然灾害、电源故障、操作系统引导失败或数据库信息丢失、设备被盗/被毁造成数据丢失或信息泄露；

通信链路安全威胁是指在传输线路上安装窃听装置或对通信链路进行干扰；

网络安全威胁是指由于互联网的开放性、国际化的特点，人们很容易通过技术手段窃取互联网信息，对网络形成严重的安全威胁；

操作系统安全威胁是指对系统平台中的软件或硬件芯片中植入威胁，如“木马”和“陷阱门”、BIOS 的万能密码；

应用系统安全威胁是指对于网络服务或用户业务系统安全的威胁，也受到“木马”和“陷阱门”的威胁；

管理系统安全威胁是指由于人员管理上疏忽而引发人为的安全漏洞，如人为的通过拷贝、拍照、抄录等手段盗取计算机信息。

具体来讲，常见的安全威胁有以下几种。

- (1) 信息泄露：信息被泄露或透露给某个非授权的实体。
- (2) 破坏信息的完整性：数据被非授权地进行增删、修改或破坏而受到损失。
- (3) 拒绝服务：对信息或其他资源的合法访问被无条件地阻止。
- (4) 非法使用(非授权访问)：某一资源被某个非授权的人或以非授权的方式使用。
- (5) 窃听：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。

(6) 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、通信总量的变化等态势进行研究，从而发现有价值的信息和规律。

(7) 假冒：通过欺骗通信系统(或用户)达到非法用户冒充成为合法用户，或者特权小的用户冒充成为特权大的用户的目的。黑客大多是采用假冒进行攻击。

(8) 旁路控制：攻击者利用系统的安全缺陷或安全性上的脆弱之处获得非授权的权利或特权。

(9) 授权侵犯：被授权以某一目的使用某一系统或资源的某个人，却将此权限用于其他非授权的目的，也称作“内部攻击”。

(10) 特洛伊木马：软件中含有一个察觉不出的或者无害的程序段，当它被执行时，会破坏用户的安全。

(11) 陷阱门：在某个系统或某个部件中设置了“机关”，使得当提供特定的输入数据时，允许违反安全策略。

(12) 抵赖：这是一种来自用户的攻击，例如，否认自己曾经发布过的某条消息、伪造一份对方来信等。

(13) 重放：所截获的某次合法的通信数据备份，出于非法的目的而被重新发送。

(14) 计算机病毒：所谓计算机病毒，是一种在计算机系统运行过程中能够实现传染和侵害的功能程序。一种病毒通常含有两个功能：一种功能是对其他程序产生“感染”；另外一种或者是引发损坏功能或者是一种植入攻击的能力。

(15) 人员渎职：一个授权的人为了钱或利益、或由于粗心，将信息泄露给一个非授权的人。

(16) 媒体废弃：信息被从废弃的磁盘或打印过的存储介质中获得。

(17) 物理侵入：侵入者通过绕过物理控制而获得对系统的访问。

(18) 窃取：重要的安全物品，如令牌或身份卡被盗。

(19) 业务欺骗：某一伪系统或系统部件欺骗合法的用户或系统自愿地放弃敏感信息。

安全架构是架构面向安全性方向上的一种细分，通常的产品安全架构、安全技术体系架构和审计架构可组成三道安全防线。

(1) 产品安全架构：构建产品安全质量属性的主要组成部分以及它们之间的关系。产品安全架构的目标是如何在不依赖外部防御系统的情况下，从源头打造自身安全的产品。

(2) 安全技术体系架构：构建安全技术体系的主要组成部分以及它们之间的关系。安全技术体系架构的任务是构建通用的安全技术基础设施，包括安全基础设施、安全工具和技术、安全组件与支持系统等，系统性地增强各产品的安全防御能力。

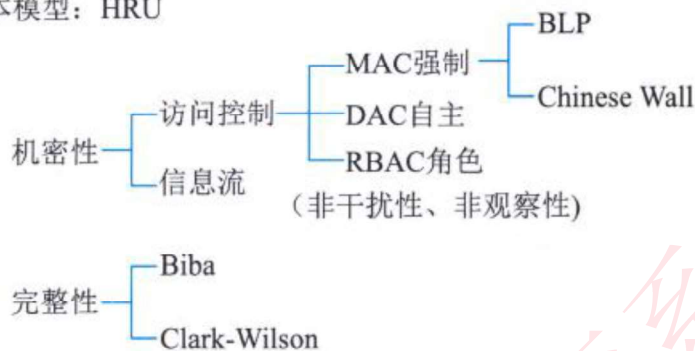
(3) 审计架构：独立的审计部门或其所能提供的风险发现能力，审计的范围主要包括安全风险在内的所有风险。

7.2 安全模型

信息系统的安全目标是控制和管理主体(含用户和进程)对客体(含数据和程序)的访问。

安全模型是准确地描述安全的重要方面及其与系统行为的关系，安全策略是从安全角度为系统整体和构成它的组件提出基本的目标。安全模型提供了实现目标应该做什么，不应该做什么，具有实践指导意义，它给出了策略的形式。如下图是模型的分类：

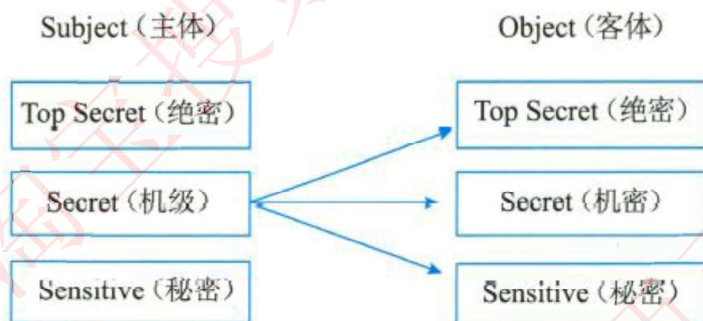
基本模型：HRU



状态机模型描述了一种无论处于何种状态都是安全的系统。它是用状态语言将安全系统描述成抽象的状态机，用状态变量表述系统的状态，用转换规则描述变量变化的过程。

状态机模型中一个状态是处于系统在特定时刻的一个快照。如果该状态所有方面满足安全策略的要求，则称此状态是安全的；一个安全状态模型系统，总是从一个安全状态启动，并且在所有迁移中保持安全状态，只允许主体以和安全策略相一致的安全方式访问资源。

Bell-LaPadula 模型使用主体、客体、访问操作(读、写、读/写)以及安全级别这些概念，当主体和客体位于不同的安全级别时，主体对客体就存在一定的访问限制。通过该模型可保证信息不被不安全主体访问。

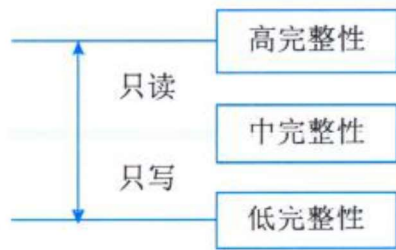


Bell-LaPadula 模型的安全规则如下：

- (1) 简单安全规则：安全级别低的主体不能读安全级别高的客体(No Read Up)；只能下读。
- (2) 星属性安全规则：安全级别高的主体不能往低级别的客体写(No Write Down)；只能上写。
- (3) 强星属性安全规则：不允许对另一级别进行读写；
- (4) 自主安全规则：使用访问控制矩阵来定义说明自由存取控制。其存取控制体现在内容相关和上下文相关。

Biba 模型不关心信息机密性的安全级别，因此它的访问控制不是建立在安全级别上，而是建立在完整性级别上。

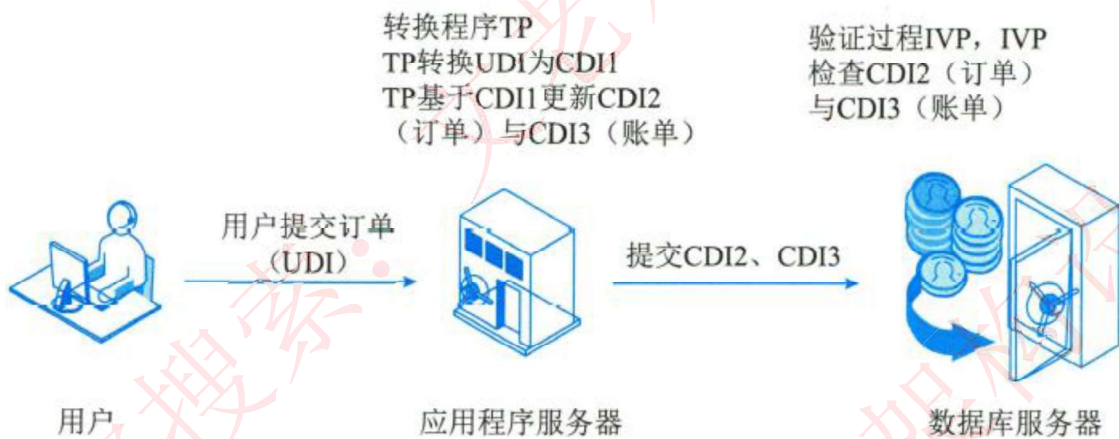
完整性的三个目标：保护数据不被未授权用户更改；保护数据不被授权用户越权修改(未授权更改)；维持数据内部和外部的一致性。



Biba 模型能够防止数据从低完整性级别流向高完整性级别，其安全规则如下：

- (1) 星完整性规则：表示完整性级别低的主体不能对完整性级别高的客体写数据；只能下写。
- (2) 简单完整性规则：表示完整性级别高的主体不能从完整性级别低的客体读取数据；只能上读。
- (3) 调用属性规则：表示一个完整性级别低的主体不能从级别高的客体调用程序或服务。

Clark-Wilson 模型是一种将完整性目标、策略和机制融为一体的模型。为了体现用户完整性，CWM 提出了职责隔离目标；为了保证数据完整性，CWM 提出了应用相关的完整性验证进程；为了建立过程完整性，CWM 定义了对于变换过程的应用相关验证。



- (1) 需要进行完整性保护的客体称之为 CDI，不需要进行完整性保护的客体称之为 UDI；
- (2) 完整性验证过程 (IVP)：确认限制数据项处于一种有效状态，如果 IVP 检验 CDI 符合完整性约束，则系统处于一个有效状态；
- (3) 转换过程 (TP)：将数据项从一种有效状态改变至另一种有效状态；
- (4) 为了确保对 CDI 的 TP 是有效的，则需要授权 User 做 TP 的认证；
- (5) 为了防止合法用户对 CDI 做非法或错误操作，将 TP 过程分为多个子过程，将每个子过程授权给不同的 User；
- (6) 但是如果 TP 的每个子过程被授权的 User 之间存在某种利益同盟，则可能存在欺骗。从而使 CDI 的完整性得不到保护。

CWM 的主要特征是：

- (1) 采用 Subject/Program/Object 三元素的组成方式。Subject 要访问 Object 只能通过 Program 进行；
- (2) 权限分离原则：将要害功能分为有 2 个或多个 Subject 完成，防止已授权用户进行未授权的修改；
- (3) 要求具有审计能力。

Chinese Wall 模型 (又名 Brew and Nash 模型) 是应用在多边安全系统中的安全模型。也就是说，是指通过行政规定和划分、内部监控、IT 系统等手段防止各部门之间出现有损客户利益的利益冲突事件。

Chinese Wall 模型的安全策略的基础是客户访问的信息不会与当前他们可支配的信息产生冲突。在投资银行中，一个银行会同时拥有多个互为竞争者的客户，一个银行家可能为一个客户工作，但他可以访问所有客户的信息。因此，应当制止该银行家访问其他客户的数据。银行家可以选择为谁工作 (DAC)，一旦选定，他就只能为该客户工作 (MAC)。

Chinese Wall 模型的访问客体控制的安全规则如下：

- (1) 与主体曾经访问过的信息属于同一公司数据集合的信息，即墙内信息可以访问；
- (2) 属于一个完全不同的利益冲突组的可以访问；
- (3) 主体能够对一个客体进行写的前提是主体未对任何属于其他公司数据集进行过访问。

定理 1：一个主体一旦访问过一个客体，则该主体只能访问位于同一公司数据集的客体或在不同利益组的客体。

定理 2：在一个利益冲突组中，一个主体最多只能访问一个公司数据集。

比如，假设 Chinese Wall 安全策略包括三个信息存储模块：某家企业的单位信息 (C)、该家企业的所有信息集合 (CD) 和该家企业与互为竞争关系企业的全部信息集合 (COI)。那么，Chinese Wall 模型规定：

- (1) 每个 C 只能唯一对应一个 CD；
- (2) 每个 CD 只能唯一对应一个利益冲突类 COI；
- (3) 一个 COI 类却可以同时包含多个 CD。

7.3 系统安全体系架构规划框架

安全技术体系架构 是对组织机构信息技术系统的安全体系结构的整体描述。安全技术体系架构的目标是建立可持续改进的安全技术体系架构的能力。

根据网络中风险威胁的存在实体划分出 5 个层次的实体对象：应用、存储、主机、网络和物理。

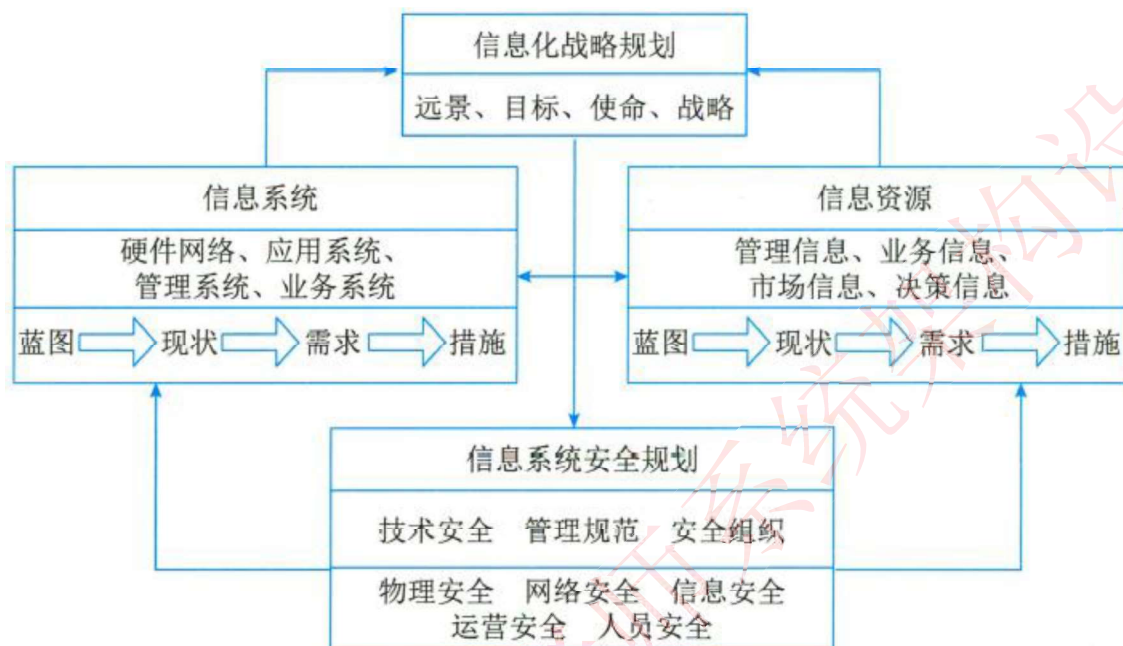
信息系统安全体系 主要是由技术体系、组织机构体系和管理体系三部分共同构成的。

技术体系是全面提供信息系统安全保护的技术保障系统，该体系由物理安全技术和系统安全技术两大类构成。

组织体系是信息系统的组织保障系统，由机构、岗位和人事三个模块构成。

管理体系由法律管理、制度管理和培训管理三部分组成。

信息系统安全规划框架



1. 信息系统安全规划依托企业信息化战略规划

信息系统安全规划依托企业信息化战略规划，对信息化战略的实施起到保驾护航的作用。信息系统安全规划的目标应该与企业信息化的目标是一致的，而且应该比企业信息化的目标更具体明确、更贴近安全。

2. 信息系统安全规划需要围绕技术安全、管理安全、组织安全考虑

规划的内容基本上应涵盖：确定信息系统安全的任务、目标、战略以及战略部门和战略人员，并在此基础上制定出物理安全、网络安全、系统安全、运营安全、人员安全的信息系统安全的总体规划。

3. 信息系统安全规划以信息系统与信息资源的安全保护为核心

规划工作需要围绕着信息系统与信息资源的开发、利用和保护工作进行，要包括蓝图、现状、需求和措施 4 个方面。

(1) 对信息系统与信息资源的规划需要从信息化建设的蓝图入手，知道企业信息化发展策略的总体目标和各阶段的实施目标，制定出信息系统安全的发展目标。

(2) 对企业的信息化工作现状进行整体的、综合、全面的分析，找出过去工作中的优势与不足。

(3) 根据信息化建设的目标提出未来几年的需求，这个需求最好可以分解成若干个小的方面，以便于今后的实施与落实。

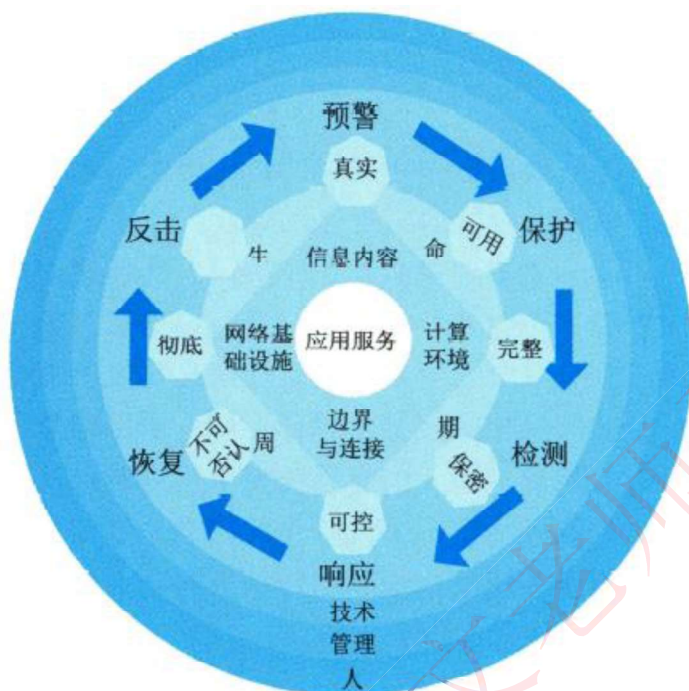
(4) 要明确在实施工作阶段的具体措施与方法，提高规划工作的执行力度。

7.4 信息安全整体架构设计

WPDRRC (Warning/Protect/Detect/React/Restore/Counterattack) 模型有 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，它们具有较强的时序性和动态性，能够较好地反映出信息系统安全保障体系的预警能力、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。人员是核心，策略是桥梁，技术是保证，落实在 WPDRRC 的 6 个环节的各个方面，将安全策预警略变为安全现实。



● W：预警主要是指利用远程安全评估系统提供的模拟攻击技术来检查系统存在的、可能被利用的薄弱环节，收集和测试网络与信息的安全风险所在，并以直观的方式进行报告，提供解决方案的建议，在经过分析后，分解网络的风险变化趋势和严重风险点，从而有效降低网络的总体风险，保护关键业务和数据。

● P：防护通常是通过采用成熟的信息安全技术及方法来实现网络与信息的安全。主要内容有加密机制，数字签名机制，访问控制机制，认证机制，信息隐藏和防火墙技术等。

● D：检测通过检测和监控网络以及系统，来发现新的威胁和弱点，强制执行安全策略。在这个过程中采用入侵检测、恶意代码过滤等技术，形成动态检测的制度，奖励报告协调机制，提高检测的实时性。主要内容有入侵检测，系统脆弱性检测，数据完整性检测和攻击性检测等。

● R：响应是指在检测到安全漏洞和安全事件之后必须及时做出正确的响应，从而把系统调整到安全状态。为此需要相应的报警、跟踪、处理系统，其中处理包括了封堵、隔离、报告等能力。主要内容有应急策略、应急机制、应急手段、入侵过程分析和安全状态评估等。

● R：恢复灾难恢复系统是当前网络、数据、服务受到黑客攻击并遭到破坏或影响后，通过必要技术手段，在尽可能短的时间内使系统恢复正常。主要内容有容错、冗余、备份、替换、修复和恢复等。

● C：反击是指采用一切可能的高新技术手段，侦察、提取计算机犯罪分子的作案线索与犯罪证据，形成强有力的取证能力和依法打击手段。

信息系统安全设计重点考虑两个方面：其一是系统安全保障体系；其二是信息安全体系架构。

1. 系统安全保障体系：是由安全服务、协议层次和系统单元等三个层面组成，且每个层都涵盖了安全管理的内容。系统安全保障体系设计工作主要考虑以下几点：

(1) 安全区域策略的确定：根据安全区域的划分，主管部门应制定针对性的安全策略。如定时审计评估、安装入侵检测系统、统一授权、认证等；

(2) 统一配置和管理防病毒系统：主管部门应当建立整体防御策略，以实现统一的配置和管理。网络防病毒的策略应满足全面性、易用性、实时性和可扩展性等方面要求；

(3) 网络安全管理：在网络安全中，除了采用一些技术措施之外，加强网络安全管理，制定有关规章制度。

2. 信息安全体系架构：具体在安全控制系统，我们可以从物理安全、系统安全、网络安全、应用安全和管理安全等 5 个方面开展分析和设计工作。

1) 物理安全：保证计算机信息系统各种设备的物理安全是保障整个网络系统安全的前提。包括：环境安全、设备安全、媒体安全等。

2) 系统安全：主要是指对信息系统组成中各个部件的安全要求。系统安全是系统整体安全的基础。它主要包括：网络结构安全、操作系统安全和应用系统安全。

3) 网络安全：是整个安全解决方案的关键。它主要包括：访问控制、通信保密、入侵检测、网络安全扫描系统和防病毒等。

4) 应用安全：主要是指多个用户使用网络系统时，对共享资源和信息存储操作所带来的安全问题。它主要包括资源共享和信息存储两个方面。

5) 安全管理：主要体现在三个方面。其一是制定健全的安全管理体制；其二是构建安全管理平台；其三是增强人员的安全防范意识。

一种面向企业的安全控制系统安全架构：



7.5 网络安全体系架构设计

OSI 定义了 7 层协议，其中除第 5 层(会话层)外，每一层均能提供相应的安全服务。实际上，最适合配置安全服务的是在物理层、网络层、运输层及应用层上，其他层都不宜配置安全服务。

OSI 开放系统互联安全体系的 5 类安全服务包括鉴别、访问控制、数据机密性、数据完整性和抗抵赖性。

OSI 定义分层多点安全技术体系架构，也称为深度防御安全技术体系架构，它通过以下三种方式将防御能力分布至整个信息系统中。

1) 多点技术防御：在对手可以从内部或外部多点攻击一个目标的前提下，多点技术防御通过对网络和基础设施、边界、计算环境这三个防御核心区域的防御达到抵御所有方式的攻击目的。

2) 分层技术防御：即使最好的可得到的信息保障产品也有弱点，其最终结果将使对手能找到一个可探查的脆弱性，一个有效的措施是在对手和目标间使用多个防御机制。

3) 支撑性基础设施：为网络、边界和计算环境中信息保障机制运行基础的支撑性基础设施，包括公钥基础设施以及检测和响应基础设施。

鉴别(Authentication) 的基本目的是防止其他实体占用和独立操作被鉴别实体的身份。鉴别有两种重要的关系背景: 一是实体由申请者来代表, 申请者与验证者之间存在着特定的通信关系(如实体鉴别); 二是实体为验证者提供数据项来源。

鉴别服务分为以下阶段:

在安装阶段, 定义申请鉴别信息和验证鉴别信息。

修改鉴别信息阶段, 实体或管理者申请鉴别信息和验证鉴别信息变更(如修改口令)。

在分发阶段, 为了验证交换鉴别信息, 把验证鉴别信息分发到各实体(如申请者或验证者)以供使用。

在获取阶段, 申请者或验证者可得到为鉴别实例生成特定交换鉴别信息所需的信息, 通过与可信第三方进行交互或鉴别实体间的信息交换可得到交换鉴别信息。

在传送阶段, 在申请者与验证者之间传送交换鉴别信息。

在验证阶段, 用验证鉴别信息核对交换鉴别信息。

在停活阶段, 将建立一种状态, 使得以前能被鉴别的实体暂时不能被鉴别。

在重新激活阶段, 使在停活阶段建立的状态将被终止。

在取消安装阶段, 实体从实体集合中被拆除。

访问控制(Access Control) 决定开放系统环境中允许使用哪些资源、在什么地方适合阻止未授权访问的过程。

ACI (访问控制信息)是用于访问控制目的的任何信息, 其中包括上下文信息。A D I (访问控制判决信息)是在做出一个特定的访问控制判决时可供 A D F 使用的部分(或全部) ACI。

A D F (访问控制判决功能)是一种特定功能, 它通过对访问请求、A D I 以及该访问请求的上下文使用访问控制策略规则而做出访问控制判决。A E F (访问控制实施功能)确保只有对目标允许的访问才由发起者执行。

涉及访问控制的有发起者、A E F 、A D F 和目标。



图 18-19 基本访问控制功能示意图

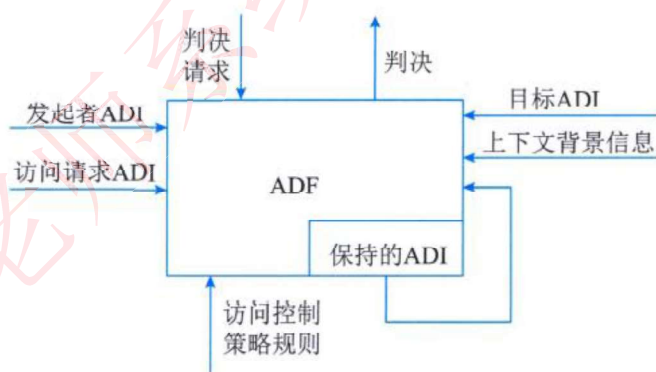


图 18-20 ADF 示意图

数据的机密性可以依赖于所驻留和传输的媒体。因此, 存储数据的机密性能通过使用隐藏数据语义(如加密)或将数据分片的机制来保证。数据在传输中的机密性能通过禁止访问的机制、通过隐藏数据语义的机制或通过分散数据的机制得以保证(如跳频等)。这些机制类型能被单独使用或者组合使用。具体机制包括: 通过禁止访问提供机密性、通过加密提供机密性、通过数据填充、虚假事件等。

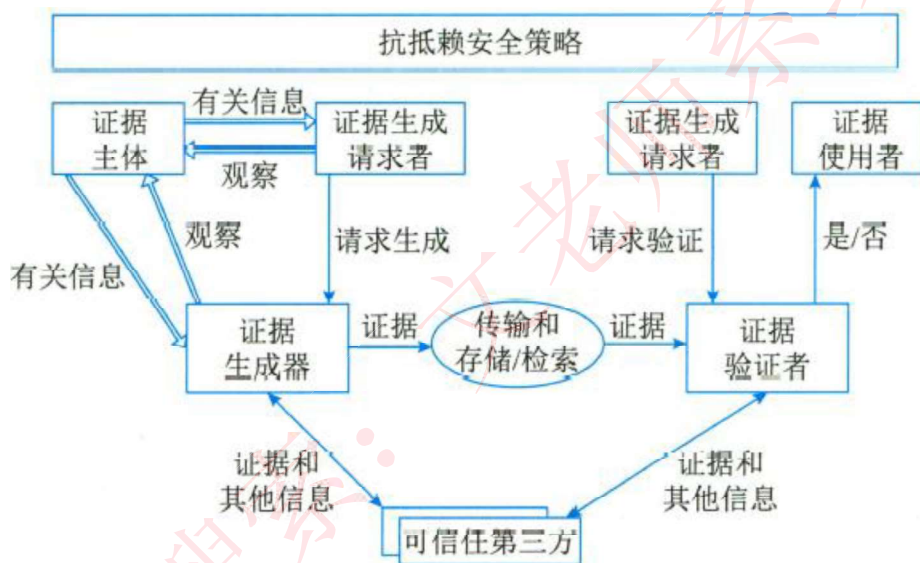
完整性框架。对于不同的媒体，数据完整性保护机制是有区别的，可概括为以下两种情况。

(1)阻止对媒体访问的机制。包括物理隔离的不受干扰的信道、路由控制、访问控制。

(2)用以探测对数据或数据项序列的非授权修改的机制。未授权修改包括未授权数据创建、数据删除以及数据重复。而相应的完整性机制包括密封、数字签名、数据重复(作为对抗其他类型违规的手段)、与密码变换相结合的数字指纹和消息序列号。

抗抵赖服务包括证据的生成、验证和记录，以及在解决纠纷时随即进行的证据恢复和再次验证。框架所描述的抗抵赖服务的目的是提供有关特定事件或行为的证据。

抗抵赖由 4 个独立的阶段组成：证据生成；证据传输、存储及恢复；证据验证和解决纠纷。如图所示。



7.6 数据库系统的安全设计

按照 TCSEC 标准，D 类产品是基本没有安全保护措施的产品，C 类产品只提供了安全保护措施，一般不称为安全产品。B 类以上产品是实行强制存取控制的产品，也是真正意义上的安全产品。所谓安全产品均是指安全级别在 B1 以上的产品，而安全数据库研究原型一般是指安全级别在 B1 级以上的以科研为目的，尚未产品化的数据库管理系统原型。

数据库完整性是指数据库中数据的正确性和相容性。数据库完整性由各种各样的完整性约束来保证，因此可以说数据库完整性设计就是数据库完整性约束的设计。数据库完整性约束可以通过 DBMS 或应用程序来实现，基于 DBMS 的完整性约束作为模式的一部分存入数据库中。

在实施数据库完整性设计时，需要把握以下基本原则：

(1)根据数据库完整性约束的类型确定其实现的系统层次和方式，并提前考虑对系统性能的影响。一般情况下，静态约束应尽量包含在数据库模式中，而动态约束由应用程序实现。

(2)实体完整性约束、引用完整性约束是关系数据库最重要的完整性约束，在不影响系统关键性能的前提下需尽量应用。用一定的时间和空间来换取系统的易用性是值得的。

(3)要慎用目前主流 D B M S 都支持的触发器功能，一方面由于触发器的性能开销较大；另一方面，触发器的多级触发难以控制，容易发生错误，非用不可时，最好使用 Before 型语句级触发器。

(4)在需求分析阶段就必须制定完整性约束的命名规范，尽量使用有意义的英文单词、缩写词、表名、列名及下画线等组合，使其易于识别和记忆。

(5)要根据业务规则对数据库完整性进行细致的测试，以尽早排除隐含的完整性约束间的冲突和对性能的影响。

(6)要有专职的数据库设计小组，自始至终负责数据库的分析、设计、测试、实施及早期维护。

(7)应采用合适的 C A S E 工具来降低数据库设计各阶段的工作量。

数据库完整性的作用：

(1)数据库完整性约束能够防止合法用户使用数据库时向数据库中添加不合语义的数据。

(2)利用基于 D B M S 的完整性控制机制来实现业务规则，易于定义，容易理解，而且可以降低应用程序的复杂性，提高应用程序的运行效率。

(3)合理的数据库完整性设计，能够同时兼顾数据库的完整性和系统的效能。

(4)在应用软件的功能测试中，完善的数据库完整性有助于尽早发现应用软件的错误。

(5)数据库完整性约束可分为 6 类：列级静态约束、元组级静态约束、关系级静态约束、列级动态约束、元组级动态约束和关系级动态约束。

一个好的数据库完整性设计，首先需要在需求分析阶段确定要通过数据库完整性约束实现的业务规则。然后在充分了解特定 DBMS 提供的完整性控制机制的基础上，依据整个系统的体系结构和性能要求，遵照数据库设计方法和应用软件设计方法，合理选择每个业务规则的实现方式。最后，认真测试，排除隐含的约束冲突和性能问题。

7.7 系统架构的脆弱性分析

脆弱性分析主要是分析信息系统中产生脆弱性的根源、脆弱性可能造成的影响、如何利用脆弱性进行攻击、如何修补脆弱性、如何防止脆弱性被利用、如何探测目标系统的脆弱性、如何预测新的脆弱性的存在等一系列问题。

从技术角度而言，漏洞的来源主要有以下几个方面：软件设计时的瑕疵、软件实现中的弱点、软件本身的瑕疵、系统和网络的错误配置。

软件脆弱性有其自身的特点，主要包括 4 个方面：

(1)脆弱性是软件系统中隐藏的一个弱点，本身不会引起危害，但被利用后会产生严重的安全后果；

(2)在软件开发过程中，自觉或不自觉引入的逻辑错误是大多数脆弱性的根本来源；

(3)与具体的系统环境密切相关，系统环境的任何差异都有可能导致不同的脆弱性问题；

(4)旧的脆弱性得到修补或纠正的同时可能引入新的脆弱性，因此脆弱性问题会长期存在。

软件脆弱性的生命周期

脆弱性的引入阶段：引入软件脆弱性的原因有：

- (1) 输入验证错误;
- (2) 权限检查错误;
- (3) 操作序列化错误;
- (4) 边界检出错误;
- (5) 软件设计时的缺陷;
- (6) 其他错误。

产生破坏效果阶段：主要包括：

- (1) 非法执行代码;
- (2) 非法修改目标对象;
- (3) 访问数据对象;
- (4) 拒绝服务攻击。

修补阶段：主要包括：

- (1) 删除伪造实体(如 IP 伪造、名字伪造等);
- (2) 增加新的实体;
- (3) 写该实体不正确的位置;
- (4) 其他情况。

软件脆弱性分析可从三个方面考虑：

- (1) 分析软件故障现象，分析故障的技术本质、总结脆弱性模式;
- (2) 分析软件开发，发现安全管理和技术的薄弱环节，提高软件安全性;
- (3) 分析软件使用，发现其脆弱性，采取相应措施，避免脆弱性转化为安全故障。

软件脆弱性分析首先要明确分析对象，脆弱性分析对象可以分为两类：脆弱性数据和软件系统。

由于软件本身具有自身的性质和特点，针对软件的脆弱性分析，我们也需要考虑软件本身的各种特点。主要考虑软件结构和实现技术两个方面。

典型软件架构的脆弱性分析

1. 分层架构的脆弱性主要表现在两个方面：

- (1) 层间的脆弱性。一旦某个底层发生错误，那么整个程序将会无法正常运行。
- (2) 层间通信的脆弱性。将系统隔离为多个相对独立的层，这就要求在层与层之间引入通信机制。

本来“直来直去”的操作现在要层层传递，势必造成性能下降。

2. C/S 架构的脆弱性主要表现在以下几个方面：

(1) 客户端软件的脆弱性。因为在用户计算机上安装了客户端软件，所以这个系统就面临着程序被分析、数据被截取的安全隐患。

(2) 网络开放性的脆弱性。目前很多传统的 C/S 系统还是采用二层结构，也就是说所有客户端直接读取服务器端中的数据，在客户端包括了数据的用户名，密码等致命的信息，这样会给系统带来安全隐患。

(3)网络协议的脆弱性。C/S 架构不便于随时与用户交流(主要是不便于数据包共享),并且 C/S 架构软件在保护数据的安全性方面有着先天的弊端。由于 C/S 架构软件的数据分布特性,客户端所发生的火灾、盗抢、地震、病毒等都将作为可怕的数据杀手。

3. B/S 架构的脆弱性主要表现在:系统如果使用 HTTP 协议, B/S 架构相对 C/S 架构而言更容易被病毒入侵,虽然最新的 HTTP 协议在安全性方面有所提升,但还是弱于 C/S。

4. 事件驱动架构的脆弱性主要表现在:

(1)组件的脆弱性。组件削弱了自身对系统的控制能力,一个组件触发事件,并不能确定响应该事件的其他组件及各组建的执行顺序。

(2)组件间交换数据的脆弱性。组件不能很好地解决数据交换问题,事件触发时,一个组件有可能需要将参数传递给另一个组件,而数据量很大的时候,如何有效传递是一个脆弱性问题。

(3)组件间逻辑关系的脆弱性。事件架构使系统中各组件的逻辑关系变得更加复杂。

(4)事件驱动容易进入死循环,这是由编程逻辑决定的。

(5)高并发的脆弱性。虽然事件驱动可实现有效利用 CPU 资源,但是存在高并发事件处理造成的系统响应问题,而且,高并发容易导致系统数据不正确、丢失数据等现象。

(6)固定流程的脆弱性。因为事件驱动的可响应流程基本都是固定的,如果操作不当,容易引发安全问题。

5. MVC 架构的脆弱性主要表现在:

(1) MVC 架构的复杂性带来脆弱性。MVC 架构增加了系统结构和实现的复杂性。比如说一个简单的界面,如果严格遵循 MVC 方式,使得模型、视图与控制器分离,会增加结构的复杂性,并可能产生过多的更新操作,降低运行效率。

(2)视图与控制器间紧密连接的脆弱性。视图与控制器是相互分离但确是联系紧密的部件,没有控制器的存在,视图应用是很有限的。反之亦然,这样就妨碍了它们的独立重用。

(3)视图对模型数据的低效率访问的脆弱性。依据模型操作接口的不同,视图可能需要多次调用才能获得足够的显示数据。对未变化数据的不必要的频繁访问也将损害操作性能。

6. 微内核架构的脆弱性主要表现在:

(1)微内核架构难以进行良好的整体化优化。由于微内核系统的核心态只实现了最基本的系统操作,这样内核以外的外部程序之间的独立运行使得系统难以进行良好的整体优化。

(2)微内核系统的进程间通信开销也较单一内核系统要大得多。从整体上看,在当前硬件条件下,微内核在效率上的损失小于其在结构上获得的收益。

(3)通信损失率高。微内核把系统分为各个小的功能块,从而降低了设计难度,系统的维护与修改也容易,但通信带来的效率损失是一个问题。

7. 微服务架构的脆弱性主要表现在:

(1)开发人员需要处理分布式系统的复杂结构。

(2)开发人员要设计服务之间的通信机制,通过写代码来处理消息传递中速度过慢或者不可用等局部实效问题。

(3)服务管理的复杂性，在生产环境中要管理多个不同的服务实例，这意味着开发团队需要全局统筹。

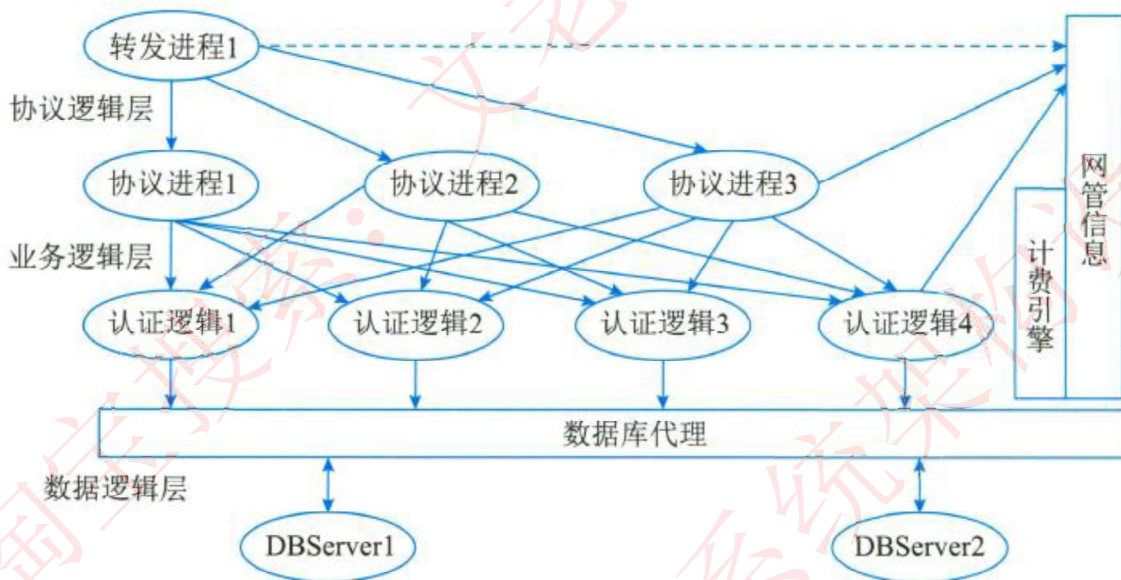
7.8 安全架构设计案例分析

电子商务系统的安全性设计

认证、授权和审计(AAA)是运行于宽带网络接入服务器上的客户端程序。AAA提供了一个用来对认证、授权和审计三种安全功能进行配置的一致框架，实际上是对网络安全的一种管理。

RADIUS服务器负责接收用户的连接请求，完成验证并把用户所需的配置信息返回给BAS建立连接，从而可以获得访问其他网络的权限时，BAS就起到了认证用户的作用。BAS负责把用户之间的验证信息传递通过密钥的参与来完成。用户的密码加密以后才能在网上传递，以避免用户的密码在不安全的网络上被窃取。

高性能的RADIUS软件架构核心如图所示。



RADIUS软件架构分为三个层面：协议逻辑层、业务逻辑层和数据逻辑层。

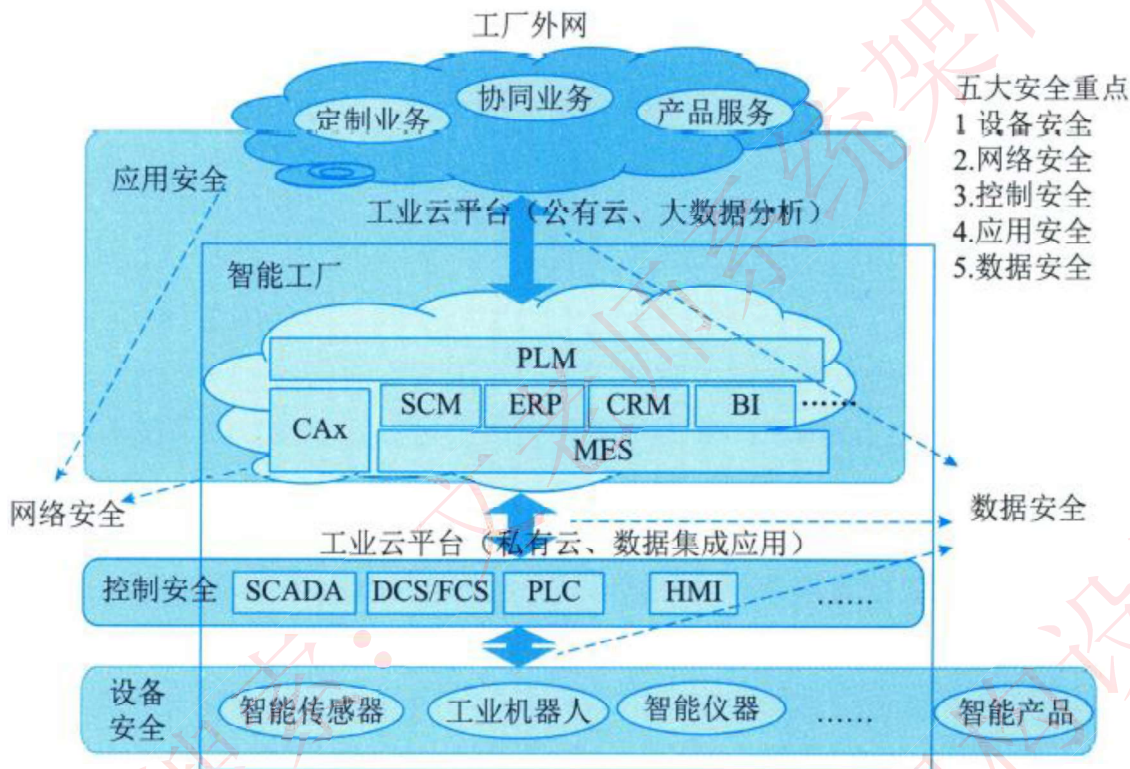
协议逻辑层主要实现RFC框架中的内容，处理网络通信协议的建立、通信和停止方面的工作。在软件功能上，这个部分主要相当于一个转发引擎，起到分发处理的内容分发到不同的协议处理过程中，这一层的功能起到了协议与业务处理的分层处理的作用。

业务逻辑层的设计是RADIUS软件架构设计的核心部分，协议处理进程主要是对转发引擎发来的包进行初步分析，并根据包的内容进一步分发到不同的业务逻辑处理进程。业务逻辑进程分为认证、计费和授权三种类型，不同的业务逻辑进程可以接收不同协议进程之间的信息并进行处理。转发进程与协议进程之间采用共享内存的方法，实现进程之间的通信。协议进程与业务逻辑处理进程之间采用进程加线程的实现方法。

数据逻辑层需要对来自业务逻辑处理线程统一管理并处理数据库代理池的数据，由数据库代理池统一连接数据库，以减少对数据库系统的压力。

基于混合云的工业安全架构设计

下图给出了大型企业采用混合云技术的安全生产管理系统的架构，企业由多个跨区域的智能工厂和公司总部组成，公司总部负责相关业务的管理、协调和统计分析，而每个智能工厂负责智能产品的设计与生产制造。智能工厂内部采用私有云实现产品设计、数据共享和生产集成等，公司总部与智能工厂间采用公有云实现智能工厂间、智能工厂与公司总部间的业务管理、协调和统计分析等。



整个安全生产管理系统架构由四层组成，设备层、控制层、设计管理层和应用层。设备层主要是指用于智能工厂生产产品所需的相关设备；

控制层主要是指智能工厂生产产品所需要建立的一套自动控制系统，控制智能设备完成生产工作；

设计/管理层是指智能工厂各种开发、业务控制和数据管理功能的集合，实现数据集成与应用；

应用层主要是指在云计算平台上进行信息处理，主要涵盖两个核心功能，一是“数据”，应用层需要完成数据的管理和数据的处理，二是“应用”，仅仅管理和处理数据还远远不够，必须将这些数据与行业应用相结合，本系统主要包括定制业务、协同业务和产品服务等。

在设计基于混合云的安全生产管理系统中，需要重点考虑 5 个方面的安全问题。设备安全、网络安全、控制安全、应用安全和数据安全。

8 大数据架构设计

8.1 大数据处理系统架构分析

大数据带来的三大挑战：

1. 如何利用信息技术等手段处理非结构化和半结构化数据

2. 如何探索大数据复杂性、不确定性特征描述的刻画方法及大数据的系统建模
3. 数据异构性与决策异构性的关系对大数据知识发现与管理决策的影响

大数据处理系统架构特征：

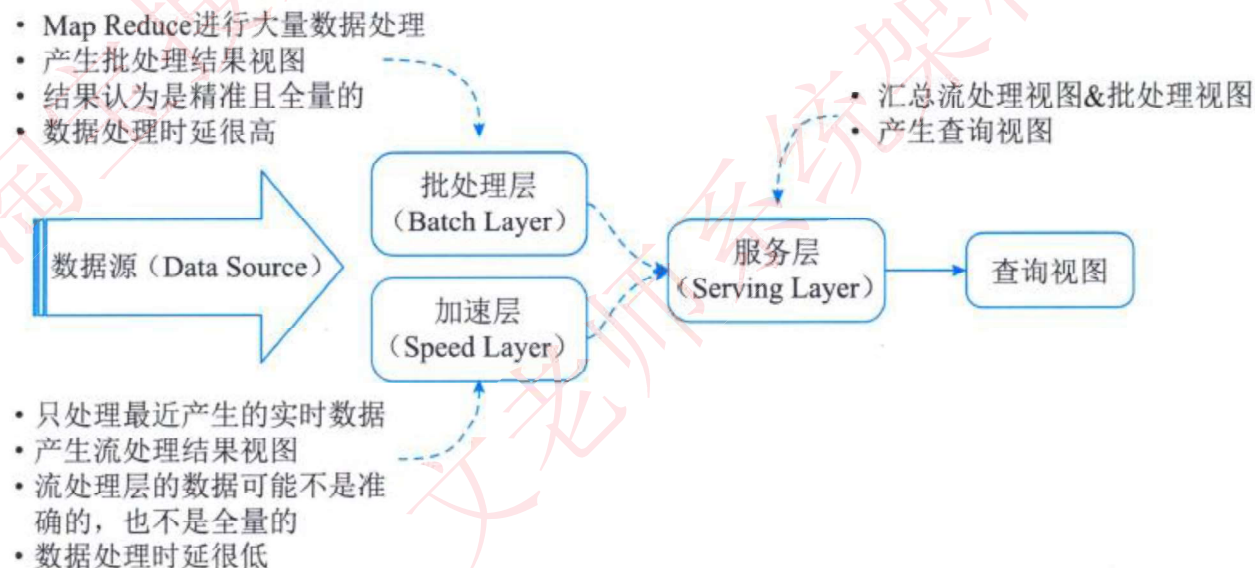
1. 鲁棒性和容错性
2. 低延迟读取和更新能力
3. 横向扩容
4. 通用性
5. 延展性
6. 即席查询能力
7. 最少维护能力
8. 可调试性

8.2 Lambda 架构

Lambda 架构设计目的在于提供一个能满足大数据系统关键特性的架构，包括高容错、低延迟、可扩展等。其整合离线计算与实时计算，融合不可变性、读写分离和复杂性隔离等原则。Lambda 是用于同时处理离线和实时数据的，可容错的，可扩展的分布式系统。它具备强鲁棒性，提供低延迟和持续更新。

