# 软件体系结构复习

A（了解（选择、判断）） B（理解（选择、判断、论述）） C（应用（论述、综合））

## 软件体系结构概论

（1）国内普遍接受的定义：软件体系结构包括构件、连接件 和约束，它是可预制和可重构的软件框架结构。

（2）体系结构的作用/意义：

• 体系结构是风险承担者(Stakeholder)进行交流的手段

• 体系结构是早期设计决策的体现

• 体系结构本身就是一种可重用的模型

（3）构件、连接件和约束的定义：

-构件是可预制和可重用的软件部件，是组成体系结构的基本计算单元或数据存储单元

-连接件也是可预制和可重用的软件部件，是构件之间的连接单元

-构件和连接件之间的关系用约束来描述

### 软件危机 B

软件危机是指落后的软件生产方式无法满足迅速增长的计算机软件需求，从而导致软件开发与维护过程中出现一系列严重问题的现象。（百度）

随着软件规模越来越大，软件生产与维护越来越困难（课件）

1. 软件危机的表现

– 软件成本日益增长

– 开发进度难以控制

– 软件质量差

– 软件维护困难

1. 原因

– 用户需求不明确

– 缺乏正确的理论指导

– 软件规模越来越大

– 软件复杂度越来越高

1. 如何克服

（课件）

• 要提高软件开发效率，提高软件产品质量，必须采用工程化的开发方法与工业化的生产技术。

• 技术问题

– 自顶而下地解决问题：

• 隐蔽细节，建立对软件系统结构的清晰描述并将其与具体实现相映射

– 基于重用（复用，reuse）的软件生产技术

• 管理问题

– 软件工程、软件过程管理

（课本）方法、工具、过程

1. 软件工程方法为软件开发提供了“如何做”的技术，是完成软件工程项目的技术手段
2. 软件工具是人们在开发软件个活动中智力和体力的扩展和延伸，为软件工程方法提供了自动或半自动的软件支持环境
3. 软件工程过程则是将软件工程的方法和工具综合起来以达到合理、及时地进行计算机软件开发的目的

### 构件与软件重用 B

1、构件是可预制和可重用的软件部件，是组成体系结构的基本计算单元或数据存储单元

2、构件模型：OMG、CORBA、DCOM（COM）、EJB

3、软件体系结构重用的是什么

（1）知识重用（例如，[软件工程](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%B7%A5%E7%A8%8B" \t "https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E9%87%8D%E7%94%A8/_blank)知识的重用）。

（2）方法和标准的重用（例如，面向对象方法或国家制定的 软件 开发规范的重用）。

（3）软件成分的重用。

4、构件的定义

（1）一个软件构件是可执行软件的一个可分离的单元

 （2）只能通过构件的接口来访问它的服务

 （3）必须能得到其接口的细节

 （4）可以与其它构件实现互操作

 （5）需要某种环境的支持

 （6）在进行了必要的组装和配置过程之后,它才能以接口规定的方式来使用

### 软件体系结构的兴起和发展 A

·起初软件设计重点：数据结构、算法

·发展4个阶段：

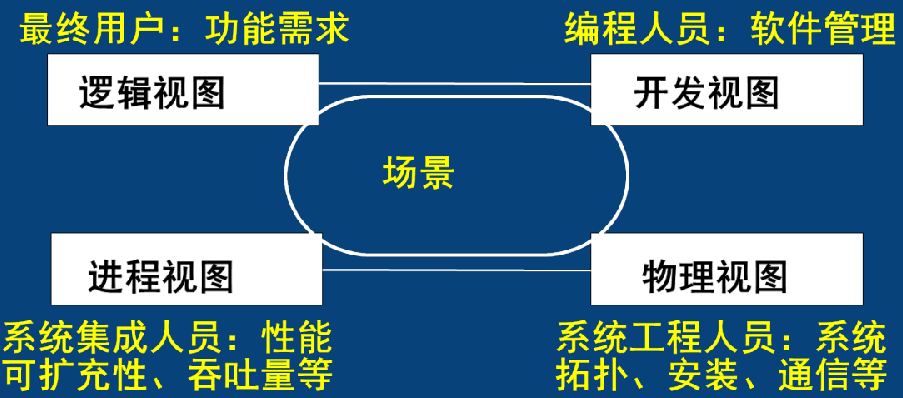
1. 无体系结构设计阶段。以汇编语言进行小规模应用程序开发
2. 萌芽阶段。出现了程序设计主题，以控制流图和数据流图构成软件结构为特征
3. 初级阶段。出现了从不同侧面描述系统的结构模型，以UML为典型代表
4. 高级阶段。以描述系统的高层抽象为中心，不关心具体的建模细节，划分了体系结构模型与传统软件架构的界限，以“4+1”模型为标志。