Lambda 架构应用场景：机器学习、物联网、流处理。

如图所示，Lambda 架构可分解为三层，即批处理层、加速层和服务层。

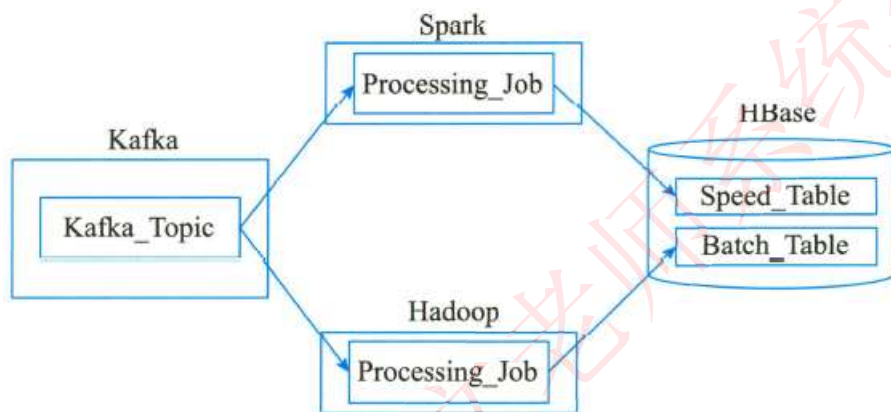


(1) 批处理层 (Batch Layer)：存储数据集，Batch Layer 在数据集上预先计算查询函数，并构建查询所对应的 View。Batch Layer 可以很好地处理离线数据，但有很多场景数据不断实时生成，并且需要实时查询处理。Speed Layer 正是用来处理增量的实时数据。

(2)加速层(Speed Layer):Batch Layer 处理的是全体数据集,而 Speed Layer 处理的是最近的增量数据流。Speed Layer 为了效率,在接收到新的数据后会不断更新 Real-time View,而 Batch Layer 是根据全体离线数据集直接得到 Batch View。

(3)服务层(Serving Layer):Serving Layer 用于合并 Batch View 和 Real-time View 中的结果数据集到最终数据集。用于响应用户的查询请求。

如图所示,在这种 Lambda 架构实现中,Hadoop(HDFS)用于存储主数据集,Spark(或 Storm)可构成速度层(Speed Layer),HBase(或 Cassandra)作为服务层,由 Hive 创建可查询的视图。



Hadoop 是被设计成适合运行在通用硬件上的分布式文件系统。HDFS 是一个具有高度容错性的系统,能提供高吞吐量的数据访问,非常适合大规模数据集上的应用。HDFS 放宽了一些约束,以达到流式读取文件系统数据的目的。

Apache Spark 是专为大规模数据处理而设计的快速通用的计算引擎。Spark 中间输出结果可以保存在内存中,从而不再需要读写 HDFS,因此 Spark 能更好地适用于数据挖掘与机器学习等需要迭代的 Map Reduce 算法。

HBase-Hadoop Database,是一个高可靠性、高性能、面向列、可伸缩的分布式存储系统,利用 HBase 技术可在廉价 PCServer 上搭建起大规模结构化存储集群。

Lambda 架构的优点:容错性好、查询灵活度高、易伸缩、易扩展。

缺点:全场景覆盖带来的编码开销。针对具体场景重新离线训练一遍益处不大。重新部署和迁移成本很高。

事件溯源(Event Sourcing)与 Lambda 架构

Event Sourcing 本质上是一种数据持久化的方式,其由三个核心观点构成:

- (1)整个系统以事件为驱动,所有业务都由事件驱动来完成。
- (2)事件是核心,系统的数据以事件为基础,事件要保存在某种存储上。
- (3)业务数据只是一些由事件产生的视图,不一定要保存到数据库中。

Lambda 架构中数据集的存储使用的概念与 Event Sourcing 中的思想完全一致,二者都是在使用统一的数据模型对数据处理事件本身进行定义。这样在发生错误的时候,能够通过模型找到错误发生的原因,对这一事件进行重新计算以丢弃错误信息,恢复到系统应该的正确状态,以此实现了系统的容错性。

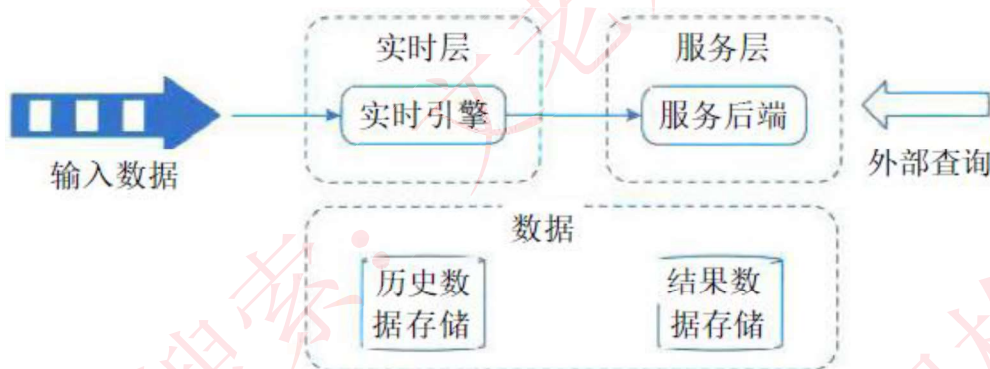
CQRS 与 Lambda 架构

CQRS 架构分离了对于数据进行的读操作(查询)和写(修改)操作。其将能够改变数据模型状态的命令和对于模型状态的查询操作实现了分离。这是领域驱动设计的一个架构模式，主要用来解决数据库报表的输出处理方式。

Lambda 架构中，数据的修改通过批处理和流处理实现，通过写操作将数据转换成查询时所对应的 View。在 Lambda 架构中，对数据进行查询时，实际上是通过读取 View 直接得到结果，读出所需的内容。这实际上是一种形式的读写分离。

8.3 Kappa 架构

Kappa 架构的原理就是：在 Lambda 的基础上进行了优化，删除了 Batch Layer 的架构，将数据通道以消息队列进行替代。因此对于 Kappa 架构来说，依旧以流处理为主，但是数据却在数据湖层面进行了存储，当需要进行离线分析或者再次计算的时候，则将数据湖的数据再次经过消息队列重播一次则可。



如图所示，输入数据直接由实时层的实时数据处理引擎对源源不断的源数据进行处理，再由服务层的服务后端进一步处理以提供上层的业务查询。而中间结果的数据都是需要存储的，这些数据包括历史数据与结果数据，统一存储在存储介质中。

从使用场景上来看，Kappa 架构与 Lambda 相比，主要有两点区别：

(1) Kappa 不是 Lambda 的替代架构，而是其简化版本，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；

(2) Lambda 直接支持批处理，因此更适合对历史数据分析查询的场景。

Kappa 架构的优点在于将实时和离线代码统一起来，方便维护而且统一了数据口径的问题，避免了 Lambda 架构中与离线数据合并的问题，查询历史数据的时候只需要重放存储的历史数据即可。

Kappa 的缺点：

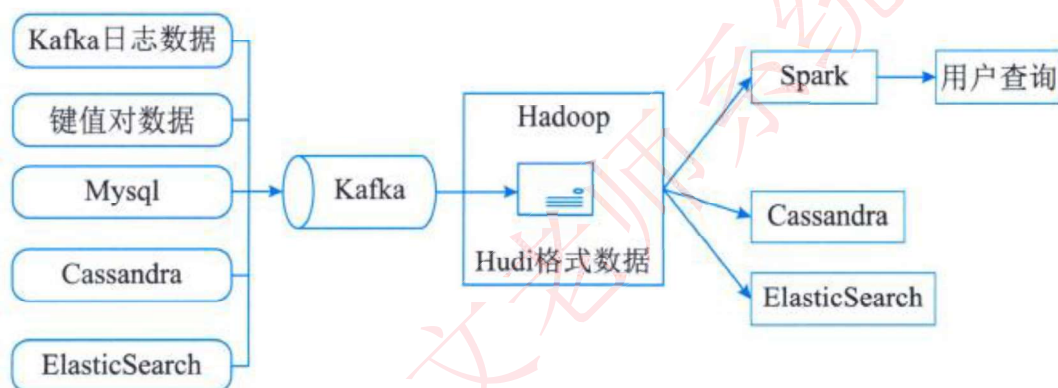
(1) 消息中间件缓存的数据量和回溯数据有性能瓶颈。通常算法需要过去 180 天的数据，如果都存在消息中间件，无疑有非常大的压力。同时，一次性回溯订正 180 天级别的数据，对实时计算的资源消耗也非常大。

(2) 在实时数据处理时，遇到大量不同的实时流进行关联时，非常依赖实时计算系统的能力，很可能因为数据流先后顺序问题，导致数据丢失。

(3)Kappa 在抛弃了离线数据处理模块的时候,同时抛弃了离线计算更加稳定可靠的特点。Lambda 虽然保证了离线计算的稳定性,但双系统的维护成本高且两套代码带来后期运维困难。

Kappa+是流式数据处理架构,它的核心思想是让流计算框架直接读 HDFS 里的数据仓库数据,一并实现实时计算和历史数据 backfl1 计算,不需要为 backfl1 作业长期保存日志或者把数据拷贝回消息队列。开发了 Apache hudi 框架来存储数据仓库数据, hudi 支持更新、删除已有 parquet 数据,也支持增量消费数据更新部分,从而系统性解决了问题 2 存储的问题。

如图所示,将不同来源的数据通过 Kafka 导入到 Hadoop 中,通过 HDFS 来存储中间数据,再通过 spark 对数据进行分析处理,最后交由上层业务进行查询。



混合分析系统的 Kappa 架构: Lambda 和 Kappa 架构都还有展示层的困难点,结果视图如何支持热点数据查询分析,一个解决方案是在 Kappa 基础上衍生数据分析流程。

在基于使用 Kafka +Flink 构建 Kappa 流计算数据架构,针对 Kappa 架构分析能力不足的问题,再利用 Kafka 对接组合 Elastic- Search 实时分析引擎,部分弥补其数据分析能力。但是 ElasticSearch 也只适合对合理数据量级的热点数据进行索引,无法覆盖所有批处理相关的分析需求,这种混合架构某种意义上属于 Kappa 和 Lambda 间的折中方案。

8.4 Lambda 架构与 Kappa 架构的对比和设计选择

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

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。而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。

1. 业务需求与技术要求：如果业务对于 Hadoop、Spark、Strom 等关键技术有强制性依赖，选择 Lambda 架构可能较为合适；如果处理数据偏好于流式计算，又依赖 Flink 计算引擎，那么选择 Kappa 架构可能更为合适。

2. 复杂度：如果项目中需要频繁地对算法模型参数进行修改，Lambda 架构需要反复修改两套代码，则显然不如 Kappa 架构简单方便。同时，如果算法模型支持同时执行批处理和流式计算，或者希望用一份代码进行数据处理，那么可以选择 Kappa 架构。

3. 开发维护成本：Lambda 架构需要有一定程度的开发维护成本，包括两套系统的开发、部署、测试、维护，适合有足够经济、技术和人力资源的开发人员。而 Kappa 架构只需要维护一套系统，适合不希望在开发维护上投入过多成本的开发人员。

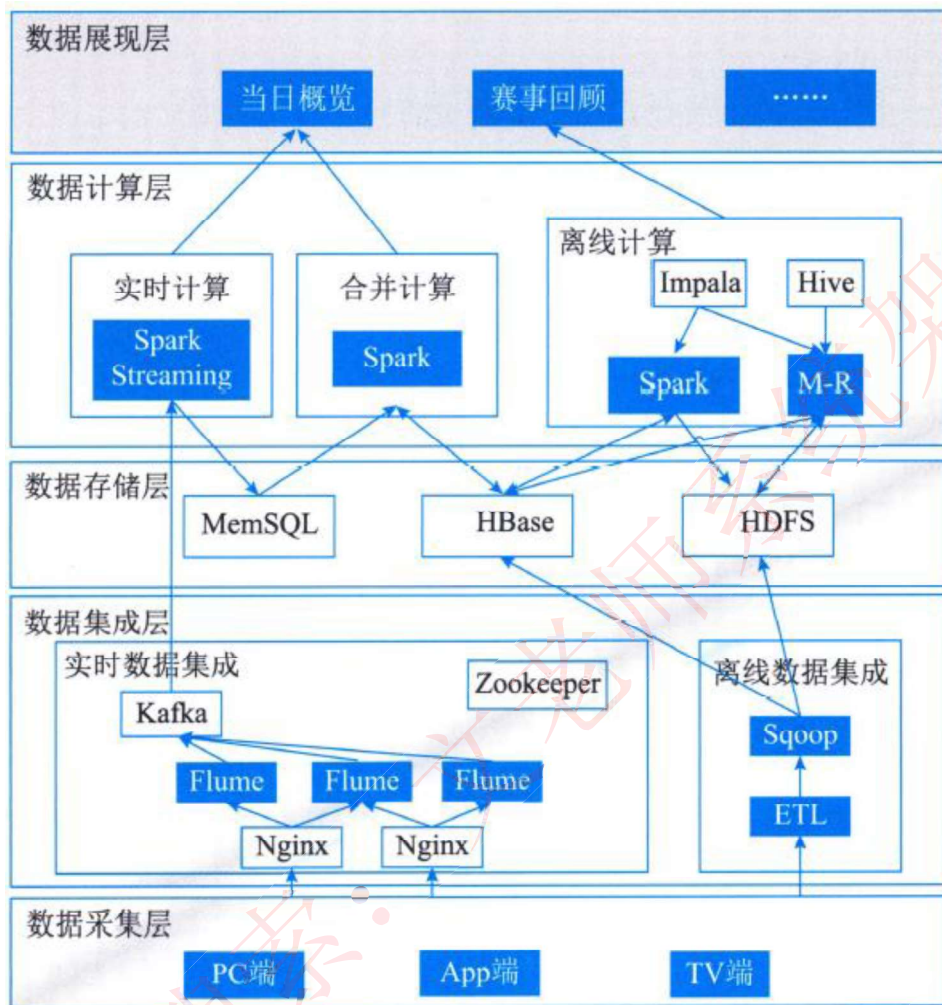
4. 历史数据处理能力：有些情况下，项目会频繁接触海量数据集进行分析，比如过往十年内的地区降水数据等，这种数据适合批处理系统进行分析，应该选择 Lambda 架构。如果始终使用小规模数据集，流处理系统完全可以使用，则应该选择 Kappa 架构。

8.5 大数据架构设计案例分析

Lambda 架构在某网奥运中的大数据应用

Lambda 架构实时处理层采用增量计算实时数据的方式，可以在集群规模不变的前提下，秒级分析出当日概览所需要的信息。赛事回顾模块需要展现自定义时间段内的历史最高在线人数、逐日播放走势、直播最高在线人数和点播视频排行等海量数据的统计信息，由于奥运期间产生的数据通常不需要被经常索引、更新，因此要求采用不可变方式存储所有的历史数据，以保证历史数据的准确性。

Lambda 架构的批处理层采用不可变存储模型，不断地往主数据集后追加新的数据，恰好可以满足对奥运数据的大规模统计分析要求。具体架构如下图：

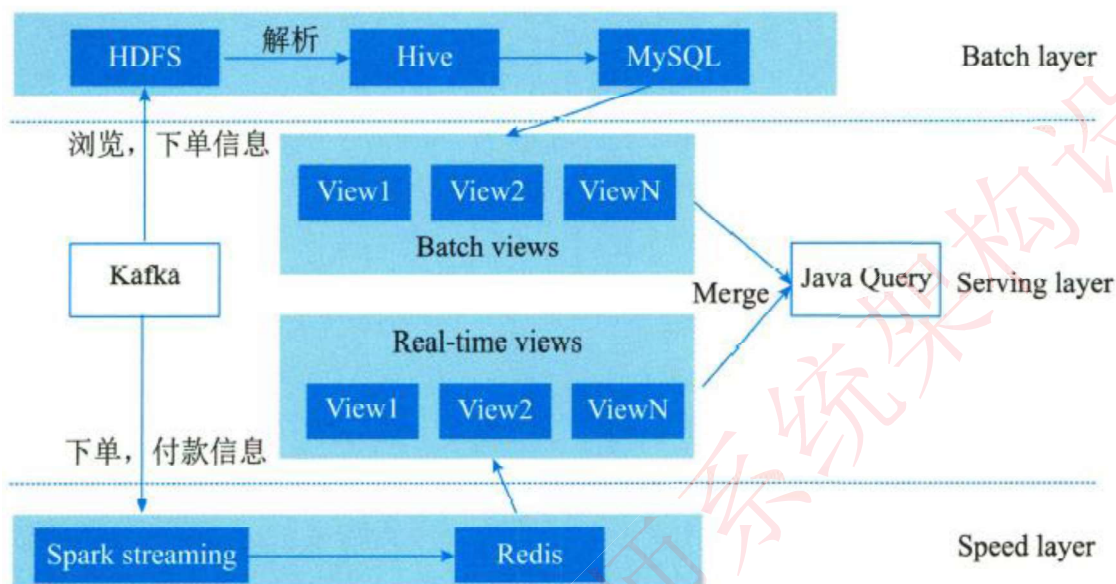


Lambda 架构在某网广告平台的应用与演进

某网广告平台展示的数据指标包含两类：曝光类(包括曝光数、点击数、点击单价、花费), 转化类(包括转化下单数、转化下单金额、转化付款数、转化付款金额)。前一类的数据主要由流量方以接口的方式提供(比如对接的腾讯广点通平台), 后一类则是某网特有的数据, 通过买家的浏览、下单、付款日志算出来。

1) 第一版架构

第一版采用了典型的 Lambda 架构形式, 架构图如图所示。



批处理层每天凌晨将 Kafka 中的浏览、下单消息同步到 HDFS 中，再将 HDFS 中的日志数据解析成 Hive 表，用 Hive Sql/Spark Sql 计算出分区的统计结果 Hive 表，最终将 Hive 表导出到 MySQL 中供服务层读取。另一方面，曝光、点击、花费等外部数据指标则是通过定时任务，调用第三方的 API，每天定时写入另一张 MySQL 表中。

实时处理层则是用 Spark Streaming 程序监听 Kafka 中的下单、付款消息，计算出每个追踪链接维度的转化数据，存储在 redis 中。

服务层则是一个 Java 服务，向外提供 http 接口。Java 服务读取两张 MySQL 表和一个 Redis 库的数据。

第一版的数据处理层比较简单，性能的瓶颈在 Java 服务层。服务层需要关联两张 MySQL 表，查询过程很复杂。

另一方面，实时数据只对接了内部的 Kafka 消息，没有实时的获取第三方的曝光、点击、浏览数据。

2) 第二版架构

针对第一版的两个问题，在第二版对数据流的结构做了一些修改。在实时处理层做了一个常驻后台的 Python 脚本，不断调用第三方 API 的小时报表，更新当日的曝光数据表。

完成第二版改动之后，Java 服务的计算压力明显下降。性能的瓶颈变成了查询 redis 数据这一块。由于 redis 里面的实时数据是业务无关的，仅统计了追踪链接维度的聚合数据。每次查询当日的转化数据，需要现在 MySQL 中查询出广告和跟踪链接的关系，找出所有的跟踪链接，再查询出这些跟踪链接的统计数据做聚合。

另一方面，离线计算的过程中涉及多次 MySQL 和 Hive 之间的导表操作，离线任务依赖链比较长，一旦出错，恢复离线任务的时间会比较久。

3) 第三版架构

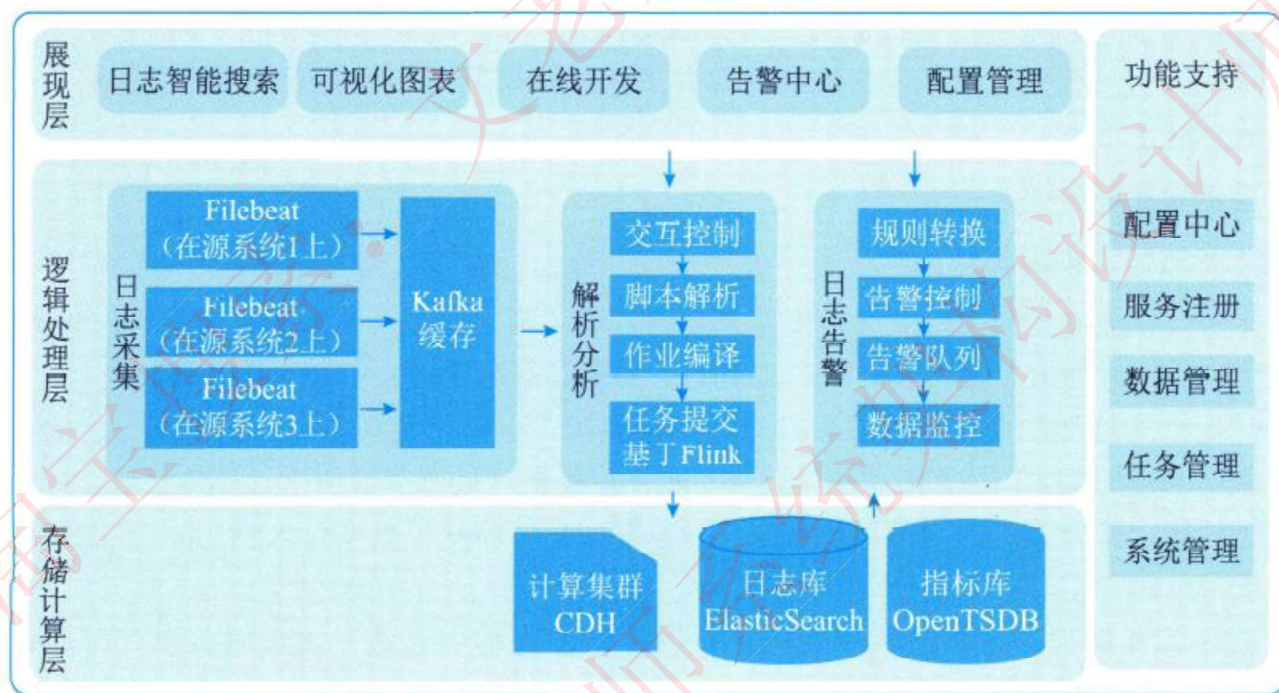
考虑到 MySQL 方便聚合、方便服务层读取的优点，在第三版中对 Lambda 架构做了一些改动，在数据层面只维护一张包含所有指标的 MySQL 表。MySQL 表的 stday (统计日期) 字段作为索引，stday= 当天的保存实时数据，st_day<当天的保存离线数据。

第三版只维护一张 MySQL 数据统计表，每天的离线任务会生成两张 hive 表，分别包含转化数据和曝光数据。这两张 Hive 表分别更新 MySQL 表的 st_day。

在实时数据这块，常驻后台的 Python 脚本更新 stday= 当天的数据的曝光类字段。spark streaming 程序在处理 kafka 中的实时下单消息时，不再统计数据到 redis，而是请求业务 Java 服务暴露出来的更新数据接口。在更新数据的接口中，找到当前下单的追踪链接所属的广告，更新 MySQL 中 stday= 当天的数据的转化类字段。这样就把查询阶段的关联操作分散在了每条订单下单的处理过程中，解决了实时数据查询的瓶颈。最终的 Java 服务层也只需要读取一个 MySQL 表，非常简洁。

某证券公司大数据系统

实时日志分析平台基于 Kappa 架构，使用统一的数据处理引擎 Flink 可实时处理全部数据，并将其存储到 Elastic-Search 与 OpenTSDB 中。实时处理过程如下：



(1) 日志采集，即在各应用系统部署采集组件 Filebeat，实时采集日志数据并输出到 Kafka 缓存。
(2) 日志清洗与解析，即基于大数据计算集群的 Flink 计算框架，实时读取 Kafka 中的日志数据进行清洗和解析，提取日志关键内容并转换成指标，以及对指标进行二次加工形成衍生指标。

(3) 日志存储，即将解析后的日志数据分类存储于 Elastic-Search 日志库中，各类基于日志的指标存储于 OpenTSDB 指标库中，供前端组件搜索与查询。

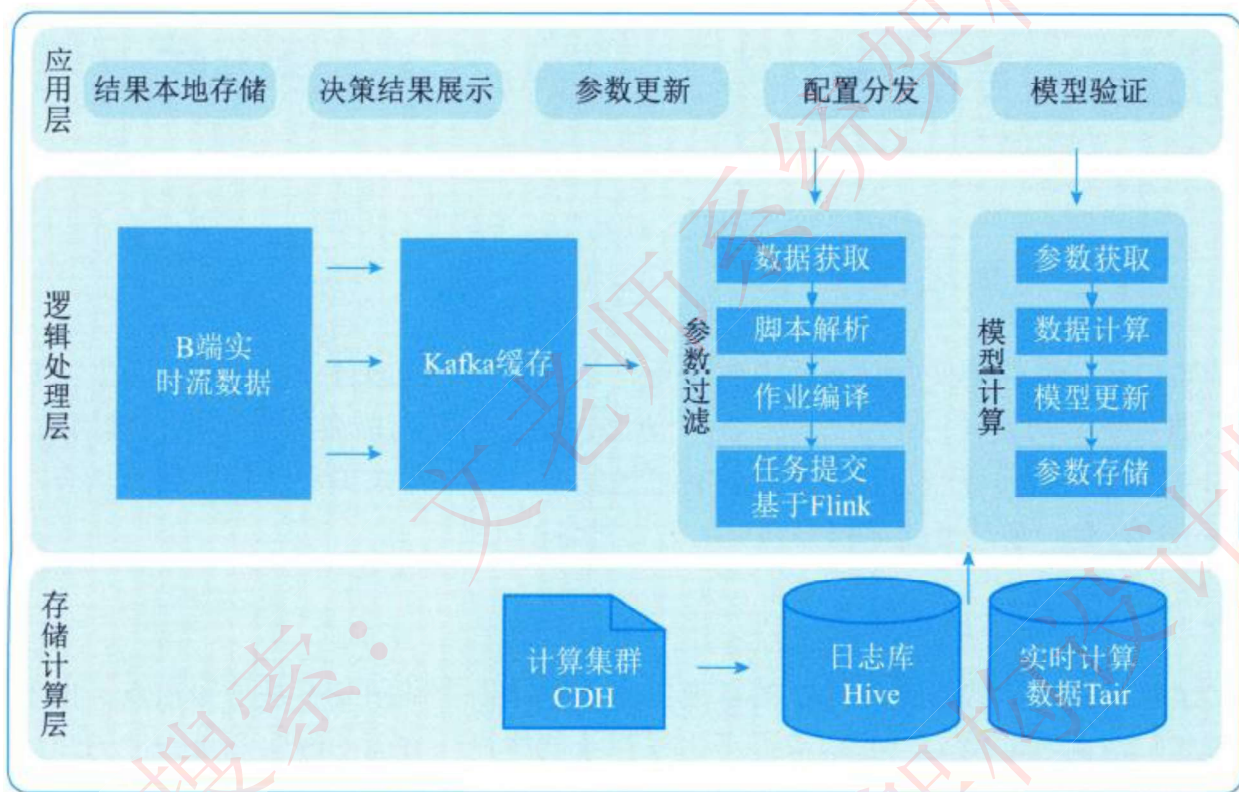
(4) 日志监控，即通过单独的告警消息队列来保持监控消息的有序管理与实时推送。

(5) 日志应用，即在充分考虑日志搜索专业需求的基础上，平台支持搜索栏常用语句保存，选择日志变量自动形成搜索表达式，以及快速按时间排序过滤、查看日志上下文等功能。同时，基于可视化分析和全息场景监控可实时展现各种指标和趋势，并在预警中心查看各类告警的优先级和详细信息，

进而结合告警信息关联查询系统日志内容来分析解决问题。此外，开发配置中心还提供了自定义日志解析开发功能，并支持告警规则、告警渠道配置。

某电商智能决策大数据系统

实时智能决策大数据平台基于 Kappa 架构，使用统一的数据处理引擎 Flink 可实时处理流数据，并将其存储到 Hive 与 Tair 中，以供后续决策服务的使用。实时处理的过程如下：



一是数据采集，即 B 端系统会实时收集用户的点击，下单以及广告的曝光和出价数据并输出到 Kafka 缓存。

二是数据的清洗与聚合，即基于大数据计算集群 Flink 计算框架，实时读取 Kafka 中的实时流数据，过滤出需要参与计算的字段，根据业务需求，聚合指定时间端的数据并转换成指标。

三是数据存储，即将 Flink 计算得到数据存储到 Hive 日志库中，需要参与模型计算计算的字段存储到 Tair 分布式缓存中。当需要进行模型计算时，决策服务会从 Tair 中读取数据，进行模型的计算，得到新的决策参数和模型。决策服务基于微服务架构，客户端部署在业务方系统中，服务端主要用于计算决策参数和模型，当服务端计算得到新的参数，此时会通过 Zookeeper 通知部署到业务方系统的客户端，客户端此时会拉取新的参数并存储到本地，并且客户端提供了获取参数的接口，业务方可以无感知调用。

四 历年案例真题

2022 年下半年试题 1

阅读以下关于软件架构设计与评估的叙述，在答题纸上回答问题 1 和问题 2。

【说明】

某电子商务公司拟升级其会员与促销管理系统，向用户提供个性化服务，提高用户的粘性。在项目立项之初，公司领导层一致认为本次升级的主要目标是提升会员管理方式的灵活性，由于当前用户规模不大，业务也相对简单，系统性能方面不做过多考虑。新系统除了保持现有的四级固定会员制度外，还需要根据用户的消费金额、偏好、重复性等相关特征动态调整商品的折扣力度，并支持在特定的活动周期内主动筛选与活动主题高度相关的用户集合，提供个性化的打折促销活动。

在需求分析与架构设计阶段，公司提出的需求和质量属性描述如下：

- (a) 管理员能够在页面上灵活设置折扣力度规则和促销活动逻辑，设置后即可生效；
- (b) 系统应该具备完整的安全防护措施，支持对恶意攻击行为进行检测与报警；
- (c) 在正常负载情况下，系统应在 0.3 秒内对用户的界面操作请求进行响应；
- (d) 用户名是系统唯一标识，要求以字母开头，由数字和字母组合而成，长度不少于 6 个字符；
- (e) 在正常负载情况下，用户支付商品费用后在 3 秒内确认订单支付信息；
- (f) 系统主站点电力中断后，应在 5 秒内将请求重定向到备用站点；
- (g) 系统支持横向存储扩展，要求在 2 人天内完成所有的扩展与测试工作；
- (h) 系统宕机后，需要在 10 秒内感知错误，并自动启动热备份系统；
- (i) 系统需要内置接口函数，支持开发团队进行功能调试与系统诊断；
- (j) 系统需要为所有的用户操作行为进行详细记录，便于后期查阅与审计；
- (k) 支持对系统的外观进行调整和配置，调整工作需要 4 人天内完成。

在对系统需求、质量属性描述和架构特性进行分析的基础上，系统架构师给出了两种候选的架构设计方案，公司目前正在组织相关专家对系统架构进行评估。

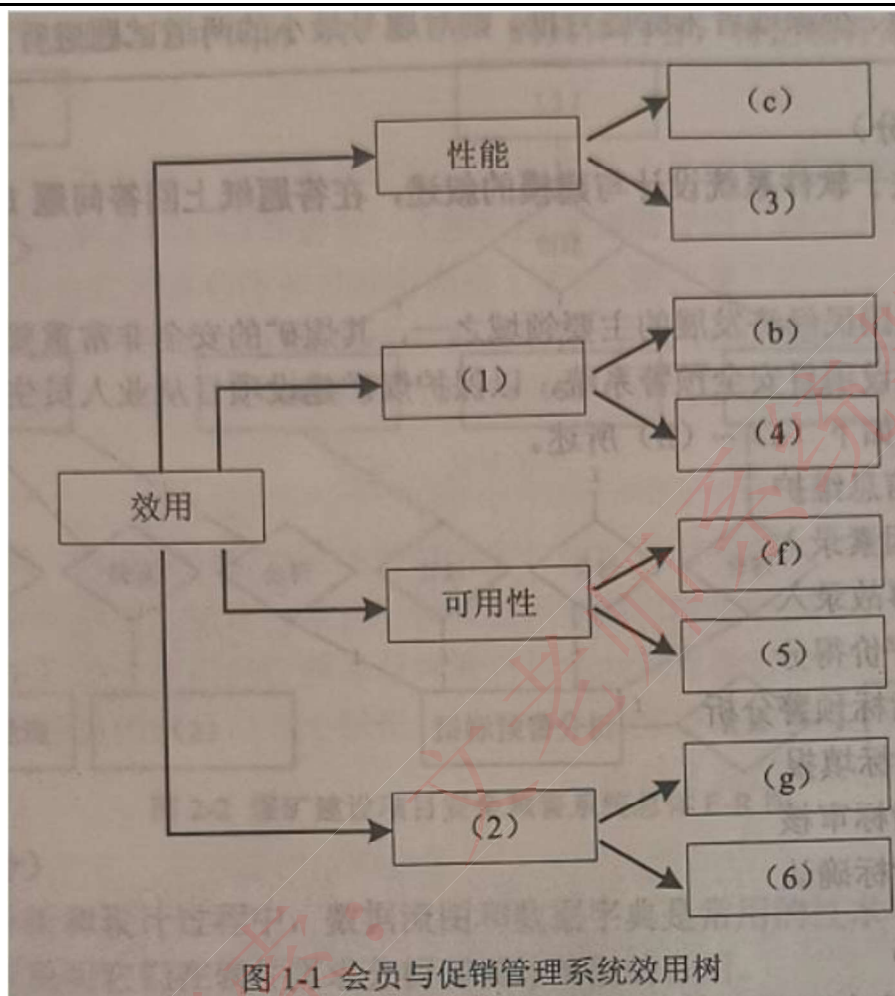


图 1-1 会员与促销管理系统效用树

【问题 1】(12 分)

在架构评估过程中，质量属性效用树（utility tree）是对系统质量属性进行识别和优先级排序的重要工具。请将合适的质量属性名称填入图 1-1 中(1)、(2)空白处，并选择题干描述的(a)~(k)填入(3)~(6)空白处，完成该系统的效用树。

【问题 2】(13 分)

针对该系统的功能，李工建议采用面向对象的架构风格，将折扣力度计算和用户筛选分别封装为独立对象，通过对象调用实现对应的功能；王工则建议采用解释器（interpreters）架构风格，将折扣力度计算和用户筛选条件封装为独立的规则，通过解释规则实现对应的功能。请针对系统的主要功能，从折扣规则的可修改性、个性化折扣定义灵活性和系统性能三个方面对这两种架构风格进行比较与分析，并指出该系统更适合采用哪种架构风格。

2022 年下半年试题 2

阅读以下关于软件系统设计与建模的叙述，在答题纸上回答问题 1 至问题 3。

【说明】

煤炭生产是国民经济发展的主要领域之一，其煤矿的安全非常重要。某能源企业拟开发一套煤矿建设项目安全预警系统，以保护煤矿建设项目从业人员生命安全。本系统的主要功能包括如下(a)~(h)所述。

- (a) 项目信息维护
- (b) 影响因素录入
- (c) 关联事故录入
- (d) 安全评价得分
- (e) 项目指标预警分析
- (f) 项目指标填报
- (g) 项目指标审核
- (h) 项目指标确认

【问题 1】(9 分)

王工根据煤矿建设项目安全预警系统的功能要求，设计完成了系统的数据流图，如图 2-1 所示。请使用题干中描述的功能(a)~(h)，补充完善空(1)~(6)处的内容，并简要介绍数据流图在分层细化过程中遵循的数据平衡原则。

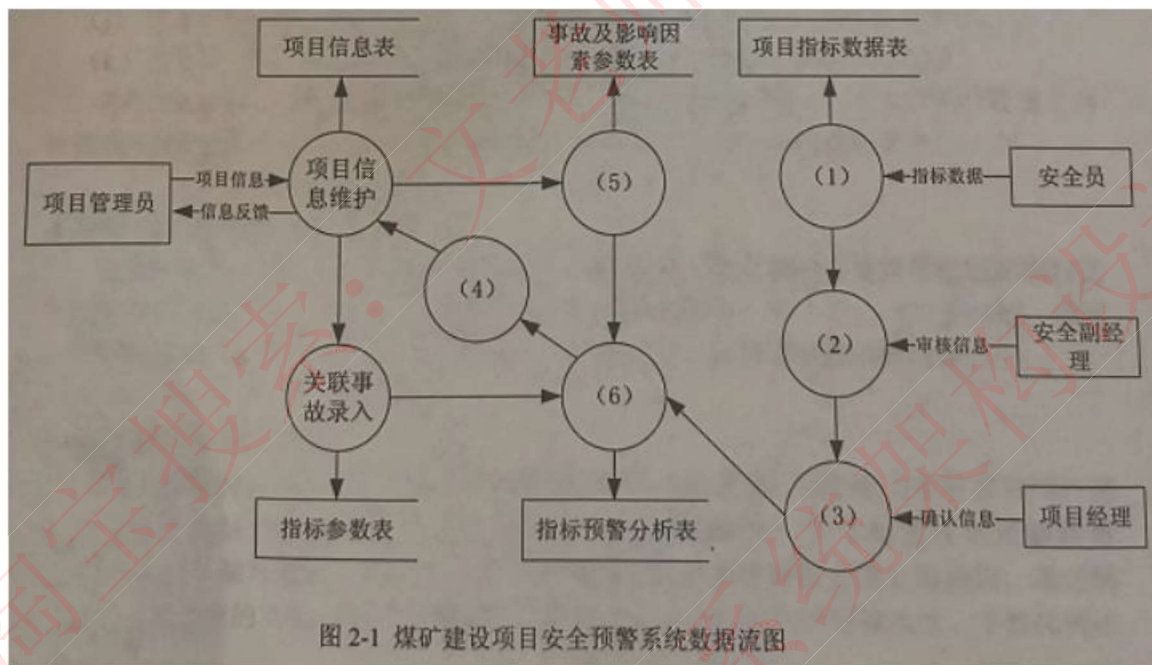
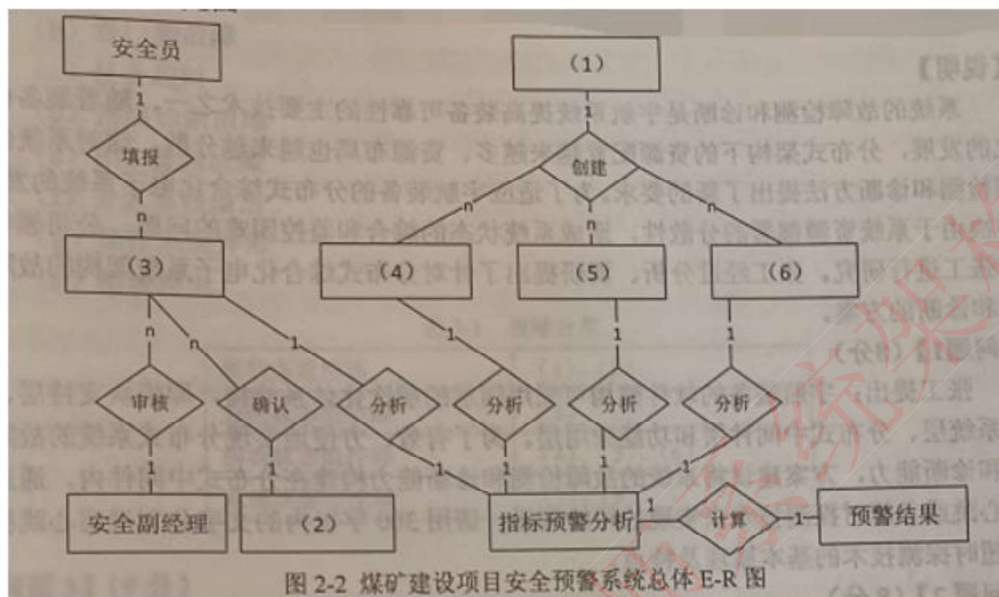


图 2-1 煤矿建设项目安全预警系统数据流图

【问题 2】(9 分)

请根据【问题 1】中数据流图表示的相关信息，补充完善煤矿建设项目安全预警系统总体 E-R 图（见图 2-2）中实体(1)~(6)的具体内容，将正确答案填在答题纸上。



【问题 3】(7 分)

在结构化分析和设计过程中，数据流图和数据字典是常用的技术手段，请用 200 字以内的文字简要说明它们在软件需求分析和设计阶段的作用。

2022 年下半年试题 3

阅读以下关于嵌入式系统故障检测和诊断的相关描述，在答题纸上回答问题 1 至问题 3。

【说明】

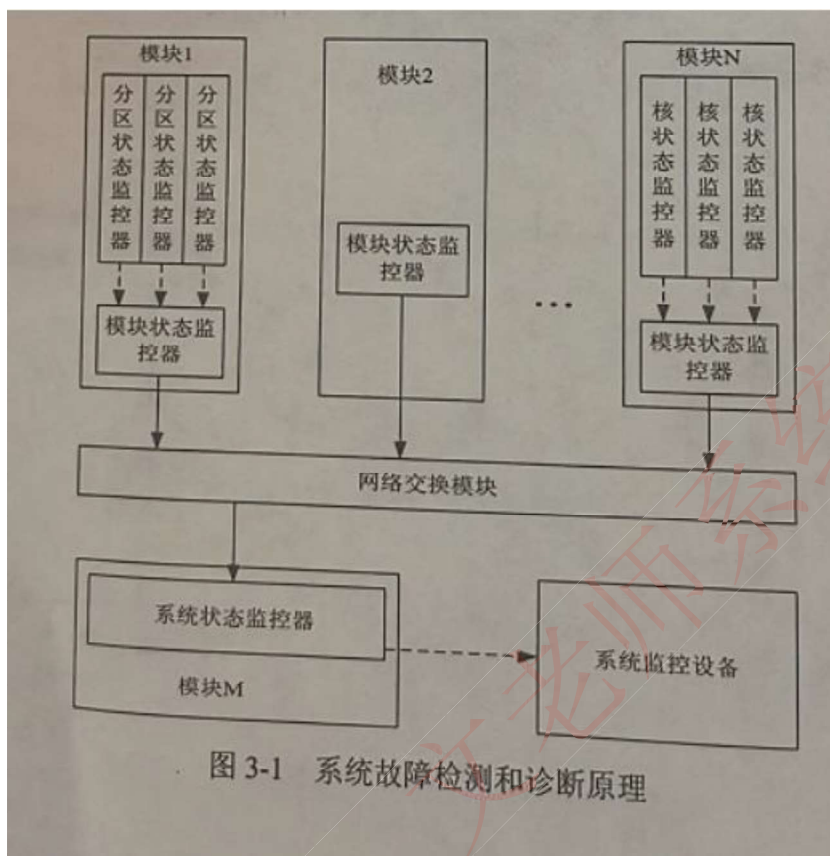
系统的故障检测和诊断是宇航系统提高装备可靠性的主要技术之一，随着装备信息化的发展，分布式架构下的资源配置越来越多、资源布局也越来越分散，这对系统的故障检测和诊断方法提出了新的要求。为了适应宇航装备的分布式综合化电子系统的发展，解决由于系统资源部署的分散性，造成系统状态的综合和监控困难的问题，公司领导安排张工进行研究。张工经过分析、调研提出了针对分布式综合化电子系统架构的故障检测和诊断的方案。

【问题 1】(8 分)

张工提出：宇航装备的软件架构可采用四层的层次化体系结构，即模块支持层、操作系统层、分布式中间件层和功能应用层。为了有效、方便地实现分布式系统的故障检测和诊断能力，方案建议将系统的故障检测和诊断能力构建在分布式中间件内，通过使用心跳或者超时探测技术来实现故障检测器。请用 300 字以内的文字分别说明心跳检测和超时探测技术的基本原理及特点。

【问题 2】(8 分)

张工针对分布式综合化电子系统的架构特征，给出了初步设计方案，指出每个节点的故障监测与诊断器主要负责监控系统中所有的故障信息，并将故障信息进行综合分析判断，使用故障诊断器分析出故障原因，给出解决方案和措施。系统可以给模块的每个处理机器核配置核状态监控器、给每个分区配置分区状态监控器、给每个模块配置模块状态监控器、给系统配置系统状态监控器，如图 3-1 所示。



请根据下面给出的分布式综合化电子系统可能产生的故障(a)~(h)，判断这些故障分别属于哪类监控器检测的范围，完善表 3-1 的(1)~(8)的空白。

- (a)应用程序除零
- (b)看门狗故障
- (c)任务超时
- (d)网络诊断故障
- (e)BIT 检测故障
- (f)分区堆栈溢出
- (g)操作系统异常
- (h)模块掉电

表 3-1 故障分类

核状态监控器	(1)、(2)
分区状态监控器	(3)
模块状态监控器	(4)、(5)、(6)
系统状态监控器	(7)、(8)

【问题 3】(9 分)

张工在方案中指出，本系统的故障诊断采用故障诊断器实现，它可综合多种故障信息和系统状态，依据智能决策数据库提供的决策策略判定出故障类型和处理方法。智能决策数据库中的策略可以对故障开展定性或定量分析。通常，在定量分析中，普遍采用基于解析模型的方法和数据驱动的方法。张

工在方案中提出该系统定量分析时应采用基于解析模型的方法。但是此提议受到王工的反驳，王工指出采用数据驱动的方法更适合分布式综合化电子系统架构的设计。请用 300 字以内的文字，说明数据驱动方法的基本概念，以及王工提出采用此方法的理由。

2022 年下半年试题 4

阅读以下关于数据库缓存的叙述，在答题纸上回答问题 1 至问题 3。

【说明】

某大型电商平台建立了一个在线 B2B 商店系统，并在全国多地建设了货物仓储中心，通过提前备货的方式来提高货物的运送效率。但是在运营过程中，发现会出现很多跨仓储中心调货从而延误货物运送的情况。为此，该企业计划新建一个全国仓储货物管理系统，在实现仓储中心常规管理功能之外，通过对在线 B2B 商店系统中订单信息进行及时的分析和挖掘，并通过大数据分析预测各地仓储中心中各类货物的配置数量，从而提高运送效率，降低成本。

当用户通过在线 B2B 商店系统选购货物时，全国仓储货物管理系统会通过该用户所在地址、商品类别以及仓储中心的货物信息和地址，实时为用户订单反馈货物起运地（某仓储中心）并预测送达时间。反馈送达时间的响应时间应小于 1 秒。

为满足反馈送达时间功能的性能要求，设计团队建议在全国仓储货物管理系统中采用数据缓存集群的方式，将仓储中心基本信息、商品类别以及库存数量放置在内存的缓存中，而仓储中心的其它商品信息则存储在数据库系统。

【问题 1】(9 分)

设计团队在讨论缓存和数据库的数据一致性问题时，李工建议采取数据实时同步更新方案，而张工则建议采用数据异步准实时更新方案。

请用 200 字以内的文字，简要介绍两种方案的基本思路，说明全国仓储货物管理系统应该采用哪种方案，并说明采取该方案的原因。

【问题 2】(9 分)

随着业务的发展，仓储中心以及商品的数量日益增加，需要对集群部署多个缓存节点，提高缓存的处理能力。李工建议采用缓存分片方法，把缓存的数据拆分到多个节点分别存储，减轻单个缓存节点的访问压力，达到分流效果。

缓存分片方法常用的有哈希算法和一致性哈希算法，李工建议采用一致性哈希算法来进行分片。请用 200 字以内的文字简要说明两种算法的基本原理，并说明李工采用一致性哈希算法的原因。

【问题 3】(7 分)

全国仓储货物管理系统开发完成，在运营一段时间后，系统维护人员发现大量黑客故意发起非法的商品送达时间查询请求，造成了缓存击穿。张工建议尽快采用布隆过滤器方法解决。请用 200 字以内的文字解释布隆过滤器的工作原理和优缺点。

2022 年下半年试题 5

阅读以下关于 Web 系统架构设计的叙述，在答题纸上回答问题 1 至问题 3。

【说明】

某公司拟开发一套基于边缘计算的智能门禁系统，用于如园区、新零售、工业现场等存在来访、被访业务的场景。来访者在来访前，可以通过线上提前预约的方式将自己的个人信息记录在后台，被访者在系统中通过此请求后，来访者在到访时可以直接通过“刷脸”的方式通过门禁，无需做其他验证。此外，系统的管理员可对正在运行的门禁设备进行管理。

基于项目需求，该公司组建项目组，召开了项目讨论会。会上，张工根据业务需求并结合边缘计算的思想，提出本系统可由访客注册模块、模型训练模块、端侧识别模块与设备调度平台模块等四项功能组成。李工从技术层面提出该系统可使用 Flask 框架与 SSM 框架为基础来开发后台服务器，将开发好的系统通过 Docker 进行部署，并使用 MQTT 协议对 Docker 进行管理。

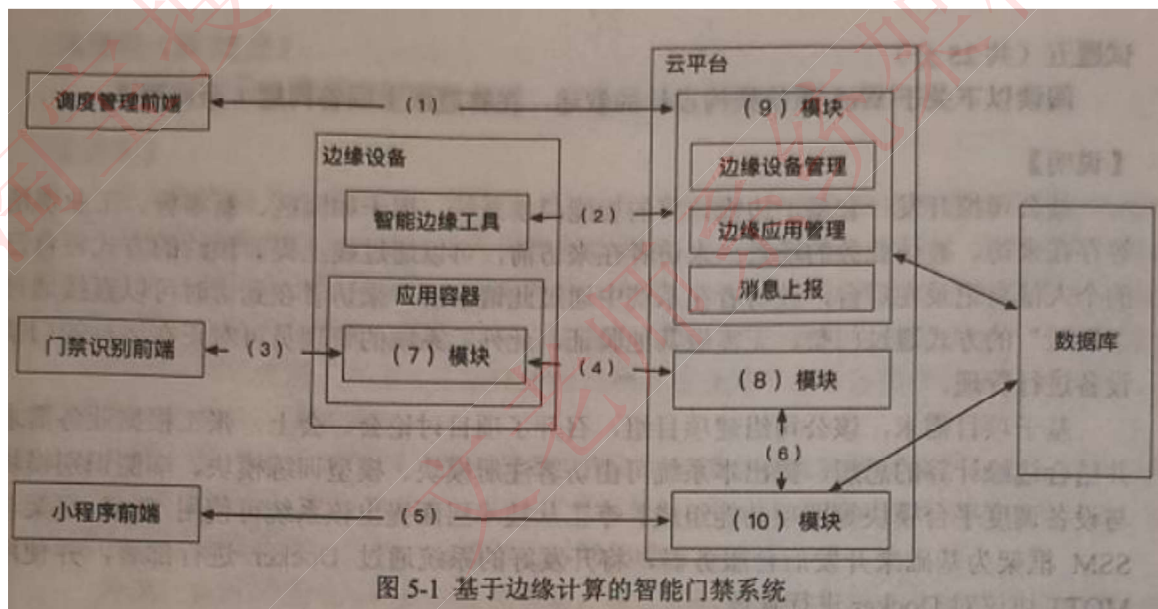
【问题 1】(5 分)