·程序结构、软件结构、软件体系结构三者有什么区别和联系？

程序结构指的是软件中的一个程序的模块及其相互关系；软件结构指的是组成某个或某类软件的模块及其相互关系；软件体系结构指的是：构成软件系统的元素的描述、元素之间的相互作用、元素的集成模式及模式约束，是一种结构、行为和属性的高级抽象。

## 软件体系结构建模

### 2.1 “4+1”视图模型 B+C



物理视图表达与客户原有环境的关系（系统的动态结构）

进程视图（并发视图）表达质量方面的属性（系统的动态结构）

逻辑视图表达系统的功能需求（系统的静态结构）

开发视图（模块视图）侧重于软件模块的组织和管理（系统的静态结构）

场景：重要系统活动的抽象，帮助设计师找到构件族之间的关系，可用文本或图形表示

#### 2.1.2 开发视图 C （类图、看图说话）

### 2.2 软件体系结构的核心模型 A

组成：构件、连接件、配置、端口、角色

### 2.3 软件体系结构的生命周期模型 A

阶段：需求分析阶段、建立软件体系结构阶段、设计阶段、实现阶段

软件体系结构的生命周期：

1. 软件体系结构的非形式化描述
2. 软件体系结构的规范描述分析
3. 软件体系结构的求精及其验证
4. 软件体系结构的实施
5. 软件体系结构的演化和扩展
6. 软件体系结构的提供、评价和度量
7. 软件体系结构的终结

## 软件体系结构风格

### 3.1 经典体系结构风格 B

§ 定义：

–描述特定领域中软件系统家族的组织方式的惯用模式，反映了领域中众多系统所共有的结构和语义特性，并指导如何将各个模块和子系统有效地组织成一个完整的系统。

§ “软件体系结构风格”的组成

– 一组组件类型。例如：数据容器，过程，对象

– 一组连接件类型/交互机制。例如：过程调用，事件，管道

– 这些组件的拓扑分布

– 一组对拓扑和行为的约束。例如：数据容器不能自己存储数据，管道不能是循环的

– 一些对风格的成本和收益的非正式描述。例如：如果你需要重用性并且性能不是很重要，那么可以使用管道风格

§ 体系结构风格的分类

（1）数据流风格：批处理序列、管道与过滤器；

（2）调用/返回风格：主程序与子程序、面向对象风格、层次结构（分层）；

（3）独立构件风格：进程通信、事件系统（隐式调用、显示调用）；

（4）虚拟机风格：解释器、基于规则的系统；

（5）仓库风格：数据库系统、超文本系统、黑板系统

**结合课件第2部分看，理解每种风格的特点和应用场景**

### 3.2 客户/服务器风格（C论述题）

（1）计算和服务主要在服务器端，部署在云的不是C/S

（2）C/S两端的主要任务

A、服务器端

①数据库安全性的要求

②数据库访问并发性的控制

③数据库前端的客户应用进程的全局数据完整性规则

④数据库的备份和恢复

1. 客户端
2. 提供用户于数据库交互的界面
3. 向数据库服务器提交用户请求并接收来自数据库服务器的信息
4. 利用客户应用程序对存在于客户端的数据执行应用逻辑要求

（3）优点：

系统的客户应用程序和服务器构件分别运行在不同的计算机上，系统中的每台服务器都可以适合各构件的要求，对于硬件和软件的变化显示出极大的适应性和灵活性，而且易于对系统进行扩充和缩小。

缺点：

* 1. 开发成本较高
  2. 客户端程序设计复杂
  3. 信息内容和形式单一
  4. 用户界面风格不一
  5. 软件移植困难
  6. 软件维护和升级困难
  7. 新技术不能轻易应用

### 3.3 三层C/S结构风格（C论述题）

1、表示层（用户接口）、功能层、数据层（对数据库管理）

2、优点：（与两层C/S结构相比）

（1）允许合理地划分三层结构功能，使之在逻辑上保持相对独立性，从而使整个系统的逻辑结构更为清楚，能提高系统和软件的可维护性和课扩展性

（2）允许更灵活有效地选用相应的平台和硬件系统，使之在处理负荷能力上与处理特性上分别适应于结构清晰的三层；并且这些平台和各个组成部分可以具有良好的可升级性和开放性。

（3）三层C/S结构中，应用的各层可以并行的开发，各层也可以选择各自最合适的开发语言

（4）允许充分利用功能层有效地隔离开表示层和数据层，未经授权的用户难以绕过功能层而利用数据库工具或黑客手段去非法地访问数据层，这就为严格的安全管理奠定了坚实的基础；整个系统的管理层次也更加合理和可控