MQTT 协议在工业物联网中得到广泛的应用，请用 300 字以内的文字简要说明 MQTT 协议。

【问题 2】(14 分)

在会议上，张工对功能模块进行了更进一步的说明：**访客注册模块**用于来访者提交申请与被访者确认申请，主要处理提交来访申请、来访申请审核业务，同时保存访客数据，为训练模块准备训练数据集；**模型训练模块**用于使用访客数据进行模型训练，为端侧设备的识别业务提供模型基础；**端侧识别模块**在边缘门禁设备上运行，使用训练好的模型来识别来访人员，与云端服务协作完成访客来访的完整业务；**设备调度平台模块**用于对边缘门禁设备进行管理，管理人员能够使用平台对边缘设备进行调度管理与状态监控，实现云端协同。

图 5-1 给出了基于边缘计算的智能门禁系统架构图，请结合 HTTP 协议和 MQTT 协议的特点，为图 5-1 中(1)~(6)处选择合适的协议；并结合张工关于功能模块的描述，补充完善图 5-1 中(7)~(10)处的空白。



【问题 3】(6 分)

请用 300 字以内的文字，从数据通信、数据安全和系统性能等方面简要分析在传统云计算模型中引入边缘计算模型的优势。

2021 年下半年试题 1

阅读以下关于软件架构设计与评估的叙述，在答题纸上回答问题 1 和问题 2。

【说明】

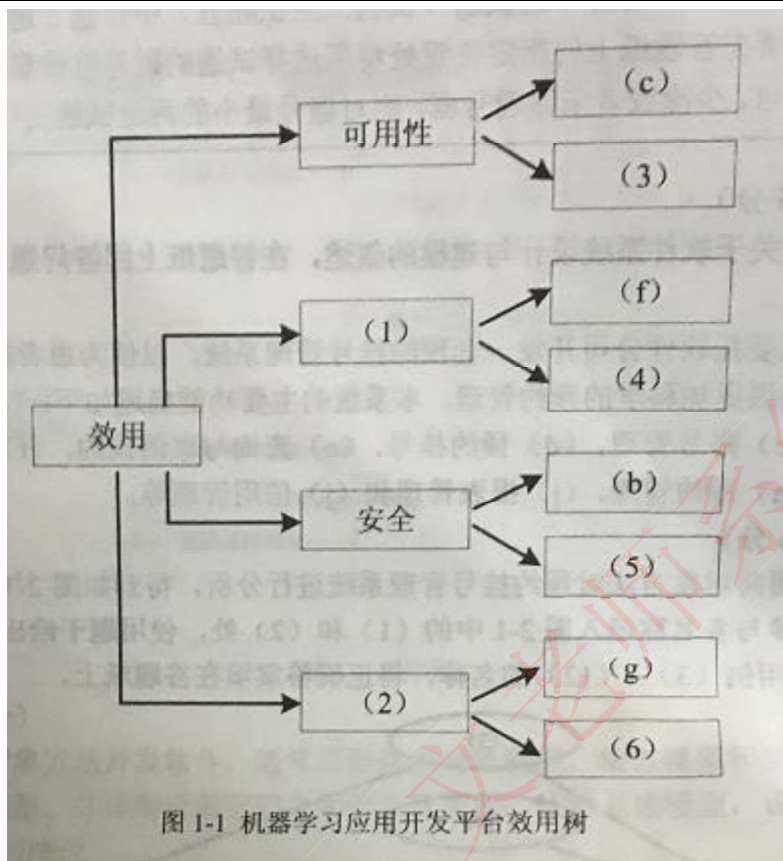
某公司拟开发一套机器学习应用开发平台，支持用户使用浏览器在线进行基于机器学习的智能应用开发活动。该平台的核心应用场景是用户通过拖拽算法组件灵活定义机器学习流程，采用自助方式进行智能应用设计、实现与部署，并可以开发新算法组件加入平台中。在需求分析与架构设计阶段，公司提出的需求和质量属性描述如下：

- (a) 平台用户分为算法工程师、软件工程师和管理员等三种角色，不同角色的功能界面有所不同；
- (b) 平台应该具备数据库保护措施，能够预防核心数据库被非授权用户访问；
- (c) 平台支持分布式部署，当主站点断电后，应在 20 秒内将请求重定向到备用站点；
- (d) 平台支持初学者和高级用户两种界面操作模式，用户可以根据自己的情况灵活选择合适的模式；
- (e) 平台主站点宕机后，需要在 15 秒内发现错误并启用备用系统；
- (f) 在正常负载情况下，机器学习流程从提交到开始执行，时间间隔不大于 5 秒；
- (g) 平台支持硬件扩容与升级，能够在 3 人天内完成所有部署与测试工作；
- (h) 平台需要对用户的所有操作过程进行详细记录，便于审计工作；
- (i) 平台部署后，针对界面风格的修改需要在 3 人天内完成；
- (j) 在正常负载情况下，平台应在 0.5 秒内对用户的界面操作请求进行响应；
- (k) 平台应该与目前国内外主流的机器学习应用开发平台的界面风格保持一致；
- (l) 平台提供机器学习算法的远程调试功能，支持算法工程师进行远程调试。

在对平台需求、质量属性描述和架构特性进行分析的基础上，公司的架构师给出了三种候选的架构设计方案，公司目前正在组织相关专家对平台架构进行评估。

【问题 1】(9 分)

在架构评估过程中，质量属性效用树(utility tree)是对系统质量属性进行识别和优先级排序的重要工具。请将合适的质量属性名称填入图 1-1 中(1)、(2)空白处，并从题干中的(a)~(l)中选择合适的质量属性描述，填入(3)~(6)空白处，完成该平台的效用树。



【问题 2】(16 分)

针对该系统的功能，赵工建议采用解释器(interpreter)架构风格，李工建议采用管道-过滤器(pipe-and-filter)的架构风格，王工则建议采用隐式调用(implicit invocation)架构风格。请针对平台的核心应用场景，从机器学习流程定义的灵活性和学习算法的可扩展性两个方面对三种架构风格进行对比与分析，并指出该平台更适合采用哪种架构风格。

2021 年下半年试题 2

阅读以下关于软件系统设计与建模的叙述，在答题纸上回答问题 1 至问题 3。

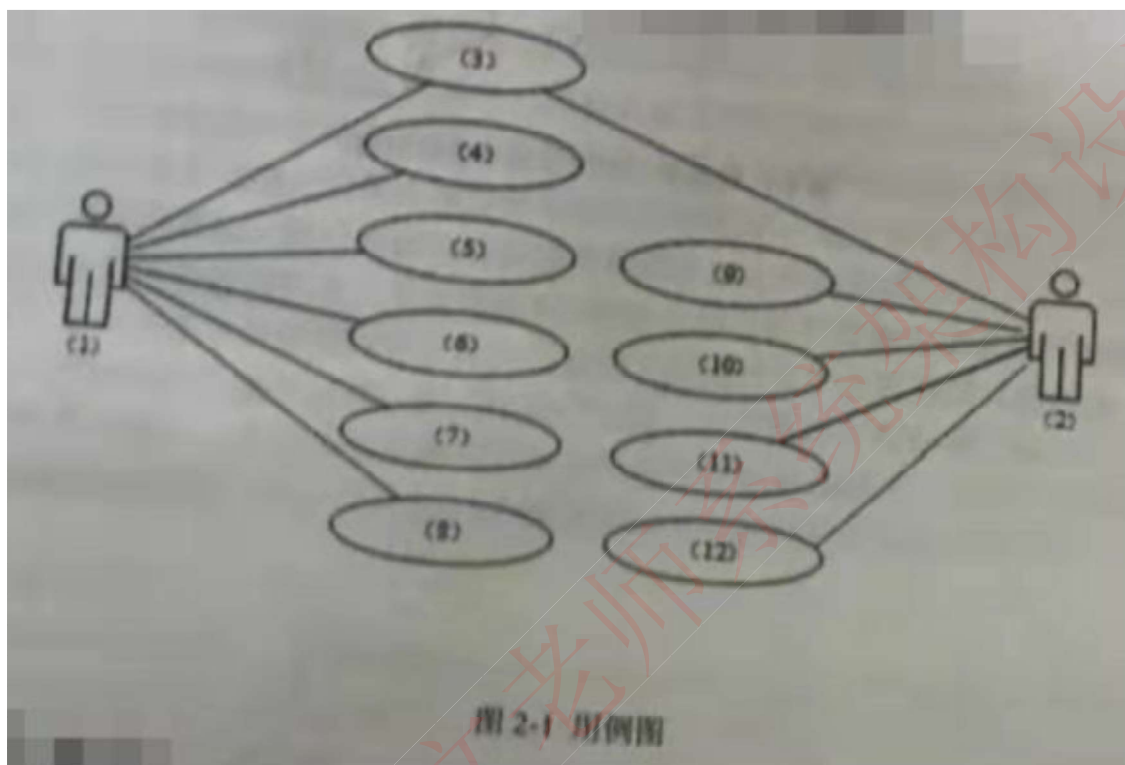
【说明】

某医院拟委托软件公司开发一套预约挂号管理系统，以便为患者提供更好的就医体验，为医院提供更加科学的预约管理。本系统的主要功能描述如下：

(a)注册登录，(b)信息浏览，(c)账号管理，(d)预约挂号，(e)查询与取消预约，(f)号源管理，(g)报告查询，(h)预约管理，(i)报表管理和(j)信用管理等。

【问题 1】(6 分)

若采用面向对象方法对预约挂号管理系统进行分析，得到如图 2-1 所示的用例图。请将合适的参与者名称填入图 2-1 中的(1)和(2)处，使用题干给出的功能描述(a)~(j)，完善用例(3)~(12)的名称，将正确答案填在答题纸上。



【问题 2】(10 分)

预约人员(患者)登录系统后发起预约挂号请求,进入预约界面。进行预约挂号时使用数据库访问类获取医生的相关信息,在数据库中调用医生列表,并调取医生出诊时段表,将医生出诊时段反馈到预约的界面,并显示给预约人员;预约人员选择医生及就诊时间后确认预约,系统返回网预约结果,并向用户显示是否预约成功。

采用面向对象方法对预约挂号过程进行分析,得到如图 2-2 所示的顺序图,使用题干中给出的描述,完善图 2-2 中对象(1),及消息(2)~(4)的名称,将正确答案填在普题纸上请简要说明在描述对象之间的动态交互关系时,协作图与顺序图存在哪些区别。



图 2-2 顺序图

【问题 3】(9 分)

采用面向对象方法开发软件，通常需要建立对象模型、动态模型和功能模型，请分别介绍这 3 种模型，并详细说明它们之间的关联关系，针对上述模型，说明哪些模型可用于软件的需求分析？

2021 年下半年试题 3

阅读以下关于嵌入式数据架构设计的相关描述，在答题纸上回答问题 1 至问题 3。

【说明】

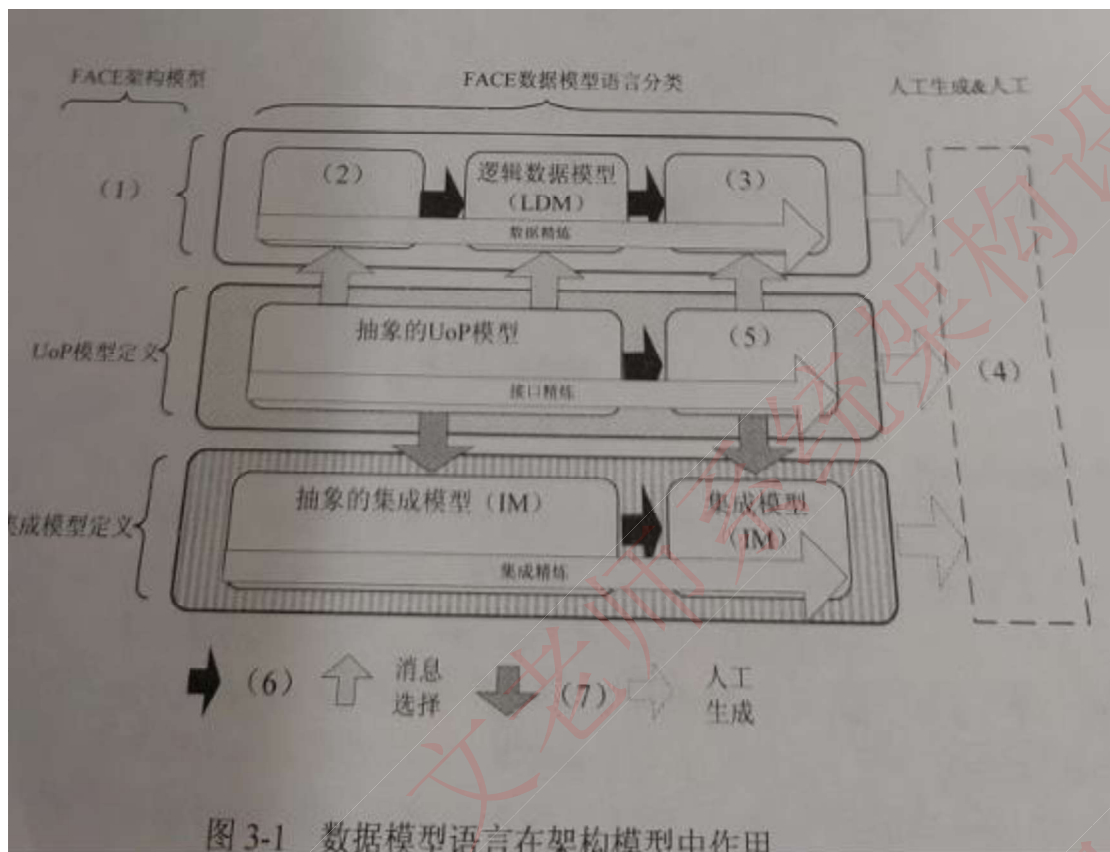
数据架构(Data architecture)是系统架构设计的主要工作之一。它主要用于描述业务数据以及数据间的关系。数据架构着重考虑“数据需求”，关注的是持久化数据的组织，数据架构的设计过程主要包括:数据定义、数据分布与数据管理。某公司为了适应宇航装备的持续发展，提升本公司的核心竞争力，改变原来事件驱动的架构设计模式。公司领导将新产品架构规划工作交给张工。张工经过分析、调研给出了本企业宇航产品的未来架构规划方案。

【问题 1】(9 分)

张工在规划方案中指出:宇航装备要实现以数据为中心的架构设计模式，就应改变传统的各个子系统独立设计方式，打破原宇航装备的生产关系。为了达到这个目标，我们首先要解决装备数据的共享、管理和存储等问题，做好顶层的数据架构规划工作。请用 300 字以内的文字说明数据定义、数据分布与数据管理的具体内涵。

【问题 2】(7 分)

张工在规划方案中提出公司未来产品设计要遵从一种开放式的架构体系，并在此基础上完善数据架构的设计工作，形成一套规格化的数据模型语言。张工给出了基于 FACE(Future Airborne Capability Environment)架构的新产品架构，其中，图 3-1 说明了数据模型语言在架构模型中作用。



2021 年下半年试题 4

阅读以下关于数据库设计的叙述，在答题纸上回答问题 1 至问题 3。

【说明】

某医药销售企业因业务发展，需要建立线上药品销售系统，为用户提供便捷的互联网药品销售服务、该系统除了常规药品展示、订单、用户交流与反馈功能外，还需要提供当前热销产品排名、评价分类管理等功能。

通过对需求的分析，在数据管理上初步决定采用关系数据库（MySQL）和数据库缓存（Redis）的混合架构实现。

经过规范化设计之后，该系统的部分数据库表结构如下所示。

供应商（供应商 ID，供应商名称，联系方式，供应商地址）；

药品（药品 ID，药品名称，药品型号，药品价格，供应商 ID）；

药品库存（药品 ID，当前库存数量）；

订单（订单号码，药品 ID，供应商 ID，药品数量，订单金额）；

【问题 1】（9 分）

在系统初步运行后，发现系统数据访问性能较差。经过分析，刘工认为原来数据库规范化设计后，关系表过于细分，造成了大量的多表关联查询，影响了性能。例如当用户查询商品信息时，需要同时显示该药品的信息、供应商的信息、当前库存等信息。

为此，刘工认为可以采用反规范化设计来改造药品关系的结构，以提高查询性能。修改后的药品关系结构为：

药品（药品 ID，药品名称，药品型号，药品价格，供应商 ID，供应商名称，当前库存数量）；
请用 200 字以内的文字说明常见的反规范化设计方法，并说明用户查询商品信息应该采用哪种反规范化设计方法。

【问题 2】（9 分）

王工认为，反规范化设计可提高查询的性能，但必然会带来数据的不一致性问题。请用 200 字以内的文字说明在反规范化设计中，解决数据不一致性问题的三种常见方法，并说明该系统应该采用哪种方法。

【问题 3】（7 分）

该系统采用了 Redis 来实现某些特定功能（如当前热销药品排名等），同时将药品关系数据放到内存以提高商品查询的性能，但必然会造成 Redis 和 MySQL 的数据实时同步问题。

（1）Redis 的数据类型包括 String、Hash、List、Set 和 ZSet 等，请说明实现当前热销药品排名的功能应该选择使用哪种数据类型。

（2）请用 200 字以内的文字解释说明解决 Redis 和 MySQL 数据实时同步问题的常见方案。

2021 年下半年试题 5

阅读以下关于 Web 系统架构设计的教述，在答题纸上回答问题 1 至问题 3。

【说明】

某公司拟开发一个智能家居管理系统，该系统的主要功能需求如下：

- 1) 用户可使用该系统客户端实现对家居设备的控制，且家居设备可向客户端反馈实时状态；
- 2) 支持家居设备数据的实时存储和查询；
- 3) 基于用户数据，挖掘用户生活习惯，向用户提供家居设备智能化使用建议。

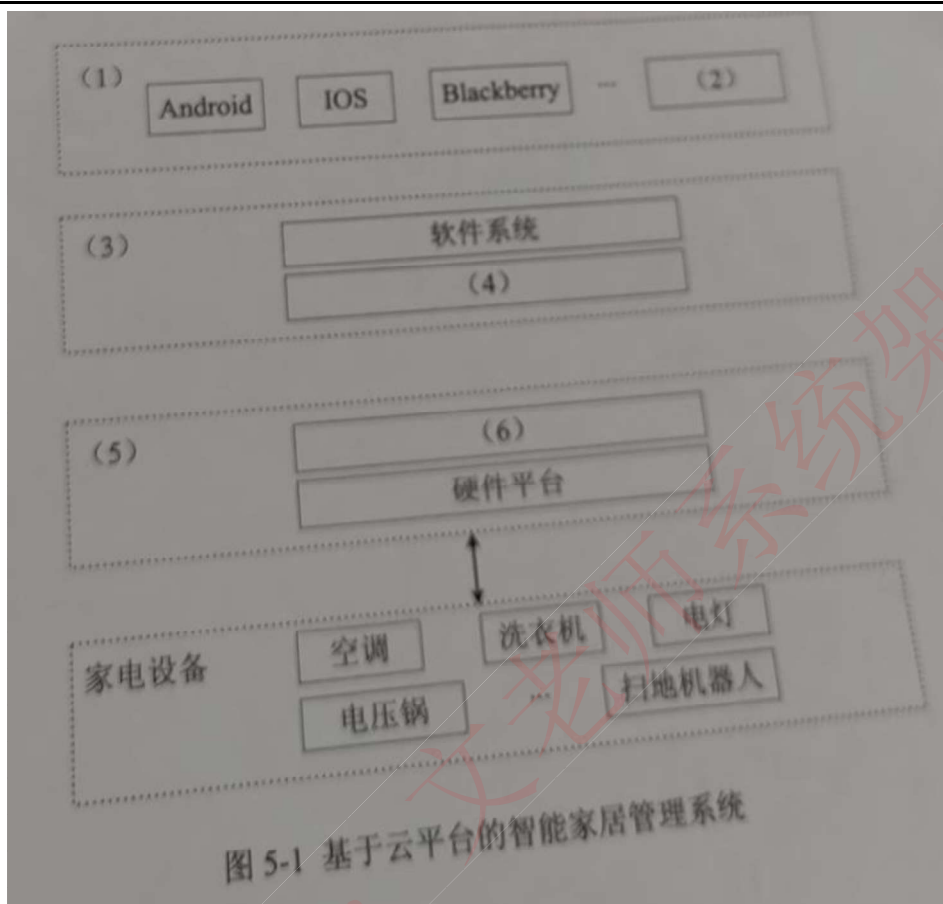
基于上述需求，该公司组建了项目组，在项目会议上，张工给出了基于家庭网关的传统智能家居管理系统的设计思路，李工给出了基于云平台的智能家居系统的设计思路。经过深入讨论，公司决定采用李工的设计思路。

【问题 1】（8 分）

请用 400 字以内的文字简要描述基于家庭网关的传统智能家居管理系统和基于云平台的智能家居管理系统在网关管理、数据处理和系统性能等方面的特点，以说明项目组选择李工设计思路的原因。

【问题 2】（12 分）

请从下面给出的（a）~（j）中进行选择，补充完善图 5-1 中空（1）~（6）处的内容，协助李工完成该系统的架构设计方案。



- (a) Wi-Fi
- (b) 蓝牙
- (c) 驱动程序
- (d) 数据库
- (e) 家庭网关
- (f) 云平台
- (g) 微服务
- (h) 用户终端
- (i) 鸿蒙
- (j) TCP/IP

【问题 3】（5 分）

该系统需实现用户终端与服务端的双向可靠通信，请用 300 字以内的文字从数据传输可靠性的角度对比分析 TCP 和 UDP 通信协议的不同，并说明该系统应采用哪种通信协议。

2020 年下半年试题 1

阅读以下关于软件架构设计与评估的叙述，在答题纸上回答问题 1 和问题 2。

【说明】

某公司拟开发一套在线软件开发系统，支持用户通过浏览器在线进行软件开发活动。该系统的主要功能包括代码编辑、语法高亮显示、代码编译、系统调试、代码仓库管理等。在需求分析与架构设计阶段，公司提出的需求和质量属性描述如下：

- (a) 根据用户的付费情况对用户进行分类，并根据类别提供相应的开发功能；
- (b) 在正常负载情况下，系统应在 0.2 秒内对用户的界面操作请求进行响应；
- (c) 系统应该具备完善的安全防护措施，能够对黑客的攻击行为进行检测与防御；
- (d) 系统主站点断电后，应在 3 秒内将请求重定向到备用站点；
- (e) 系统支持中文昵称，但用户名必须以字母开头，长度不少于 8 个字符；
- (f) 系统宕机后，需要在 15 秒内发现错误并启用备用系统；
- (g) 在正常负载情况下，用户的代码提交请求应该在 0.5 秒内完成；
- (h) 系统支持硬件设备灵活扩容，应保证在 2 人·天内完成所有的部署与测试工作；
- (i) 系统需要为针对代码仓库的所有操作情况进行详细记录，便于后期查阅与审计；
- (j) 更改系统的 Web 界面风格需要在 4 人·天内完成；
- (k) 系统本身需要提供远程调试接口，支持开发团队进行远程排错。

在对系统需求、质量属性和架构特性进行分析的基础上，该公司的系统架构师给出了两种候选的架构设计方案，公司目前正在组织相关专家对候选系统架构进行评估。

【问题 1】(13 分)

针对该系统的功能，李工建议采用管道-过滤器(pipe and filter)的架构风格，而王工则建议采用仓库(repository)架构风格。请指出该系统更适合采用哪种架构风格，并针对系统的主要功能，从数据处理方式、系统的可扩展性和处理性能三个方面对这两种架构风格进行比较与分析，填写表 1-1 中的(1)~(4)空白处。

表 1-1 两种架构风格的比较与分析

架构风格名称	数据处理方式	系统可扩展性	处理性能
管道-过滤器	数据驱动机制，处理流程事先确定，交互性差	(2)	劣势：需要数据格式转换，性能降低 优势：支持过滤器并发调用，性能提高
仓库	(1)	数据与处理解耦合，可动态添加和删除处理组件	劣势：(3) 优势：(4)

【问题 2】(12 分)

在架构评估过程中，质量属性效用树(utility tree)是对系统质量属性进行识别和优先级排序的重要工具。请将合适的质量属性名称填入图 1-1 中(1)、(2)空白处，并选择题干描述的(a)~(k)填入(3)~(6)空白处，完成该系统的效用树。

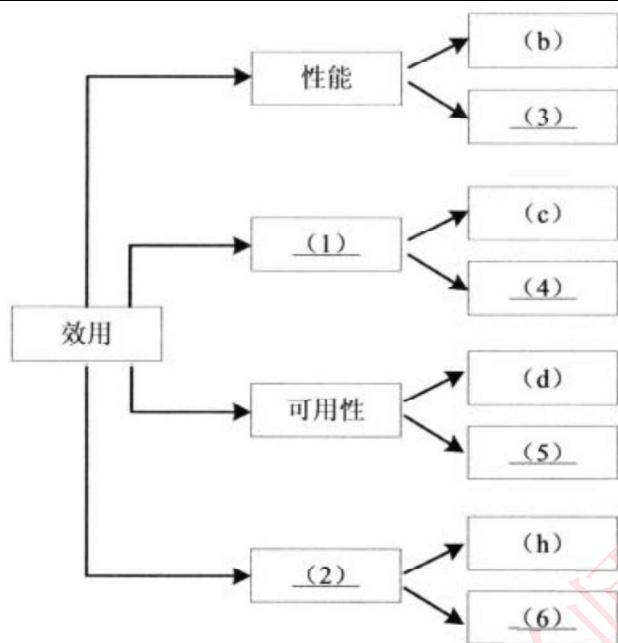


图 1-1 在线软件开发系统效用树

2020 年下半年试题 2

某企业委托软件公司开发一套包裹信息管理系统，以便于对该企业通过快递收发的包裹信息进行统一管理，在系统设计阶段，需要对不同快递信息的包裹单信息进行建模，其中，邮政包裹单如图 2-1 所示：

国内普通包裹详情单（通知单联）				邮1106	
				接收局号码：	
PA 05837748444		内件品名及数量			
收件人	姓名： 电话：		是否保价 是 ☐ 否 ☐		收寄人名章：
	单位名称：		保价金额： 元		
	详细地址：				
寄件人	姓名： 电话：		寄件人声明：同意并遵守背面的“使用须知”。如包裹无法投递，按如下选择处理： ☐ 退还寄件人 ☐ 抛弃处理	重量： 克	
	单位名称：		签字：	资费： 元	
	详细地址：			挂号费： 元	
	用户代码： 邮政编码：			保价费： 元	
PA 0583-7748 4 44		检查人员名章	回执费： 元	总计： 元	

图 2-1 包裹单示意图

[问题 1]

请说明关系型数据库开发中，逻辑数据模型设计过程包含哪些任务？根据图 2-1 包裹详情单应该涉及出哪些关系模式的名称，并指出每个关系模式的主键属性。

[问题 2]

请说明什么是超类实体？结合图中包裹单信息，试设计一种超类实体，给出完整的属性列表。

[问题 3]

请说明什么是派生属性？结合图中包裹单信息说明哪个属性是派生属性。

2020 年下半年试题 3

阅读以下关于开放式嵌入式软件架构设计的相关描述，回答问题 1 至问题 3

【说明】

某公司一直从事宇航系统研制任务，随着宇航产品综合化、网络化技术发展的需要，公司的业务量急剧增加，研制新的软件架构已迫在眉睫。公司架构师王工广泛调研了多种现代架构的基础，建议采用基于 FACE（Future Airborne Capability Environment）的宇航系统开放式软件架构，以实现宇航系统的跨平台复用，实现宇航软件高质量、低成本的开发。公司领导肯定了王工的提案，并指出公司要全面实施基于 FACE 的开放式软件架构，应注意每个具体项目在实施中如何有效实现从需求到架构设计的关系，掌握基于软件需求的软件架构设计方法，并做好开放式软件架构中各段间的接口标准化设计工作。

【问题 1】（9 分）

王工指出，软件开发中需求分析是根本，架构设计是核心，不考虑软件需求便进行软件架构设计很可能导致架构设计的失败，因此，如何把软件需求映射到软件架构至关重要。请从描述语言、非功能性需求描述、需求和架构的一致性等三个方面，用 300 字以内的文字说明软件需求到架构的映射存在哪些难点。

【问题 2】（10 分）

图 3-1 是王工给出的 FACE 架构布局，包括操作系统、I/O 服务、平台服务、传输服务和可移植组件等 5 个段；操作系统、I/O 和传输等 3 个标准接口。请分析图 3-1 给出的 FACE 架构的相关信息，用 300 字以内的文字简要说明 FACE 5 个段的含义。

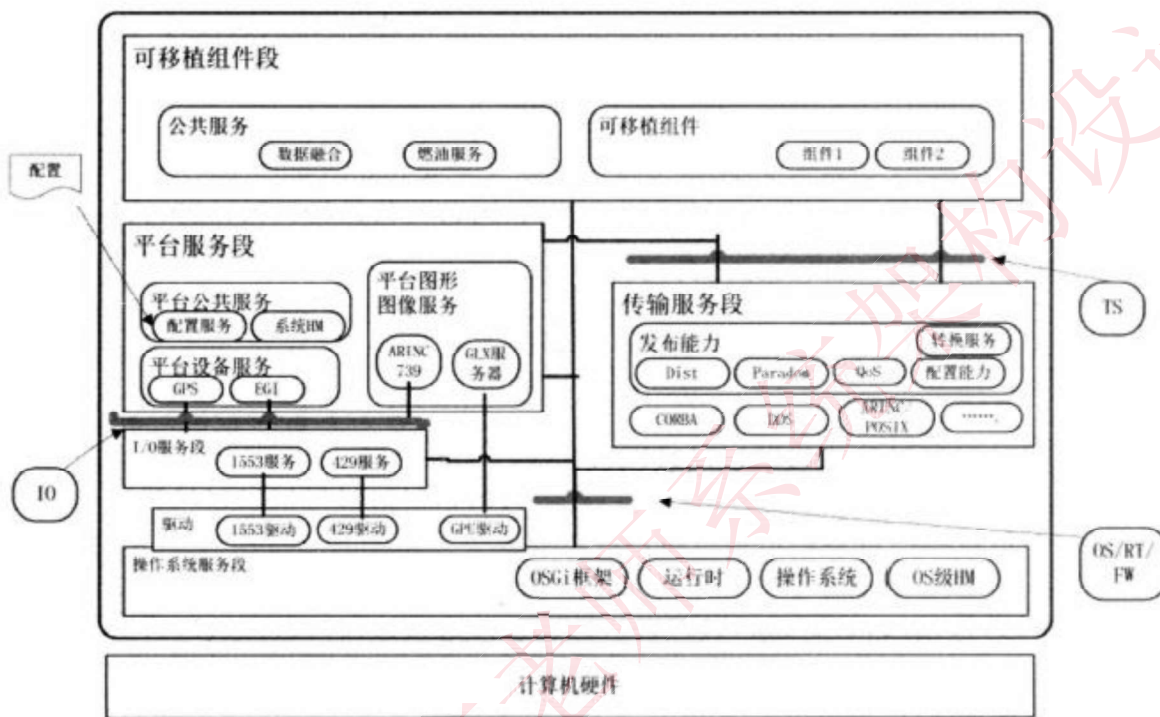


图 3-1 FACE 架构

【问题 3】（6 分）

FACE 架构的核心能力是可支持应用程序的跨平台执行和可移植性, 要达到可移植能力, 必须解决应用程序的紧耦合和封装的障碍。请用 200 字以内的文字简要说明在可移植性上, 应用程序的紧耦合和封装问题的主要表现分别是什么, 并给出解决方案。

2020 年下半年试题 4

阅读以下关于数据库缓存的叙述, 在答题纸上回答问题 1 至问题 3。

【说明】

某互联网文化发展公司因业务发展, 需要建立网上社区平台, 为用户提供一个对网络文化产品(如互联网小说、电影、漫画等)进行评论、交流的平台。该平台的部分功能如下:

- (a) 用户帖子的评论计数器;
- (b) 支持粉丝列表功能;
- (c) 支持标签管理;
- (d) 支持共同好友功能等;
- (e) 提供排名功能, 如当天最热前 10 名帖子排名、热搜榜前 5 排名等;
- (f) 用户信息的结构化存储;
- (g) 提供好友信息的发布/订阅功能。

该系统在性能上需要考虑高性能、高并发, 以支持大量用户的同时访问。开发团队经过综合考虑, 在数据管理上决定采用 Redis+数据库(缓存+数据库)的解决方案。

【问题 1】（10 分）

Redis 支持丰富的数据类型，并能够提供一些常见功能需求的解决方案。请选择题干描述的(a)~(g)功能选项，填入表 4-1 中(1)~(5)的空白处。

表 4-1 Redis 数据类型与业务功能对照表

数据类型	存储的值	可实现的业务功能
STRING	字符串、整数或浮点数	(1)
LIST	列表	(2)
SET	无序集合	(3)
HASH	包括键值对的无序散列表	(4)
ZSET	有序集合	(5)

【问题 2】(7 分)

该网上社区平台需要为用户提供 7*24 小时的不间断服务。同时在系统出现宕机等故障时，能在最短时间内通过重启等方式重新建立服务。为此，开发团队选择了 Redis 持久化支持。Redis 有两种持久化方式，分别是 RDB(Redis DataBase)持久化方式和 AOF(Append Only File)持久化方式。开发团队最终选择了 RDB 方式。

请用 200 字以内的文字，从磁盘更新频率、数据安全、数据一致性、重启性能和数据文件大小五个方面比较两种方式，并简要说明开发团队选择 RDB 的原因。

【问题 3】(8 分)

缓存中存储当前的热点数据，Redis 为每个 KEY 值都设置了过期时间，以提高缓存命中率。为了清除非热点数据，Redis 选择“定期删除+惰性删除”策略。如果该策略失效，Redis 内存使用率会越来越高，一般应采用内存淘汰机制来解决。

请用 100 字以内的文字简要描述该策略的失效场景，并给出三种内存淘汰机制。

2020 年下半年试题 5

阅读以下关于 Web 系统架构设计的叙述，在答题纸上回答问题 1 至问题 3。

【说明】

某公司拟开发一款基于 Web 的工业设备监测系统，以实现对多种工业设备数据的分类采集、运行状态监测以及相关信息的管理。该系统应具备以下功能：

现场设备状态采集功能：根据数据类型对设备监测指标状态信号进行分类采集；

设备采集数据传输功能：利用可靠的传输技术，实现将设备数据从制造现场传输到系统后台；

设备监测显示功能：对设备的运行状态、工作状态以及报警状态进行监测并提供相应的图形化显示界面；

设备信息管理功能：支持设备运行历史状态、报警记录、参数信息的查询。

同时，该系统还需满足以下非功能性需求：

(a) 系统应支持大于 100 个工业设备的并行监测；

(b) 设备数据从制造现场传输到系统后台的传输时间小于 1s；

(c) 系统应 7*24 小时工作；

- (d)可抵御常见 XSS 攻击；
- (e)系统在故障情况下，应在 0.5 小时内恢复；
- (f)支持数据审计。

面对系统需求，公司召开项目组讨论会议，制定系统设计方案，最终决定采用三层拓扑结构，即现场设备数据采集层、Web 监测服务层和前端 Web 显示层。

【问题 1】(6 分)

请按照性能、安全性和可用性等三类非功能性需求分类，选择题干描述的(a)～(f)填入(1)～(3)

表 5-1 非功能性需求归类表

非功能性需求类别	非功能性需求
性能	(1)
安全性	(2)
可用性	(3)

【问题 2】(14 分)

该系统的 Web 监测服务层拟采用 SSM (spring+spring MVC+Mybatis)框架进行系统研发。SSM 框架的工作流程图如图 5.1 所示，请从下面给出的(a)～(k)中进行选择，补充完善图 5-1 中(1)～(7)处空白的内容。

- (a)Connection Pool
- (b)Struts2
- (c)Persistent Layer
- (d)Mybatis
- (e)HTTP
- (f)MVC
- (g)Kafka
- (h)View Layer
- (i)JSP
- (j)Controller Layer
- (k)Spring

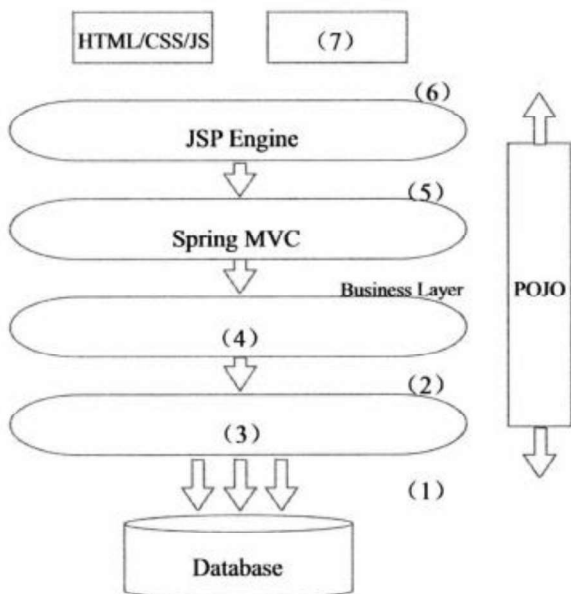


图 5-1 SSM 框架工作流程图

【问题 3】(5 分)

该工业设备检测系统拟采用工业控制领域中统一的数据访问机制，实现与多种不同设备的数据交互，请用 200 字以内的文字说明采用标准的数据访问机制的原因。

2019 年下半年试题 1

【说明】

某电子商务公司为了更好地管理用户，提升企业销售业绩，拟开发一套用户管理系统。该系统的基本功能是根据用户的消费级别、消费历史、信用情况等指标将用户划分为不同的等级，并针对不同等级的用户提供相应的折扣方案。在需求分析与架构设计阶段，电子商务公司提出的需求、质量属性描述和架构特性如下：

- (a) 用户目前分为普通用户、银卡用户、金卡用户和白金用户四个等级，后续需要能够根据消费情况进行动态调整；
- (b) 系统应该具备完善的安全防护措施，能够对黑客的攻击行为进行检测与防御；
- (c) 在正常负载情况下，系统应在 0.5 秒内对用户的商品查询请求进行响应；
- (d) 在各种节假日或公司活动中，针对所有级别用户，系统均能够根据用户实时的消费情况动态调整折扣力度；
- (e) 系统主站点断电后，应在 5 秒内将请求重定向到备用站点；
- (f) 系统支持中文昵称，但用户名要求必须以字母开头，长度不少于 8 个字符；
- (g) 当系统发生网络失效后，需要在 15 秒内发现错误并启用备用网络；
- (h) 系统在展示商品的实时视频时，需要保证视频画面具有 1024x768 像素的分辨率，40 帧/秒的速率；
- (i) 系统要扩容时，应保证在 10 人月内完成所有的部署与测试工作；
- (j) 系统应对用户信息数据库的所有操作都进行完整记录；

(k)更改系统的 Web 界面接口必须在 4 人周内完成；

(l)系统必须提供远程调试接口，并支持远程调试。

在对系统需求、质量属性描述和架构特性进行分析的基础上，该系统架构师给出了两种候选的架构设计方案，公司目前正在组织相关专家对系统架构进行评估。

问题 1 （13 分）

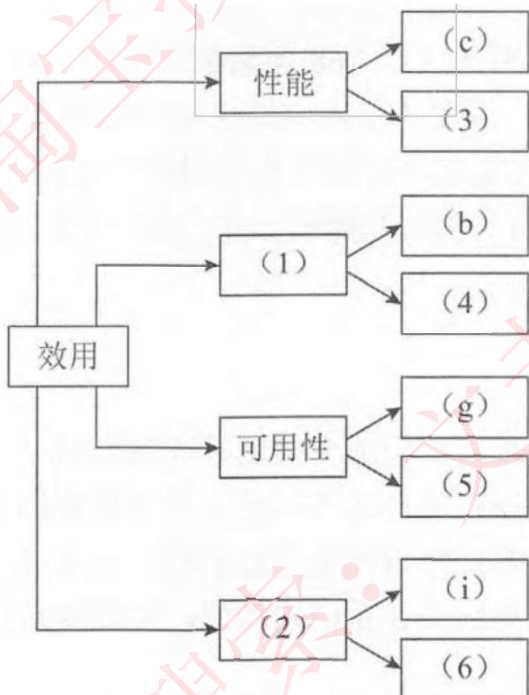
针对用户级别与折扣规则管理功能的架构设计问题，李工建议采用面向对象的架构风格，而王工则建议采用基于规则的架构风格。请指出该系统更适合采用哪种架构风格，并从用户级别、折扣规则定义的灵活性、可扩展性和性能三个方面对这两种架构风格进行比较与分析，填写表 1-1 中的 (1)~(3)空白处。

表 1-1 两种架构风格的比较与分析

架构风格名称	灵活性	可扩展性	性能
面对对象	将用户级别、折扣规则等封装为对象，在系统启动时加载	(2)	(3)
基于规则	(1)	加入新的用户级别和折扣规则时只需要定义新的规则，解释规则即可进行扩展	需要对用户级别与折扣规则进行实时解释、性能较差

问题 2 （12 分）

在架构评估过程中，质量属性效用树 (utility tree) 是对系统质量属性进行识别和优先级排序的重要工具。请将合适的质量属性名称填入图 1-1 中 (1)、(2) 空白处，并选择题干描述的 (a)~(l) 填入 (3)~(6) 空白处，完成该系统的效用树。



2019 年下半年试题 2

【说明】

某软件企业为快餐店开发一套在线订餐管理系统，主要功能包括：

(1) 在线订餐：已注册客户通过网络在线选择快餐店所提供的餐品种类和数量后提交订单，系统显示订单费用供客户确认，客户确认后支付订单所列各项费用。

(2) 厨房备餐：厨房接收到客户已付款订单后按照订单餐品列表选择各类食材进行餐品加工。

(3) 食材采购：当快餐店某类食材低于特定数量时自动向供应商发起采购信息，包括食材类型和数量，供应商接收到采购信息后按照要求将食材送至快餐店并提交已采购的食材信息，系统自动更新食材库存。

(4) 生成报表：每个周末和月末，快餐店经理会自动收到系统生成的统计报表，报表中详细列出了本周或本月订单的统计信息以及库存食材的统计信息。

现采用数据流图对上述订餐管理系统进行分析与设计，系统未完成的 0 层数据流图如图 2-1 所示。

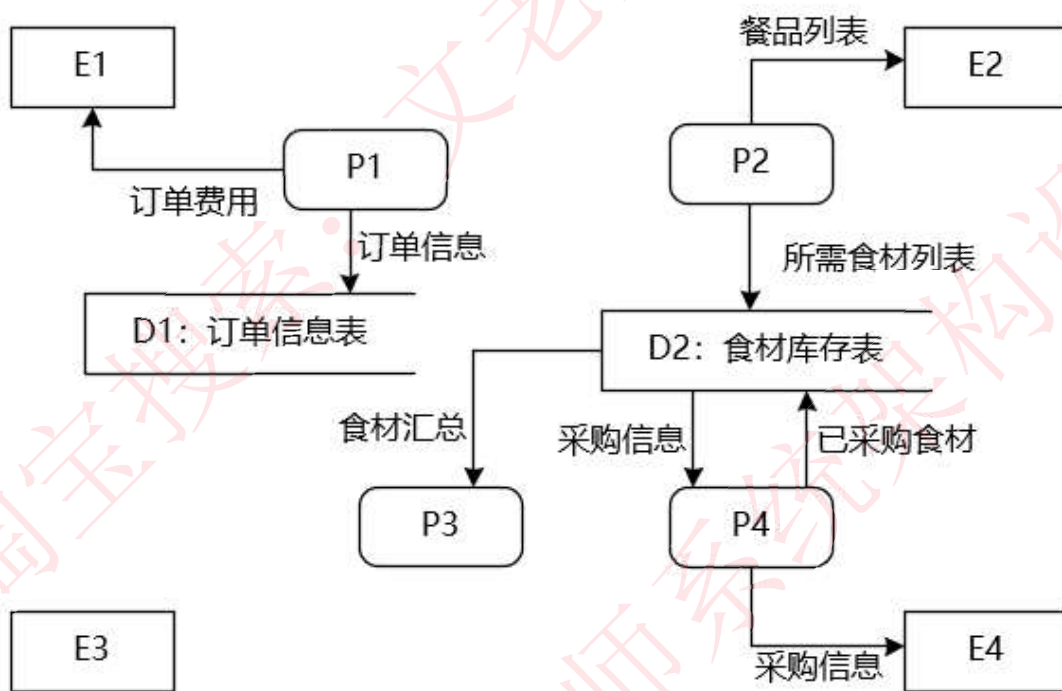


图2-1

问题 1（8 分）

根据订餐管理系统功能说明，请在图 2-1 所示数据流图中给出外部实体 E1~E4 和加工 P1~P4 的具体名称。

问题 2（8 分）

根据数据流图规范和订餐管理系统功能说明，请说明在图 2-1 中需要补充哪些数据流可以构造出完整的 0 层数据流图。

问题 3（9 分）

根据数据流图的含义，请说明数据流图和系统流程图之间有哪些方面的区别。

2019 年下半年试题 3

【说明】

信息物理系统 (Cyber Physical Systems, CPS) 技术已成为未来宇航装备发展的重点关键技术之一。某公司长期从事嵌入式系统的研制工作，随着公司业务范围不断扩展，公司决定进入宇航装备的研制领域。为了做好前期准备，公司决定让王工程师负责编制公司进军宇航装备领域的战略规划。王工经调研和分析，认为未来宇航装备将向着网络化、智能化和综合化的目标发展，CPS 将会是宇航装备的核心技术，公司应构建基于 CPS 技术的新产品架构，实现超前的技术战略储备。

问题 1 (9 分)

通常 CPS 结构分为感知层、网络层和控制层，请用 300 字以内文字说明 CPS 的定义，并简要说明各层的含义。

问题 2 (10 分)

王工在提交的战略规划中指出：飞行器中的电子设备是一个大型分布式系统，其传感器、控制器和采集器分布在飞机各个部位，相互间采用高速总线互连，实现子系统间的数据交换，而飞行员或地面指挥系统根据飞行数据的汇总决策飞行任务的执行。图 3-1 给出了飞行器系统功能组成图。

请参考图 3-1 给出的功能图，依据你所掌握的 CPS 知识，说明以下所列的功能分别属于 CPS 结构中的哪层，哪项功能不属于 CPS 任何一层。

- | | | |
|------------|-----------|------------|
| 1. 飞行传感器管理 | 2. 步进电机控制 | 3. 显控 |
| 4. 发电机控制 | 5. 环控 | 6. 配电管理 |
| 7. 转速传感器 | 8. 传感器总线 | 9. 飞行员 |
| | | 10. 火警信号探测 |



图 3-1 飞行器系统功能组成图

问题 3（6 分）

王工在提交的战略规划中指出：未来宇航领域装备将呈现网络化、智能化和综合化等特征，形成集群式的协同能力，安全性尤为重要。在宇航领域的 CPS 系统中，不同层面上都会存在一定的安全威胁。请用 100 字以内文字说明 CPS 系统会存在哪三类安全威胁，并对每类安全威胁至少举出两个例子说明。

2019 年下半年试题 4

【说明】

某初创企业的主营业务是为用户提供高度个性化的商品订购业务，其业务系统支持 PC 端、手机 App 等多种访问方式。系统上线后受到用户普遍欢迎，在线用户数和订单数量迅速增长，原有的关系数据库服务器不能满足高速并发的业务要求。

为了减轻数据库服务器的压力，该企业采用了分布式缓存系统，将应用系统经常使用的数据放置在内存，降低对数据库服务器的查询请求，提高了系统性能。在使用缓存系统的过程中，企业碰到了一系列技术问题。

问题 1（11 分）

该系统使用过程中，由于同样的数据分别存在于数据库和缓存系统中，必然会造成数据同步或数据不一致性的问题。该企业团队为解决这个问题，提出了如下解决思路：应用程序读数据时，首先读缓存，当该数据不在缓存时，再读取数据库；应用程序写数据时，先写缓存，成功后再写数据库；或者先写数据库，再写缓存。

王工认为该解决思路并未解决数据同步或数据不一致性的问题，请用 100 字以内的文字解释其原因。

王工给出了一种可以解决该问题的数据读写步骤如下：

读数据操作的基本步骤：

1. 根据 key 读缓存；
2. 读取成功则直接返回；
3. 若 key 不在缓存中时，根据 key (a) ；
4. 读取成功后， (b) ；
5. 成功返回。

写数据操作的基本步骤：

1. 根据 key 值写 (c) ；
2. 成功后 (d) ；
3. 成功返回。

请填写完善上述步骤中 (a)~(d) 处的空白内容。

问题 2 (8 分)

缓存系统一般以 key/value 形式存储数据，在系统运维中发现，部分针对缓存的查询，未在缓存系统中找到对应的 key，从而引发了大量对数据库服务器的查询请求，最严重时甚至导致了数据库服务器的宕机。

经过运维人员的深入分析，发现存在两种情况：

- (1) 用户请求的 key 值在系统中不存在时，会查询数据库系统，加大了数据库服务器的压力；
- (2) 系统运行期间，发生了黑客攻击，以大量系统不存在的随机 key 发起了查询请求，从而导致了数据库服务器的宕机。

经过研究，研发团队决定，当在数据库中也未查找到该 key 时，在缓存系统中为 key 设置空值，防止对数据库服务器发起重复查询。

请用 100 字以内文字说明该设置空值方案存在的问题，并给出解决思路。

问题 3 (6 分)

缓存系统中的 key 一般会存在有效期，超过有效期则 key 失效；有时也会根据 LRU 算法将某些 key 移出内存。当应用软件查询 key 时，如 key 失效或不在内存，会重新读取数据库，并更新缓存中的 key。

运维团队发现在某些情况下，若大量的 key 设置了相同的失效时间，导致缓存在同一时刻众多 key 同时失效，或者瞬间产生对缓存系统不存在 key 的大量访问，或者缓存系统重启等原因，都会造成数据库服务器请求瞬时爆量，引起大量缓存更新操作，导致整个系统性能急剧下降，进而造成整个系统崩溃。

请用 100 字以内文字，给出解决该问题的两种不同思路。

2019 年下半年试题 5

【说明】

某公司拟开发一个物流车辆管理系统，该系统可支持各车辆实时位置监控、车辆历史轨迹管理、违规违章记录管理、车辆固定资产管理、随车备品及配件更换记录管理、车辆寿命管理等功能需求。其非功能性需求如下：

- (1) 系统应支持大于 50 个终端设备的并发请求；
- (2) 系统应能够实时识别车牌，识别时间应小于 1s；
- (3) 系统应 7X24 小时工作；
- (4) 具有友好的用户界面；
- (5) 可抵御常见 SQL 注入攻击；
- (6) 独立事务操作响应时间应小于 3s；
- (7) 系统在故障情况下，应在 1 小时内恢复；
- (8) 新用户学习使用系统的时间少于 1 小时。

面对系统需求，公司召开项目组讨论会议，制订系统设计方案，最终决定基于分布式架构设计实现该物流车辆管理系统，应用 Kafka、Redis 数据缓存等技术实现对物流车辆自身数据、业务数据进行快速、高效的处理。

问题 1（4 分）

请将上述非功能性需求 (1)~(8) 归类到性能、安全性、可用性、易用性这四类非功能性需求。

问题 2（14 分）

经项目组讨论，完成了该系统的分布式架构设计，如图 5-1 所示。请从下面给出的 (a)~(j) 中进行选择，补充完善图 5-1 中 (1)~(7) 处空白的内容。

- | | | | |
|-------------|----------------|----------------|--------------|
| (a) 数据存储层 | (b) Spuct2 | (c) 负载均衡层 | (d) 表现层 |
| (e) HTTP 协议 | (f) Rcdis 数据缓存 | (g) Kafka 分发消息 | (h) 分布式通信处理层 |
| (i) 逻辑处理层 | (j) CDN 内容分发 | | |

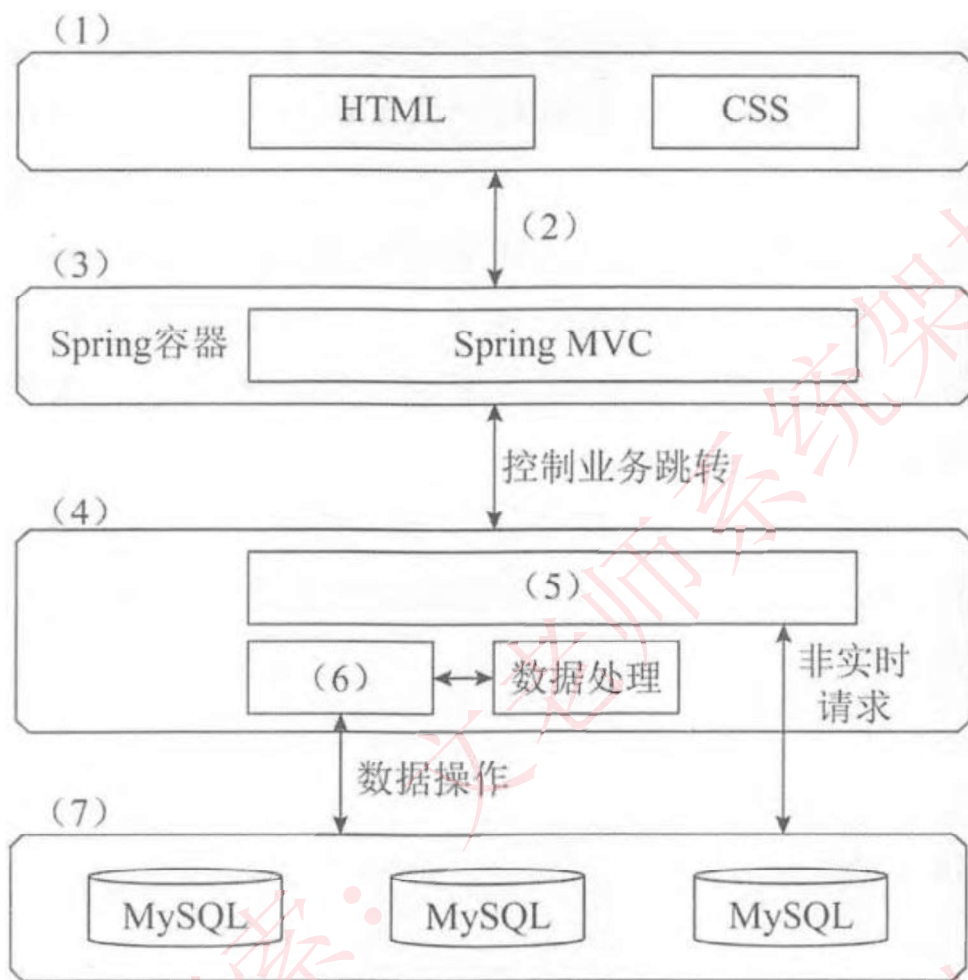


图 5-1 物流车辆管理系统架构设计图

问题 3 (7 分)

该物流车辆管理系统需抵御常见的 SQL 注入攻击, 请用 200 字以内的文字说明什么是 SQL 注入攻击, 并列举出两种抵御 SQL 注入攻击的方式。

2018 年下半年试题 1

试题一

阅读以下关于软件系统设计的叙述, 在答题纸上回答问题 1 至问题 3。

【说明】

某文化产业集团委托软件公司开发一套文化用品商城系统, 业务涉及文化用品销售、定制、竞拍和点评等板块, 以提升商城的信息化建设水平。该软件公司组织项目组完成了需求调研, 现已进入到系统架构设计阶段。考虑到系统需求对架构设计决策的影响, 项目组先列出了可能影响系统架构设计的部分需求如下:

- (a) 用户界面支持用户的个性化定制;
- (b) 系统需要支持当前主流的标准和服务, 特别是通信协议和平台接口;
- (c) 用户操作的响应时间应不大于 3 秒, 竞拍板块不大于 1 秒;
- (d) 系统具有故障诊断和快速恢复能力;

- (e) 用户密码需要加密传输；
- (f) 系统需要支持不低于 2G 的数据缓存；
- (g) 用户操作停滞时间超过一定时限需要重新登录验证；
- (h) 系统支持用户选择汉语、英语或法语三种语言之一进行操作。

项目组提出了两种系统架构设计方案：瘦客户端 C/S 架构和胖客户端 C/S 架构，经过对上述需求逐条分析和讨论，最终决定采用瘦客户端 C/S 架构进行设计。

【问题 1】（8 分）

在系统架构设计中，决定系统架构设计的非功能性需求主要有四类：操作性需求、性能需求、安全性需求和文化需求。请简要说明四类需求的含义。

【问题 2】（8 分）

根据表 1-1 的分类，将题干所给出的系统需求 (a) ~ (h) 分别填入 (1) ~ (4)。

表 1-1 需求分类

需求类别	系统需求
操作性需求	(1)
性能需求	(2)
安全性需求	(3)
文化需求	(4)

【问题 3】（9 分）

请说明瘦客户端 C/S 架构能够满足题干中给出的哪些系统需求（只需要回答出三个系统需求）。

2018 年下半年试题 2

阅读以下关于软件系统建模的叙述，在答题纸上回答问题 1 至问题 3。

【说明】

某公司欲建设一个房屋租赁服务系统，统一管理房主和租赁者的信息，提供快捷的租赁服务。本系统的主要功能描述如下：

1. 登记房主信息。记录房主的姓名、住址、身份证号和联系电话等信息，并写入房主信息文件。
2. 登记房屋信息。记录房屋的地址、房屋类型（如平房、带阳台的楼房、独立式住宅等）、楼层、租金及房屋状态（待租赁、已出租）等信息，并写入房屋信息文件。一名房主可以在系统中登记多套待租赁的房屋。
3. 登记租赁者信息。记录租赁者的个人信息，包括：姓名、性别、住址、身份证号和电话号码等，并写入租赁者信息文件。
4. 安排看房。已经登记在系统中的租赁者，可以从待租赁房屋列表中查询待租赁房屋信息。租赁者可以提出看房请求，系统安排租赁者看房。对于每次看房，系统会生成一条看房记录并将其写入看房记录文件中。
5. 收取手续费。房主登记完房屋后，系统会生成一份费用单，房主根据费用单交纳相应的费用。

2018 年下半年试题 3

阅读以下关于嵌入式实时系统相关技术的叙述，在答题纸上回答问题 1 和问题 2。

【说明】

某公司长期从事宇航领域嵌入式实时系统的软件研制任务。公司为了适应未来嵌入式系统网络化、智能化和综合化的技术发展需要，决定重新考虑新产品的架构问题，经理将论证工作交给王工负责。王工经调研和分析，完成了新产品架构设计方案，提交公司高层讨论。

【问题 1】(14 分)

王工提交的设计方案中指出：由于公司目前研制的嵌入式实时产品属于简单型系统，其嵌入式子系统相互独立，功能单一，时序简单。而未来满足网络化、智能化和综合化的嵌入式实时系统将是一种复杂系统，其核心特征体现为实时任务的机理、状态和行为的复杂性。简单任务和复杂任务的特征区分主要表现在十个方面。请参考表 3-1 给出的实时任务特征分类，用题干中给出的 (a) ~ (t) 20 个实时任务特征描述，补充完善表 3-1 给出的空 (1) ~ (14)。

- (a) 任务属性不会随时间变化而改变；
- (b) 任务的属性与时间相关；
- (c) 任务仅可以从非连续集中获取特征变量；
- (d) 任务变量域是连续的；
- (e) 功能原理不依赖于上下文；
- (f) 功能原理依赖于上下文；
- (g) 任务行为可以用 step-by-step 顺序分析方法来理解；
- (h) 许多任务在产生访问活动时相互间是并发处理的，很难用 step-bystep 方法分析；
- (i) 因果关系相互影响；
- (j) 行为特征依赖于大量的反馈机制；
- (k) 系统内构成、策略和描述是相似的；
- (l) 系统内存在许多不同的构成、策略和描述；
- (m) 功能关系是非线性的；
- (n) 功能关系是线性的；
- (o) 不同的子任务是相互独立的，任务内部仅存在少量的交互操作；
- (p) 不同的子任务有很高的交互操作，要把一个单任务的行为隔离开是困难的；
- (q) 域特征有非常整齐的原则和规则；
- (r) 许多不同的上下文依赖于规则；
- (s) 原理和规则在表面属性上很容易被识别；
- (t) 原理被覆盖、抽象，而不会在表面属性上被识别。

表 3-1 简单任务和复杂任务特征比较

特征分类	简单任务 (simple task)	复杂任务 (complex task)
静态/动态	(a)	(b)
连续/非连续	(1)	(2)
子系统的独立性	(3)	(4)
顺序/并行执行	(5)	(6)
单一性/混合性	(7)	(8)
工作原理	(9)	(10)
线性/非线性	(11)	(12)
上下文相关性	(13)	(14)
规律/不规律	(q)	(r)
表面属性	(s)	(t)

【问题 2】(11 分)

王工设计方案中指出：要满足未来网络化、智能化和综合化的需求，应该设计一种能够充分表达嵌入式系统行为的、且具有一定通用性的通信架构，以避免复杂任务的某些特征带来的通信复杂性。通常为了实现嵌入式系统中计算组件间的通信，在架构上需要一种简单的架构风格，用于屏蔽不同协议、不同硬件和不同结构组成所带来的复杂性。图 3-1 给出了一种“腰 (Waistline)”型通信模式的架构风格。腰型架构的关键是基本消息通信 (BMTS)，通常 BMTS 的消息与时间属性相关，支持事件触发消息、速率约束消息和时间触发消息。

请说明基于 BMTS 的消息通信网络的主要特征和上述三种消息的基本含义，并举例给出两种具有时间触发消息能力的网络总线。

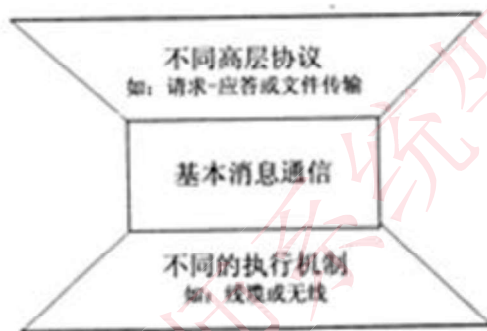


图 3-1 “腰”型通信模式架构风格

2018 年下半年试题 5

阅读以下关于 Web 系统设计的叙述，在答题纸上回答问题 1 至问题 3。

【说明】

某银行拟将以分行为主体的银行信息系统，全面整合为由总行统一管理维护的银行信息系统，实现统一的用户账户管理、转账汇款、自助缴费、理财投资、贷款管理、网上支付、财务报表分析等业务功能。但是，由于原有以分行为主体的银行信息系统中，多个业务系统采用异构平台、数据库和中间件，使用的报文交换标准和通信协议也不尽相同，使用传统的 EAI 解决方案根本无法实现新的业务模式下异构系统间灵活的交互和集成。因此，为了以最小的系统改进整合现有的基于不同技术实现

的银行业务系统，该银行拟采用基于 ESB 的面向服务架构（SOA）集成方案实现业务整合。

【问题 1】（7 分）

请说明什么是面向服务架构（SOA）以及 ESB 在 SOA 中的作用与特点。

【问题 2】（12 分）

基于该信息系统整合的实际需求，项目组完成了基于 SOA 的银行信息系统架构设计方案。该系统架构图如图 5-1 所示：

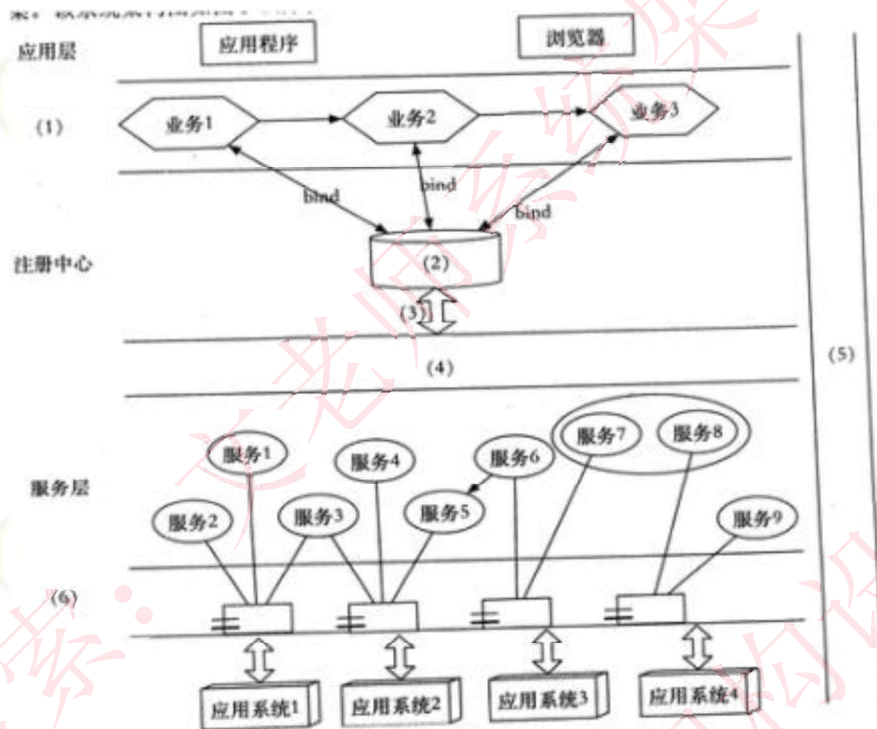


图 5-1 基于 SOA 的银行信息系统架构设计

请从 (a) ~ (j) 中选择相应内容填入图 5-1 的 (1) ~ (6)，补充完善架构设计图。

- (a) 数据层
- (b) 界面层
- (c) 业务层
- (d) bind
- (e) 企业服务总线 ESB
- (f) XML
- (g) 安全验证和质量管理
- (h) publish
- (i) UDDI\
- (j) 组件层
- (k) BPEL

【问题 3】（6 分）

针对银行信息系统的数据交互安全性需求，列举 3 种可实现信息系统安全保障的措施。

2017 年下半年试题 1

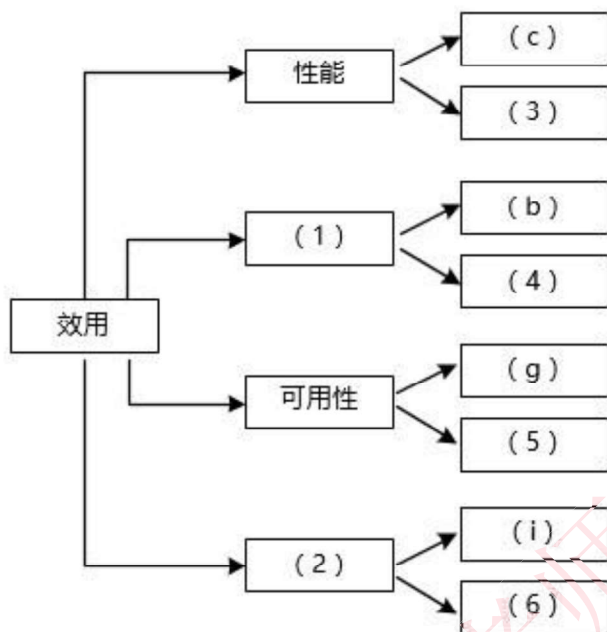
阅读以下关于软件架构评估的叙述，在答题纸上回答问题 1 和问题 2。

【说明】某单位为了建设健全的公路桥梁养护管理档案，拟开发一套公路桥梁在线管理系统。在系统的需求分析与架构设计阶段，用户提出的需求、质量属性描述和架构特性如下：

- (a) 系统用户分为高级管理员、数据管理员和数据维护员等三类；
- (b) 系统应该具备完善的安全防护措施，能够对黑客的攻击行为进行检测与防御；
- (c) 正常负载情况下，系统必须在 0.5 秒内对用户的查询请求进行响应；
- (d) 对查询请求处理时间的要求将影响系统的数据传输协议和处理过程的设计；
- (e) 系统的用户名不能为中文，要求必须以字母开头，长度不少于 5 个字符；
- (f) 更改系统加密的级别将对安全性和性能产生影响；
- (g) 网络失效后，系统需要在 10 秒内发现错误并启用备用系统；
- (h) 查询过程中涉及到的桥梁与公路的实时状态视频传输必须保证画面具有 1024*768 的分辨率，40 帧 / 秒的速率；
- (i) 在系统升级时，必须保证在 10 人月内可添加一个新的消息处理中间件；
- (j) 系统主站点断电后，必须在 3 秒内将请求重定向到备用站点；
- (k) 如果每秒钟用户查询请求的数量是 10 个，处理单个请求的时间为 30 毫秒，则系统应保证在 1 秒内完成用户的查询请求；
- (l) 对桥梁信息数据库的所有操作都必须进行完整记录；
- (m) 更改系统的 Web 界面接口必须在 4 人周内完成；
- (n) 如果“养护报告生成”业务逻辑的描述尚未达成共识，可能导致部分业务功能模块规则的矛盾，影响系统的可修改性
- (o) 系统必须提供远程调试接口，并支持系统的远程调试。

在对系统需求，质量属性描述和架构特性进行分析的基础上，系统的架构师给出了三个候选的架构设计方案，公司目前正在组织系统开发的相关人员对系统架构进行评估。

【问题 1】(12 分) 在架构评估过程中，质量属性效用树 (utility tree) 是对系统质量属性进行识别和优先级排序的重要工具。请给出合适的质量属性，填入图 1-1 中 (1)、(2) 空白处；并选择题干描述的 (a)~(o)，填入 (3)~(6) 空白处，完成该系统的效用树。



【问题 2】(13 分) 在架构评估过程中，需要正确识别系统的架构风险、敏感点和权衡点，并进行合理的架构决策。请用 300 字以内的文字给出系统架构风险、敏感点和权衡点的定义，并从题干 (a) ~ (o) 中分别选出 1 个对系统架构风险、敏感点和权衡点最为恰当的描述。

2017 年下半年试题 2

阅读以下关于软件系统设计的叙述，在答题纸上回答问题 1 至问题 3。

【说明】某软件企业受该省教育部门委托建设高校数字化教育教学资源共享平台，实现以众筹众创的方式组织省内普通高校联合开展教育教学资源内容建设，实现全省优质教学资源整合和共享。该资源共享平台的主要功能模块包括：

(1) 统一身份认证模块：提供统一的认证入口，为平台其他核心业务模块提供用户管理、身份认证、权限分级和单点登录等功能；

(2) 共享资源管理模块：提供教学资源申报流程服务，包括了资源申报、分类定制、资料上传、资源审核和资源发布等功能；

(3) 共享资源展示模块：提供教育教学共享资源的展示服务，包括资源导航、视频点播、资源检索、分类展示、资源评价和推荐等功能；

(4) 资源元模型管理模块：依据资源类型提供共享资源的描述属性、内容属性和展示属性，包括共享资源统一标准和规范、资源加工和在线编辑工具、数字水印和模板定制等功能；

(5) 系统综合管理模块：提供系统管理和维护服务，包括系统配置、数据备份恢复、资源导入导出和统计分析等功能。

项目组经过分析和讨论，决定采用基于 Java EE 的 MVC 模式设计资源共享平台的软件架构，如图 2-1 所示。

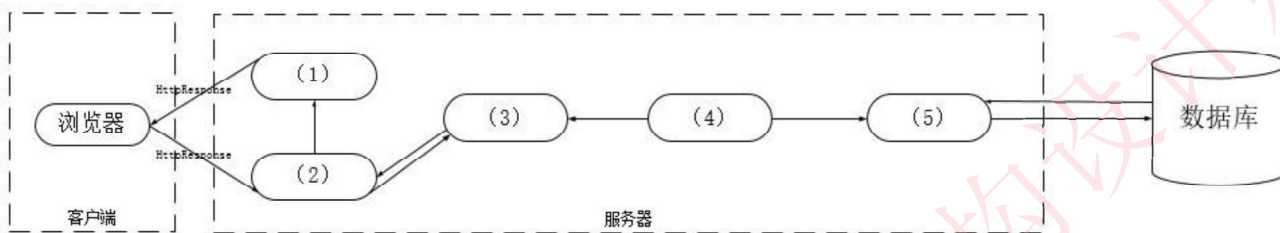


图 2-1 资源共享平台软件架构

【问题 1】 (9 分)

MVC 架构中包含哪三种元素，它们的作用分别是什么？请根据图 2-1 所示架构将 JavaEE 中 JSP、Servlet、Service、JavaBean、DAO 五种构件分别填入空 (1)~(5) 所示位置。

【问题 2】 (6 分)

项目组架构师王工提出在图 2-1 所示架构设计中加入 EJB 构件，采用企业级 JavaEE 架构开发资源共享平台。请说明 EJB 构件中的 Bean (构件) 分为哪三种类型，每种类型 Bean 的职责是什么。

【问题 3】 (10 分)

如果采用王工提出的企业级 JavaEE 架构，请说明下列 (a)~(e) 所给出的业务功能构件中，有状态和无状态构件分别包括哪些。

- (a) Identification Bean (身份认证构件)
- (b) ResPublish Bean (资源发布构件)
- (c) ResRetrieval Bean (资源检索构件)
- (d) OnlineEdit Bean (在线编辑构件)
- (e) Statistics Bean (统计分析构件)

2017 年下半年试题 3

阅读以下关于机器人操作系统架构的描述，回答问题 1 至问题 3

【说明】 随着人工智能技术的发展，工业机器人已成为当前工业界的热点研究对象。某宇航设备公司为了扩大业务范围，决策层研究决定准备开展工业机器人研制新业务。公司将论证工作交给了软件架构师王工，王工经过分析和调研，从机器人市场现状、领域需求、组成及关键技术和风险分析等方面开展了综合论证。论证报告指出：首先，为了保障本公司机器人研制的持续性，应根据领域需求选择一种适应的设计架构；其次，为了规避风险，公司的研制工作不能从零开始，应该采用国际开源社区所提供机器人操作系统 (Robot Operating System, ROS) 作为机器人开发的基本平台。

在讨论会上，架构师李工提出不同意见，他认为公司针对宇航领域已开发了某款嵌入式实时操作系统，且被多种宇航装备使用，可靠性较高。因此应该采用现有架构体系作为机器人的开发平台。会上王工说明了机器人操作系统与该款操作系统的差别，要沿用需要进行改造，投入较大。经过激烈讨论，公司领导同意了王工采用 ROS 的意见。

【问题 1】（5 分）

王工拟采用的 ROS 具有分布式进程框架，以点对点设计以及服务和节点管理器方式，使得执行程序可以各自独立地设计，松散地、实时地组合起来。这些进程可以按照功能包和功能包集的方式分组，因而可以容易地分享和发布。请用 400 字以内文字说明 ROS 与嵌入式实时操作系统的共同点，以及在实时性和任务通信方式两个方面的差异。

【问题 2】（10 分）

ROS 为应用程序间通信提供了主题(Topic)、服务 (Service)和动作 (Action) 三种消息通信方式，每种通信方式都有其特点。请将以下给出的三类通信的主要特点填入表 3-1 中(1)-(5) 的空白处，将答案写在答题纸上。

- (a) 适合用于传输传感器信息（数据流）
- (b) 能够知道是否调用成功
- (c) 一对多模式
- (d) 有握手信号
- (e) 服务执行完会有反馈
- (f) 可以监控长时间执行的进程
- (g) 较复杂
- (h) 可能让系统过载(数据太多)
- (i) 服务执行完之前，程序会等待
- (j) 建立通信较慢
- (k) 可能丢失数据

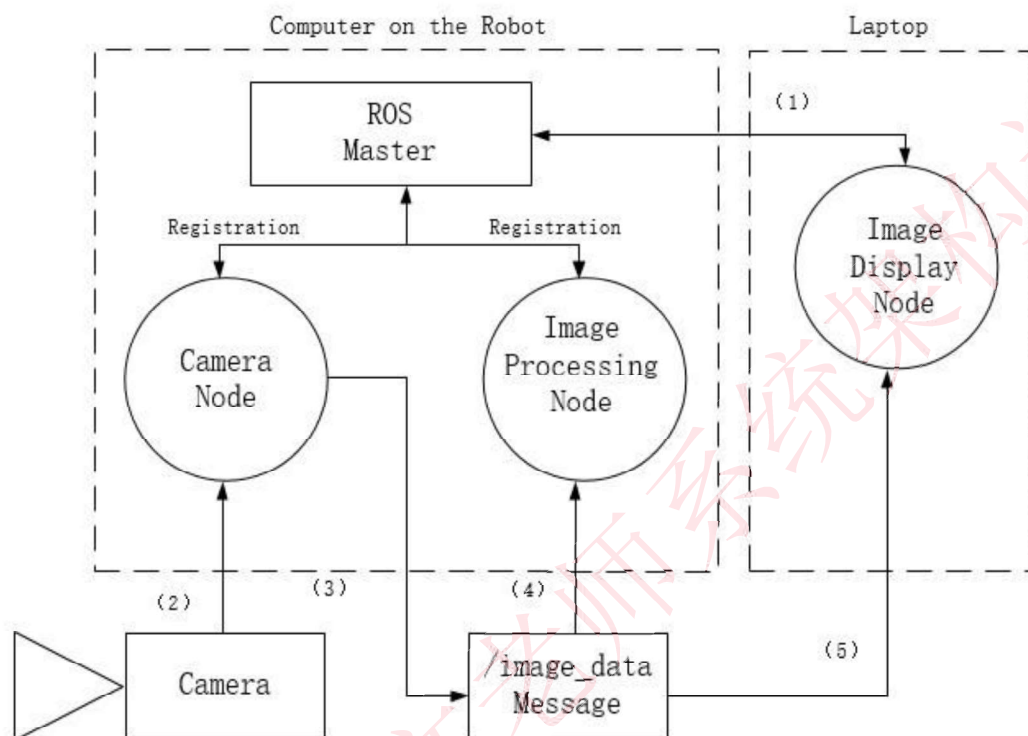
表 3-1 ROS 三类通信的主要特点

类型	特点
主题 (Topic)	(a) 适合用于传输传感器信息 (数据流)
	(1)
	(2)
	(h) 可能让系统过载 (数据太多)
服务 (Service)	(b) 能够知道是否调用成功
	(3)
	(e) 服务执行完会有反馈
	(4)
动作 (Action)	(5)
	(g) 较复杂
	(d) 有握手信号

【问题 3】 (10 分)

ROS 的架构定义了 ROS 系统由多个各自独立的节点(组件)组成,并且各个节点之间可以通过发布/订阅(Publish/Subscribe)消息模型进行通信。图 3-1 给出一个简单机器人结构实例,请根据以下文字描述,补充图 3-1 中(1)~(5)处空白,将答案写在答题纸上。

“机器人开始阶段,所有节点都要注册(Registration)到 Master 上,注册后,摄像头节点声明它要发布(Publish)一个叫做/image_data 的消息。另外两个节点(图像处理节点和图像显示节点)声明它们需要订阅(Subscribe)这个/image_data 消息。因此,一旦摄像头节点收到相机发送的数据(Data),就立即将数据/image_data 直接发送到另外两个节点。

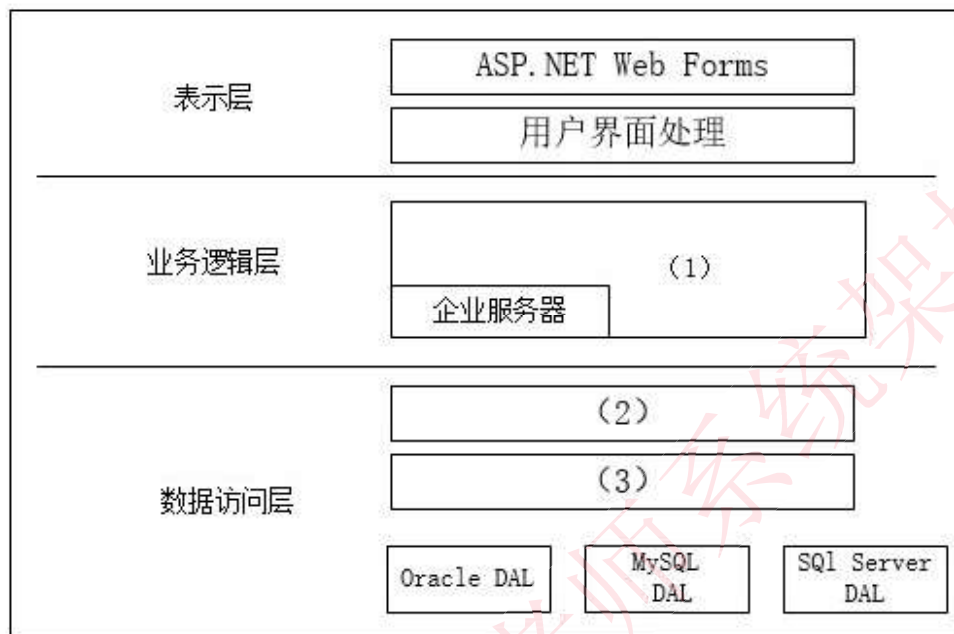


2017 年下半年试题 4

阅读以下关于数据库设计的叙述，在答题纸上回答问题 1 至问题 3。

【说明】某制造企业为拓展网上销售业务，委托某软件企业开发一套电子商务网站。初期仅解决基本的网上销售、订单等功能需求。该软件企业很快决定基于 .NET 平台和 SQL Server 数据库进行开发，但在数据库访问方式上出现了争议。王工认为应该采用程序在线访问的方式访问数据库；而李工认为本企业内部程序员缺乏数据库开发经验，而且应用简单，应该采用 ORM（对象关系映射）方式。最终经过综合考虑，该软件企业采用了李工的建议。

随着业务的发展，该电子商务网站逐渐发展成一个通用的电子商务平台，销售多家制造企业的产品，电子商务平台的功能也日益复杂。目前急需对该电子商务网站进行改造，以支持对多种异构数据库平台的数据访问，同时满足复杂的数据管理需求。该软件企业针对上述需求，对电子商务网站的架构进行了重新设计，新增加了数据访问层，同时采用工厂设计模式解决异构数据库访问的问题。新设计的系统架构如图 4-1 所示。



【问题 1】（9 分）

请用 300 字以内的文字分别说明数据库程序在线访问方式和 ORM 方式的优缺点，说明该软件企业采用 ORM 的原因。

【问题 2】（9 分）

请用 100 字以内的文字说明新体系架构中增加数据访问层的原因。请根据图 4-1 所示，填写图中空白处(1) - (3)。

【问题 3】（7 分）

应用程序设计中，数据库访问需要良好的封装性和可维护性，因此经常使用工厂设计模式来实现对数据库访问的封装。请解释工厂设计模式，并说明其优点和应用场景；请解释说明工厂模式在数据访问层中的应用。

2017 年下半年试题 5

阅读以下关于 Web 系统架构设计的叙述，在答题纸上回答问题 1 至问题 3。

【说明】

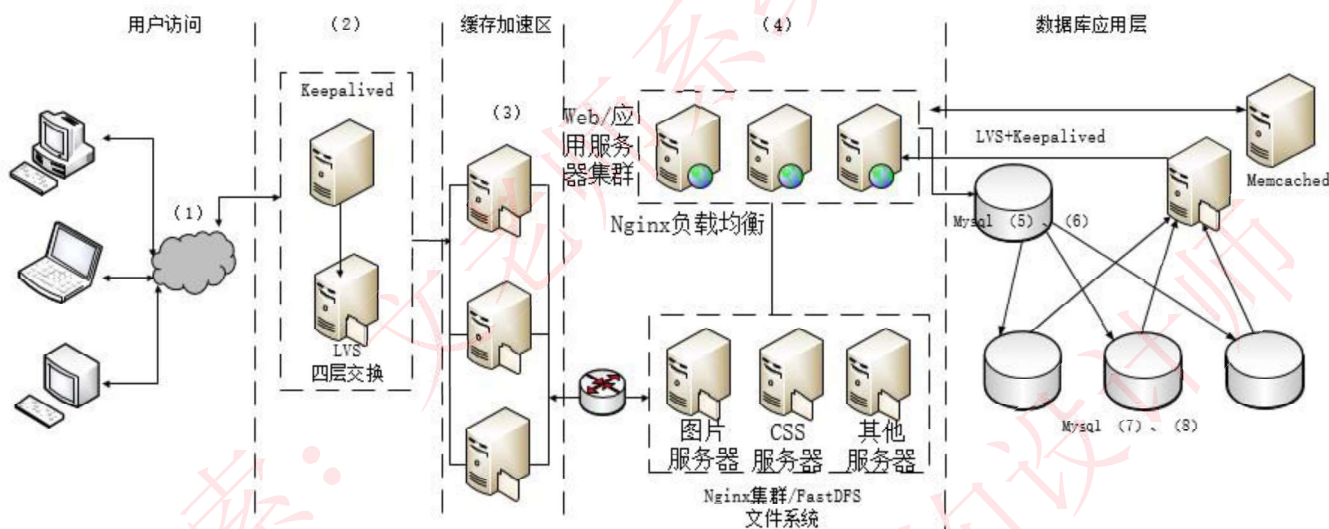
某电子商务企业因发展良好，客户量逐步增大，企业业务不断扩充，导致其原有的 B2C 商品交易平台已不能满足现有业务需求。因此，该企业委托某软件公司重新开发一套商品交易平台。该企业要求新平台应可适应客户从手机、平板设备、电脑等不同终端设备访问系统，同时满足电商定期开展“秒杀”、“限时促销”等活动的系统高并发访问量的需求。面对系统需求，软件公司召开项目组讨论会议，制定系统设计方案。讨论会议上，王工提出可以应用响应式 Web 设计满足客户从不同设备正确访问系统的需求。同时，采用增加镜像站点、CDN 内容分发等方式解决高并发访问量带来的问题。李工在王工的提议上补充，仅仅依靠上述外网加速技术不能完全解决高用户并发访问问题，如果访问量持续

增加，系统仍存在崩溃可能。李工提出应同时结合负载均衡、缓存服务器、Web 应用服务器、分布式文件系统、分布式数据库等方法设计系统架构。经过项目组讨论，最终决定综合王工和李工的思路，完成新系统的架构设计。

【问题 1】（5 分）

请用 200 字以内的文字描述什么是“响应式 Web 设计”，并列举 2 个响应式 Web 设计的实现方式。

【问题 2】（16 分）综合王工和李工的提议，项目组完成了新商品交易平台的系统架构设计方案。新系统架构图如图 5-1 所示。请从选项 (a) - (j) 中为架构图中 (1) - (8) 处空白选择相应的内容，补充支持高并发的 Web 应用系统架构设计图。



- (a) Web 应用层
- (b) 界面层
- (c) 负载均衡层
- (d) CDN 内容分发
- (e) 主数据库
- (f) 缓存服务器集群
- (g) 从数据库
- (h) 写操作
- (i) 读操作
- (j) 文件服务器集群

【问题 3】（4 分）

根据李工的提议，新的 B2C 商品交易平台引入了主从复制机制。请针对 B2C 商品交易平台的特点，简要叙述引入该机制的好处。

2016 年下半年试题 1

阅读以下关于软件架构设计的叙述，在答题纸上回答问题 1 至问题 3。

【说明】

某软件公司为某品牌手机厂商开发一套手机应用程序集成开发环境，以提高开发手机应用程序的质量和效率。在项目之初，公司的系统分析师对该集成开发环境的需求进行了调研和分析，具体描述如下：

a. 需要同时支持该厂商自行定义的应用编程语言的编辑、界面可视化设计、编译、调试等模块，这些模块产生的模型或数据格式差异较大，集成环境应提供数据集成能力。集成开发环境还要支持以适配方式集成公司现有的应用模拟器工具。

b. 经过调研，手机应用开发人员更倾向于使用 Windows 系统，因此集成开发环境的界面需要与 Windows 平台上的主流开发工具的界面风格保持一致

c. 支持相关开发数据在云端存储，需要在云端存储数据的机密性和完整性。

d. 支持用户通过配置界面依据自己的喜好修改界面风格，包括颜色、布局、代码高亮方式等，配置完成后无需重启环境。

e. 支持不同模型的自动转换。在初始需求中定义的机器性能条件下，对于一个包含 50 个对象的设计模型，将其转换为相应代码框架时所消耗时间不超过 5 秒。

f. 能够连续运行的时间不小于 240 小时，意外退出后能够在 10 秒之内自动重启。

g. 集成开发环境具有模块化结构，支持以模块为单位进行调试、测试与发布。

h. 支持应用开发过程中的代码调试功能：开发人员可以设置断点，启动调试，编辑器可以自动卷屏并命中断点，能通过变量监视器查看当前变量取值。

在对需求进行分析后，公司的架构师小张查阅了相关的资料，认为该集成开发环境应该采用管道一过滤器(Pipe-Filter)的架构风格，公司的资深架构师王工在仔细分析后，认为应该采用数据仓储(Data Repository)的架构风格。公司经过评审，最终采用了王工的方案。

【问题 1】(10 分) 识别软件架构质量属性是进行架构设计的重要步骤。请分析题干中的需求描述，填写表 1-1 中(1)~(5)处的空白。

表 1-1 质量属性识别表

质量属性名称需求描述编号

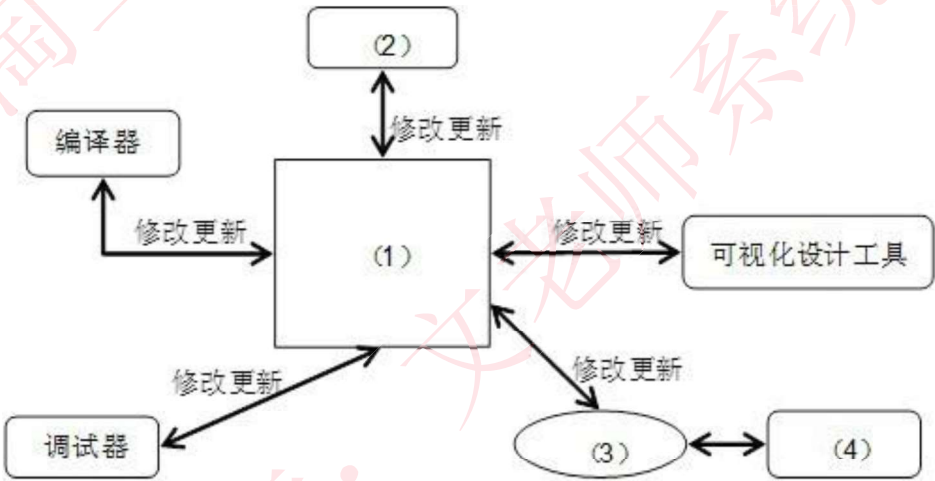
可用性	(1)
(2)	e
可修改性	(3)
可测试性	(4)
安全性	c
易用性	(5)

【问题 2】（7 分）请在阅读题干需求描述的基础上，从交互方式、数据结构、控制结构和扩展方法 4 个方面对两种架构风格进行比较，填写表 1-2 中(1)～(4)处的空白。

表 1-2 两种架构的比较

比较因素	管道—过滤器风格	数据仓储风格
交互方式	顺序结构或有限的循环结构	(1)
数据结构	(2)	文件或模型
控制结构	(3)	业务功能驱动
扩展方法	接口适配	(4)

【问题 3】（8 分）在确定采用数据仓库架构风格后，王工给出了集成开发环境的架构图。请填写图 1-1 中(1)～(4)处的空白，完成该集成开发环境的架构图。



1-1 集成开发环境架构图

2016 年下半年试题 2

阅读以下关于软件系统建模的叙述，在答题纸上回答问题 1 至问题 3。

【说明】某软件公司计划开发一套教学管理系统，用于为高校提供教学管理服务。该教学管理系统基本的需求包括：

- (1) 系统用户必须成功登录到系统后才能使用系统的各项功能服务；
- (2) 管理员(Registrar)使用该系统管理学校(University)、系(Department)、教师(Lecturer)、学生(Student)和课程(Course)等教学基础信息；
- (3) 学生使用系统选择并注册课程，必须通过所选课程的考试才能获得学分；如果考试不及格，必须参加补考，通过后才能获得课程学分；
- (4) 教师使用该系统选择所要教的课程，并从系统获得选择该课程的学生名单；
- (5) 管理员使用系统生成课程课表，维护系统所需的有关课程、学生和教师的信息；
- (6) 每个月到了月底系统会通过打印机打印学生的考勤信息。

项目组经过分析和讨论，决定采用面向对象开发技术对系统各项需求建模。

【问题 1】(7 分) 用例建模用来描述待开发系统的功能需求，主要元素是用例和参与者。请根据题目所述需求，说明教学服务系统中有哪些参与者。

【问题 2】(7 分) 用例是对系统行为的动态描述，用例获取是需求分析阶段的主要任务之一。请指出在面向对象系统建模中，用例之间的关系有哪几种类型？对题目所述教学服务系统的需求建模时，“登录系统”用例与“注册课程”用例之间、“参加考试”用例与“参加补考”用例之间的关系分别属于哪种类型？

【问题 3】(11 分) 类图主要用来描述系统的静态结构，是组件图和配置图的基础。请指出在面向对象系统建模中，类之间的关系有哪几种类型？对题目所述教学服务系统的需求建模时，类 University 与类 Student 之间、类 University 和类 Department 之间、类 Student 和类 Course 之间的关系分别属于哪种类型？

2016 年下半年试题 3

阅读以下关于嵌入式实时系统设计的描述，回答问题 1 至问题 3。

【说明】嵌入式系统是当前航空、航天、船舶及工业、医疗等领域的核心技术，嵌入式系统可包括实时系统与非实时系统两种。某宇航公司长期从事航空航天飞行器电子设备的研制工作，随着业务的扩大，需要大量大学毕业生补充到科研生产部门。按照公司规定，大学毕业生必须进行相关基础知识培训，为此，公司经理安排王工对他们进行了长达一个月的培训。

【问题 1】(7 分) 王工在培训中指出：嵌入式系统主要负责对设备的各种传感器进行管理与控制。而航空航天飞行器的电子设备由于对时间具有很强的敏感性，通常由嵌入式实时系统进行管控，请用 300 字以内文字说明什么是实时系统，实时系统有哪些主要特性。

【问题 2】(8 分) 实时系统根据应用场景、时间特征以及工作方式的不同，存在多种实时特性，大致有三种分类方法，即时间类别、时间需求和工作方式结构。根据自己所掌握的“实时性”知识，将图 3-1 给出的实时特性按三种分类方式，填写图 3-1 中(1)~(8)处空白。

备选答案：时限的危害程度；时间角色；弱；时间响应；固定；时限 / 反应时间；时间明确；输入 / 输出激励；时间触发；强；周期 / 零星 / 非周期；事件触发。

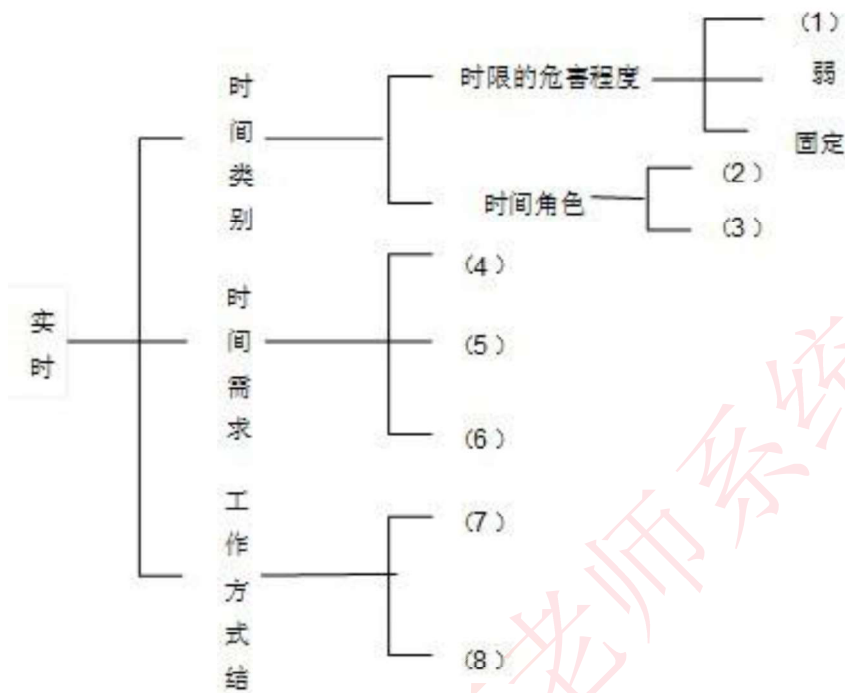


图 3-1 实时特性分类图

【问题 3】（10 分）可靠性是实时系统的关键特性之一，区分软件的错误(Error)、缺陷(Defect)、故障(Fault)和失效(Failure)概念是软件可靠性设计工作的基础。请简要说明错误、缺陷、故障和失效的定义；并在图 3-2 中标出错误、缺陷和失效出现阶段，说明缺陷、故障和失效的表现形式，填写图 3-2 中(1)~(6)处的空白。

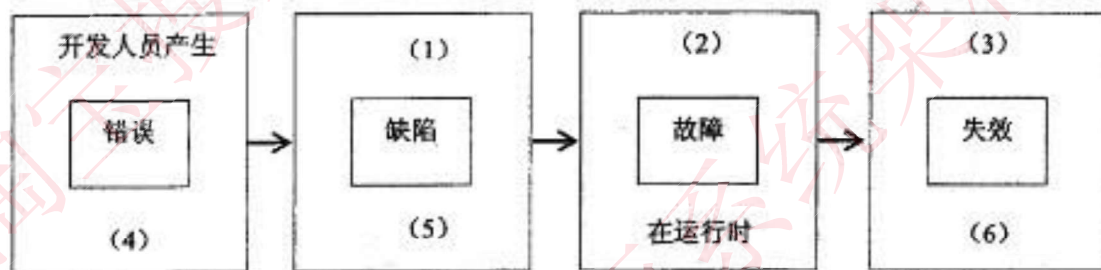


图 3-2 错误、缺陷、故障和失效关系图

2016 年下半年试题 4

阅读以下关于应用服务器的叙述，，在答题纸上回答问题 1 至问题 3。

【说明】

某电子产品制造公司，几年前开发建设了企业网站系统，实现了企业宣传、产品介绍、客服以及售后服务等基本功能。该网站技术上采用了 Web 服务器、动态脚本语言 PHP。随着市场销售渠道变化以及企业业务的急剧拓展，该公司急需建立完善的电子商务平台。

公司张工建议对原有网站系统进行扩展，增加新的功能(包括订单系统、支付系统、库存管理 等)，这样有利于降低成本、快速上线；而王工则认为原有网站系统在技术上存在先天不足，不能满足企业业务

的快速发展，尤其是企业业务将服务全球，需要提供 24 小时不间断服务，系统在大负荷和长时间运行下的稳定性至关重要。建议采用应用服务器的 Web 开发方法，例如 J2EE，为企业重新开发新的电子商务平台。

【问题 1】（7 分）王工认为原有网站在技术上存在先天不足，不能满足企业业务的快速发展，根据你的理解，请用 300 字以内的文字说明原系统存在哪几个方面的不足。

【问题 2】（8 分）请简要说明应用服务器的概念，并重点说明应用服务器如何来保障系统在大负荷和长时间运行下的稳定性以及可扩展性。

【问题 3】（10 分）J2EE 平台采用了多层分布式应用程序模型，实现不同逻辑功能的应用程序被封装到不同的构件中，处于不同层次的构件可被分别部署到不同的机器中。请填写图 4-1 中 (1)~(5) 处的空白，完成 J2EE 的 N 层体系结构。

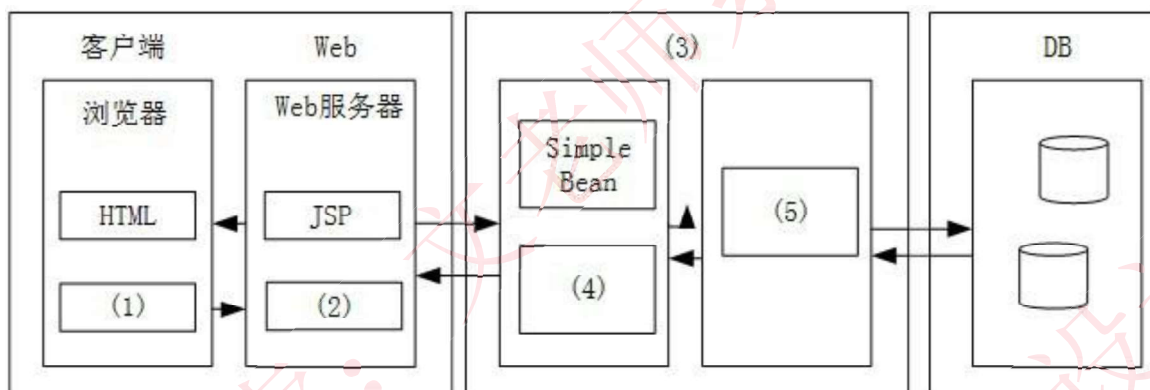


图 4-1 J2EE 的 N 层体系结构示意图

2016 年下半年试题 5

阅读以下关于 Scrum 敏捷开发过程的叙述，在答题纸上回答问题 1 至问题 3。

【说明】

Scrum 是一个增量的、迭代的敏捷软件开发过程。某软件公司计划开发一个基于 Web 的 Scrum 项目管理系统，用于支持项目团队采用 Scrum 敏捷开发方法进行软件开发，辅助主管智能决策。此项目管理系统提供的主要服务包括项目团队的管理、敏捷开发过程管理和工件的管理。

Scrum 敏捷开发中，项目团队由 Scrum 主管、产品负责人和开发团队人员三种不同的角色组成，其开发过程由若干个 Sprint（短的迭代周期，通常为 2 到 4 周）活动组成。

Product Backlog 是在 Scrum 过程初期产生的一个按照商业价值排序的需求列表，该列表条目的体现形式通常为用户故事。在每一个 Sprint 活动中，项目团队从 Product Backlog 中挑选最高优先级的用户故事进行开发。被挑选的用户故事在 Sprint 计划会议上经过细化分解为任务，同时初步估算每一个任务的预计完成时间，编写 Sprint Backlog。

在 Sprint 活动期间，项目团队每天早晨需举行每日站立会议，重新估算剩余任务的预计完成时间，更新 Sprint Backlog、Sprint 燃尽图和 Release 燃尽图。在每个 Sprint 活动结束后，项目团队召开评审会议和回顾会议，交付产品增量，总结 Sprint 期间的工作情况和问题。此时，如果 Product Backlog 中还有未完成的用户故事，则项目团队将开始筹备下一个 Sprint 活动迭代。

为完成 Scrum 项目管理系统，考虑到系统的智能决策需求，公司决定使用 MVC 架构模式开发该项目管理系统。具体来说，系统采用轻量级 J2EE 架构和 SSH 框架进行开发，使用 MySQL 数据库作为底层 存储。

【问题 1】（10 分）

Scrum 项目管理软件需真实模拟 Scrum 敏捷开发流程，请根据你的理解完成图 5-1 给出的 Scrum 敏捷 开发状态图，填写其中(1)～(5)的内容。

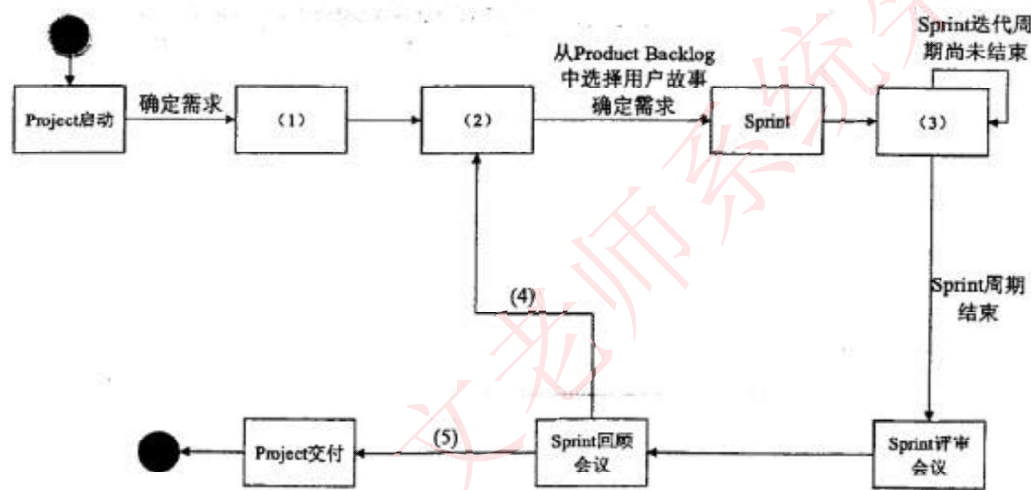


图 5-1 Scrum 敏捷开发状态图

【问题 2】（6 分）根据题干描述，本系统采用 MVC 架构模式，请从各选答案 a~n 中分别选出属于 MVC 架构模型中的模型 (Model)、视图 (View) 和控制器 (Controler) 的相关内容描述填入表 5-1 的空 (1) ~ (3) 处。

表5-1 架构模式中包含的内容

架构模式	包含内容
模型 (Model)	(1)
视图 (View)	(2)
控制器 (Controler)	(3)

备选答案:

- | | |
|-------------------|------------------|
| a Sprint 燃尽图 | h 用户 |
| b Project | i 交付产品增量 |
| c Product Backlog | j 新建项目 |
| d 用户故事 | k Task |
| e估算任务预计完成时间 | l Sprint |
| f Release 燃尽图 | m 产品负责人 |
| g Sprint 回顾会议 | n Sprint Backlog |

【问题3】（9分）

根据项目组给出的系统设计方案，将备选答案 a~l 的内容填写在图 5-2 中的空 (1)~(9)，完成系统架构图。

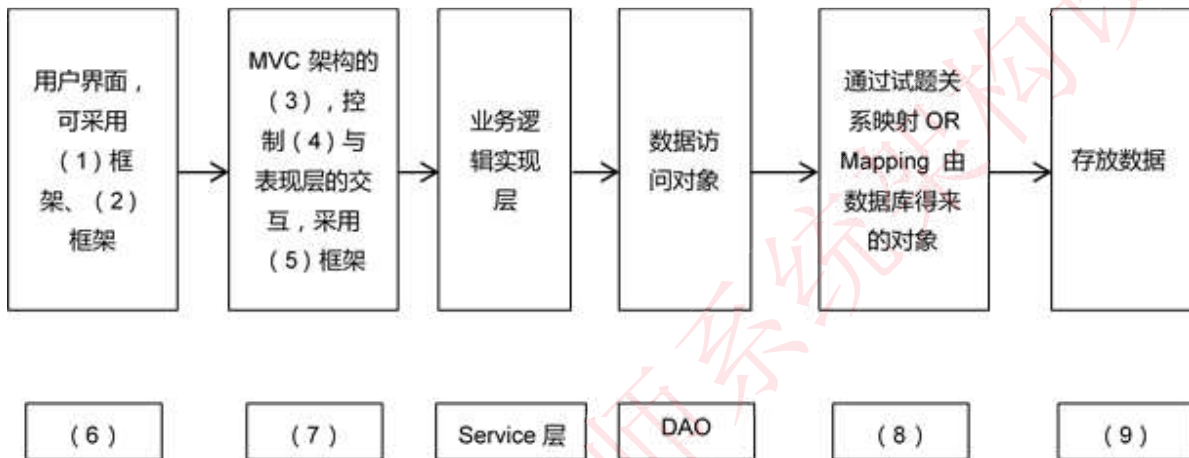


图 5-2 系统架构图

备选答案：

a	Struts 2	g	模型层
b	Hibernate 持久层	h	控制层
c	数据库服务 (MySQL)	i	EJB
d	Sitemesh	j	Web 层
e	业务逻辑层	k	视图层
f	JQuery	l	PostgreSQL

2015 年下半年试题 1

阅读以下关于软件架构评估的说明，在答题纸上回答问题 1 和问题 2。

【说明】

某软件公司拟为某市级公安机关开发一套特种车辆管理与监控系统，以提高特种车辆管理的效率和准确性。在系统需求分析与架构设计阶段，用户提出的部分需求和关键质量属性场景如下：

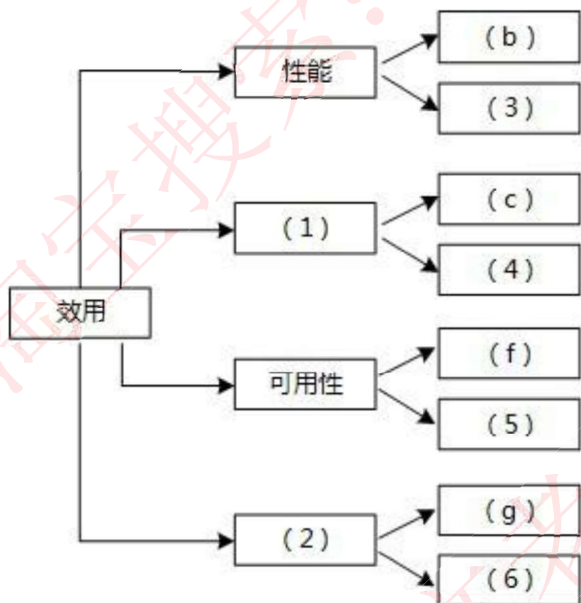
- (a) 系统用户分为管理员、分管领导和普通民警等三类；
- (b) 正常负载情况下，系统必须在 0.5 秒内对用户的车辆查询请求进行响应；
- (c) 系统能够抵御 99.999% 的黑客攻击；
- (d) 系统的用户名必须以字母开头，长度不少于 5 个字符；
- (e) 对查询请求处理时间的要求将影响系统的数据传输协议和处理过程的设计；
- (f) 网络失效后，系统需要在 2 分钟内发现并启用备用网络系统；

- (g) 在系统升级时，需要保证在 1 个月内添加一个新的消息处理中间件；
- (h) 查询过程中涉及到的车辆实时视频传输必须保证 20 帧/秒的速率，且画面具有 600*480 的分辨率；
- (i) 更改系统加密的级别将对安全性和性能产生影响；
- (j) 系统主站点断电后，需要在 3 秒内将请求重定向到备用站点；
- (k) 假设每秒中用户查询请求的数量是 10 个，处理请求的时间为 30 毫秒，则“在 1 秒内完成用户的查询请求”这一要求是可以实现的；
- (l) 对用户信息数据的授权访问必须保证 99.999% 的安全性；
- (m) 目前对“车辆信息实时监控”业务逻辑的描述尚未达成共识，这可能导致部分业务功能模块的重复，影响系统的可修改性；
- (n) 更改系统的 Web 界面接口必须在 1 周内完成；
- (o) 系统需要提供远程调试接口，并支持系统的远程调试。

在对系统需求和质量属性场景进行分析的基础上，系统的架构师给出了三个候选的架构设计方案。公司目前正在组织系统开发的相关人员对系统架构进行评估。

【问题 1】（12 分）

在架构评估过程中，质量属性效用树（utility tree）是对系统质量属性进行识别和优先级排序的重要工具。请给出合适的质量属性，填入图 1-1 中 (1)、(2) 空白处；并选择题干描述中的 (a) ~ (o)，将恰当的序号填入 (3) ~ (6) 空白处，完成该系统的效用树。



【问题 2】（13 分） 在架构评估过程中；需要正确识别系统的架构风险、敏感点和权衡点，并进行合理的架构决策。请用 300 字以内的文字给出系统架构风险、敏感点和权衡点的定义，并从题干描述中的 (a) ~ (o) 各选 出 1 个属于系统架构风险、敏感点和权衡点的描述。

2015 年下半年试题 2

阅读以下关于系统设计建模的说明，在答题纸上回答问题 1 至问题 3。

【说明】

某公司拟研制一款高空监视无人直升机，该无人机采用遥控一自主复合型控制实现垂直升降。该直升机飞行控制系统由机上部分和地面部分组成，机上部分主要包括无线电传输设备、飞控计算机、导航设备等，地面部分包括遥控操纵设备、无线电传输设备以及地面综合控制计算机等。其主要工作原理是地面综合控制计算机负责发送相应指令，飞控计算机按照预定程序实现相应功能。经过需求分析，对该无人直升机控制系统纵向控制基本功能整理如下：

- (a) 飞控计算机加电后，应完成系统初始化，飞机进入准备起飞状态；
- (b) 在准备起飞状态中等待地面综合控制计算机发送起飞指令，飞控计算机接收到起飞指令后，进入垂直起飞状态；
- (c) 垂直起飞过程中如果飞控计算机发现飞机飞行异常，飞行控制系统应转入无线电遥控飞行状态，地面综合控制计算机发送遥控指令；
- (d) 垂直起飞达到预定起飞高度后，飞机应进入高度保持状态；
- (e) 飞控计算机在收到地面综合控制计算机发送的目标高度后，飞机应进入垂直升降状态，接近目标高度；垂直升降过程中出现飞机飞行异常，控制系统应转入无线电遥控飞行；
- (f) 飞机到达目标高度后，应进入高度保持状态，完成相应的任务；
- (g) 飞机在接到地面综合控制计算机发送的任务执行结束指令后，进入飞机降落状态；
- (h) 飞机降落过程中如果出现飞机飞行异常，控制系统应转入无线电遥控飞行；
- (i) 飞机降落到指定着陆高度后，进入飞机着陆状态，应按照预定着陆算法，进行着陆；
- (j) 无线电遥控飞行中，地面综合控制计算机发送着陆指令，飞机进入着陆状态，应按照预定着陆算法，进行着陆。

【问题 1】（6 分）

状态图和活动图是软件系统设计建模中常用的两种手段，请用 200 字以内文字简要说明状态图和活动图的含义及其区别。

【问题 2】（10 分） 根据题干中描述的基本功能需求，架构师王工通过对需求的分析和总结给出了无人直升机控制系统纵向控制状态图(图 2-1)。请根据题干描述，提炼出相应状态及条件，并完善图 2-1 所示状态图中的 (1)~(5)，将答案填写在答题纸中。

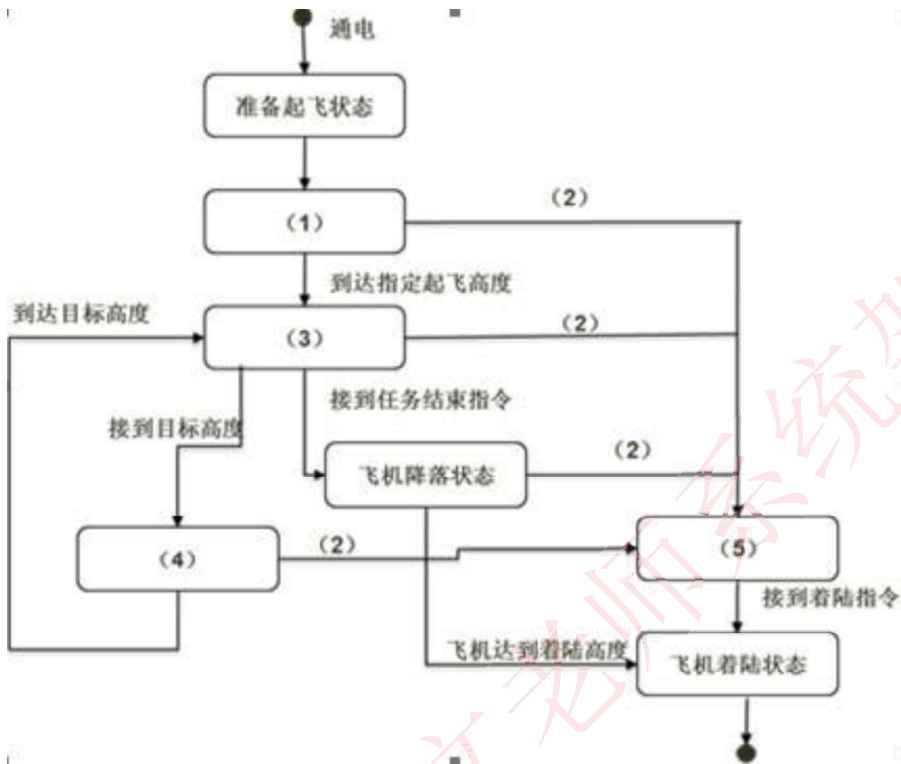


图 2-1 无人直升机控制系统纵向控制状态图

【问题 3】（9 分） 根据题目中描述的基本功能需求，架构师王工给出了无人直升机控制系统纵向控制的顶层活动图（图 2-2）。请根据题干描述，完善图 2-2 活动图的（1）-（9），将答案填写在答题纸中。

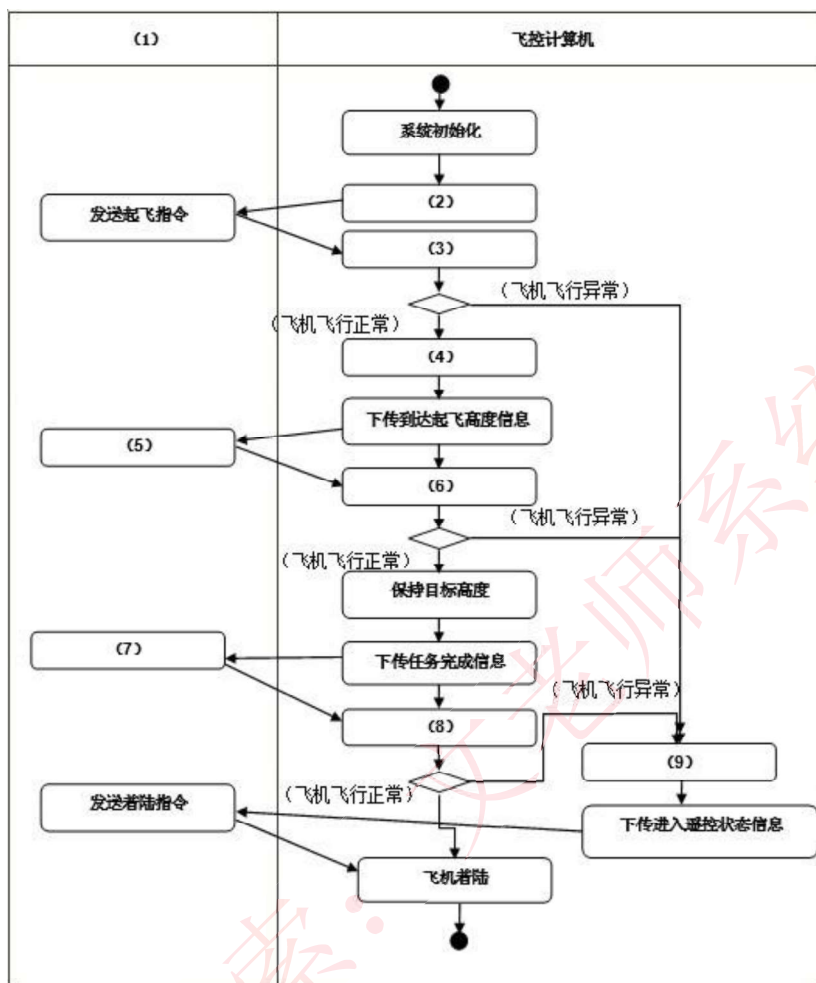


图 2-2 无人直升机控制系统纵向控制顶层活动图

2015 年下半年试题 3

阅读以下关于嵌入式系统可靠性设计方面的描述，回答问题 1 至问题 3。

【说明】 某宇航公司长期从事宇航装备的研制工作，嵌入式系统的可靠性分析与设计已成为该公司产品研制中的核心工作，随着宇航装备的综合化技术发展，嵌入式软件规模发生了巨大变化，代码规模已从原来的几十万扩展到上百万，从而带来了由于软件失效而引起系统可靠性降低的隐患。公司领导非常重视软件可靠性工作，决定抽调王工程师等 5 人组建可靠性研究团队，专门研究提高本公司宇航装备的系统可靠性和软件可靠性问题，并要求在三个月内，给出本公司在系统和软件设计方面如何考虑可靠性设计的方法和规范。可靠性研究团队很快拿出了系统及硬件的可靠性提高方案，但对于软件可靠性问题始终没有研究出一种普遍认同的方法。

【问题 1】（共 9 分）

请用 200 字以内文字说明系统可靠性的定义及包含的 4 个子特性，并简要指出提高系统可靠性一般采用哪些技术？

【问题 2】（共 8 分）

王工带领的可靠性研究团队之所以没能快速取得软件可靠性问题的技术突破，其核心原因是他们没有搞懂高可靠性软件应具备的特点。软件可靠性一般致力于系统性地减少和消除对软件程序性能有

不利影响的系统故障。除非被修改，否则软件系统不会随着时间的推移而发生退化。请根据你对软件可靠性的理解，给出表 3-1 所列出的硬件可靠性特征对应的软件可靠性特征之间的差异或相似之处，将答案写在答题纸上。

表3-1 硬件和软件可靠性对比

序号	硬件可靠性	软件可靠性
1	失效率服从浴缸曲线。老化状态类似于软件调试状态	(1)
2	即使不适用，材料劣化也会导致失效	(2)
3	硬件维修会恢复原始状态	(3)
4	硬件失效之前会有报警	(4)

【问题 3】（共 8 分）王工带领的可靠性研究团队在分析了大量相关资料基础上，提出软件的质量和可靠性必须在开发过程构建到软件中，也就是说，为了提高软件的可靠性，必须在需求分析、设计阶段开展软件可靠性筹划和设计。研究团队针对本公司承担的飞行控制系统制定出了一套飞控软件的可靠性设计要求。飞行控制系统是一种双余度同构型系统，输入采用了独立的两路数据通道，在系统内完成输入数据的交叉对比、表决‘制导率计算，输出数据的交叉对比、表决、一输出等功能，系统的监控模块实现对系统失效或失步的检测与冗余。其软件的可靠性设计包括恢复块方法和 N 版本程序设计方法。请根据恢复块方法工作原理完成图 3-1，在(1)～(4)中填入恰当的内容。并比较恢复块方法与 N 版本程序设计方法，将比较结果(5)～(8)填入表 3-2 中。

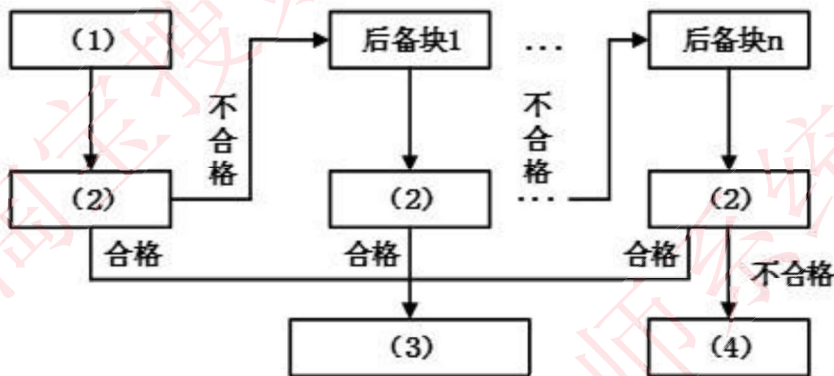


图 3-1 恢复块方法

表 3-2 恢复块方法与 N 版本程序设计的比较

	恢复块方法	N版本程序设计
硬件运行环境	单机	多机
错误检测方法	验证测试程序	(5)
恢复策略	(6)	向前恢复
实时性	(7)	(8)

2015 年下半年试题 4

阅读以下关于应用系统数据架构的说明，在答题纸上回答问题 1 至问题 3。

【说明】某软件公司拟开发一套贸易综合管理系统，包括客户关系管理子系统和商品信息管理子系统两部分。客户关系管理子系统主要管理客户信息，并根据贸易业务需要频繁向客户发送相关的电子邮件、短信等提醒信息。商品信息管理子系统主要为客户提供商品信息在线查询功能，包括商品基本信息、实时库存与价格等。

在对系统进行数据架构设计时，公司项目组的架构师王工主张采用文件系统进行数据管理，原因是目前公司客户和商品数量不大，且系统功能较为简单，采用文件系统进行数据管理简单直观，开发周期短。架构师李工则建议采用关系数据库进行数据管理，原因在于公司目前正处在高速扩张期，虽然目前的客户和商品数量不大，但随着公司快速发展，需要管理的数据必然飞速膨胀，采用关系数据库作为数据存储层，系统的扩展性更强，并能够对未来可能增加的复杂业务提供有效支持。经过讨论，项目组初步采纳了李工的意见，决定采用关系数据库存储客户数据，并针对业务特征对系统性能进行优化。

【问题 1】（8 分）

请从设计难度、数据冗余程度、数据架构、应用扩展性等 4 个方面对关系型数据库管理系统和文件系统两种数据存储方式进行比较，填写表 4-1 中(1)~(4)。

表 4-1 关系型数据库管理系统和文件系统存储方式比较

	设计难度	数据冗余程度	数据架构	应用扩展性
关系型数据库	(1)	遵守数据库范式，数据冗余较少	以数据库为中心组织、管理数据	(4)
文件系统	针对特定应用系统设计，难度较小	(2)	(3)	符合特定应用系统要求的文件数据很难在不同的应用系统之间共享

【问题 2】（13 分）对系统的核心业务需求进行认真分析后，公司的资深架构师张工提出一种内存数据库和关系数据库的混合存储架构，其核心思想是将需要频繁读写的数据存入内存数据库，而将相对固定不变的数据存入关系数据库。请首先分析比较内存数据库和关系数据库在数据模型、读写性能、存储容量、可靠性等方面的差异，填写表 4-2 中(1)~(4)的空白，并根据张工的思路指定各种业务数据的存储方式，填写表 4-3 中(5)~(9)中的空白。

表 4-2 内存数据库和关系数据库比较

	只要数据模型	读写性能	存储容量	可靠性
内存数据库	(1)	内存直接读写，性能相对较高	(3)	(4)
关系数据库	关系模式	(2)	基于磁盘存储，存储容量大	内建恢复机制，可靠性较高

表 4-3 业务数据存储方式

业务数据	存储方式
客户基本信息	关系数据库
客户电子邮件	(5)
客户联系电话	(6)
商品基本信息	(7)
商品库存信息	(8)
商品价格信息	(9)

【问题 3】（4 分） 系统开发完成进行压力测试时，发现在大数据量的情况下，部分业务查询响应时间过长，经过分析发现其主要原因是部分 SQL 查询语句效率低下。请判断表 4-4 中的 SQL 语句设计策略哪些可能会提升查询效率，哪些可能会降低查询效率，在(1)~(4)中填入“提升”或“降低”。

表 4-4 SQL 设计策略对性能的影响

SQL 语句设计策略	查询性能影响情况
查询时不要返回不需要的行、列	(1)
需要进行多表连接查询时，尽量采用子查询结构	(2)
尽量采用 NOT IN、NOT EXIST、LIKE 等操作	(3)
尽量避免使用 DISTINCT 关键字	(4)

2015 年下半年试题 5

阅读以下关于 Web 应用系统架构设计的叙述，在答题纸上回答问题 1 至问题 3。

【说明】 某信息技术公司计划开发一套在线投票系统，用于为市场调研、信息调查和销售反馈等业务提供服务。该系统计划通过大量宣传和奖品鼓励的方式快速积累用户，当用户规模扩大到一定程度时，开始联系相关企业提供信息服务，并按照信息服务种类和用户投票数量收取费用。

为了降低开发成本和提高开发效率，项目组经过讨论后决定采用轻量级 Java EE 开发框架设计系统应用架构。在应用架构设计中，除了满足系统主要功能需求，还需要考虑的因素包括：

- (1) 项目开发采用 MySQL 数据库存储数据，一旦将来可能移植到其它数据库平台；
- (2) 系统开发过程中尽可能降低或者消除 SQL 语句开发的工作量；
- (3) 投票系统中数据之间的关系复杂，需要支持数据对象的聚合和继承等关系。

项目组基于 MVC 模式设计出了投票系统的架构，包括表示层、业务逻辑层、数据持久层和数据层。在具体讨论数据持久层采用哪种技术方案时，老王建议采用成熟的 Hibernate 框架，小李则认为 iBatis 更加灵活，更适合作为投票系统数据持久层开发技术。

【问题 1】（5 分）

请用 300 以内文字说明什么是数据持久层，使用数据持久层能够为项目开发带来哪些好处？

【问题 2】（12 分） 针对在线投票系统的实际应用需求和要求，项目组应选用哪种技术实现数据持久层？请用 200 字以内文字说明其采用该技术的原因。

【问题 3】（8 分） 数据持久层是 Web 应用系统框架中重要的组成部分，主流的数据持久层技术分别基于不同的技术方案，请在表 5-1 中（1）-（4）处分别根据（a）~（d）所列技术的方案类别填入其序号。

表 5-1 数据持久层技术分类

技术方案	实现技术
JDBC封装	(1)
SQL Mapping	(2)
O/R Mapping	(3)
Entity Bean	(4)

(a) BMP, CMP (b) iBatis/MyBatis (c) SpringJdbcTemplate (d) TopLink, JDO, Hibernate

2014 年下半年试题 1

请仔细阅读以下关于网络设备管理系统架构设计的说明，在答题纸上回答问题 1 和问题 2。

【说明】 某软件公司欲开发一个网络设备管理系统，对管理区域内的网络设备（如路由器和交换机等）进行远程监视和控制。公司的系统分析师首先对系统进行了需求分析，识别出如下 3 项核心需求：

（a）目前需要管理的网络设备确定为 10 类 20 种，未来还将有新类别的网络设备纳入到该设备管理系统中；

（b）不同类别的网络设备，监视和控制的内容差异较大；同一类网络设备，监视和控制的内容相似，但不同厂商的实现方式（包括控制接口格式、编程语言等）差异较大；

（c）网络管理员能够在一个统一的终端之上实现对这些网络设备的可视化呈现和管理操作。

针对上述需求，公司研发部门的架构师对网络设备管理系统的架构进行了分析与设计，架构师王工认为该系统可以采用 MVC 架构风格实现，即对每种网络设备设计一个监控组件，组件通过调用网络设备厂商内置的编程接口对监控指令进行接收和处理；系统管理员通过管理模块向监控组件发送 监控指令，对网络设备进行远程管理；网络状态、监控结果等信息会在控制终端上进行展示。针对 不同网络设备的差异，王工认为可以对当前的 20 种网络设备接口进行调研与梳理，然后通过定义统一操作接口屏蔽设备差异。李工同意王工提出的 MVC 架构风格和定义统一操作接口的思路，但考虑到未来还会有新类别的网络设备接入，认为还需要采用扩展接口的方式支持系统开发人员扩展或修改现有操作接口。公司组织专家进行架构评审，最终同意了王工的方案和李工的改进意见。

【问题 1】（10 分） 请用 300 字以内的文字解释什么是 MVC 架构风格以及其中的组件交互关系，并根据题干描述，指出该系统中的 M、V、C 分别对应什么。

【问题 2】（15 分） 扩展接口模式结构通常包含四个角色：基础接口、组件、扩展接口和客户端，

它们之间的关系如图 1-1 所示。

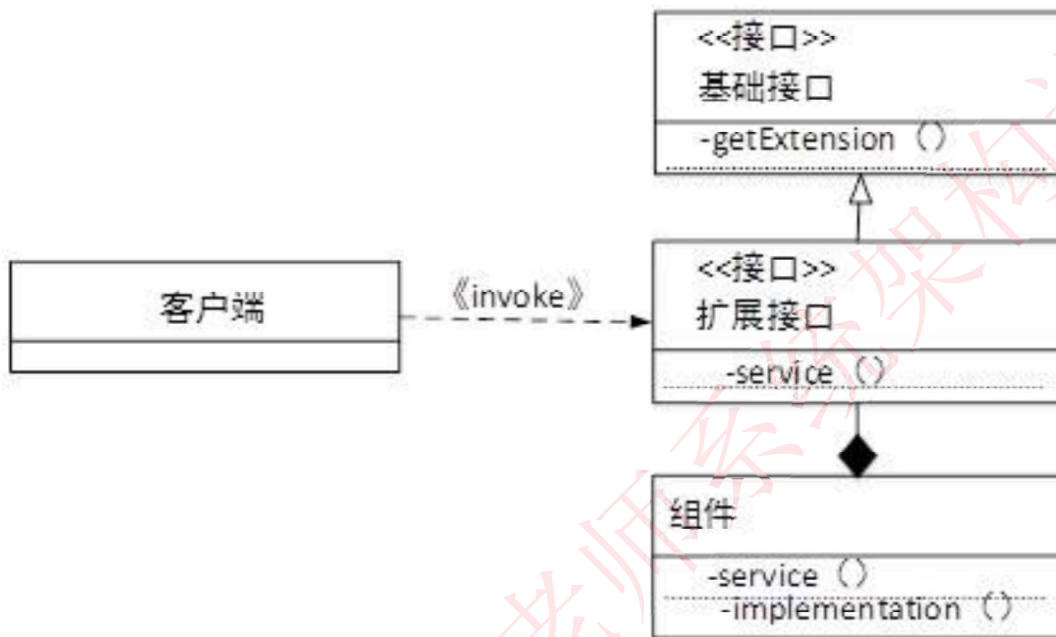


图 1-1 扩展接口模式角色关系

其中每个扩展接口需要通过扩展基础接口获得基本操作能力，然后加入自己特有的操作接口，并通过设置全局唯一接口 ID 对自身接口进行标识；每个具体的组件需要实现扩展接口完成实际操作；客户端不与组件直接交互，而需要通过与扩展接口交互提出调用请求，扩展接口根据请求查找并选择合适的实现组件响应客户端请求。请根据上图所示和题干描述，指出扩展接口模式结构中的四个角色分别对应网络设备管理系统的哪些部分；并以客户端发起调用操作这一场景为例，填写表 1-1 中的（1）～（5）。

表 1-1 客户端发起调用操作过程描述

序号	操作
1	客户端调用某个 (1) A 上的操作接口，该操作接口可能是基础接口，也可能是扩展接口
2	若实现 A 的 (2) 存在被执行请求的操作接口，则调用该操作接口向用户返回结果
3	如果所有组件均没有实现 (3)，则客户端调用 A 上的 <code>getExtension</code> 方法，传入需要的 (4)，通过查找与定位，找到实现该操作接口的 (5) B，并将 B 的引用传回客户端
4	客户端调用 B 上的操作接口，通过相应的实现组件返回结果

备选答案：基础接口、扩展接口、操作接口、接口 ID、客户端、组件。

2014 年下半年试题 2

请详细阅读以下关于系统过程建模的说明，在答题纸上回答问题 1 至问题 3。

【说明】某公司正在研发一套新的库存管理系统。系统中一个关键事件是接收供应商供货。项目组系统分析员小王花了大量时间在仓库观察了整个事件的处理过程，并开发出该过程所执行活动的列表：供应商发送货物和商品清单，公司收到商品后执行收货处理，包括卸载商品、确定收到了订单上的商品、处理与供应商的分歧等。对于已有商品，调整其库存信息，对于新采购的商品，在库存中添加新的商品记

录。收货完成后，系统执行入库处理，将商品放到仓库对应的货架上。在付款处理活动中，自动生成应付账款信息，如果查询到该供应商有待付款记录，则进行合并付款，付款完成后消除应付账款记录。最后，仓库管理员根据最新的库存商品，调整出货信息。小王根据自己观察的过程创建了该事件的 1 层数据流图，如下图所示。

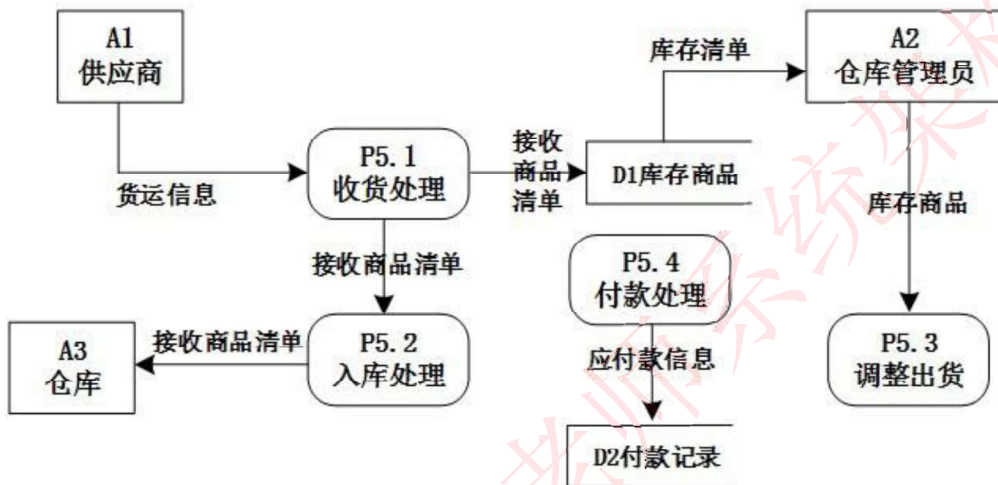


图 2-1 接收供应商供货的 1 层数据流图

【问题 1】（8 分）

请用 300 以内文字说明数据流图(Data Flow Diagram)的基本元素及其作用。

【问题 2】（12 分）

数据流图在绘制过程中可能出现多种语法错误，请分析图 2-1 所示数据流图中哪些地方有错误，并分别说明错误的类型。

【问题 3】（5 分）

系统建模过程中为了保证数据模型和过程模型的一致性，需要通过数据-过程-CRUD 矩阵来实现数据模型和过程模型的同步，请在表 2-1 所示 CRUD 矩阵（1）～（5）中填入相关操作。

表 2-1 接收供应商供货的 CRUD 矩阵

	P5.1 收货处理	P5.2 入库处理	P5.3 调整出货	P5.4 付款处理
供应商	(1)			(2)
库存商品		(3)	(4)	
付款记录				(5)

2014 年下半年试题 3

请详细阅读有关嵌入式构件设计方面的说明，在答题纸上回答问题 1 至问题 3。

【说明】 构件(component)也称为组件，是一个功能相对独立的具有可复用价值的软硬件单元。近年来，构件技术正在逐步应用于大型嵌入式系统的软件设计。某公司长期从事飞行器电子设备研制工作，已积累了大量成熟软件。但是，由于当初管理和设计等原因，公司的大量软件不能被复用，严重影响了公司后续发展。公司领导层高度重视软件复用问题，明确提出了要将本公司的成熟软件进行改造，建立公司可复用的软件构件库，以提升开发效率、降低成本。公司领导层决定将此任务交给技术部门的王工程师负

责组织实施。两个月后，王工程师经过调研、梳理和实验，提交了一份实施方案。此方案得到了公司领导层的肯定，但在实施过程中遇到了许多困难，主要表现在公司软件架构的变更和构件抽取的界面等方面。

【问题 1】（共 9 分） 请用 200 字以内文字说明获取构件的方法有哪几种？开发构件通常采用哪几种策略？并列举出两种主流构件标准。

【问题 2】（共 6 分） 由于该公司已具备大量的成熟软件，王工程师此次的主要工作就是采用遗留工程(Legacy Engineering)方法，将具有潜在复用价值的软件提取出来，得到可复用的构件。因此，在设计软件时与原开发技术人员产生了重大意见分歧，主要分歧焦点在于大家对构件概念理解上的差异。请根据你对构件的理解，判断表 3-1 给出的有关构件的说法是否正确，将答案写在答题纸上。

表 3-1 有关构件的 6 种说法

序号	关于构件的说明	正确：√ 不正确：×
1	构件是系统中的一个封装了设计与实现，而只披露接口的可更换的部分	(1)
2	构件是解决软件复用的基础，复用的形式可分为垂直式复用和水平式复用。而水平式复用的主要关键在于领域分析，具有领域特征和相似性，受到广泛关注	(2)
3	构件构建在平台之上，平台提供核心平台服务，是构件实现与构件组装的基础。构件组装通常采用基于功能的组装技术、基于数据的组装技术和基于配置的组装技术等三种技术	(3)
4	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，由构件的描述、构件的相互作用（连接件）、指导构件集成的模式以及这些模式的约束组成	(4)
5	构件可分为硬件构件、软件构件、系统构件和应用构件。RTL（运行时库）属于软件构件，由于 RTL 与应用领域相关，所以 RTL 应属于垂直式复用构件	(5)
6	硬件构件的功能被给定的硬件结构如 ASIC 预先确定，是不能修改的。同样，软件构件的功能由在 FPGA 或者 CPU 上的软件确定的，也是不能修改的	(6)

【问题 3】（共 10 分） 王工程师的实施方案指出：本公司的大部分产品是为用户提供标准计算平台的，而此平台中的主要开发工作是为嵌入式操作系统研制板级支持软件(BSP)。为了提高 BSP 软件的复用，应首先开展 BSP 构件的开发，且构件架构应符合国外 GENESYS 规范定义的嵌入式系统架构风格。图 3-1 给出了架构风格定义的构件通用接口，其中：链接接口 (LIF) 是构件对外提供的功能服务接口；局部接口建立了构件和它的局部环境的连接，如传感器、作动器或人机接口；技术相关接口(TDI)提供了查看构件内部、观察构件的内部变量的手段，如诊断等；技术无关接口(TII)用来在运行时配置、复使、重启构件的接口。现需要针对 BSP 中常用的 RS-232 串行驱动程序设计一个可复用的软构件，请说明该软构件四类接口的具体功能。

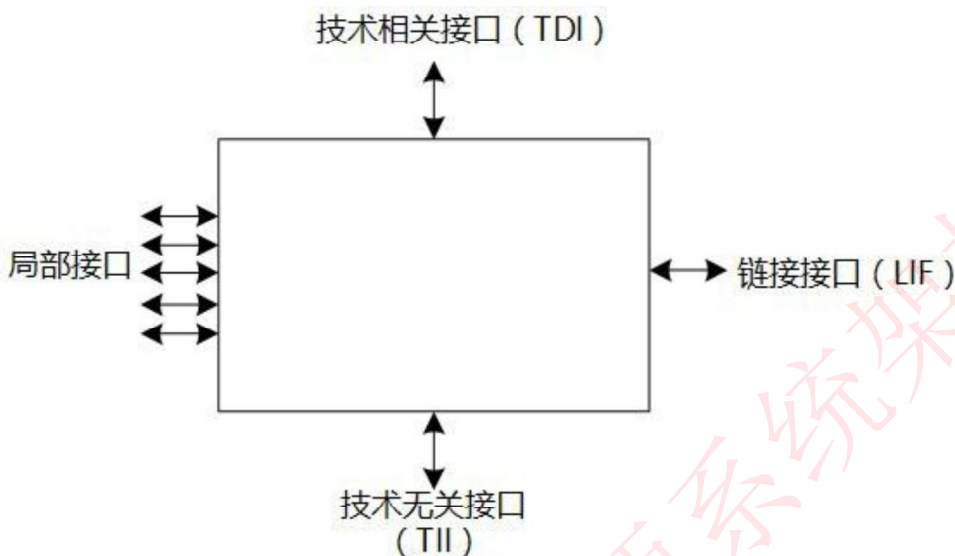


图 3-1 GENESYS 规范定义的构件接口

2014 年下半年试题 4

请仔细阅读有关软件架构评估方面的说明，在答题纸上回答问题 1 至问题 2。

【说明】某电子商务公司拟升级目前正在使用的在线交易系统，以提高客户网上购物时在线支付环节的效率和安全性。公司研发部门在需求分析的基础上，给出了在线交易系统的架构设计。公司组织相关人员召开了针对架构设计的评估会议，会上用户提出的需求、架构师识别的关键质量属性场景和评估专家的意见等内容部分列举如下：

- (a) 在正常负载情况下，系统必须在 0.5 秒内响应用户的交易请求；
- (b) 用户的信用卡支付必须保证 99.999%的安全性；
- (c) 系统升级后用户名要求至少包含 8 个字符；
- (d) 网络失效后，系统需要在 2 分钟内发现错误并启用备用系统；
- (e) 在高峰负载情况下，用户发起支付请求后系统必须在 10 秒内完成支付功能；
- (f) 系统拟采用新的加密算法，这会提高系统安全性，但同时会降低系统的性能；
- (g) 对交易请求处理时间的要求将影响系统数据传输协议和交易处理过程的设计；
- (h) 需要在 30 人月内为系统添加公司新购买的事务处理中间件；
- (i) 现有架构设计中的支付部分与第三方支付平台紧耦合，当系统需要支持新的支付平台时，这种设计会导致支付部分代码的修改，影响系统的可修改性；
- (j) 主站点断电后，需要在 3 秒内将访问请求重定向到备用站点；
- (k) 用户信息数据库授权必须保证 99.999%可用；
- (l) 系统需要对 Web 界面风格进行修改，修改工作必须在 4 人月内完成；
- (m) 系统需要为后端工程师提供远程调试接口，并支持远程调试。

【问题 1】（12 分）

在架构评估过程中，质量属性效用树(utility tree)是对系统质量属性进行识别和优先级排序的重要工具。请给出合适的质量属性，填入图 4-1 中 (1)、(2) 空白处；并选择题干描述的 (a)～(m)，填入 (3)～(6) 空白处，完成该系统的效用树。

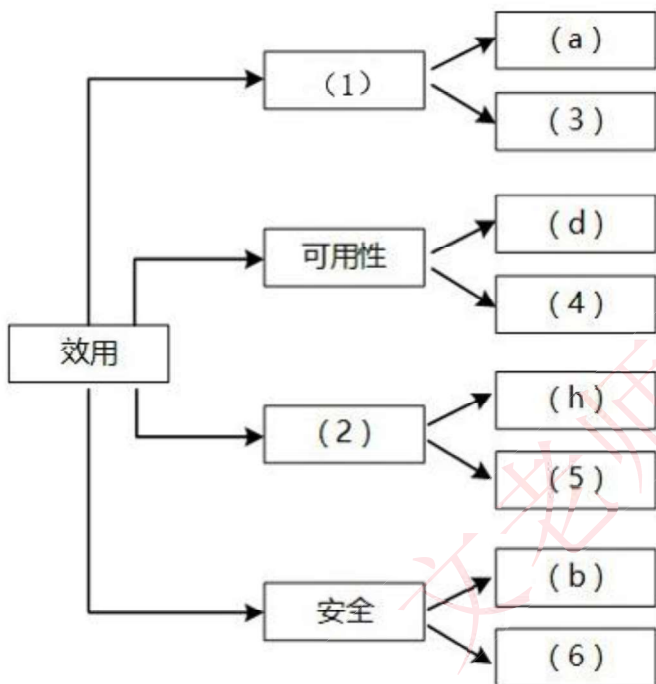


图 4-1 在线交易系统效用树

【问题 2】(13 分) 在架构评估过程中，需要正确识别系统的架构风险、敏感点和权衡点，并进行合理的架构决策。请用 300 字以内的文字给出系统架构风险、敏感点和权衡点的定义，并从题干 (a)～(m) 中各选 出 1 个对系统架构风险、敏感点和权衡点最为恰当的描述。

2014 年下半年试题 5

请仔细阅读有关 Web 应用架构设计方面的说明，在答题纸上回答问题 1 至问题 3。

【说明】 某软件公司开发运维了一个社交网站系统，该系统基于开源软件平台 LAMP

(Linux+Apache+MySQL+PHP) 构建，运行一段时间以来，随着用户数量及访问量的增加，系统在 Web 服务器负载、磁盘 I/O 等方面出现了明显瓶颈，已不能满足大量客户端并发访问的要求，因此公司成立了专门的项目组，拟对系统架构进行调整以提高系统并发处理能力。目前系统采用了传统的三层结构，系统架构如图 5-1 所示。

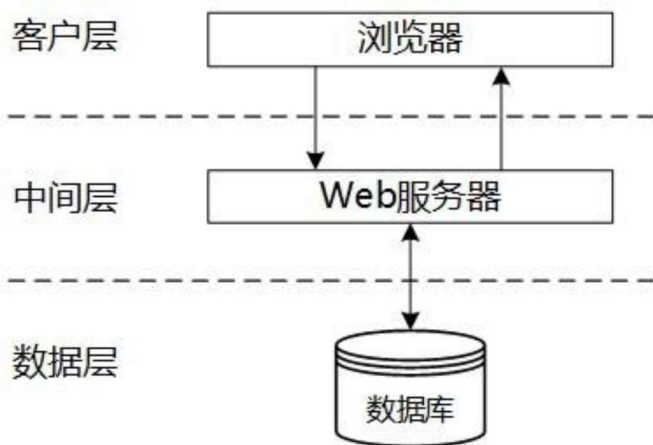


图 5-1 某社交网站系统架构

【问题 1】（10 分）针对目前出现的 Web 服务器负载过大问题，项目组决定在客户端与中间层 Web 服务器之间引入负载均衡器，通过中间层 Web 服务器集群来提高 Web 请求的并发处理能力。在讨论拟采用的负载均衡机制时，王工提出采用基于 DNS 的负载均衡机制，而李工则认为应采用基于反向代理的负载均衡机制，项目组经过讨论，最终确定采用李工提出的方案。请用 200 字以内的文字，分别简要说明两个机制的基本原理；并从系统执行效率、安全性及简易性等方面将两种机制进行对比，将对比结果填入表 5-1 中。

表 5-1 两种负载均衡机制对比分析表

特性		基于 DNS 的负载均衡	基于反向代理的负载均衡
系统执行效率	是否考虑内容服务器性能差异及实时负载情况	(1)	(2)
	是否可对内部服务器静态资源进行缓存	(3)	(4)
安全性	是否能屏蔽客户端对真实 Web 服务器的直接访问	(5)	(6)
简易性	是否具有实现简单、容易实施及低成本的特性	(7)	(8)

注：请在表格 (1) ~ (8) 处填入“是”或“否”

【问题 2】（7 分）针对并发数据库访问所带来的磁盘 I/O 瓶颈问题，项目组决定在数据层引入数据库扩展机制。经过调研得知系统数据库中存储的主要数据为以用户标识为索引的社交网络数据，且系统运行时发生的大部分数据库操作为查询操作。经过讨论，项目组决定引入数据库分区和 MySQL 主从复制两种扩展机制。数据库分区可采用水平分区和垂直分区两种方式，请用 350 字以内的文字说明在本系统中应采用哪种方式及其原因，并分析引入主从复制机制给系统带来的好处。

【问题 3】（8 分）为进一步提高数据库访问效率，项目组决定在中间层与数据层之间引入缓存机制。赵工开始提出可直接使用 MySQL 的查询缓存(query cache)机制，但项目组经过分析好友动态显示等典型业务的操作需求，同时考虑已引入的数据库扩展机制，认为查询缓存尚不能很好地提升系统的查询操作效率，项目组最终决定在中间层与数据层之间引入 Memcached 分布式缓存机制。

(a) 请补充下述关于引入 Memcached 后系统访问数据库的基本过程：系统需要读取后台数据时，先检查数据是否存在于 (1) 中，若存在则直接从其中读取，若不存在则从 (2) 中读取并保存

在 (3) 中；当 (4) 中数据发生更新时，需要将更新后的内容同步到 (5) 实例中。（备选答案：

数据库、Memcached 缓存)

(b) 请结合已知信息从缓存架构、缓存有效性及缓存数据类型等方面分析使用 Memcached 代替数据库查询缓存的原因。

2013 年下半年试题 1

阅读以下关于企业应用系统集成架构设计的说明，在答题纸上回答问题 1 和问题 2。

【说明】某航空公司希望对构建于上世纪七、八十年代的主要业务系统进行改造与集成，提高企业的竞争力。由于集成过程非常复杂，公司决定首先以 Ramp Coordination 系统为例进行集成过程的探索与验证。

在航空业中，Ramp Coordination 是指飞机从降落到起飞过程中所需要进行的各种业务活动的协调过程。通常每个航班都有一位员工负责 Ramp Coordination，称之为 Ramp Coordinator 由 Ramp Coordinator 协调的业务活动包括检查机位环境、卸货和装货等。

由于航班类型、机型的不同，Ramp Coordination 的流程有很大差异。图 1-1 (a) 所示的流程主要针对短期中转航班，这类航班在机场稍作停留后就起飞；图 1-1 (b) 所示的流程主要针对到达航班，通常在机场过夜后第二天起飞；图 1-1 (c) 所示的流程主要针对离港航班，这类航班是每天的第一班飞机。这三种类型的航班根据长途/短途、国内/国外等因素还可以进一步细分，每种细分航班类型的 Ramp Coordination 的流程也略有不同。

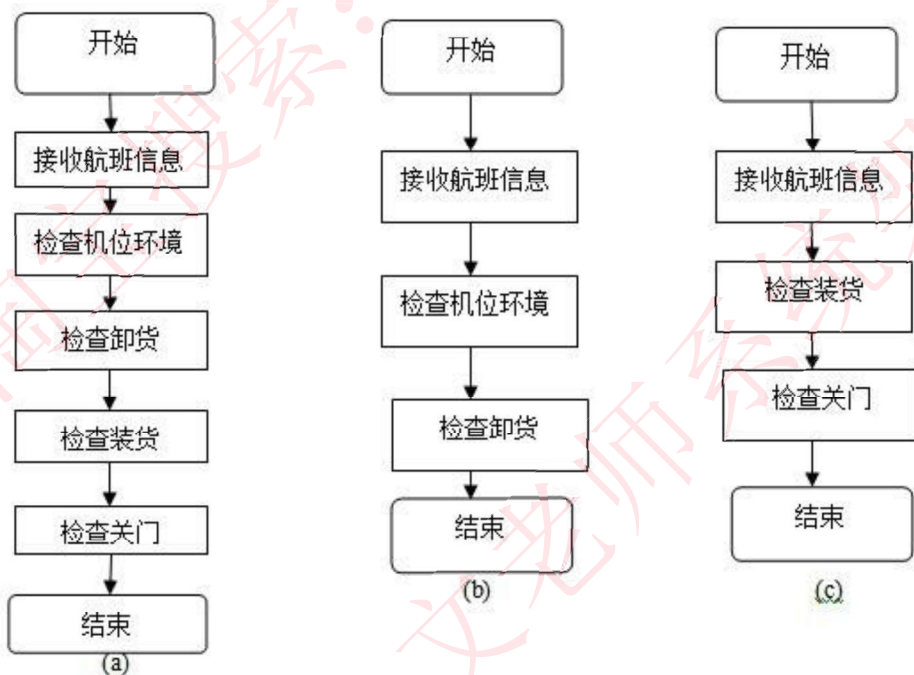


图 1-1 Ramp Coordination 业务流程

为了完成上述业务，Ramp Coordination 信息系统需要从乘务人员管理系统中提取航班乘务员的信息、从订票系统中提取乘客信息、从机务人员管理系统中提取机务人员信息、接收来自航班调度系统的航班到达事件。其中乘务人员管理系统和航班调度系统运行在大型主机系统中，机务人员管理系统运行在 Unix 操作系统之上，订票系统基于 Java 语言，具有 Web 界面，运行在 Linux 操作系统之上。

目前 Ramp Coordination 信息系统主要由人工完成所有协调工作，效率低且容易出错。公司领导要求集成后的 Ramp Coordination 信息系统能够针对不同需求迅速开展业务流程，灵活、高效地完成协调任务。针对上述要求，公司 IT 部门的架构师经过分析与讨论，最终采用面向服务的架构，以服务为中心进行 Ramp Coordination 信息系统的集成工作。

【问题 1】(10 分)

服务建模是对 Ramp Coordination 信息系统进行集成的首要工作，公司的架构师首先对 Ramp Coordination 信息系统进行服务建模，识别出系统中的两个主要业务服务组件：

- (1) Ramp Control：负责 Ramp Coordination 信息系统中相关各种业务活动的组件；
- (2) Flight Management：负责航班相关信息的管理，包括航班日程，乘客信息等。

针对上述服务模型，结合题干描述，请为每个业务服务组件提供的服务进行分析与整理，完成表 1-1 中的空白部分。

表 1-1 业务组件服务提供的服务

业务服务组件	提供服务名称
Ramp Control	
Flight Management	

【问题 2】(15 分)

对 Ramp Coordination 信息系统的集成涉及到对乘务人员管理系统、航班调度系统、机务人员管理系统和订票系统的组织与协调，公司架构师决定采用企业服务总线(Enterprise Service Bus, ESB)技术进行系统集成，请用 200 字以内的文字对 ESB 的定义进行描述，给出 ESB 的五个主要功能，并针对题干描述，将恰当的内容填入图 1-2 中的 (1) ~ (6)。

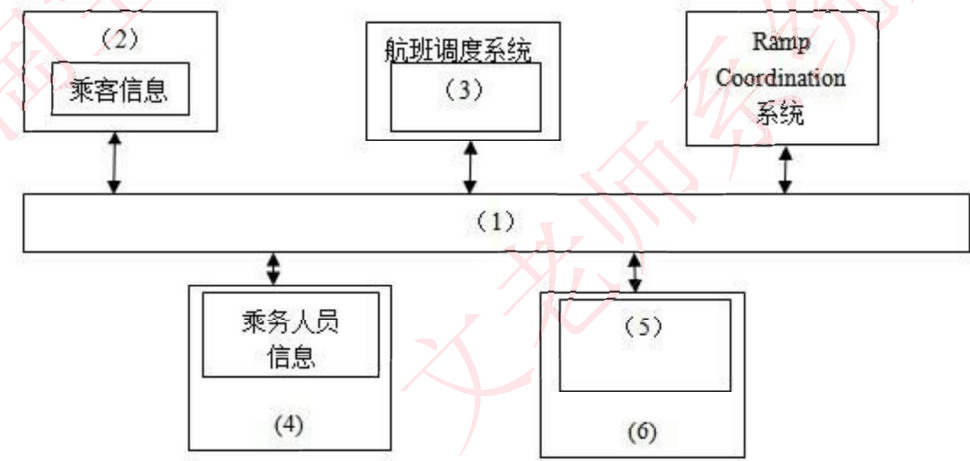


图 1-2 系统集成框架图

2013 年下半年试题 2

阅读以下关于某项目开发计划的说明，在答题纸上回答问题 1 至问题 4。

【说明】某软件公司拟开发一套电子商务系统，王工作为项目组负责人负责编制项目计划。由于该企业业务发展需要，CEO 急于启动电子商务系统，要求王工尽快准备一份拟开发系统的时间和成本估算报告。

项目组经过讨论后，确定出与项目相关的任务如表 2-1 所示。其中，根据项目组开发经验，分别给出了正常工作及加班赶工两种情况下所需的时间和费用。

表 2-1 项目开发任务进度及费用

任务名称	正常工作	加班工作	前置任务
A. 系统调研	4 天/7200 元	3 天/8400 元	-
B. 提交项目计划	2 天/1600 元	1 天/1900 元	A
C. 需求分析	6 天/9600 元	4 天/14200 元	B
D. 系统设计	12 天/22200 元	8 天/27600 元	C
E. 数据库开发	3 天/5100 元	2 天/5700 元	D
F. 网页开发	6 天/8700 元	5 天/10000 元	D
G. 报表开发	4 天/6000 元	任务外包无法赶工	D
H. 测试修改	7 天/9800 元	4 天/12800 元	E, F, G
I. 安装部署	4 天/4000 元	2 天/5000 元	H

【问题 1】（7 分）

请用 400 字以内文字说明王工拟编制的项目计划中应包括哪些内容。

【问题 2】（8 分）

请根据表 2-1，分别给出正常工作和最短工期两种情况下完成此项目所需的时间和费用。

【问题 3】（4 分）如果项目在系统调研阶段用了 7 天时间才完成，公司要求尽量控制成本，王工可在后续任务中采取什么措施来保证项目能按照正常工作进度完成？

【问题 4】（6 分）

如果企业 CEO 想在 34 天后系统上线，王工应该采取什么措施来满足这一要求？这种情况下完成项目所需的费用是多少？

2013 年下半年试题 3

阅读以下有关嵌入式软件 FMEA 方法和相关案例的说明，在答题纸上回答问题 1 至问题 3。

【说明】故障（失效）模型影响分析 FMEA 是分析产品所有可能的故障模式及其可能产生的影响，并按每个故障模式产生影响的严重程度及其发生概率予以分类的一种归纳分析方法。近年来，FMEA 方法已被广泛用于安全关键系统的嵌入式软件可靠性分析工作。

某软件公司承担了一项通信软件的开发项目。该项目由 FC 系统、DY 系统和 GD 系统组成，而 DY 系统（TMS320C25S）软件负责按系统的通信协议完成与 FC 系统的通信，图 3-1 给出了该通信软件的约定层次图。

公司高层将项目交给王工程师，王工认为此项目是安全关键系统，安全等级应为Ⅱ类（致命的），因此应开展软件的 FMEA 分析。

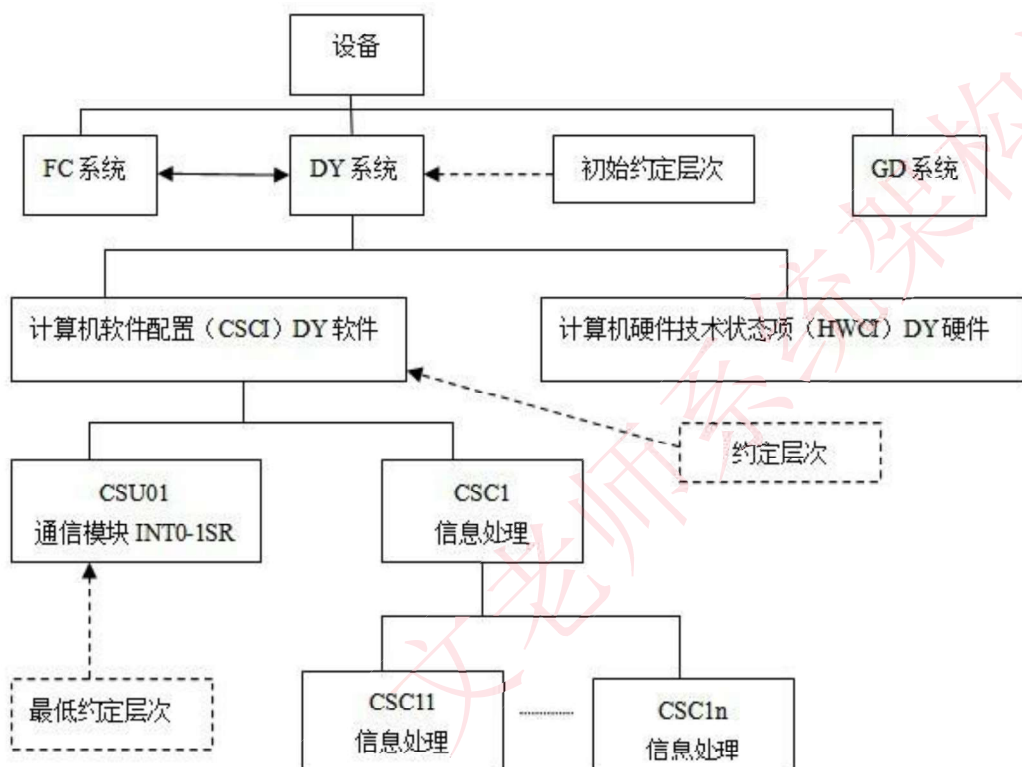


图 3-1 某设备通信软件的约定层次图

【问题 1】（8 分） 请阅读以下有关 FMEA 的描述，将恰当的内容填入（1）～（7）。FMEA 是 FMA（故障模式分析）和 FEA（故障影响分析）的组合，它对系统各种可能的风险进行评价、分析后，在现有技术的基础上消除这些风险或将这些风险降低到可接受的水平。为达到最佳效益，FMEA 必须在产品研制初期进行。

FMEA 实际是一组系列化的活动，其主要活动包括：

- （1）_____；
- （2）_____；
- （3）_____。

由于产品故障可能与设计、制造过程、使用、承包商/供应商以及服务有关，因此 FMEA 又细分为（4）FMEA、（5）FMEA、（6）FMEA 和（7）FMEA 四类。

【问题 2】（10 分）

从图 3-1 可以看出，CSU01 信模块是该项目的关键模块，主要功能定义为：总线通信控制器自动完成一帧数据的接收，存入数据缓冲区，并产生中断（INT0）通知 CPU 从数据缓冲区中读取数据；CPU 读完数据后，将准备好的发送数据写至数据缓存区，写完后通知总线通信控制器自动完成一帧数据的发送。CRC 校验由外部电路完成判别，其结果通过数据线上的相应位进行标识。针对 CSU01 通信模块，简要描述实施 FMEA 的具体内容，填写完成表 3-1 的（1）～（5）。

表 3-1 CSU01 通信模块 FMEA 步骤的主要内容

序号	主要步骤	具体内容
1	故障模式确定	(1)
2	故障原因分析	(2)
3	故障影响分析	(3)
4	危害性分析	(4)
5	改进措施	(5)

【问题 3】（7 分）

表 3-2 给出针对该项目的 CSU01 通信模块的软件故障（失效）模型影响分析 FMECA 表（局部），请根据此题描述情况填写表 3-2 中的（1）~（7）。

注：表 3-2 中的 SRPN（软件风险优先数）=SESR（软件故障模式的严酷度等级）×SOPR（软件故障模式的发生概率等级）×SDDR（软件故障模式的被检测难度等级）。

表3-2 通信模块INT0-ISR的软件FMECA表（局部）

序号	单元	功能	故障模式	故障原因	故障影响			危害性分析				改进措施
					局部影响	高一层次影响	最终影响	SE SR	SO PR	SD DR	SR PN	
1	INT0- ISR	数据接收	通信接口非接收状态	(1)	模块单元无法进入	无法产生INT0中断	通信功能丧失	8	7	4	224	初始化时写 0C300H 地址单元后,读 0C300H 的 D7 位,直到确认通信接口为接收状态。判别中加以计数限制,以保证规定时间到时,记录故障标志并报错
2			中断允许处于禁止状态	程序使用 DINT 和 EINT 不当	模块单元无法进入	(2)	通信功能丧失	8	7	4	224	严格检查 DINT 和 EINT 的语句位置
3			CRC 错误	(3)	接收数据异常	接收数据错误	(4)	7	5	6	210	首先读 0C200H 地址单元的 D0 位,判别 CRC 是否正确,若 CRC 错误,则放弃此帧
4		数据发送	尚未发送就强行设置接收状态	(5)	影响发送数据的正确性	发送数据错误	通信错误	7	6	5	(6)	写 0C200H 地址单元后,读 0C200H 地址单元的 D7 位,判别是否已发送完,再通过写 0C300H 地址单元设置通信接口为接收状态。注:此措施与模式 10 和 11 相结合
5			(7)	总线通信控制器错误	发送数据失败,如果程序处理不当可能造成死循环	发送数据失败,处理不当此单元可能无法退出	通信功能丧失,处理不当可能死机	8	6	7	336	写 0C200H 地址单元后,读 0C200H 地址单元的 D7 位,判别是否已发送完,并加以计数限制,以保证规定时间到时,记录故障标志并退出此模块

2013 年下半年试题 4

阅读以下有关表现层设计方面的说明,在答题纸上回答问题 1 至问题 3。

【说明】某商业银行欲开发一套个人银行系统,为用户提供常见的金融服务,包括转账、查询、存款变更和个人信息管理等功能。该软件除了业务需求外,还有一些特殊的表现层需求:

- (1) 根据用户级别的不同,界面和可用功能是不同的;
- (2) 支持 Web、Windows、手机 App 等多种不同类型的界面;
- (3) 考虑到将来功能的扩展,需要系统支持界面的定制以及动态生成等功能,以降低系统维护和新功能发布的成本。

经过对需求的讨论,该银行初步决定采用 MVC 模式设计该个人银行系统的表现层,采用 XML 作为 GUI 的描述语言,并应用 XML 的界面管理技术来实现灵活的界面配置、界面动态生成和界面定制。

【问题 1】（9 分） MVC 模式强制性地将一个应用处理流程按照模型、视图、控制的方式进行分离，三者的协作关系如图 4-1 所示。

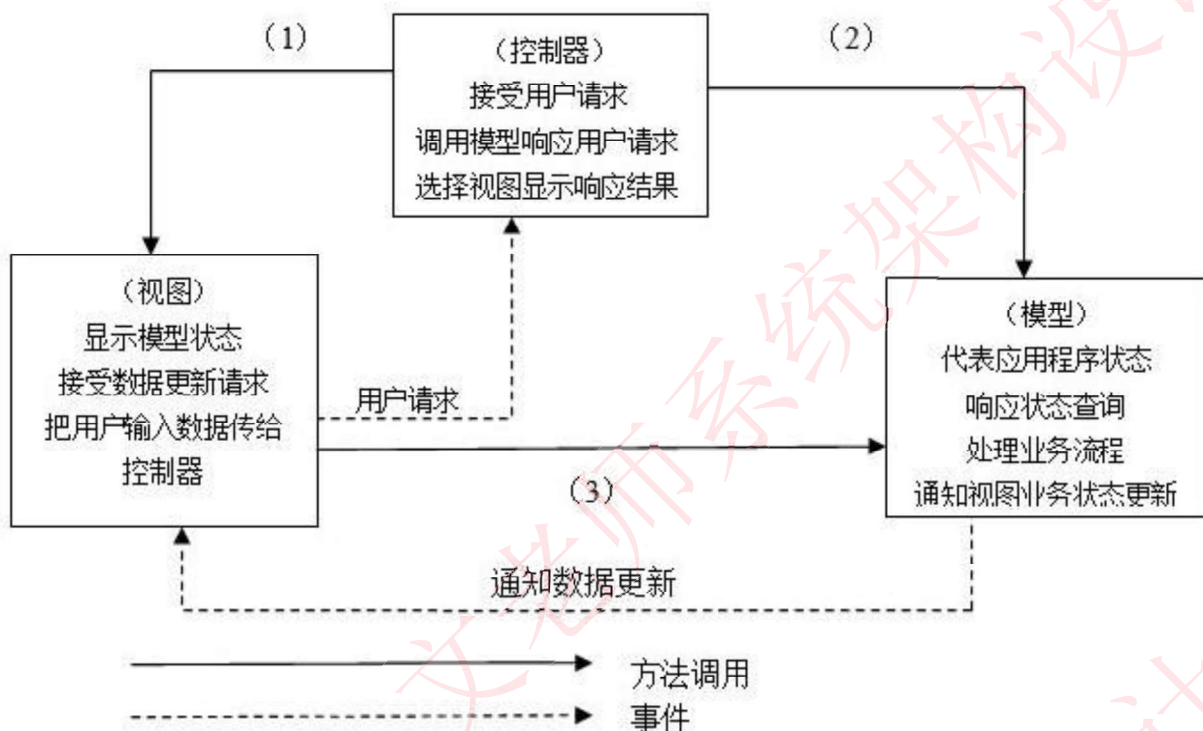


图 4-1 MVC 设计模式

请填写图 4-1 中的 (1) ~ (3)，并简要说明在该个人银行系统中采用 MVC 模式对界面设计的作用。

【问题 2】（4 分） 请从设计模式的角度，简要说明设计方案采用 XML 作为 GUI 描述语言的机制。

【问题 3】（12 分） 基于 XML 的界面管理技术可实现灵活的界面配置、界面动态生成和界面定制，其思路是用 XML 生成配置文件及界面所需的元数据，按不同需求生成界面元素及软件界面，其技术框图如图 4-2 所示。

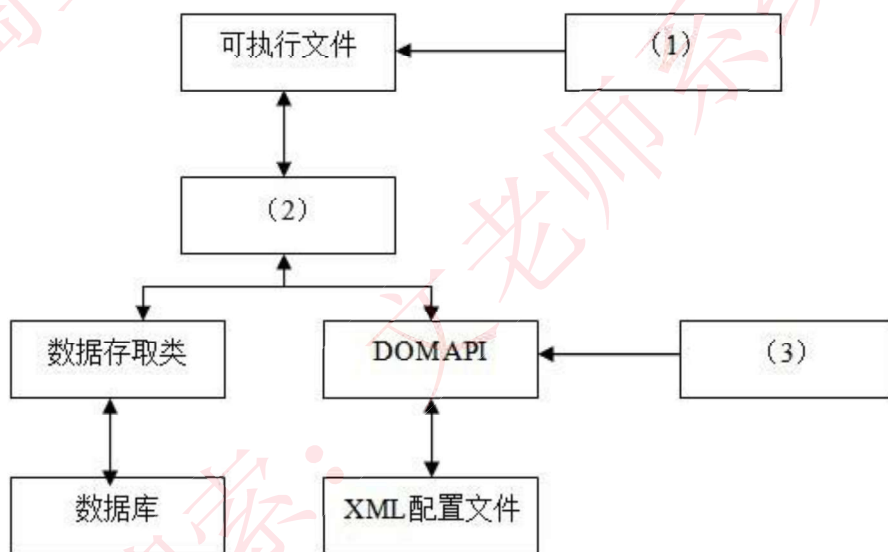


图 4-2 基于 XML 的界面管理技术框架

请将恰当的内容填入图 4-2 中的 (1) ~ (3)，并简要解释说明其含义。

2013 年下半年试题 5

阅读以下有关软件与信息安全方面的说明，在答题纸上回答问题 1 至问题 3。

【说明】某软件公司拟开发一套信息安全支撑平台，为客户的局域网业务环境提供信息安全保护。该支撑平台的主要需求如下：

- (1) 为局域网业务环境提供用户身份鉴别与资源访问授权功能；
- (2) 为局域网环境中交换的网络数据提供加密保护；
- (3) 为服务器和终端机存储的敏感持久数据提供加密保护；
- (4) 保护的主要实体对象包括局域网内交换的网络数据包、文件服务器中的敏感数据文件、数据库服务器中的敏感关系数据和终端机用户存储的敏感数据文件；
- (5) 服务器中存储的敏感数据按安全管理员配置的权限访问；
- (6) 业务系统生成的单个敏感数据文件可能会达到数百兆的规模；
- (7) 终端机用户存储的敏感数据为用户私有；
- (8) 局域网业务环境的总用户数在 100 人以内。

【问题 1】（9 分）在确定该支撑平台所采用的用户身份鉴别机制时，王工提出采用基于口令的简单认证机制，而李工则提出采用基于公钥体系的认证机制。项目组经过讨论，确定采用基于公钥体系的机制，请结合上述需求具体分析采用李工方案的原因。

【问题 2】（7 分）针对需求（7），项目组经过讨论，确定了基于数字信封的加密方式，其加密后的文件结构如图 5-1 所示。请结合需求说明对文件数据进行加密时，应采用对称加密的块加密方式还是流加密方式，为什么？并对该机制中的数据加密与解密过程进行描述。

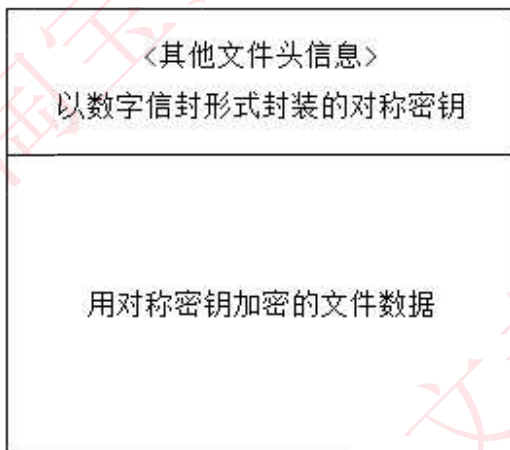


图 5-1 加密数据文件结构

【问题 3】（9 分）对数据库服务器中的敏感关系数据进行加密保护时，客户业务系统中的敏感关系数据主要是特定数据库表中的敏感字段值，客户要求对不同程度的敏感字段采用不同强度的密钥进行防护，且加密方式应尽可能减少安全管理与应用程序的负担。目前数据库管理系统提供的基本数据加密方式主要包括加解密 API 和透明加密两种，请用 300 字以内的文字对这两种方式进行解释，并结合需求

说明应 采用哪种加密方式。

五 案例真题答案解析

2022 年下半年试题 1

答案：

【问题 1】

- (1) 安全性
- (2) 可修改性
- (3) (e)
- (4) (j)
- (5) (h)
- (6) (k)

【问题 2】

应该选择解释器架构风格。

折扣规则的可修改性：解释器风格比面向对象方式实现强。因为解释器风格折扣规则是独立的语法规则，由解释器可对变化的规则进行解析，修改更容易。而面向对象相对固定，有变化需要修改具体的类。

个性化折扣定义灵活性：解释器强于面向对象，解释器可以根据用户灵活解释执行规则，做到千人千面。

系统性能：面向对象优于解释器。面向对象的实现相对固定，而解释器是运行期动态绑定执行。

解析：

第一题必做，架构第一题果然固定了，都是质量属性+架构风格对比，很简单，第一问满分 12 分，第二问选解释器没什么问题吧，很明显的倾向性，然后三个方向按文老师直播说的三个方向都说一下带点倾向性，至少 9 分是能拿到的，本题应该拿到 9+12=21 分。

2022 年下半年试题 2

答案：

【问题 1】

- (1) f
- (2) g
- (3) h
- (4) d
- (5) b
- (6) e

分层细化的数据平衡原则：

1、子图与父图之间的平衡：

(1) 父图与子图之间平衡是指任何一张 DFD 子图边界上的输入/输出数据流必须与其父图对应加工的输入/输出数据流保持一致。

(2) 如果父图中某个加工的一条数据流对应于子图中的几条数据流，而子图中组成这些数据流的数据项全体正好等于父图中的这条数据流，那么它们仍然是平衡的。

2、子图内部：加工的输入和输出需要平衡。

【问题 2】

(1) 项目管理员

(2) 项目经理

(3) 项目指标数据

(4) ~ (6) 指标参数、项目信息、事故及影响因素参数

【问题 3】

数据流图分析阶段：建立系统的功能模型，从而完成需求分析。

数据流图设计阶段：为模块划分与模块之间接口设计提供依据。

数据字典在分析与设计阶段的作用为：

是所有人员工作的依据，统一的标准。它可以确保数据在系统中的完整性和一致性。

具体作用包括：按各种要求列表、相互参照、由描述内容检索名称、一致性检验和完整性检验。

解析：

毫无疑问必选题，而且比较简单，文老师感觉架构历年系统分析的题型也固定了，近三年基本都是数据流图、ER 图或者 UML 的那些图，都是选词填空这种类型的，跟软设的下午题前三题很像而且更简单，那后面可以把软设这块题拿出来也给架构专项联系一下。这个题目整体应该是可以拿到 18 分以上的。

2022 年下半年试题 3

答案：

【问题 1】

心跳是一种用于故障检测的手段。分布式系统中，各种异常，如：宕机、磁盘损坏、网络故障等，时有发生，通过心跳可以快速有效的定位集群中的错误结点，并做及时的处理保证集群正常服务

通常探针会不断发送健康检查来检查服务是否健康。当远程节点没有响应时，我们只能猜测数据包在过程中的某个地方丢失了。下一个操作将是重试或等待一段时间，直到超时。

【问题 2】

(1) (2) : be

(3): f

(4) (5) (6): adh

(7) (8) gc

【问题 3】

通过对系统运行过程中的监测数据进行分析，从而在无精准系统数学模型情况下，对系统进行故障诊断，具体方法包括机器学习、统计分析法和信号分析法。

因为宇航系统是一个非常复杂的系统，如果采用张工的方案（基于解析模型的方法），这一类方法需要建立再精准数学模型的基础上来进行故障诊断。但是对于宇航系统这种非常复杂的系统难以精确建模。所以王工提出了数据驱动的方法，不需要精准系统数学模型。

解析：

这个题目不应该选，大家刚看到问题，心跳倒是讲过，超时检测也能猜出来啥意思，问题二是选词填空，但是问题三那些就没听说过了，很明显超纲了，不建议选。

2022 年下半年试题 4

答案：

【问题 1】

实时方案：当数据库数据更新时，同时更新内存的缓存数据。

异步准实时更新方案：当数据库数据更新时，不立即更新缓存数据，而是将需要更新的操作记录成日志，再逐步排队完成更新。

本题中，建议采用准实时方案，理由是：题目中对性能有严格要求，要求 1s 内完成。实时同步方案最大的问题在于同步并发时的性能不可控。所以准实时方案才能确保该要求能实现。

【问题 2】

哈希分片：通过对 key 进行 hash 操作，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，把存储结点和需要存储的数据都存放在一个 hash 环上，数据根据 hash 值在 hash 环上按正时针方向找到对应的数据存储结点上。

一致性哈希分片的方式在扩充缓存结点时，只需要对少量数据进行存储位置的更新，而哈希分片需要对几乎所有数据进行存储位置更新。

【问题 3】

布隆过滤器通过一个很长的二进制向量和一系列随机映射函数来记录与识别某个数据是否在一个集合中。如果数据不在集合中，能被识别出来，不需要到数据库中进行查找，所以能将数据库查询返回值为空的查询过滤掉。

优点：

- 1、占用内存小
- 2、查询效率高
- 3、不需要存储元素本身，在某些对保密要求比较严格的场合有很大优势

缺点：

- 1、有一定的误判率，即存在假阳性，不能准确判断元素是否在集合中。
- 2、不能获取元素本身
- 3、一般情况下不能从布隆过滤器中删除元素

解析：

这个题目不知道大家有没有选，也是不建议选，如果选择的话，问题 2 和问题 3 基本都是没法做的，那些概念都是超纲的，基本是全部丢分，问题一也得有相关经验的才能做对。

2022 年下半年试题 5

答案：

【问题 1】

MQTT（消息队列遥测传输）是一个基于发布/订阅模式的消息协议。它工作在 TCP/IP 协议族上，是为硬件性能低下的远程设备以及网络状况糟糕的情况下而设计的发布/订阅型消息协议。MQTT 协议是轻量、简单、开放和易于实现的。

【问题 2】

- (1) HTTP
- (2) MQTT
- (3) MQTT
- (4) MQTT
- (5) HTTP
- (6) HTTP
- (7) 端侧识别模块（在边缘设备上运行的只有 7）
- (8) 模型训练模块（承上启下，一方面跟访客注册模块通信，另一方满跟端侧识别模块通信）
- (9) 设备调度平台模块（对边缘设备管理的是 9）
- (10) 访客注册模块（小程序前端接受访客注册）

【问题 3】

数据通信：通信更快捷，数据量更少。因为数据处理对比在边缘设备上完成，通信更多时候只传输匹配与结果的指令。

数据安全：数据以加密方式存储在需要用到的边缘设备上，本地化处理比对，减少原始信息在网上的传递带来的安全隐患。黑客也无法通过攻破一个结点使整个系统瘫痪。

系统性能：性能更高，以人脸识别为例，在进行识别时，只在本地进行比对不用把人脸数据传递到远程服务器对比。

解析：

其实，本题是建议选的，相比较前面两题可以拿多分，因为其问题二是完全的阅读理解题，与技术无关，前面六分选协议，主要是分外部还是内部交互，可能会选错一半，但是后面八分四个空功能选择是有比较明显指向性的，很容易选择，然后关键 MQTT 这个东西，虽然没讲过，但是大家写论文的时候普遍都抄了这玩意，不记得了吗？在观察者模式还是隐式调用的实例那里，而且都抄的 MQTT 是个什么东西，别说抄完不记得了啊，这样问题一至少能有点分，至于问题三三个方面都是很通俗易懂的方向，扯点定义就有分，本题拿 12 分以上不难。

2021 年下半年试题 1

答案：

【问题 1】

1. 性能
 2. 可修改性
 3. e
 4. j
 5. h
 6. i

【问题 2】

应采取解释器风格。

(1) 解释器风格是自定义了一套规则供使用者使用，使用者基于这个规则来开发构件，能够跨平台适配。

(2) 管道-过滤器风格每个构件都有一组输入和输出，构件读取输入的数据流，经过内部处理（计算或增值），产生输出数据流。前一个构件的输出作为后一个构件的输入，前后数据流关联。过滤器就是构件，连接件就是管道。

(3) 隐式调用风格是构件不直接调用一个过程，而是触发或广播一个或多个事件。构件中的过程在一个或多个事件中注册，当某个事件被触发时，系统自动调用在这个事件中注册的所有过程。一个事件的触发就导致了另一个模块中的过程调用。

平台支持初学者和高级用户两种界面操作模式，用户可以根据自己的情况灵活选择合适的模式：从灵活性上解释器可以通过灵活的自定义规则实现规则的重组。

从可扩展性上解释器可以包括一个完成解释工作的解释引擎、一个包含将被解释的代码的存储区、一个记录解释引擎当前工作状态的数据结构，以及一个记录源代码被解释执行的进度的数据结构。可以通过新建规则实现可扩展性。

解析：本题的题型和我考前发布的案例冲刺题预测类似，可以猜测以后架构第一题就是这种题型了，质量属性+架构风格对比，对架构学员来说是一件好事，以后有了主攻方向，案例会更简单。本问题一没问题，问题二就看个人发挥了，整体拿 18 分应该没问题。

2021 年下半年试题 2

答案：

【问题 1】

- (1) 系统管理员 (2) 患者
- (3) a (4) c (5) f (6) h (7) i (8) j
- (9) b (10) d (11) e (12) g
- (4) - (8) 答案可互换。(9) - (12) 答案可互换。

【问题 2】

(1) 预约人员 (2) 发起预约挂号请求 (3) 显示医生出诊时段 (4) 显示是否预约成功
顺序图强调交互的消息时间顺序。

协作图强调接受和发送消息的对象的结构组织，强调通信的方式。

【问题 3】

对象模型描述系统中对象的静态结构、对象之间的关系、属性和操作，主要用对象图来实现。

动态模型描述与时间和操作顺序有关的系统特征，例如，激发事件、事件序列、确定事件先后关系的状态等，主要用状态图来实现。

功能模型描述一个计算如何从输入值得到输出值，它不考虑计算的次序，主要用 DFD 来实现。

功能模型指发生了什么，动态模型确定什么时候发生，而对象模型确定发生的客体。

对象设计建立基于分析模型的设计模型并考虑实现细节，可用于软件的需求分析。

解析：本题也在我们考前发布的案例冲刺题里能找到类似题型，问题 1 和问题 2 是很容易的，问题 3 的三个模型没详细说过，需要自己顾名思义去发散，整体拿 18 分应该可以。

2021 年下半年试题 3

答案：

【问题 1】

【问题 2】

解析：题目不全，暂无答案，待补充，从已有信息来看，本题不应该选。

2021 年下半年试题 4

答案：

【问题 1】

常见反规范化技术如下：

(1) 增加冗余列：在多个表中保留相同的列，通过增加数据冗余减少或避免查询时的连接操作。

(2) 增加派生列：在表中增加可以由本表或其它表中数据计算生成的列，减少查询时的连接操作并避免计算或使用集合函数。

(3) 重新组表：如果许多用户需要查看两个表连接出来的结果数据，则把这两个表重新组成一个表来减少连接而提高性能。

(4) 水平分割表：根据一列或多列数据的值，把数据放到多个独立的表中，主要用于表数据规模很大、表中数据相对独立或数据需要存放到多个介质上时使用。

(5) 垂直分割表：对表进行分割，将主键与部分列放到一个表中，主键与其它列放到另一个表中，在查询时减少 I/O 次数。

用户查询商品信息采用的是增加冗余列的方式。

【问题 2】

批处理维护、应用逻辑和触发器。

(1) 批处理维护：指对复制列或派生列的修改积累一定的时间后，运行一批处理作业或存储过程对复制或派生列进行修改，这只能在对实时性要求不高的情况下使用。

(2) 应用逻辑: 要求必须在同一事务中对所有涉及的表进行增、删、改操作。用应用逻辑来实现数据的完整性风险较大, 因为同一逻辑必须在所有的应用中使用和维护, 容易遗漏, 特别是在需求变化时, 不易于维护。

(3) 触发器: 对数据的任何修改立即触发对复制列或派生列的相应修改。触发器是实时的, 而且相应的处理逻辑只在一个地方出现, 易于维护。一般来说, 是解决这类问题比较好的办法。

该系统应该采用触发器。

【问题 3】

(1) ZSet

(2) 一、对强一致要求比较高的, 应采用实时同步方案, 即查询缓存查询不到再从 DB 查询, 保存到缓存; 更新缓存时, 先更新数据库, 再将缓存的设置过期(建议不要去更新缓存内容, 直接设置缓存过期)。

二、对于并发程度较高的, 可采用异步队列的方式同步, 可采用 kafka 等消息中间件处理消息生产和消费。

三、使用阿里的同步工具 canal, canal 实现方式是模拟 mysql slave 和 master 的同步机制, 监控 DB binlog 的日志更新来触发缓存的更新, 此种方法可以解放程序员双手, 减少工作量, 但在使用时有些局限性。

四、采用 UDF 自定义函数的方式, 面对 mysql 的 API 进行编程, 利用触发器进行缓存同步。

解析: 本题可以选, 不过只能拿低分, 只有问题一是有把握的, 问题二、三就要看缘分, 整体可能只能得到 10-12 分左右。

2021 年下半年试题 5

答案:

【问题 1】

在网关管理方面, 基于云平台的智能家居管理系统可以将分散的智能家居网关数据集中起来, 实现对智能家居网关的远程高效管理。

在数据处理方面, 云端服务器对智能家居网数据进行备份存储, 当家庭网关由于故障等原因导致数据丢失时, 可以通过云端管理系统对网关数据进行恢复, 从而提高数据的容灾性。

在系统性能方面, 基于云服务平台的智能家居管理系统将数据信息存储在云端, 减少了数据请求时间, 提高了通信效率。

【问题 2】

(1) h 用户终端 (2) i 鸿蒙 (3) f 云平台 (4) d 数据库 (5) e 家庭网关 (6) c 驱动程序

【问题 3】

TCP 在 IP 协议提供的不可靠数据服务的基础上, 采用了重发技术, 为应用程序提供了一个可靠的、面向连接的、全双工的数据传输服务。TCP 协议一般用于传输数据量比较少, 且对可靠性要求高的场合。

UDP 是一种不可靠的、无连接的协议,可以保证应用程序进程间的通信,与 TCP 相比,UDP 是一种无连接的协议,它的错误检测功能要弱得多。

该系统应采用 TCP 协议

解析: 本题应该选,典型的选图填空题,而且也有熟悉的知识点,整体可以拿 15 分以上。

2020 年下半年试题 1

试题一分析

本题考查软件架构评估方面的知识与应用,主要包括质量属性效用树和架构分析两个部分。

此类题目要求考生认真阅读题目对系统需求的描述,经过分类、概括等方法,从中确定软件功能需求、软件质量属性、架构风险、架构敏感点、架构权衡点等内容,并采用效用树这一工具对架构进行评估。

【问题 1】

本问题考查考生对影响系统架构风格选型的理解与掌握。根据系统要求,李工建议采用管道-过滤器(pipe and filter)的架构风格,而王工则建议采用仓库(repository)架构风格。考生需要从系统的主要功能和要求,从数据处理方式、系统的可扩展性和处理性能三个方面对这两种架构风格的优势和劣势进行比较与分析。具体如下表所示。

架构风格名称	数据处理方式	系统可扩展性	处理性能
管道-过滤器	数据驱动机制,处理流程事先确定,交互性差	数据与处理紧密关联,调整处理流程需要系统重新启动	劣势: 需要数据格式转换,性能降低 优势: 支持过滤器并发调用,性能提高
仓库	数据存储为中心仓库,处理流程独立,支持交互式处理	数据与处理耦合,可动态添加和删除处理组件	劣势: 数据与处理分离,需要加载数据,性能降低 优势: 数据处理组件之间一般无依赖关系,可并发调用,提高性能

经过综合比较与分析,可以看出该系统更适合使用仓库风格。

【问题 2】

在架构评估过程中,质量属性效用树(utility tree)是对系统质量属性进行识别和优先级排序的重要工具。质量属性效用树主要关注性能、可用性、安全性和可修改性等四个用户最为关注的质量属性,考生需要对题干的需求进行分析,逐一找出这四个质量属性对应的描述,然后填入空白处即可。

经过对题干进行分析,可以看出:

- (a) 根据用户的付费情况对用户进行分类,并根据类别提供相应的开发功能(功能需求);
- (b) 在正常负载情况下,系统应在 0.2 秒内对用户的界面操作请求进行响应(性能);
- (c) 系统应该具备完善的安全防护措施,能够对黑客的攻击行为进行检测与防御(安全性);
- (d) 系统主站点断电后,应在 3 秒内将请求重定向到备用站点(可用性);
- (e) 系统支持中文昵称,但用户名必须以字母开头,长度不少于 8 个字符(功能需求);
- (f) 系统宕机后,需要在 15 秒内发现错误并启用备用系统(可用性);
- (g) 在正常负载情况下,用户的代码提交请求应该在 0.5 秒内完成(性能);

- (h) 系统支持硬件设备灵活扩容, 应保证在 2 人·天内完成所有的部署与测试工作(可修改性);
- (i) 系统需要为针对代码仓库的所有操作情况进行详细记录, 便于后期查阅与审计(安全性);
- (j) 更改系统的 Web 界面风格需要在 4 人·天内完成(可修改性);
- (k) 系统本身需要提供远程调试接口, 支持开发团队进行远程排错(可测试性)。

参考答案

【问题 1】

该系统更适合采用仓库架构风格。

- (1) 数据存储在中心仓库, 处理流程独立, 支持交互式处理。
- (2) 数据与处理紧密关联, 调整处理流程需要系统重新启动。
- (3) 数据与处理分离, 需要加载数据, 性能降低。
- (4) 数据处理组件之间一般无依赖关系, 可并发调用, 提高性能。

【问题 2】

- (1) 安全性
- (2) 可修改性
- (3) (g)
- (4) (i)
- (5) (f)
- (6) (j)

2020 年下半年试题 2

试题二分析

本题考查数据库设计与建模相关知识及应用。

数据库设计过程包括了逻辑数据建模和物理数据建模, 逻辑数据建模阶段主要构造实体 联系图表达实体及其属性和实体间的联系, 物理数据建模阶段主要根据所选数据库系统设计 数据库模式。实体联系图(Entity Relationship Diagram)指以实体、联系、属性三个基本概念概括数据的基本结构, 从而描述静态数据结构的模式。实体是具有公共性质的可相互 区别的现实世界对象的集合, 可以是具体的, 也可以是抽象的概念或联系。属性是实体所具有的模拟特性, 一个实体可由若干个属性来刻画。联系是数据对象彼此之间存在的相互关系。

此类题目要求考生认真阅读题目对问题的描述, 准确理解数据库设计的主要任务和实体 联系图中各个元素的含义, 结合图中所给出的包裹单示意图中所描述的数据项, 分析其关系 确定实体、属性和联系。

【问题 1】

在关系型数据库开发中, 逻辑数据模型建设的主要任务是构建实体联系图。构建过程中,

首先通过上下文数据模型确定实体及其联系, 为每个实体确定其标识性属性并添加完整属性, 在此基础上利用规范化技术对所建立逻辑数据模型进行优化, 一般需要满足第三范式 3NF 要求。对图 2-1 所示包裹单中所有数据项进行分析, 主要涉及的实体包括收件人、寄件人及 其之间的关联实

体包裹单，其余数据项设计为上述三个实体的属性即可。

【问题 2】

数据库建模中可以对属性相似的实体进行进一步的抽象，通过将多个实体中相同的属性组合起来构造出新的抽象实体，即超类实体，原有多个实体称之为子类实体，通过两者之间的继承关系来表达抽象实体和具体实体的关系。图 2-1 中收件人和寄件人的属性都包括了姓名、电话、单位名称、详细地址和邮政编码等信息，可以设计出一个超类实体“用户”来实现通用属性的抽象表示。

【问题 3】

在数据库优化过程中，第三范式要求消除派生属性，即某个实体的非主键属性由该实体其他非主键属性决定，那么该属性可以称之为派生属性。图 2-1 所示属性中，包裹单的属性“费用总计”是由资费、挂号费、保价费、回执费等计算得出，所以是派生属性。

参考答案

【问题 1】

逻辑数据模型设计过程包含的任务：

- (1) 构建系统上下文数据模型，包含实体及实体之间的联系；
- (2) 绘制基于主键的数据模型，为每个实体添加主键属性；
- (3) 构建全属性数据模型，为每个实体添加非主键属性；
- (4) 利用规范化技术建立系统规范化数据模型。

包裹单的逻辑数据模型中包含的实体：

- (1) 收件人（主键：电话）；
- (2) 寄件人（主键：电话）；
- (3) 包裹单（主键：编号）。

【问题 2】

超类实体是将多个实体中相同的属性组合起来构造出的新实体。

用户（姓名、电话、单位名称、详细地址）

【问题 3】

派生属性是指某个实体的非主键属性由该实体其他非主键属性决定。

包裹单中的总计是由资费、挂号费、保价费、回执费计算得出，所以是派生属性。

2020 年下半年试题 3

试题三分析

FACE 是近年来宇航领域提出的一种面向服务的、安全可靠、可移植、可扩展的开放式嵌入式系统架构，可实现宇航软件的跨平台复用以及高质量、低成本的开发工作。从图 3-1 可以看出，FACE 将宇航软件分为 5 个功能服务段，各段之间通过标准的服务接口或传输服务实现功能间的相互调用。架构设计是软件系统开发中的重要环节，其架构的优劣直接影响着软件系统的功能实现，因此，架构能否全面反映需求是架构设计的重中之重。

【问题 1】

通常在软件开发过程中，需求会随着开发深入而有所变化，而架构又不能完全地将需求全部反映出来，因此，如何把软件需求映射到软件架构是至关重要一个问题。在架构设计时，架构设计师应密切关注需求到架构的映射存在以下 5 方面的难点：

(1) 需求和架构描述语言存在差异：软件需求是频繁获取的非正规的自然语言，而软件架构常用某种正式语言。

(2) 非功能属性难以在架构中描述：系统属性中描述的非功能性需求通常很难在架构模型中形成规约。

(3) 需求和架构的一致性难以保障：从软件需求映射到软件架构的过程中，保持一致性和可追溯性很难，且复杂程度很高，因为单一的软件需求可能定位到多个软件架构的关注点。反之，架构元素也可能有多个软件需求。

(4) 用迭代和同步演化方法开发软件时，由于需求的不完整而带来的架构设计困难：架构设计必须基于一个准确的需求开展，而有些软件需求只能在建模后甚至是在架构实现时才能被准确理解。

(5) 难以确定和细化包含这些需求的架构相关信息：大规模系统必须满足数以千计的需求，会导致很难确定和细化包含这些需求的架构相关信息。

【问题 2】

从图 3-1 可知，FACE 架构由 5 个基本段组成，每段内又分为多个功能服务，从这些服务可以看出每段的基本能力。例如，操作系统段是 FACE 架构的基本功能，除基本操作系统外，还涵盖了运行库，操作系统的健康监控（HM），图中所给出的 OSGi 框架，实现功能组件“即插即用”能力。如果考生掌握了面向服务的架构风格，就不难给出各个段的具体含义。

(1) 操作系统服务段：为 FACE 架构其他段提供操作系统、运行时和操作系统级健康监控等服务。通过开放式 OSGi 框架为上层功能提供 OS 标准接口，并可实现上层组件的即插即用能力。本段是 FACE 架构的基本服务段。

(2) I/O 服务段：主要针对专用 I/O 设备进行抽象，屏蔽平台服务段软件与硬件设备的关系，形成一种虚拟设备，这里隐含着对系统中的所有硬件 I/O 的虚拟化。由于图形服务软件和 GPU 处理器紧密相关，因此 I/O 服务段不对 GPU 驱动进行抽象。

(3) 平台服务段：主要是指平台/用户需要的共性服务软件，主要涵盖跨平台的系统管理、共享设备服务，以及健康管理等。如：系统级健康监控（HM）、配置、日志和流媒体等服务。本段主要包括平台公共服务、平台设备服务和平台图像服务等三类。

(4) 传输服务段：通过使用传统跨平台中间件软件（如 CORBA、DDA 等），为平台上层可移植组件段提供平台性的数据交换服务，可移植组件将通过传输服务段提供的服务实现交换，禁止组件间直接调用。本段应具备 QoS 质量特征服务、配置能力服务以及分布式传输服务等。

(5) 可移植组件段：为用户软件段，提供了多组件使用能力和功能服务。主要包括公共服务和可移植组件两类。

【问题 3】

紧耦合和封装是软件模块化设计中最难以解决的两个问题，要使软件具备良好的可移植性、可复用性，就必须清楚其问题的表现形式。

紧耦合是应用程序移植的一个障碍，进一步说，就是计算平台的硬件设备和软件模块及其沟通之间的耦合代表了一个应用程序的可移植性方面的障碍。原因是便携性使得每个平台设备都有一个接口控制文件（ICD），描述了由硬件所支持的消息和协议，应用程序对消息和协议的支持将紧密耦合于硬件。若要移植，需要太多的工作来修改应用程序以支持不同的结构元素。

为了尽量减少支持新的硬件设备所需要的工作，可采用分离原则，通过隔离实现硬件特定信息和少数模块的代码，来减少耦合性。

通常紧耦合问题主要表现在 I/O 问题、业务逻辑问题和表现问题。

传统的应用程序不可移植的另一个原因是这些应用程序被紧密耦合到一组固定的接口，而这些数据的每个数据源或槽（sinks）都暴露出了设备的特殊接口，这些特殊接口在每个平台中都是不同的。这样，支持平台设备的接口控制文件（ICD）是被硬编码到应用程序中，就导致应用程序不能成功在不同计算平台上执行。

为了解决这种接口控制文件（ICD）被硬编码而难以封装的问题，可以通过提供数据源或槽的软件服务的方法，从紧耦合组件分解出应用程序，并将平台相关部分加入计算环境中，在计算平台内提供数据源或槽的软件服务，并实现接口标准化。

通常封装问题主要表现在：ICD 硬编码问题、组件的紧耦合问题、直接调用问题。

参考答案

【问题 1】

（1）需求和架构描述语言存在差异：软件需求是频繁获取的非正规的自然语言，而软件架构常用的是一种正式语言。

（2）非功能属性难于在架构中描述：系统属性中描述的非功能性需求通常很难在架构模型中形成规约。

（3）需求和架构的一致性难以保障：从软件需求映射到软件架构的过程中，保持一致性和可追溯性很难，且复杂程度很高，因为单一的软件需求可能定位到多个软件架构的关注点。反之，架构元素也可能有多个软件需求。

【问题 2】

操作系统服务段：为 FACE 架构其他段提供操作系统、运行时和操作系统级健康监控等服务。通过开放式 OSGi 框架为上层功能提供 OS 标准接口，并可实现上层组件的即插即用能力。

I/O 服务段：主要针对专用 I/O 设备进行抽象，屏蔽平台服务段软件与硬件设备的关系。由于图形服务软件和 GPU 处理器紧密相关，因此 I/O 服务段不对 GPU 驱动进行抽象。

平台服务段：主要是指用户需要的共性软件，如：系统级健康监控（HM）、配置、日志和流媒体等服务。本段可包括平台公共服务、平台设备服务和平台图像服务等三类。

传输服务段：主要为上层可移植组件段提供平台性的数据交换服务。可移植组件将通过传输服务段提供的服务实现交换，禁止组件间直接调用。

可移植组件段：提供了多组件使用能力和功能服务。主要包括公共服务和可移植组件两类。

【问题 3】

紧耦合问题主要表现在：I/O 问题、业务逻辑问题和表现问题。

解决方案：可采用分离原则，通过隔离实现硬件特定信息和少数模块的代码，减少耦合性。

封装问题主要表现在：ICD 硬编码问题、组件的紧耦合问题、直接调用问题。

解决方案：可以通过提供数据源或槽的软件服务的方法，将紧耦合组件分解出应用程序，并将平台相关部分加入计算环境中，在计算平台内提供数据源或槽的软件服务，并实现接口标准化。

2020 年下半年试题 4

试题四分析

本题考查数据库缓存的基本概念和具体应用。

【问题 1】

本问题考查 Redis 数据库缓存产品基本数据类型的常见应用。

- (1) STRING 类型：常规的 key/value 缓存应用，常规计数如粉丝数等；
- (2) LIST 类型：各类列表应用，如关注列表、好友列表、订阅列表等；
- (3) SET 类型：与 LIST 类似，但提供去重操作，也提供集合操作，可实现共同关注、共同喜好、共同好友等功能；
- (4) HASH 类型：存储部分变更数据，如用户数据等；
- (5) ZSET 类型：类似 SET 但提供自动排序，也可实现带权重的队列，如各类排行榜等。

【问题 2】

本问题考查 Redis 持久化存储的基本概念及应用。

Redis 提供了两种持久化存储的机制，分别是 RDB (Redis DataBase)持久化方式和 AOF (Append Only File)持久化方式。RDB 持久化方式是指在指定的时间间隔内将内存中的数据 集快照写入磁盘，是 Redis 默认的持久化方式。AOF 方式是指 redis 会将每一个收到的写命令都通过 write 函数追加到日志文件中。

两种方式各有优缺点，大致的比较如下：

- (1) 磁盘更新频率：AOF 比 RDB 文件更新频率高。
- (2) 数据安全：AOF 比 RDB 更安全。
- (3) 数据一致性：RDB 间隔一段时间存储，可能发生数据丢失和不一致；AOF 通过 append 模式写文件，即使发生服务器宕机，也可通过 redis-check-aof 工具解决数据一致性问题。
- (4) 重启性能：RDB 性能比 AOF 好。
- (5) 数据文件大小：AOF 文件比 RDB 文件大。

该项目的实际需求是：在系统出现宕机等故障时，需要在最短时间内通过重启等方式重新建立服务，因此重启性能是最需要考虑的因素，故该开发团队选择 RDB 方式。

【问题 3】

本问题考查 Redis 使用过程中数据清除相关的概念。

缓存中 般用来存储当前的热点数据，Redis 为每个 KEY 值都设置了过期时间，以提高缓存命中率。为了清除非热点数据，Redis 选择“定期删除+惰性删除”策略。

“定期删除+惰性删除”策略也会存在失效的可能。比如，如果“定期删除”没删除 KEY，也没

即时去请求 KEY, 也就是说“惰性删除”也没生效。这样, Redis 默认的“定期删除+ 惰性删除”策略就失效了。

如果该策略失效, Redis 内存使用率会越来越高, 一般应采用内存淘汰机制来解决。常见的内存淘汰机制有:

- (1) 从已设置过期时间的数据集最近最少使用的数据淘汰。
- (2) 从已设置过期时间的数据集将要过期的数据淘汰。
- (3) 从已设置过期时间的数据集任意选择数据淘汰。
- (4) 从数据集最近最少使用的数据淘汰。
- (5) 从数据集任意选择数据淘汰。

参考答案

【问题 1】

- (1) (a)
- (2) (b)、(g)
- (3) (c)、(d)
- (4) (f)
- (5) (e)

【问题 2】

磁盘更新频率: AOF 比 RDB 文件更新频率高。

数据安全: AOF 比 RDB 更安全。

数据一致性: RDB 间隔一段时间存储, 可能发生数据丢失和不一致; AOF 通过 append 模式写文件, 即使发生服务器宕机, 也可通过 redis-check-aof 工具解决数据一致性问题。

重启性能: RDB 性能比 AOF 好。

数据文件大小: AOF 文件比 RDB 文件大。

综合上述五个方面的比较, 考虑在系统出现宕机等故障时, 需要在最短时间内通过重启等方式重新建立服务, 因此开发团队最终选择了 RDB 方式。

【问题 3】

失效场景: 如果“定期删除”没删除 KEY, 也没即时去请求 KEY, 也就是说“惰性删除”也没生效。这样, Redis 默认的“定期删除+惰性删除”策略就失效了。

对此, 可采用内存淘汰机制解决:

- (1) 从已设置过期时间的数据集最近最少使用的数据淘汰。
- (2) 从已设置过期时间的数据集将要过期的数据淘汰。
- (3) 从已设置过期时间的数据集任意选择数据淘汰。
- (4) 从数据集最近最少使用的数据淘汰。
- (5) 从数据集任意选择数据淘汰。

2020 年下半年试题 5

试题五分析

本题考查 Web 系统架构设计相关知识及如何在实际问题中综合应用。

此类题目要求考生认真阅读题目对现实系统需求的描述，结合 Web 系统设计相关知识、实现技术等完成 Web 系统分析设计。

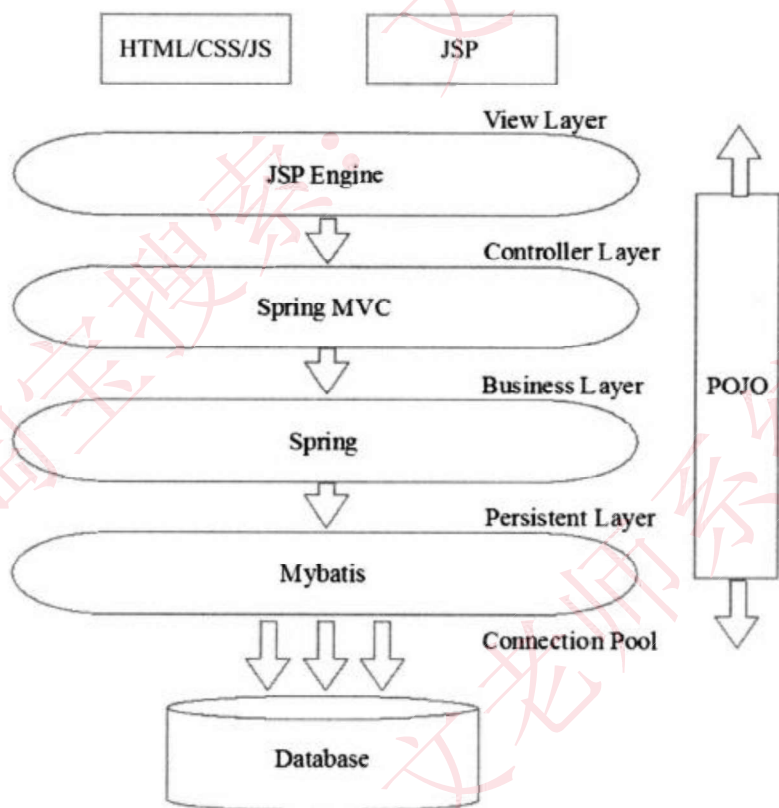
【问题 1】

软件质量属性有可用性、可修改性、性能、安全性、可测试性、易用性等。可用性关注的是系统产生故障的可能性和从故障中恢复的能力；性能关注的是系统对事件的响应时间；安全性关注的是系统保护合法用户正常使用系统、阻止非法用户攻击系统的能力；可测试性关注的是系统发现错误的能力；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。

【问题 2】

SSM 框架是 spring MVC, spring 和 Mybatis 框架的整合，是标准的 MVC 模式。其使用 spring MVC 负责请求的转发和视图管理；spring 实现业务对象管理；Mybatis 作为数据对象的持久化引擎。

因此，基于 SSM 的工业设备监测系统设计架构如下图所示。



【问题 3】

标准的数据访问机制可以在硬件供应商和软件开发商之间建立一套完整的规则。只要遵循这套规则，数据交互对两者来说都是透明的，硬件供应商只需考虑应用程序的多种需求和传输协议，软件开发商也不必了解硬件的实质和操作过程，实现对设备数据采集的统一管理。

例如，OPC (OLE for Process Control)即用于过程控制的 OLE, 是一个工业标准。OPC 是为了不同供应厂商的设备和应用程序之间的软件接口标准化，使其间的数据交换更加简单化的目的而提出的。作为结果，可以向用户提供不依赖于特定开发语言和开发环境且可以自由组合使用的过程控制软件组件产品。利用 OPC 的系统，是由按照应用程序(客户程序)的要求提供数据采集服务的 OPC 服务器，使用 OPC 服务器所必需的 OPC 接口，以及接受服务的 OPC 应用程序所构成。OPC 服务器是按照各个供应厂商的硬件所开发的，使之可以吸收各个供应厂商硬件和系统的差异，从而实现不依存于硬件的系统构成。同时利用一种叫作 Variant 的数据类型，可以不依存于硬件中固有数据类型。

参考答案

【问题 1】

- (1) (a) (b)
- (2) (d) (f)
- (3) (c) (e)

【问题 2】

- (1) (a)
- (2) (c)
- (3) (d)
- (4) (k)
- (5) (j)
- (6) (h)
- (7) (i)

【问题 3】

该工业设备检测系统需与不同设备进行数据交互，采用标准的数据访问机制可以在硬件供应商和软件开发商之间建立一套完整的规则。只要遵循这套规则，数据交互对两者来说都是透明的，硬件供应商只需考虑应用程序的多种需求和传输协议，软件开发商也不必了解硬件的实质和操作过程，实现对设备数据采集的统一管理。

2019 年下半年试题 1

答案：

【问题 1】

用户级别与折扣规则管理功能更适合采用基于规则的架构风格。

- (1) 将用户级别、折扣规则等描述为可动态改变的规则数据；
- (2) 加入新的用户级别和折扣规则时需要重新定义新的对象，并需要重启系统；
- (3) 用户级别和折扣规则已经在系统内编码，可直接运行，性能较好。

【问题 2】

- (1) 安全性
- (2) 可修改性

- (3) (h)
- (4) (j)
- (5) (e)
- (6) (k)

2019 年下半年试题 2

答案：

【问题 1】

E1：客户 E2：厨房 E3：经理 E4：供应商
P1：在线订餐 P2：厨房备餐 P3：生成报表 P4：食材采购

【问题 2】

- (1) 增加 E1 到 P1 数据流“餐品订单”；
- (2) 增加 P1 到 P2 数据流“餐品订单”；
- (3) 增加 D1 到的数据流“订单汇总”；
- (4) 增加 P3 到 E3 数据流“统计报表”。

【问题 3】

- (1) 数据流图中的处理过程可并行；系统流程图在某个时间点只能处于一个处理过程。
- (2) 数据流图展现系统的数据流；系统流程图展现系统的控制流。
- (3) 数据流图展现全局的处理过程，过程之间遵循不同的计时标准；系统流程图中处理过程遵循一致的计时标准。

2019 年下半年试题 3

答案：

【问题 1】

信息物理系统（CyberPhysical Systems, CPS）作为计算进程和物理进程的统一体，是集计算、通信与控制于一体的下一代智能系统。信息物理系统通过人机交互接口实现程的交互，使用网络化空间，以远程的、可靠的、实时的、安全的、协作的方式操控一个物理实体。

感知层：主要由传感器、控制器和采集器等设备组成，它属于信息物理系统中的末端设备。

网络层：主要是连接信息世界和物理世界的桥梁，实现的是数据传输，为系统提供实时的网络服务，保证网络分组传输的实时可靠。

控制层：主要是根据认知结果及物理设备传回来的数据进行相应的分析，将相应的结果返回给客户端。

【问题 2】

感知层：2、4、7、10

网络层：8

控制层：1、3、5、6

不属于 CPS 结构中的功能：9

【问题 3】

- (1) 感知层安全威胁：感知数据破坏、信息窃听、节点捕获。
- (2) 网络层安全威胁：拒绝服务攻击、选择性转发、方向误导攻击。
- (3) 控制层安全威胁：用户隐私泄露、恶意代码、非授权访问。

2019 年下半年试题 4

答案：

【问题 1】

存在双写不一致问题，在写数据时，可能存在缓存写成功，数据库写失败，或者反之，从而造成数据不一致。当多个请求发生时，也可能产生读写冲突的并发问题。

- (a) 从数据库中读取数据或读数据库
- (b) 更新缓存中 key 值或更新缓存
- (c) 数据库
- (d) 删除缓存 key 或使缓存 key 失效或更新缓存 (key 值)

【问题 2】

存在问题：不在系统中的 key 值是无限的，如果均设置 key 值为空，会造成内存资源的极大浪费，引起性能急剧下降。

解决思路：查询缓存之前，对 key 值进行过滤，只允许系统中存在的 key 进行后续操作（例如采用 key 的 bitmap 进行过滤）。

【问题 3】

思路 1：缓存失效后，通过加排它锁或者队列方式控制数据库写缓存的线程数量，使得缓存更新串行化；

思路 2：给不同 key 设置随机或不同的失效时间，使失效时间的分布尽量均匀；

思路 3：设置两级或多级缓存，避免访问数据库服务器。

2019 年下半年试题 5

答案：

【问题 1】

性能：(1)、(2)、(6)

安全性：(5)

可用性：(3)、(7)

易用性：(4)、(8)

【问题 2】

- (1) (d)
- (2) (e)
- (3) (i)
- (4) (h)

(5) (g)

(6) (f)

(7) (a)

【问题 3】

SQL 注入攻击，就是通过把 SQL 命令插入到 Web 表单提交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的 SQL 命令。

可以通过以下方式抵御 SQL 注入攻击：

- 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；
- 存储过程来执行所有的查询；
- 使用专业的漏洞扫描工具。

2018 年下半年试题 1

答案：

【问题 1】

系统性能需求 (Performance Requirements)：指响应时间、吞吐量、准确性、有效性、资源利用率等与系统完成任务效率相关的指标。可靠性、可用性等指标归为此类。

安全性需求 (Security Requirements)：系统向合法用户提供服务并阻止非授权用户使用服务方面的系统需求。

操作性需求 (Operational Requirements)：与用户操作使用系统相关的一些需求。

文化需求 (Cultural Requirements)：带有文化背景因素的系统需求。

【问题 2】

(1) (a) (b)

(2) (c) (d) (f)

(3) (e) (g)

(4) (h)

【问题 3】

(a) 瘦的把业务逻辑从客户端放到了服务上。

(b) 胖和瘦无明显差异

(c) 胖客户端，在客户端的运算能力强一些。瘦客户端可以在服务端用集群做支持

(d) 瘦客户端将业务逻辑迁移到应用服务器上，所以故障只要修复服务器上的内容，而胖客户端要更新所有客户端，工作量大，所以此情况下瘦客户端有优势。

(e) 胖客户端的后端是数据库，没有业务逻辑，此时要做加密传输没有基础，但瘦客户端可以做到。

(f) 胖客户端做到 2G 数据缓存很容易，而瘦客户端不现实。

(g) 瘦客户端与胖客户端均可做到。

(h) 瘦客户端与胖客户端均可做到。

解析：

系统性能需求 (Performance Requirements)：系统扩展性 (Scalability Requirements)、系统可靠性 (Reliability Requirements)、系统可用性 (Availability Requirements)

安全性需求 (Security Requirements)：用户安全需求 (User Security Requirements)、数据安全需求 (Data Security Requirements)

系统运营需求 (Operational Requirements)：硬件平台需求 (Hard Requirements)、软件平台需求 (Software Requirements)、网络需求 (Network Requirements)、产品支持需求 (Product Support Requirements)

国际化需求 (Globalization Requirements)：中国文化及政策需求 (Cultural and Political Requirements)、本地化需求 (Localization Requirements)

2018 年下半年试题 2

答案：

【问题 1】

E1：房主

E2：租赁者

P1：登记房主信息

P2：登记房屋信息

P3：登记租赁者信息

P4：查询租赁房屋信息

P5：安排看房

P6：变更房屋状态

D1：房主信息文件

D2：租赁者信息文件

D3 房屋信息文件

D4：看房记录文件

【问题 2】

(1) 房主

(2) 房屋

(3) 房主

(4) 租赁者

(5) 看房记录

【问题 3】

(1) 实体用于数据建模，而类用于面向对象建模。实体只有属性，而类有属性和操作。

(2) Essential Use Cases 可翻译为抽象用例，而 Real Use Cases 可翻译为基础用例。他们是区别在于：基础用例是实实在在与用户需求有对应关系用例，是从用户需求获取的渠道得到的，而抽象用例是从基础用例中抽取的用例的公共部分，是为了避免重复工作，优化结构而提出的用例。

2018 年下半年试题 3

答案：

【问题 1】

- | | | | |
|----------|----------|----------|----------|
| (1) (d) | (2) (c) | (3) (e) | (4) (f) |
| (5) (g) | (6) (h) | (7) (i) | (8) (j) |
| (9) (k) | (10) (l) | (11) (n) | (12) (m) |
| (13) (o) | (14) (p) | | |

【问题 2】

BMTS 的消息通信网络主要特征为: 能适配不同的传输介质, 以及适配不同的协议, 屏蔽不同协议之间的差异, 简化通信过程降低系统复杂度。

事件触发消息: 以事件作为触发方式, 事件发生便触发相应消息。

速率约束消息: 传输速率固定的消息。

时间触发消息: 以时间作为触发方式, 到达时间点便触发相应消息。

具有时间触发消息能力的网络总线:

航空电子全双工交换式以太网 (Avionics Full Duplex Switched Ethernet, AFDX)

时间触发以太网 (Time-Triggered Ethernet, TTE)

2018 年下半年试题 4

答案:

【问题 1】

- (1) key/value, list, set, hash, sorted
- (2) 不支持
- (3) 不支持
- (4) 不支持
- (5) 有
- (6) 不支持

【问题 2】

(1) Memcache 没有持久化功能, 所以掉电数据会全部丢失, 而且无法直接恢复, 这存在可靠性问题。

(2) Memcache 不支持事务, 所以操作过程中可能产生数据的不一致性。

同步方案: 读取数据时, 先读取 Redis 中的数据, 如果 Redis 没有, 则从原数据库中读取, 并同步更新 Redis 中的数据, 写回时, 写入到原数据库中, 并同步更新至 Redis 中。

【问题 3】

Redis 分布式存储的 2 种常见方案:

- 1. 主从
- 2. Cluster

Redis 集群切片的常见方式:

- 1. 客户端切片, 即在客户端就通过 key 的 hash 值对应到不同的服务器。

2. 对数据根据 key 散列到不同的 slot 上，不同的 slot 对应到不同服务器。

2018 年下半年试题 5

答案：

【问题 1】

SOA 是一个组件模型，它将应用程序的不同功能单元（称为服务）通过这些服务之间定义良好的接口和契约联系起来。接口是采用中立的方式进行定义的，它应该独立于实现服务的硬件平台、操作系统和编程语言。这使得构建在各种这样的系统中的服务可以一种统一和通用方式进行交互。

ESB 作用于特点：

1、SOA 的一种实现方式，ESB 在面向服务的架构中起到的是总线作用，将各种服务进行连接与整合；

2、描述服务的元数据和服务注册管理；

3、在服务请求者和提供者之间传递数据，以及对这些数据进行转换的能力，并支持由实践中总结出来的一些模式如同步模式、异步模式等；

4、发现、路由、匹配和选择的能力，以支持服务之间的动态交互，解耦服务请求者和提供者。高级一些的能力，包括对安全的支持、服务质量保证、可管理性和负载平衡等。

【问题 2】

(1) (c) 业务层

(2) (i) UDDI

(3) (h) publish

(4) (e) 企业服务总线 ESB

(4) (g) 安全验证和质量管理

(6) (j) 组件层

【问题 3】

1、引入 https 协议或采用加密技术对数据先加密再传输

2、采用信息摘要技术对重要信息进行完整性验证

3、交易类敏感信息采用数字签名机制

解析：

从应用的角度定义，可以认为 SOA 是一种应用框架，它着眼于日常的业务应用，并将它们划分为单独的业务功能和流程，即所谓的“服务”。SOA 使用户可以构建、部署和整合这些服务，且无需依赖应用程序及其运行平台，从而提高业务流程的灵活性。这种业务灵活性可使企业加快发展速度，降低总体拥有成本，改善对及时、准确性信息的访问。SOA 有助于实现更多的资产重用、更轻松的管理和更快的开发与部署。

从软件的基本原理定义，可以认为 SOA 是一个组件模型，它将应用程序的不同功能单元（称为服务）通过这些服务之间定义良好的接口和启用联系起来。接口是采用中立的方式进行定义的，它应该独立于实现服务的硬件平台、操作系统和编程语言。这使得构建在各种这样的系统中的服务可以一种统一和通用的方式进行交互。

2017 年下半年试题 1

试题分析

本题考查的是软件质量属性，要求考生掌握常见的软件质量属性的含义与具体的应用场景。

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。

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。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、完整性、不可否认性及可控性等特性。设计策略：追踪审计、信息隐藏

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。此外风险点、非风险点、敏感点与权衡点要能正确区分。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。敏感点是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。权衡点是影响多个质量属性的特性，是多个质量属性的敏感点。

试题答案

【问题 1】（1）安全性 （2）可修改性 （3）（h） （4）（1） （5）（j） （6）（m）

【问题 2】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。

敏感点是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。

权衡点是影响多个质量属性的特性，是多个质量属性的敏感点。

风险点：（n） 敏感点：（d） 权衡点：（f）

2017 年下半年试题 2

试题分析

MVC 是一种目前广泛流行的软件设计模式。近年来，随着 J2EE（Java 2Enterprise Edition）的成熟，MVC 成为了 J2EE 平台上推荐的一种设计模式。MVC 强制性地把一个应用的输入、处理、输出流程按照视图、控制、模型的方式进行分离，形成了三个核心模块：控制器、模型、视图。

（1）控制器（Controller）：控制器接受用户的输入并调用模型和视图去完成用户的需求。该部分是用户界面与 Model 的接口。一方面它解释来自于视图的输入，将其解释成为系统能够理解的对象，同时它也识别用户动作，并将其解释为对模型特定方法的调用；另一方面，它处理来自于模型的事件和模型逻辑执行的结果，调用适当的视图为用户提供反馈。

（2）模型（Model）：模型是应用程序的主体部分。模型表示业务数据和业务逻辑。一个模型能为多个视图提供数据。由于同一个模型可以被多个视图重用，所以提高了应用的可重用性。

（3）视图（View）：视图是用户看到并与之交互的界面。视图向用户显示相关的数据，并能接收用户的输入数据，但是它并不进行任何实际的业务处理。视图可以向模型查询业务状态，但不能改

变模型。视图还能接受模型发出的数据更新事件，从而对用户界面进行同步更新。

Session Bean 描述了与客户端的一个短暂的会话。当客户端的执行完成后，Session Bean 和它的数据都将消失；

Entity Bean 描述了存储在数据库表中的一行持久稳固的数据，如果客户端终止或者服务结束，底层的服务会负责 entity Bean 数据的存储。

Message-driven bean 结合了 Session Bean 和 Java 信息服务（JMS）信息监听者的功能，它允许一个商业组件异步地接受 JMS 消息。

试题答案

【问题 1】 MVC 架构包含：视图、控制器、模型

视图（View）：视图是用户看到并与之交互的界面。视图向用户显示相关的数据，并能接收用户的输入数据，但是它并不进行任何实际的业务处理。

控制器（Controller）：控制器接受用户的输入并调用模型和视图去完成用户的需求。该部分是用户界面与 Model 的接口。一方面它解释来自于视图的输入，将其解释成为系统能够理解的对象，同时它也识别用户动作，并将其解释为对模型特定方法的调用；另一方面，它处理来自于模型的事件和模型逻辑执行的结果，调用适当的视图为用户提供反馈。

模型（Model）：模型是应用程序的主体部分。模型表示业务数据和业务逻辑。一个模型能为多个视图提供数据。

(1) JSP (2) Servlet (3) Service (4) JavaBean (5) DAO

【问题 2】

EJB 中的 Bean 分三种类型：Session Bean、Entity Beans 和 Message-Driven Bean。

Session Bean 的职责是：维护一个短暂的会话

Entity Beans 的职责是：维护一行持久稳固的数据

Message-Driven Bean 的职责是：异步接受消息

【问题 3】 有状态：（a）、（b）、（d） 无状态：（c）、（e）

2017 年下半年试题 3

试题答案

【问题 1】 ROS 与嵌入式实时操作系统的共同点：（1）系统微型化

（2）系统专用性强（3）软硬件依赖性强（4）系统资源受限

ROS 与嵌入式实时操作系统的差异： 实时性：ROS 弱于嵌入式实时操作系统

通信方式：ROS 的通信方式较为丰富，嵌入式实时操作系统通信方式单一。

【问题 2】（1）（2）（c）（k） （3）（4）（f）（i） （5）（j）

【问题 3】（1）Registration （2）Data （3）Publish

（4）Subscribe （5）Subscribe

2017 年下半年试题 4

试题分析

ORM，即 Object-Relationl Mapping，它在关系型数据库和对象之间作一个映射，这样，我们在具体的操作数据库的时候，就不需要再去和复杂的 SQL 语句打交道，只要像平时操作对象一样操作即可。

当你开发一个应用程序的时候（不使用 OR Mapping），可能会涉及许多数据访问层的代码，用来从数据库保存、删除、读取对象信息等等，然而这些代码写起来总是重复的。一个更好的办法就是引入 OR Mapping。实质上，一个 OR Mapping 会为你生成 DAL。与其自己写 DAL 代码，不如用 OR Mapping，你只需要关心对象就好。使用 ORM 可以大大降低学习和开发成本。而在实际的开发中，真正对客户有价值的是其独特的业务功能，而不应该把大量时间花费在编写数据访问、CRUD 方法、后期的 Bug 查找和维护上。在使用 ORM 之后，ORM 框架已经把数据库转变成了我们熟悉的对象，我们将只需要了解面向对象开发就可以实现数据库应用程序的开发，不需要浪费时间在 SQL 上。同时也可减少代码量，减少数据层出错机会。通过 Cache 的实现，能够对性能进行调优，实现了 ORM 区隔了实际数据存储和业务层之间的关系，能够对每一层进行单独跟踪，增加了性能优化的可能。

试题答案

【问题 1】 数据库程序在线访问方式优点：

- 1、性能比直接 SQL 好
- 2、可以处理复杂查询语句

数据库程序在线访问方式缺点：

- 1、要求程序员懂 SQL 语句
- 2、修改与维护相对困难

ORM 优点：

- 1、使用 ORM 可以大大降低学习和开发成本。
- 2、程序员不用再写 SQL 来进行数据库操作。
- 3、减少程序的代码量。
- 4、降低由于 SQL 代码质量差而带来的影响。

ORM 缺点

- 1、不太容易处理复杂查询语句。

2、性能较直接用 SQL 差。 本题中的场景之所以选择 ORM，主要考虑的是程序缺数据库开发经验，这样 SQL 语句质量有很大风险。同时学习成本很高。此外应用简单，不也担心 ORM 对性能的影响。

【问题 2】 增加数据访问层的原因：

- (1) 由于涉及到多种异构数据库平台，数据访问复杂性增加，不宜与业务逻辑混合在一起
- (2) 数据管理变复杂之后，需要使用的代码量增加，分单独层次有利于让逻辑更清晰。

(3) 业务逻辑应以相同的方式应对异构的数据库，此时需要单独的数据访问层屏蔽差异性。

(1) 执行业务逻辑 (2) 工厂层 (3) 数据访问接口层

【问题 3】 工厂模式分抽象工厂与工厂方法，题目中的场景适合采用抽象工厂设计模式。

抽象工厂设计模式提供一个接口，可以创建一系列相关或相互依赖的对象，而无需指定它们具体的类。其优点是可以非常方便的创建一系列的对象，其使用场景也是创建系列对象的情况。在本题中，可以针对 Oracle、MySQL、SQLServer 分别建立抽象工厂，若指定当前工厂为 Oracle 工厂，则创建出来的数据库连接，数据集等一系列的对象都是符合 Oracle 操作要求的。这样便于数据库之间的切换。

2017 年下半年试题 5

试题分析

响应式 web 设计是指我们设计与开发的页面可以根据用户的行为（比如改变浏览器的窗口大小）和不同的设备环境（比如系统平台、屏幕分辨率以及横竖屏等）做出相应的响应来调整页面的布局，以提供用户可感知的、流畅的阅读和操作体验。响应式设计一般遵循“设计先行，内容优先，移动优先”的原则。就是说，如果一个页面想要响应 PC 端和移动终端包括用户的一些行为等，那么设计师至少需要设计两套以上页面的设计图（PC 端一套，移动终端一套），交互设计师也需先根据终端进行交互设计，把需求中最重要的内容展示在移动小屏幕上，然后前端工程师据此设计开发出响应式框架。

试题答案

【问题 1】 响应式 web 设计是指我们设计与开发的页面可以根据用户的行为和不同的设备环境做出相应的响应来调整页面的布局，以提供用户可感知的、流畅的阅读和操作体验。

实现方式：(1) 流式布局 (2) 弹性布局加媒体查询

【问题 2】

(1) (d) (2) (c) (3) (f) (4) (a)

(5) (6) (e) (h) (7) (8) (g) (i)

【问题 3】 1、提升性能 交易平台要求高并发，主从复制方式一主多从，不同的用户请求可以从不同的从数据库读取数据，提高并发度。

2、可扩展性更优 如果采用单台数据库服务器，则访问量持续增加时，数据库瓶颈暴露，且无法迅速解决问题。而主从结构可以快速增加从服务器数量，以满足需求。

3、提升可用性 一主多从，一台从服务器出现故障不影响整个系统正常工作。

4、相当于负载均衡 一主多从分担任务，相当于负载均衡

5、提升数据安全性系统中的数据冗余存放多份，不会因为某台机器硬件故障而导致数据丢失。

2016 年下半年试题 1

试题分析

本题考查的是架构设计过程中涉及到的一些质量属性，以及架构风格的对比。常用的质量属性包括：

1、性能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。

2、可靠性是软件系统在设计或系统错误面前，在意外或错误使用的情况下维持软件系统的功能特性的基本能力。

3、可用性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。

4、安全性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、完整性、不可否认性及可控性等特性。

5、可修改性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。

6、易用性软件开发工具应有十分友好的用户界面，用户乐于使用；工具应能剪裁和定制，以适应特定用户的需要；工具应能提示用户的交互操作，提供简单有效的执行方式；工具还应能检查用户的操作错误，尽可能自动改正错误。

题目中还涉及到管道过滤器风格与仓库风格的对比。这两种风格的基本概念为：在管道/过滤器风格的软件体系结构中，每个构件都有一组输入和输出，数据输入构件，经过内部处理，然后产生数据输出。因此，这里的构件被称为过滤器，这种风格的连接件就像是数据流传输的管道，将一个过滤器的输出传到另一过滤器的输入。

在仓库(repository)风格中，有两种不同的构件：中央数据结构说明当前状态，独立构件在中央数据 存储上执行。一方面，若构件控制共享数据，则仓库是一传统型数据库；另一方面，若中央数据结构的当前状态触发进程执行的选择，则仓库是一黑板系统。

题目中对比的方面分别是：交互方式：管理过滤器很明显是顺序结构或循环结构，数据在管道中进行传递。而仓库结构是数据在中心位置，所有的处理均是中心结点与周边结点之间的交互，从形态来看，是星型的。

数据结构：从数据结构来看，仓库风格会使用一个文件将数据保存起来，所有的操作围绕这个文件 进行。而管道过滤器则是在过滤器之间传递数据流。

控制结构：从控制结构来说仓库风格是业务功能驱动，而管道过滤器是由数据流驱动的。

扩展方法：从扩展方法来讲，管道过滤器是通过过滤器提供标准接口与其它过滤器对接，而数据仓库风格，要共享数据，扩展功能，只要功能的操作与数据模型本身是匹配的就行了，就像我们要共享一个数据库做系统集成，此时共享同一数据库的多个应用系统所用的数据模型一定会是一致的，否则无法去共享。

本题最后一问是分析开发环境架构图。要填充这个图，首先得想明白的是，这个环境中，仓库的中心是数据，数据对应的是语法结构树，因为一系列的处理都是以此为依据来进行的。接下来找出与编译器同等地位的其它处理工具即可。

试题答案

【问题 1】 (1) f (2) 性能 (3) g (4) h (5) b

【问题 2】 (1) 星型 (2) 数据流 (3) 数据流驱动 (4) 模型适配

【问题 3】 (1) 语法结构树 (2) 编辑器 (3) 适配器 (4) 应用模拟器工具

2016 年下半年试题 2

试题分析

参与者是指系统以外的，需要使用系统或与系统交互的事物，包括：人或组织、设备、外部系统等。在本题中，较为容易识别的参与者包括：学生、教师、管理员，比较隐晦的参与者包括：时间、打印机。

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

试题答案

【问题 1】学生、教师、管理员、时间、打印机。

【问题 2】用例之间的关系包括：包含、扩展、泛化。

“登录系统”用例与“注册课程”用例之间的关系为：包含关系。

“参加考试”用例与“参加补考”用例之间的关系为：扩展关系。

【问题 3】类之间的关系包括：关联、聚合、组合、依赖、泛化、实现（可写可不写，因为实现是接口与类之间的关系，而接口是一种特殊的类）

类 University 与类 Student 之间的关系是：聚合关系。

类 University 与类 Department 之间的关系是：组合关系。

类 Student 与类 Course 之间的关系是：关联关系。

2016 年下半年试题 3

试题分析

实时系统的特性包括：

一、时间约束性

实时系统的任务具有一定的时间约束（截止时间）。根据截止时间，实时系统的实时性分为“硬实时”和“软实时”。硬实时是指应用的时间需求能够得到完全满足，否则就造成重大安全事故，甚至造成重大的生命财产损失和生态破坏，如在航空航天、军事、核工业等一些关键领域中的应用。软实时是指某些应用虽然提出时间需求，但实时任务偶尔违反这种需求对系统运行及环境不会造成严重影响，如监控系统等和信息采集系统等。

二、可预测性

可预测性是指系统能够对实时任务的执行时间进行判断，确定是否能够满足任务的时限要求。由于实时系统对时间约束要求的严格性，使可预测性称为实时系统的一项重要性能要求。除了要求硬件延迟的可预测性以外，还要求软件系统的可预测性，包括应用程序的响应时间是可预测的，即在有限的时间内完成必须的工作；以及操作系统的可预测性，即实时原语、调度函数等运行开销应是有界的，以保证应用程序执行时间的有界性。

三、可靠性

大多数实时系统要求有较高的可靠性。在一些重要的实时应用中，任何不可靠因素和计算机的一个微小故障，或某些特定强实时任务（又叫关键任务）超过时限，都可能引起难以预测的严重后果。为此，系统需要采用静态分析和保留资源的方法及冗余配置，使系统在最坏情况下都能正常工作或避免损失。可靠性已成为衡量实时系统性能不可缺少的重要指标。

四、与外部环境的交互作用性

实时系统通常运行在一定的环境下，外部环境是实时系统不可缺少的一个组成部分。计算机子系统一般是控制系统，它必须在规定的时间内对外部请求做出反应。外部物理环境往往是被控子系统，两者相互作用构成完整的实时系统。大多数控制子系统必须连续运转以保证子系统的正常工作或准备对任何异常行为采取行动。

五、多任务类型

在实时系统中，不但包括周期任务、偶发任务、非周期任务，还包括非实时任务。实时任务要求要满足时限，而非实时任务要求要使其响应时间尽可能的短。多种类型任务的混合，使系统的可调度性分析更加困难。

六、约束的复杂性

任务的约束包括时间约束、资源约束、执行顺序约束和性能约束。时间约束是任何实时系统都固有的约束。资源约束是指多个实时任务共享有限的资源时，必须按照一定的资源访问控制协议进行同步，以避免死锁和高优先级任务被低优先级任务堵塞的时间（即优先级倒置时间）不可预测。执行顺序约束是指各任务的启动和执行必须满足一定的时间和顺序约束。例如，在分布式端到端（end-to-end）实时系统很重，同一任务的各子任务之间存在前驱/后驱约束关系，需要执行同步协议来管理子任务的启动和控制子任务的执行，使它们满足时间约束和系统可调度要求。性能约束是指必须满足如可靠性、可用性、可预测性、服务质量（Quality of Service, QoS）等性能指标。

七、具有短暂超载的特点

在实时系统中，即使一个功能设计合理、资源充足的系统也可能由于一下原因超载：

- 1) 系统元件出现老化，外围设备错误或系统发生故障。随着系统运行时间的增长，系统元件出现老化，系统部件可能发生故障，导致系统可用资源降低，不能满足实时任务的时间约束要求。
- 2) 环境的动态变化。由于不能对未来的环境、系统状态进行正确有效地预测，因此不能从整体角度上对任务进行调度，可能导致系统超载。
- 3) 应用规模的扩大。原先满足实时任务时限要求的系统，随着应用规模的增大，可能出现不能满足任务时限要求的情况，而重新设计、重建系统在时间和经济上又不允许。

软件失效的机理可描述为：软件错误→软件缺陷→软件故障→软件失效

1、软件错误：在可以预见的时期内，软件仍将由人来开发。在整个软件生存期的各个阶段，都贯穿着人的直接或间接的干预。然而，人难免犯错误，这必然给软件留下不良的痕迹。软件错误是指在软件生存期内的不希望或不可接受的人为错误，其结果是导致软件缺陷的产生。可见，软件错误是一种人为过程，相对于软件本身，是一种外部行为。

2、软件缺陷：软件缺陷是存在于软件（文档、数据、程序）之中的那些不希望或不可接受的偏差，如少一个逗号、多一语句等。其结果是软件运行于某一特定条件时出现软件故障，这时称软件缺陷被激活。

3、软件故障：软件故障是指软件运行过程中出现的一种不希望或不可接受的内部状态。譬如，软件处于执行一个多余循环过程时，我们说软件出现故障。此时若无时当的措施（容错）加以及时处理，便产生软件失效。显然，软件故障是一种动态行为。

4、软件失效：软件失效是指软件运行时产生的一种不希望或不可接受的外部行为结果。

试题答案

【问题 1】 实时系统是指向系统发出一指令后，在一个极短的时间内，系统回复结果。实时系统的特性：

- (1) 时间约束性（及时性）
- (2) 可预测性
- (3) 高可靠性
- (4) 与外部环境的交互作用性
- (5) 多任务类型
- (6) 约束的复杂性
- (7) 具有短暂超载的特点

【问题 2】

- (1) 强
- (2) (3) 时间响应、时间明确
- (4) (5) (6) 时限/反应时间、输入/输出激励、周期/零星/非周期
- (7) (8) 时间触发、事件触发

【问题 3】 软件错误：软件错误是指在软件生存期内的不希望或不可接受的人为错误，其结果是导致软件缺陷的产生。

软件缺陷：软件缺陷是存在于软件（文档、数据、程序）之中的那些不希望或不可接受的偏差。

软件故障：软件故障是指软件运行过程中出现的一种不希望或不可接受的内部状态。

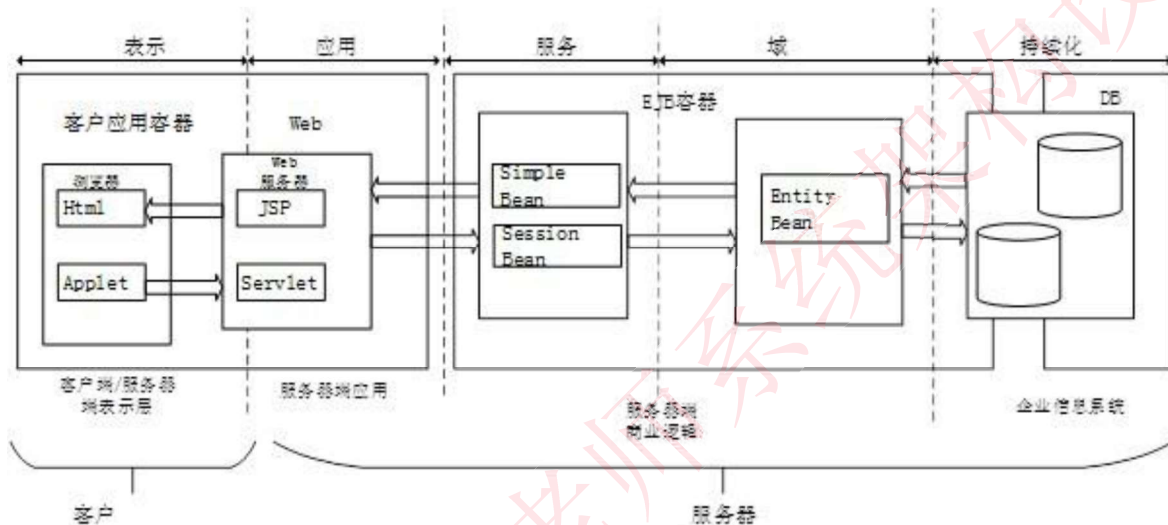
软件失效：软件失效是指软件运行时产生的一种不希望或不可接受的外部行为结果。

- (1) 存在
- (2) 引起
- (3) 用户经历
- (4) 在开发过程中
- (5) 在产品中
- (6) 在运行时

2016 年下半年试题 4

试题分析

应用服务器是指通过各种协议把商业逻辑暴露给客户端的程序。它提供了访问商业逻辑的途径以供客户端应用程序使用。应用服务器使用此商业逻辑就像调用对象的一个方法一样。简单的说能实现动态网页技术的服务器叫做 web 应用服务器。



试题答案

【问题 1】 1、PHP 只能实现简单的分布式两层或三层的架构，而 JAVA 在这方面就比较强大，可以实现多层的网络架构。数据库层(持久化层)、应用(业务)逻辑层、表示逻辑层彼此分开，而且现在不同的层都已经有一些成熟的开发框架的支持。

2、PHP 是面向过程的语言，Java 是面向对象的，面向过程语言开发的程序只要业务流程发生变化，修改工作量很大，所以可修改性差，同时可复用性也差。

3、PHP 语言在可靠性方面比 J2EE 平台差，J2EE 平台有大量增强可靠性的成熟解决方案，而 PHP 只是一种简单的脚本语言，在可靠性方面缺乏成熟解决方案。

4、PHP 对于不同的数据库采用不同的数据库访问接口，而 Java 通过 JDBC 来访问数据库，通过不同的数据库厂商提供的数据库驱动方便地访问数据库，访问数据库的接口比较统一。所以原架构在数据库连接方面修改起来工作量也是很大的。

5、PHP 适合于小型项目，所以本项目中以前采用 PHP 是合适的，但目前大量功能需要增加，PHP 在稳定性方面也达不到要求。

6、PHP 比 Java 的可维护性差。

7、PHP 比 Java 的扩展性差。

8、PHP 比 Java 的安全性差。

【问题 2】 应用服务器是指通过各种协议把商业逻辑暴露给客户端的程序。

1、若系统负荷很大，可以布署多台应用服务，多台应用服务器分担任务，以达到性能要求。

2、应用服务器可以通过灵活的增加服务器完成扩展，所以可扩展性很好。

3、应用服务器可长时间稳定运行。因为当一台应用服务器出现故障时，可以将当前运行的事务转移至正常应用服务器上完成执行，不影响业务正常执行，从而保障高可靠性与稳定性。

【问题 3】(1)Applet (2)Servlet (3)EJB 容器 (4)SessionBean (5)EntityBean

2016 年下半年试题 5

试题分析

本题看似是考敏捷+MVC，但事实上是考状态图+MVC。很多考生在回答问题 1 时，被 Scrum 的详细流程给吓倒了，认为自己没有用 Scrum 做过项目，对这种细节并不清楚，肯定做不对。事实并非如此。本题只是一个状态图的题而已，知道状态图的基本表达方式就能解决问题。因为题目已详细介绍 Scrum 的处理流程，把这个文字版的转化为图片版的即可。

问题 2 与问题 3 是关于 MVC 的概念考查，首先我们得了解 MVC 各个层级的主要功能用途，这样才能把众多的选项进行归类，归类之后再对应到具体位置就容易了，具体答案参看参考答案。

试题答案

【问题 1】(1)Product Backlog (2)Sprint 计划会议 (3)每日站立会议 (4)还有未完成的用户故事 (5)交付产品增量

【问题 2】(1)c、e、n (2)a、f、j (3)g

【问题 3】(1) (2)d f (3)h (4)g (5)a (6)k (7)h (8)b (9)c

2015 年下半年试题 1

试题答案

【问题 1】(1) 安全性 (2) 可修改性 (3) (h) (4) (1) (5) (j) (6) (n)

【问题 2】架构风险：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。敏感点：是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：是影响多个质量属性的特性，是多个质量属性的敏感点。架构风险：(m)；敏感点：(e)；权衡点：(i)。

2015 年下半年试题 2

试题分析

【问题 1】本问题考查系统建模中常用的活动图与状态图的概念知识。

状态图用来描述一个特定对象的所有可能状态以及其引起状态转移的事件，一个状态图包括一系列的状态以及状态之间的转移，状态图通常用于表示单个对象在其生命周期中的行为。活动图用来描述操作的行为，也用于描述用例和对象内部的工作过程。状态图和活动图都是用来描述系统的动态行为特征的，主要用于描述事物的状态变化和处理过程。

但是两者还是有本质区别：状态图和活动图用于不同的目的，状态图着重描述一系列的状态及状态间的转移，状态间的变迁需要外部事件的触发。活动图用于捕获动作（将要执行的工作或活动）及动作的结果，活动图中一个活动结束将立即进入下一个活动，是内部处理驱动的流程。

【问题 2】本问题考查系统建模中状态图的设计与应用。考生应该在熟记基本概念的基础上结合实际问题灵活掌握并应用这些概念。在解答本题时，首先需要对题目中描述的基本功能需求 (a) ~ (j) 进行分析与梳理，确定系统控制中的所有状态以及状态间的转换条件，再结合问题 2 中已经给出的状

态，完成其余状态及条件的设计。

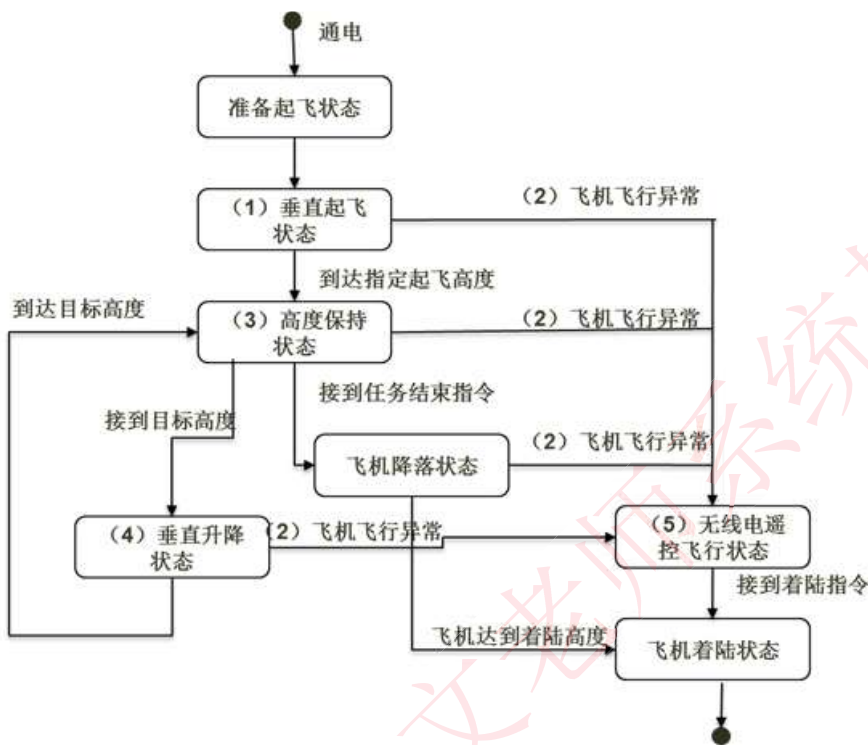
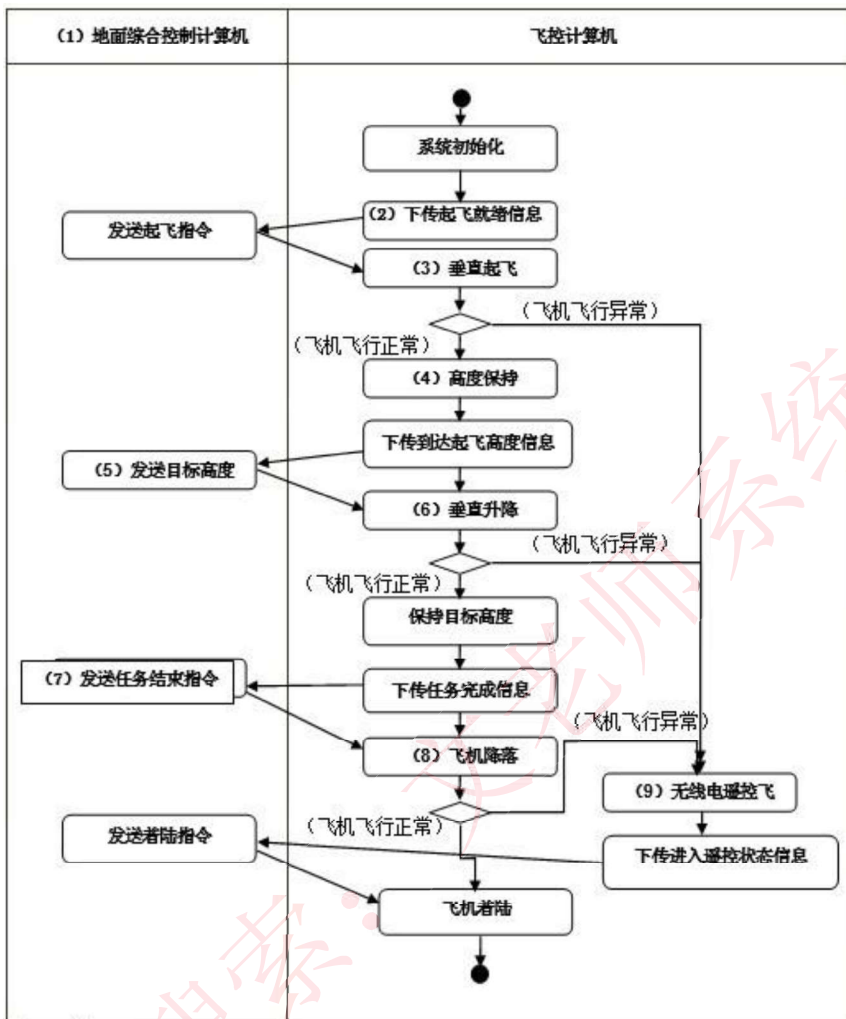


图2-1 无人直升机控制系统纵向状态图

【问题 3】

本问题考查系统建模中活动图的设计与应用。考生应该掌握泳道活动图的概念并且学会应用。泳道活动图，是将一个活动图中放入活动状态进行分组，每一组表示一个特定的类或者对象，它们负责完成组内的活动。每个活动都明确属于一个泳道，不可以跨越泳道，而转移则可以跨越泳道。在解答本题时，首先需要对题目中描述的基本功能需求进行分析与梳理，确定题目中存在哪些硬件设备与飞控计算机进行交互，以及设备间的交互关系，再结合问题 3 中已经给出的活动，完成其余活动及设备的设计。



试题答案

【问题 1】 状态图主要用于描述一个对象在其生存期间的动态行为，表现一个对象所经历的状态序列，引起状态转移的事件（event），以及因状态转移而伴随的动作（action）。

活动图可以用于描述系统的工作流程和并发行为。活动图其实可看作状态图的特殊形式，活动图中一个活动结束后将立即进入下一个活动（在状态图中状态的转移可能需要事件的触发）。

两者最大的区别是：状态图侧重于描述行为的结果，而活动图侧重描述行为的动作。其次活动图可描述并发行为，而状态图不能。

【问题 2】

- (1) 垂直起飞状态
- (2) 飞机飞行异常
- (3) 高度保持状态
- (4) 垂直升降状态
- (5) 无线遥控飞行状态

【问题 3】

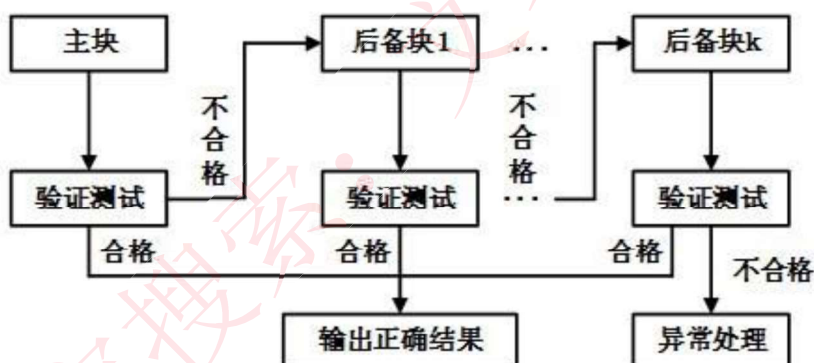
- (1) 地面综合控制计算机
- (2) 下传完成系统初始化信息
- (3) 飞机垂直起飞
- (4) 飞行高度保持
- (5) 发送目标高度
- (6) 飞机垂直升降
- (7) 发送任务执行结束指令
- (8) 飞机降落
- (9) 无线电遥控飞行

2015 年下半年试题 3

试题分析



	恢复块方法	N 版本程序设计
硬件运行环境	单机	多机
错误检测方法	验证测试程序	表决
恢复策略	后向恢复	前向恢复
实时性	差	好



试题答案

【问题 1】 可靠性 (Reliability) 是指产品在规定的条件下和规定的时间内完成规定功能的能力。子特性：成熟性，容错性，易恢复性，可靠性的依从性。

提高可靠性的技术： (1) N 版本程序设计 (2) 恢复块方法
(3) 防卫式程序设计 (4) 双机热备或集群系统 (5) 冗余设计

【问题 2】

- (1) 不考虑软件演化的情况下，失效率在统计上是非增的
- (2) 如果不使用该软件，永远不会发生失效
- (3) 软件维护会创建新的软件代码
- (4) 软件失效之前很少会有报警

【问题 3】

- (1) 主块
- (2) 验证测试
- (3) 输出正确结果
- (4) 异常处理
- (5) 表决
- (6) 后向恢复
- (7) 差
- (8) 好

2015 年下半年试题 4

试题分析

本题考查文件系统、关系型数据库、内存型数据库的主要特点。

【问题 1】 影响数据存储方式的主要考虑因素包括：设计难度、数据冗余程度、数据架构、应用扩展性；文件系统具有以下特点：

- 针对特定应用系统设计，难度较小；
- 数据冗余较大，可能在多个文件中复制相同的数据属性；
- 以应用系统为中心组织、管理数据；
- 符合特定应用系统要求的文件数据很难在不同的应用系统之间共享。

关系型数据库具有以下特点。

- 数据结构需要符合关系模式，设计难度较大；
- 遵守数据库范式，数据冗余较少；

- 以数据库为中心组织、管理数据;
- 数据独立于应用系统, 很容易在不同的应用系统之间共享数据。

【问题 2】 内存数据库型数据库是将数据放在内存中直接操作的数据库, 使用内存型数据库将极大地提高应用 的性能, 同时通过数据缓存、快速算法、并行操作等的改进, 使内存型数据库相对于传统的关系型数据库数据处理性能提高 10 倍以上, 同时内存型数据库的应用受到内存大小, 数据恢复要求的限制。 关系型数据库和内存数据的主要特点如下:

	主要数据模型	读写性能	存储容量	可靠性
内存数据库	Key-Value 模式 (键-值对模式)	内存直接读写, 性能相对较高	基于内存存储, 存储容量受限	恢复机制复杂, 可靠性较低
关系数据库	关系模式	外存读写, 性能 相对较低	基于存盘存储, 存储容量大	内建恢复机制, 可靠性较高

根据贸易综合管理系统的需求描述, 需要频繁向客户发送相关的电子邮件、短信等提醒信息, 并实时更新商品库存信息和商品价格信息, 因此混合存储架构中, 将客户电子邮件、客户联系电话、商品库存信息、商品价格信息等数据存入内存数据库; 客户基本信息, 商品基本信息相对稳定、访问 频率较低, 存入关系型数据库。

【问题 3】 SQL 语句设计时, 影响查询效率的设计原则是:

- 查询时尽量不要返回不需要的行、列;
- 需要进行多表连接查询时, 尽量使用连接查询, 避免使用子查询结构;
- 尽量避免采用 NOT IN、NOT EXIST、LIKE 等使用全表查询的操作;
- 尽量避免使用 DISTINCT 关键字

试题答案

【问题 1】

- (1) 针对特定应用系统, 难度较大
- (2) 可能在多个文件中复制相同的数据属性, 数据冗余较大
- (3) 以应用为中心管理数据
- (4) 数据库独立于应用系统, 数据库系统接口标准化, 易于在不同应用之间共享数据

【问题 2】

- (1)Key-Value 模式 (2) 外存读写, 性能相对较低
- (3) 运行时整个数据库基本全调入内存, 数据库容量受内存容量限制, 容量较小
- (4) 虽然也有恢复机制, 但并不是所有故障都能恢复, 可靠性较低
- (5) 内存数据库 (6) 内存数据库 (7) 关系数据库 (8) 内存数据库 (9) 内存数据库

【问题3】 (1)提升(2)降低(3)降低(4)提升

2015年下半年试题5

试题分析

iBatis 是 apache 的一个开源项目，一个 O/R Mapping 解决方案，iBatis 最大的特点就是小巧，上手很快。如果不需要太多复杂的功能，iBatis 是能够满足你的要求又足够灵活的最简单的解决方案，现在的 iBatis 已经改名为 Mybatis 了。

EJB 有两种主要类型 BMP (Bean managed persistence) 和 CMP (Container managed persistence)，这两种类型各有优缺点。BMP 是在 Bean 中完成对数据库 JDBC 的各种调用，也就是说，在你的实体 bean(entity bean)中，明确写入了 SQL 语句，如“insert ..”或“select ..”，并且使用 Datasource 获得一个数据库资源以及连接 (connection) 从而对数据库直接进行增加删除修改。CMP 是由 EJB 容器自动完成对数据库的操作，你所有做的，就是在实体 bean 重写入 SetXXX 或 getXXX 方法，然后在 ejb-jar.xml 中定义 cmp-field 就可以。

Hibernate 对数据库结构提供了较为完整的封装，Hibernate 的 O/R Mapping 实现了 POJO 和数据库表之间的映射，以及 SQL 的自动生成和执行。程序员往往只需定义好了 POJO 到数据库表的映射关系，即可通过 Hibernate 提供的方法完成持久层操作。程序员甚至不需要对 SQL 的熟练掌握，Hibernate/OJB 会根据制定的存储逻辑，自动生成对应的 SQL 并调用 JDBC 接口加以执行。

iBATIS 的着力点，则在于 POJO 与 SQL 之间的映射关系。然后通过映射配置文件，将 SQL 所需的参数，以及返回的结果字段映射到指定 POJO。相对 Hibernate “O/R”而言，iBATIS 是一种“Sql Mapping”的 ORM 实现。

Hibernate 的调优方案制定合理的缓存策略；尽量使用延迟加载特性；采用合理的 Session 管理机制；使用批量抓取，设定合理的批处理参数 (batch_size)；进行合理的 O/R 映射设计

Mybatis 调优方案 MyBatis 在 Session 方面和 Hibernate 的 Session 生命周期是一致的，同样需要合理的 Session 管理机制。MyBatis 同样具有二级缓存机制。MyBatis 可以进行详细的 SQL 优化设计。

SQL 优化方面 Hibernate 的查询会将表中的所有字段查询出来，这一点会有性能消耗。Hibernate 也可以自己写 SQL 来指定需要查询的字段，但这样就破坏了 Hibernate 开发的简洁性。而 Mybatis 的 SQL 是手动编写的，所以可以按需求指定查询的字段。

Hibernate HQL 语句的调优需要将 SQL 打印出来，而 Hibernate 的 SQL 被很多人嫌弃因为太丑了。MyBatis 的 SQL 是自己手动写的所以调整方便。但 Hibernate 具有自己的日志统计。Mybatis 本身不带日志统计，使用 Log4j 进行日志记录。

扩展性方面 Hibernate 与具体数据库的关联只需在 XML 文件中配置即可，所有的 HQL 语句与具体使用的数据库无关，移植性很好。MyBatis 项目中所有的 SQL 语句都是依赖所用的数据库的，所以不同数据库类型的支持不好。

对象管理

Hibernate 是完整的对象/关系映射解决方案，它提供了对象状态管理 (state management) 的

功能,使开发者不再需要理会底层数据库系统的细节。也就是说,相对于常见的 JDBC/SQL 持久层方案中需要管理 SQL 语句, Hibernate 采用了更自然的面向对象的视角来持久化 Java 应用中的数据。

换句话说,使用 Hibernate 的开发者应该总是关注对象的状态 (state),不必考虑 SQL 语句的执行。这部分细节已经由 Hibernate 掌管妥当,只有开发者在进行系统性能调优的时候才需要进行了了解。而 MyBatis 在这一块没有文档说明,用户需要对对象自己进行详细的管理。

抓取策略 Hibernate 对实体关联对象的抓取有着良好的机制。对于每一个关联关系都可以详细地设置是否延迟加载,并且提供关联抓取、查询抓取、子查询抓取、批量抓取四种模式。它是详细配置和处理的。而 Mybatis 的延迟加载是全局配置的。

试题答案

【问题 1】 数据持久层是一组软件服务,将应用程序与该程序所使用的数据源分离,为整个项目提供一个统一、安全、并发的数据持久机制。

好处: 1、程序代码重用性强,即使更换数据库,只需要更改配置文件,不必重写程序代码。

2、业务逻辑代码可读性强,在代码中不会有大量的 SQL 语言,提高程序的可读性。

3、持久化技术可以自动优化,以减少对数据库的访问量,提高程序运行效率。

4、简化开发工作,让开发人员更关注于业务逻辑的开发。

5、通过对象/关系映射向业务逻辑提供面向对象的数据访问。

【问题 2】 1、项目组应选 Hibernate 框架 2、选择该技术的原因是:

(1)从移植的角度来看使用 Hibernate 更容易移植到其它数据库平台。Hibernate 与具体数据库的关联只需在 XML 文件中配置即可,所有的 HQL 语句与具体使用的数据库无关,移植性很好。MyBatis 项目中所有的 SQL 语句都是依赖所用的数据库的,所以不同数据库类型的支持不好。

(2)使用 Hibernate 能降低或者消除 SQL 语句开发工作量, Hibernate 提供了方法完成持久层操作,程序员不需要对 SQL 的熟练掌握,便可完成任务。

(3)Hibernate 提供了对象状态管理的功能,使开发者不再需要理会底层数据库系统的细节,而 MyBatis 在这一块没有文档说明,用户需要对对象自己进行详细的管理。

【问题 3】 (1) (c) (2) (b) (3) (d) (4) (a)

2014 年下半年试题 1

试题分析

本题主要考查 MVC 架构风格的定义以及扩展接口模式结构的分析与理解。

【问题 1】

MVC 架构风格最初是 Smalltalk-80 中用来构建用户界面时采用的架构设计风格。其中 M 代表模式 (Model), V 代表视图 (View), C 代表控制器 (Controller)。在该风格中,模型表示待展示的对象,视图表示模型的展示,控制器负责把用户的动作转成针对模型的操作。模型通过更新视图的数据来反映自

身的变化。在本系统中，模型（M）代表监控组件、视图（V）代表控制终端、控制器（C）代表管理模块。

【问题 2】 扩展接口模式结构通常包含四个角色：基础接口、组件、扩展接口和客户端。其中每个扩展接口需要通过扩展基础接口获得基本操作能力，然后加入自己特有的操作接口，并通过设置全局唯一接口 ID 对自身接口进行标识。每个具体的组件需要实现扩展接口完成实际操作，客户端不与组件直接交互，而需要通过扩展接口交互提出调用请求，扩展接口根据请求查找并选择合适的实现组件响应客户端请求。根据题干描述，可以看出基础接口这一角色应该对应统一操作接口，组件这一角色应该对应监控组件，扩展接口这一角色应该对应新网络设备的操作接口，客户端这一角色应该对应控制终端。

试题答案

【问题 1】（10 分） MVC 架构风格：用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。MVC 架构将整个软件系统划分为模型、视图和控制器 3 个部分。模型负责维护并保存具有持久性的业务数据，实现业务处理功能，并将业务数据的变化情况及时通知视图；视图负责呈现模型中包含的业务数据，响应模型变化通知，更新呈现形式，并向控制器传递用户的界面动作；控制器负责将用户的界面动作映射为模型中的业务处理功能并实际调用之，然后根据模型返回的业务处理结果选择新的视图。

在本题中：M：监控组件 V：控制终端 C：管理模块。

【问题 2】（15 分） 基础接口：统一操作接口； 组件：监控组件 扩展接口：新网络设备的操作接口 客户端：控制终端

（1）扩展接口

（2）组件

（3）操作接口

（4）接口 ID

（5）扩展接口

2014 年下半年试题 2

试题答案

【问题 1】（8 分）

数据流：数据流是数据在系统内传播的路径，因此由一组成分固定的数据组成。

外部实体：代表系统之外的实体，可以是人、物或其他软件系统。

加工（处理）：加工是对数据进行处理的单元，它接收一定的数据输入，对其进行处理，并产生输出。

数据存储：表示信息的静态存储，可以是文件、文件的一部分、数据库的元素等。

【问题 2】（12 分） 1、加工 P5.3 仅有输入流，缺少输出流，这属于黑洞。

2、加工 P5.4 仅有输出流，缺少输入流，这属于奇迹。

3、D1 到 A2：缺少处理数据流的加工。

4、D2：数据存储没有输出的数据流。

【问题 3】（5 分）

(1) R (2) R (3) CRU (4) RU (5) CRUD

2014 年下半年试题 3

试题分析

构件组装是指将库中的构件经适当修改后相互连接，或者将它们与当前开发项目中的软件元素相连接，最终构成新的目标软件。构件组装技术大致可分为基于功能的组装技术、基于数据的组装技术和面向对象的组装技术。

(1) 基于功能的组装技术 基于功能的组装技术采用子程序调用和参数传递的方式将构件组装起来。它要求库中的构件以子程序/过程/函数的形式出现，并且接口说明必须清晰。当使用这种组装技术进行软件开发时，开发人员首先应对目标软件系统进行功能分解，将系统分解为强内聚、松耦合的功能模块。然后根据各模块的功能需求提取构件，对它进行适应性修改后再挂接在上述功能分解框架（framework）中。

(2) 基于数据的组装技术 基于数据的组装技术首先根据当前软件问题的核心数据结构设计出一个框架，然后根据框架中各结 点的需求提取构件并进行适应性修改，再将构件逐个分配至框架中的适当位置。此后，构件的组装方式仍然是传统的子程序调用与参数传递。这种组装技术也要求库中构件以子程序形式出现，但它所依赖的软件设计方法不再是功能分解，而是面向数据的设计方法，例如 Jackson 系统开发方法。

(3) 面向对象的组装技术 由于封装和继承特征，面向对象方法比其他软件开发方法更适合支持软件重用。在面向对象的软件开发方法中，如果从类库中检索出来的基类能够完全满足新软件项目的需求，则可以直接应用。否则，必须以类库中的基类为父类采用构造法或子类法生成子类。

试题答案

【问题 1】（9 分） 获取构件的方法包括：

(1) 从现有构件中获得符合要求的构件，直接使用或作适应性（flexibility）修改，得到可重用的构件；

(2) 通过遗留工程（legacy engineering），将具有潜在重用价值的构件提取出来，得到可重用的构件；

(3) 从市场上购买现成的商业构件，即 COTS（Commercial Off-The-Shell）构件；

(4) 开发新的符合要求的构件。

开发构件的策略：

分区：指的是将问题情景的空间分割成几乎可以独立研究的部分；

抽象：是对给定实践内执行指定计算的软/硬件单元的一种抽象；

分割：是将结构引入构件的行为，支持对行为性质进行时序推理。

主流构件标准包括：COBRA、COM/DCOM/COM+、EJB（答出其中两种即可）。

【问题 2】（6 分）

(1) √ (2) × (3) × (4) √ (5) × (6) ×

【问题 3】（10 分）

局部接口：将 RS-232 芯片的外部中断向量连接到 CPU，从 RS-232 芯片中读取寄存器的内容，将控制信息定稿 RS-232 芯片寄存器。

技术相关接口（TDI）：诊断数据传输校验的错误。

链接接口（LIF）：对 RS-232 芯片初始化，并向其发送数据与从芯片接收数据。技术无关接口（TII）：RS-232 芯片或驱动构件复位和重启，重新配置传输速率。

2014 年下半年试题 4

试题分析

本题主要考查考生对软件架构评估、软件质量属性以及架构评估中相关概念的理解与掌握。考生应该在熟记基础概念的基础上结合实际问题灵活掌握并应用这些概念。

在解答本题时，首先需要对题干中的所有软件需求描述进行分析与梳理，区分并找出其中的需求分析、软件质量属性描述，或者可能的风险、权衡点或敏感点描述。具体列举如下：

- (a) 在正常负载情况下，系统必须在 0.5 秒内响应用户的交易请求，对应性能属性。
- (b) 用户的信用卡支付必须保证 99.999%的安全性，对应安全性属性。
- (c) 系统升级后用户名要求至少包含 8 个字符，对应软件业务需求。
- (d) 网络失效后，系统需要在 2 分钟内发现错误并启用备用系统，对应可用性属性。
- (e) 在高峰负载情况下，用户发起支付请求后系统必须在 10 秒内完成支付功能，对应性能属性。
- (f) 系统拟采用新的加密算法，这会提高系统安全性，但同时会降低系统的性能，系统权衡点。
- (g) 对交易请求处理时间的要求将影响系统数据传输协议和交易处理过程的设计，对应系统敏感点。
- (h) 需要在 30 人月内为系统添加公司新购买的事务处理中间件，对应可修改性属性。
- (i) 现有架构设计中的支付部分与第三方支付平台紧耦合，当系统需要支持新的支付平台时，这种设计会导致支付部分代码的修改，影响系统的可修改性，对应系统风险。
- (j) 主站点断电后，需要在 3 秒内将访问请求重定向到备用站点，对应可用性属性。
- (k) 用户信息数据库授权必须保证 99.999%可用，对应安全性属性。
- (l) 系统需要对 Web 界面风格进行修改，修改工作必须在 4 人月内完成，对应可修改性属性。
- (m) 系统需要为后端工程师提供远程调试接口，并支持远程调试，对应可测试性属性。

【问题 1】

在架构评估过程中，质量属性效用树（unility tree）是对系统质量属性进行识别和优先级排序的重要工具。效用树主要关注性能、可修改性、可用性和安全 4 个方面，考生首先必须填入对应的质量属性名称，并根据上述分析依次填入合适的标号。

【问题 2】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。敏感点是为了实现某种特定质量属性，一个或多个系统组件所具有的特性。权衡点是影响多个质量属性，并对多个质量属性来说都是敏感点的系统属性。根据上述分析可知题干描述中，（i）描述的是系统架构风险；（g）描述的是敏感点；（f）描述的是权衡点。

试题答案

【问题 1】（12 分）

- （1）性能
- （2）可修改性
- （3）（e）
- （4）（j）
- （5）（1）
- （6）（k）

【问题 2】（13 分）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。敏感点是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。权衡点是影响多个质量属性的特性，是多个质量属性的敏感点。风险点：（i）；敏感点：（g）；权衡点：（f）。

2014 年下半年试题 5

试题分析

基于 DNS 的负载均衡。基于 DNS 的负载均衡是在 DNS 服务器中为同一个主机名配置多个 IP 地址，在应答 DNS 查询时，DNS 服务器对每个查询将以 DNS 文件中主机记录的 IP 地址按顺序返回不同的解析结果，将客户端的访问引导到不同的节点上去，使得不同的客户端访问不同的节点，从而达到负载均衡的目的。

DNS 负载均衡的优点是经济、简单易行，并且节点可以位于 Internet 上任意的地方。但它也存在不少缺点，例如，为了保证 DNS 数据及时更新，一般都要将 DNS 的刷新时间设置得较小，但太小会造成太大的额外网络流量，并且更改了 DNS 数据之后也不能立即生效；DNS 负载均衡采用的是简单的轮转算法，不能区分节点之间的差异，不能反映节点的当前运行状态，不能做到为性能较好的节点多分配请求，甚至会出现客户请求集中在某一个节点上的情况。另外，要给每个节点分配一个 Internet 上的 IP 地址，这势必会占用过多的 IP 地址。

反向代理负载均衡。反向代理负载均衡是将来自 Internet 上的连接请求以反向代理的方式动态地转发给内部网络上的多个节点进行处理，从而达到负载均衡的目的。反向代理负载均衡既能以软件方式实现，也能在高速缓存器和负载均衡器等硬件设备上实现。反向代理负载均衡可以将优化的负载均衡

策略和代理服务器的高速缓存技术结合在一起,提升静态网页的访问速度,提高系统性能。另外,由于网络外部用户不能直接访问真实的节点计算机,反向代理负载均衡还具备额外的安全性(同理,基于 NAT 的负载均衡也有此优点)。

反向代理负载均衡的缺点主要表现在两个方面。首先,反向代理处于 OSI 参考模型应用层,因此,必须为每种应用服务专门开发一个反向代理服务器,这样,就限制了反向代理负载均衡技术的应用范围,现在一般都用于对 Web 服务器的负载均衡;其次,针对每一次代理,代理服务器都必须打开两个连接,一个对外,一个对内。在并发连接请求数量非常大的时候,代理服务器的负载也就非常大,代理服务器本身会成为服务的瓶颈。

Memcached 是一个高性能的分布式内存对象缓存系统,用于动态 Web 应用以减轻数据库负载。它通过在内存中缓存数据和对象来减少读取数据库的次数,从而提高动态、数据库驱动网站的速度。Memcached 基于一个存储键/值对的 hashmap。其守护进程(daemon)是用 C 写的,但是客户端可以用任何语言来编写,并通过 memcached 协议与守护进程通信。

Mysql 查询缓存的作用就是当查询接收到一个和之前同样的查询,服务器将会从查询缓存中检索结果,而不是再次分析和执行上次的查询。这样就大大提高了性能,节省时间。

试题答案

【问题 1】(10 分)

基于 DNS 的负载均衡是在 DNS 服务器中为同一个主机名配置多个 IP 地址,在应答 DNS 查询时,DNS 服务器对每个查询将以 DNS 文件中主机记录的 IP 地址按顺序返回不同的解析结果,将客户端的访问引导到不同的节点上去,使得不同的客户端访问不同的节点,从而达到负载均衡的目的。

反向代理负载均衡是将来自 Internet 上的连接请求以反向代理的方式动态地转发给内部网络上的多个节点进行处理,从而达到负载均衡的目的。

(1) 否 (2) 是 (3) 否 (4) 是 (5) 否 (6) 是 (7) 是 (8) 否

【问题 2】(7 分) 本系统应采用水平分区,因为社交网络数据库的数据表记录数量非常庞大,而且记录的访问,大多集中于本地区域,所以水平分区能极大提高处理效率。主从复制机制使得同样的数据,存在多个副本,这样让用户查询数据时,可以选择该数据最近的副本进行访问,提高效率,降低资源使用时的冲突。

【问题 3】(8 分)

(1) Memcached (2) 数据库 (3) Memcached (4) 数据库

(5) Memcached Memcached 相比数据库查询缓存:

缓存架构:数据库缓存只是将查询结果进行缓存,适用面很窄,而 Memcached 是将数据库中的表进行缓存,对于在这些表之上的操作均可适用。缓存有效性:Memcached 缓存时效较长,只要未更新,就属于有效状态,而数据库查询缓存时效较短(具体时效与配置有关),所以在此方面 Memcached 有优势。缓存数据类型:Memcached 缓存数据为表级,而数据库查询缓存为元组级。

2013 年下半年试题 1

试题分析

本案例题的主题是企业应用系统集成，使用的技术主要是 SOA 中的 ESB。所以要求考生对 ESB 有一定的熟悉程度。试题中问题 2 的前两问为概念题，在此不作分析，请直接参看答案。

问题 1 要求指出业务服务组件 RampControl 和 Flight Management 分别提供的服务名称。很多考生在看到这类问题时，都觉得自己没有做过面向服务架构设计中的服务设计，觉得题目难度已经超出自己的能力范围，而无法答题。其实不然，因为服务的划分，与传统开发中的功能模块划分一样，只是粒度大一些而已。只要认真看题，并分析系统提供了哪些功能，哪些功能归属于 RampControl，哪些应归属于 Flight Management，答案是很容易得出的。如题目“通常每个航班都有一位员工负责 Ramp Coordination，称之为 Ramp Coordinator 由 Ramp Coordinator 协调的业务活动包括检查机位环境、卸货和装货等。”从此就可以看出 Ramp Control 提供的服务包括：机位环境查询服务、卸货检查服务、装货检查服务。从流程图可以看出此组件还应包括检查关门服务。这样，整个流程图中，只余下接收航班信息服务适合划分至 Flight Management 组件。

对于问题 2 中的填图问题，主要是通过题目中给出的信息，以及图中其它同类位置的信息来判断。如：图中中心模块连接了各个分支模块，每个分支模块结构相同，所以可从分支模块表达的信息看出，外框要填写的是“**系统”，而内框是“**信息”，依据这个规则，在题干中很容易得出答案。中心部分，自然就是连接件 ESB 了。

试题答案

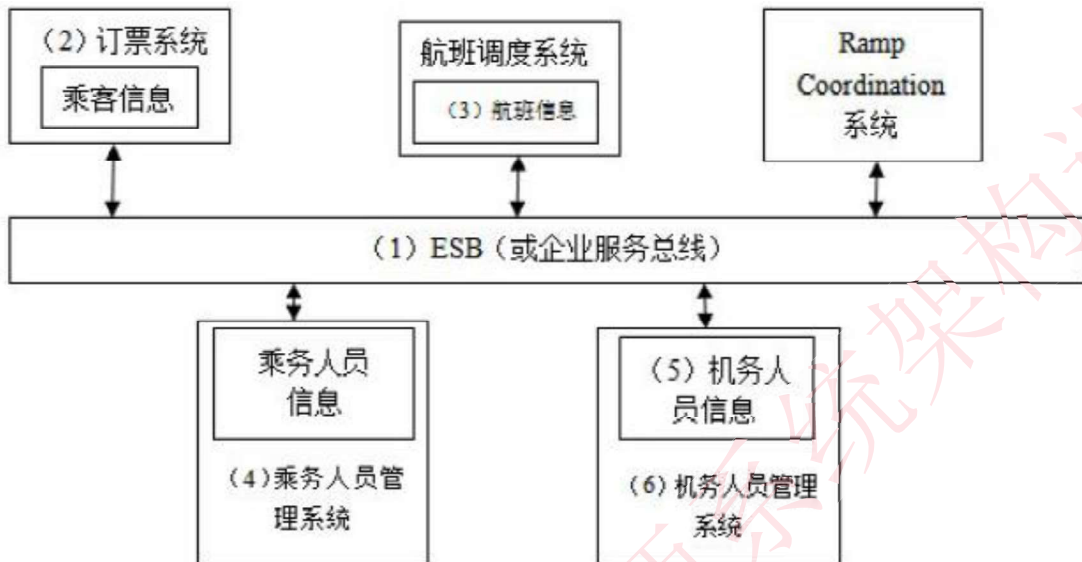
问题 1

- (1) 检查机位环境、检查卸货、检查装货、检查关门
- (2) 接收航班信息

【问题 2】 ESB 是传统中间件技术与 XML、Web 服务等技术结合的产物，主要支持异构系统集成。ESB 基于内容的路由和过滤，具备复杂数据的传输能力，并可以提供一系列的标准接口。

ESB 的主要功能：

- (1) 服务位置透明性；(2) 传输协议转换；(3) 消息格式转换；
- (4) 消息路由；(5) 消息增强；(6) 安全性；(7) 监控与管理。

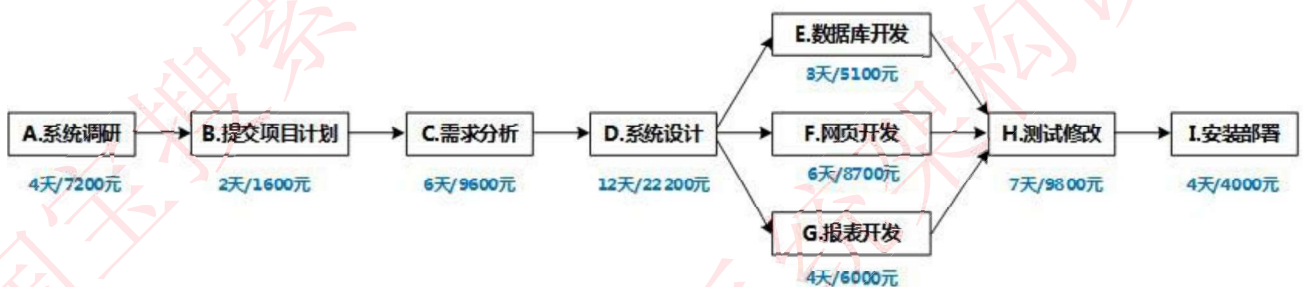


2013 年下半年试题 2

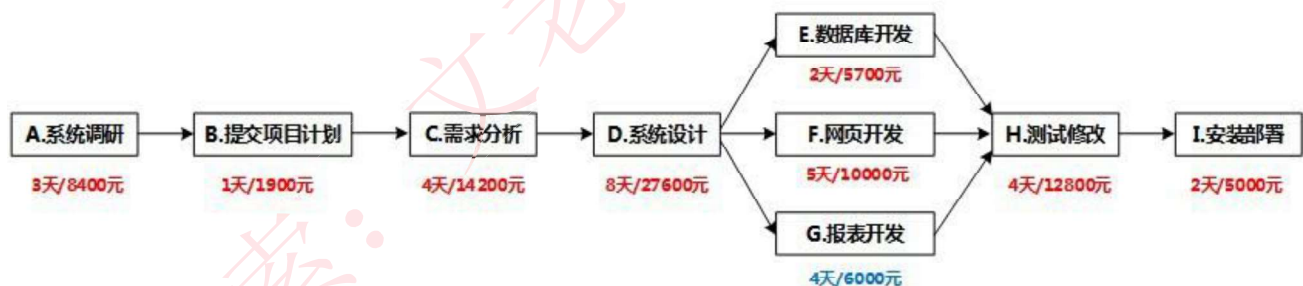
试题分析

正常工作成本=7200 元+1600 元+9600 元+22200 元+5100 元+8700 元+6000 元+9800 元+4000 元=74200 元。

正常工作工期=4+2+6+12+6+7+4=41 天。



最短工期成本=8400 元+1900 元+14200 元+27600 元+5700 元+10000 元+6000 元+12800 元+5000 元=91600 元。 最短工期=3+1+4+8+5+4+2=27 天。



要缩短项目的工期，主要有两种方法：

赶工：对成本和进度进行权衡，确定如何尽量少增加费用的前提下最大限度地缩短项目所需要的时间，

称为赶进度也称赶工。

快速跟进：调整逻辑关系，通过对各种逻辑关系并行确定来缩短项目周期。在进行项目设计中，当风险不大时，通过精心安排而使项目的前后阶段相互搭接以加快项目进展速度的做法叫快速跟进。其中快速跟进由于只是将部分工作提前开始，所以不会明显增加成本，在当前的环境中，是比较合适的方法。

任务名称	正常工作	加班工作	可压缩天数	压缩 1 天增加费用
A.系统调研	4 天/7200 元	3 天/8400 元	1	1200 元
B.提交项目计划	2 天/1600 元	1 天/1900 元	1	300 元
C.需求分析	6 天/9600 元	4 天/14200 元	2	2300 元
D.系统设计	12 天/22200 元	8 天/27600 元	4	1350 元
E.数据库开发	3 天/5100 元	2 天/5700 元	1	600 元
F.网页开发	6 天/8700 元	5 天/10000 元	1	1300 元
G.报表开发	4 天/6000 元	任务外包无法赶工	0	
H.测试修改	7 天/9800 元	4 天/12800 元	3	1000 元
I.安装部署	4 天/4000 元	2 天/5000 元	2	500 元

试题答案

(1) 项目背景

(2) 项目经理、项目经理的主管领导、客户方联系人、客户方的主管领导，项目领导小组(项目管理团队)和项目实施小组人员

(3) 项目的总体技术解决方案

(4) 所选择的项目管理过程及执行水平

(5) 对这些过程的工具、技术和输入输出的描述

(6) 选择的项目的生命周期和相关的项目阶段

(7) 项目最终目标和阶段性目标

(8) 进度计划

(9) 项目预算

(10) 变更流程和变更控制委员会

(11) 对于内容、范围和时间的关键管理评审，以便于确定悬留问题和未决决策

【问题 2】 正常工作成本=74200 元。 正常工作工期=41 天。 最短工期成本=91600 元。 最短工期=27 天。

【问题 3】 在“B 提交项目计划”和“I 安装部署”任务中采用加班工作措施，以使得能够按照正常工作进度完成。

【问题 4】 标准时长 41 天的任务，要 34 天完成，应赶工 7 天。具体赶工的任务包括：将 A、B、H、I 四个任务加班完成，这样正好弥补之前延误的 7 天工期，最终以 79700 元完成项目。

2013 年下半年试题 3

试题分析

本题考查考生对软件故障（失效）模型影响分析（FMEA）方法的概念理解和技术掌握程度。FMEA 是安全关键系统常用的一种失效分析方法，它可帮助系统设计者找出系统中的薄弱环节，通过预防措施的制定，可及早消除系统可能产生的失效点。

问题 1 主要考查考生对 FMEA 的理解程度。在做 FMEA 分析之前考生必须清楚三种主要活动，其一是分析系统或产品有可能存在故障模式，其二是对每一种故障模式开展风险评估（量化分析），其三对故障机理进行分析，制定出预防和改进措施。风险评估可采用多种方法，对软件而言，严酷度等级（SESR）、发生概率等级（SOPR）和被检测难度等级（SDDR）是评估的主要评价参数，这些等级通常以经验或专家评判方法获取。考生应该清楚，FMEA 方法可适用产品整个生命周期的各类活动，不同活动可采用不同的评估方法。通常存在四种 FMEA：设计 FMEA、过程 FMEA、使用 FMEA 和服务 FMEA。

根据图给出的实例，完成 FMEA 的分析，重点考查考生灵活使用理论方法对实际问题的分析能力。问题 2 给出了 CSU01 通信模块的工作原理，考生可根据自己原先掌握的嵌入式数据采集的基本工作经验，回答出 FMEA 分析工作 5 个步骤中具体内容。回答的要点可以这样考虑：

（1）影响数据交换的主要失效原因不外乎发生在输入（接收）、输出（发送）操作上，因此可从这两方面入手；

（2）故障原因分析可将产生故障的点定位在三方面，即硬件控制器、发送方和自身缺陷；

（3）故障影响分析，按照 FMEA 的分层思想，根据题目中图的结构，影响范围可以包括对本软件模块（INT0-ISR）的影响、对上一层软件（DY 软件）的影响和对最终 DY 系统的影响；

（4）危害分析和改进措施，软件 FMEA 通常采用风险优先数 RPN 方法分析，改进措施通常就是根据危害性分析结果和风险优先数等情况，逐一对故障模式制定改进措施。

如果清楚了问题 1 和问题 2 所给出一些分析概念的话，则回答问题 3 的问题就相对容易，只要有实践经验，可通过表中的相关信息正确判断出相应内容。这里要说明的是故障模式 5 由于其软件风险优先数（SRPN）很高（336），在设计时应重点关注。

试题答案

- （1）找出产品/过程中潜在的故障模式
- （2）根据相应的评价体系对找出的潜在故障模式进行风险量化评估
- （3）列出故障起因/机理，寻找预防或改进措施
- （4）设计（5）过程（6）使用（7）服务

【问题 2】

- （1）根据通信协议，可按接收数据功能和发送数据功能分别确定故障模式
- （2）故障原因分为总线通信控制器原因、对方发送的原因和自身程序的原因
- （3）针对每个故障模式分析基对本模块直至整个 DY 系统造成的影响

(4) 采用风险优先数 RPN 方法进行该通信模块的危害性分析

(5) 根据以上故障模式、原因、影响及危害性的分析结果，综合考虑故障的影响及 SRPN 值等情况，对每个故障模式制定了相应的改进措施。

【问题 3】

(1) 程序写 0C300H 地址单元不当

(2) 无法响应 INT0 中断

(3) 线路误码

(4) 通信错误

(5) 程序控制错误

(6) 210

(7) 数据发送始终不成功数据发送始终不成功

2013 年下半年试题 4

试题分析

本题考查 MVC 模式以及 XML 技术的应用。

MVC 全名是 Model View Controller，是模型(model)－视图(view)－控制器(controller)的缩写，一种软件设计典范，用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。

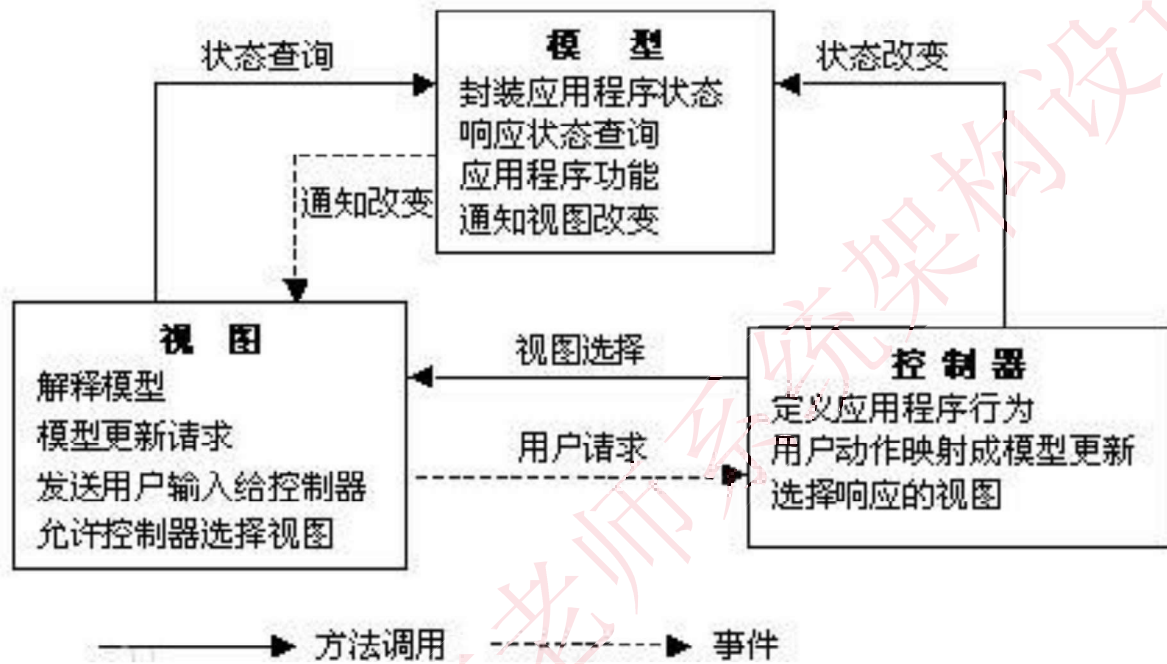
Model（模型）是应用程序中用于处理应用程序数据逻辑的部分。通常模型对象负责在数据库中存取数据。

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。

MVC 分层有助于管理复杂的应用程序，因为您可以在一个时间内专门关注一个方面。例如，您可以在不依赖业务逻辑的情况下专注于视图设计。同时也让应用程序的测试更加容易。

MVC 分层同时也简化了分组开发。不同的开发人员可同时开发视图、控制器逻辑和业务逻辑。



有了上面的介绍与图示，已经可以解决问题 1 了。问题 2 与问题 3 属于概念性问题，具体情况请参看参考答案。

试题答案

问题 1

- (1) 选择视图 (2) 业务视图 (3) 状态查询

MVC 模式对该个人银行系统的作用：

- (1) 允许多种界面的扩展，视图的变更与增加，与模型无关；
- (2) 易于维护，控制器和视图随着模型的扩展而扩展，只要保持公共接口，控制器和视图的旧版本可以继续使用；
- (3) 可支持功能强大的用户界面。

【问题 2】

从设计模式的角度来说，整个 XML 表现层解析的机制是一种策略模式。在调用显示 GUI 时，不是直接调用特定的表现技术的 API，而是装载 GUI 对应的 XML 配置文件，然后根据特定的表现技术的解析器解析 XML，得到 GUI 视图实例对象。这样，对于 GUI 开发人员来说，GUI 视图只需要维护一套 XML 文件即可。

【问题 3】

- (1) 界面定制模块
- (2) 界面动态生成模块

(3) 界面配置模块 界面配置是对用户界面的静态定义，通过读取配置文件的初始值对界面配置。由界面配置对软件功能进行裁剪、重组和扩充，以实现特殊需求。 界面定制是对用户界面的动态修改过程，在软件运行过程中，用户可按需求和使用习惯，对界面元素（如菜单、工具栏、键盘命令）的属性（如

文字、图标、大小、位置等)进行修改。软件运行结束,界面定制的结果被保存。

系统通过 DOM API 读取 XML 配置文件的表示层信息(初始界面大小、位置等),通过数据存取类读取数据库中的数据层信息,运行时由界面元素动态生成界面。界面配置和定制模块在软件运行前后,修改配置文件,更改界面内容。

2013 年下半年试题 5

试题分析

本题考查的是安全性方面的知识,解决相关的问题,需要对一些基础知识有一定了解。如:对称加密与非对称加密的基本流程,优缺点;口令认证与基于公钥的数字证书认证做法有什么不同;数据库加密中的加解密 API 与透明加密。

试题答案

(1) 基于口令的认证方式实现简单,但由于口令复杂度及管理方面的原因,易受到认证攻击;而在基于公钥体系的认证方式中,由于其密钥机制的复杂性,同时在认证过程中私钥不在网络上传输,因此可以有效防止认证攻击,与基于口令的认证方式相比更为安全。

(2) 按照需求描述,在完成用户身份鉴别后,需依据用户身份进一步对业务数据进行安全保护,且受保护数据中包含用户私有的终端机数据文件,在基于口令的认证方式中,用户口令为用户和认证服务器共享,没有用户独有的直接秘密信息,而在基于公钥的认证方式中,可基于用户私钥对私有数据进行加密保护,实现更加简便。

(3) 基于公钥体系的认证方式协议和计算更加复杂,因此其计算复杂度要高于基于口令的认证方式,但业务环境的总用户数据在 100 人以内,用户规模不大,运行环境又为局域网环境,因此基于公钥体系的认证方式可以满足平台效率要求。

【问题 2】

应采用流加密方式。因为需求中提及“单个敏感数据文件可能会达到数百兆的规模”,文件数据量较大,使用流加密方式可以获得更高的加解密效率。

数据加密与解密过程如下: 其加密过程为:首先生成一个对称密钥,使用用户公钥加密这个对称密钥后存储在文件头,然后用生成的对称密钥加密文件数据存储。其解密过程为:用户首先使用自己的私钥解密被加密的对称密钥,再用该对称密钥解密出数据原文。

【问题 3】

目前数据库管理系统提供的基本数据加密支持主要有以下两种:

(1) 加解密 API:数据库管理系统提供可在 SQL 语句中调用的加解密 API,应用可以利用这些 API 构建自己的基础架构,对数据进行加密保护。

(2) 透明加密:安全管理员为数据库敏感字段选择加密方式及密钥强度,应用访问受保护数据时只需使用口令打开或关闭密钥表,对数据的加密和解密由数据库管理系统自动完成。

加解密 API 方式的灵活性强,但构建和管理复杂;而透明加密方式管理简单,应用程序负担轻,但灵

活性较差。用户要求尽可能减少安全管理与应用程序的负担，因此应选择透明加密方式。