3、缺点：

三层C/S结构个层间的通信效率若不高，即是分配给个层的硬件能力很强，做为整体也达不到所要求的性能

### 3.4 浏览器/服务器风格（C论述题）

1、优点

（1）系统安装、修改和维护都在服务器端解决

（2）结合www浏览器技术，实现了原来需要复杂的专用软件才能实现的强大功能，并节约了成本

（3）用户无需安装客户端，极大的方便了用户使用

2、缺点

（1）B/S体系结构缺乏对动态页面的支持能力，没有集成有效的数据库处理能力

（2）B/S体系结构的系统扩展能力差安全性难以控制

（3）采用B/S体系结构的系统，在数据查询等响应速度上要远低于C/S体系结构

（4）B/S体系结构的数据提交一般以页面为单位，数据交互性不强，不利于在线事务处理应用

### 3.5 - 3.10 不同的软件体系结构风格（A 选择）

1、公共对象请求代理体系结构CORBA

2、正交软件体系结构风格

3、基于层次消息总线的体系结构风格

4、异构结构风格

5、互连系统构成的系统及其体系结构

6、特定领域软件体系结构DSSA

## 软件体系结构描述

### 4.1 软件体系结构描述方法 A

软件体系结构描述方法可以分为文字表达工具、数学表达工具和图形表达工具。其中产业界多用图形，学术界多用数学

### 4.3 - 4.4 软件体系结构描述语言（ADL）A

1、ADL三要素：构件、连接件、体系结构配置

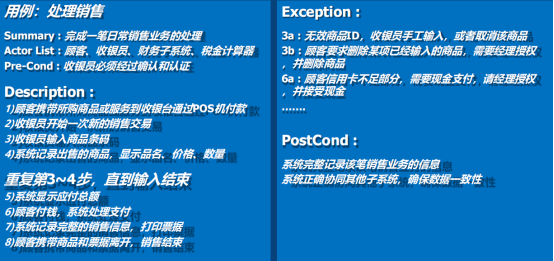
2、典型的体系结构语言（选择题）

1. UniCon
2. Wright
3. C2
4. Rapide
5. SADL
6. Aesop
7. ACME

## 统一建模语言

### 5.1 - 5.2 UML（统一建模语言）A

### 5.3 用例图 C



1. 在这个图中有几个角色（参与者）
2. 描述系统边界（省）
3. 结合背景描述有几个用例，分别是干什么的
4. 选择某一个熟悉的用例进行描述（如图）
5. 用例编号
6. 用例名称
7. 执行者
8. 用例说明（大致描述，对应summary）
9. 前置条件：（可以写无）
10. 基本事件流（对应description）
11. 异常事件流
12. 后置条件：（一般不能写无，都会有，包括系统的反应、数据库表有什么变化；系统（页面）有什么变化）

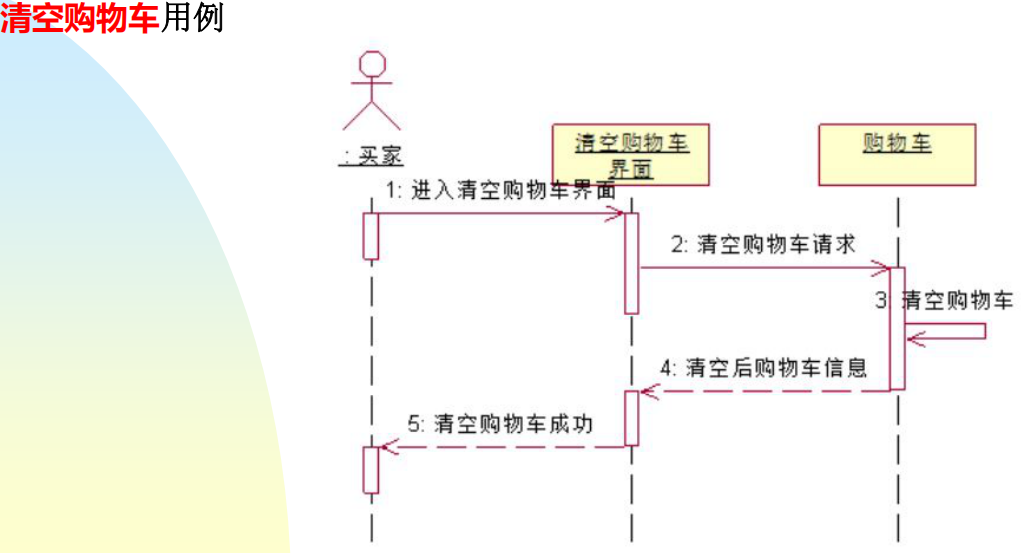
### 5.4 类图 C

1、有几个类，类名，类的属性和方法

2、类与类之间的关系（关联关系<组合（实心菱形）、聚合（空心菱形，指向部分类）>、继承关系、实现<虚线空三角，指向基类>）

### 5.5 交互图 C

#### 5.5.1 顺序图 C



1、消息（简单、同步、异步、返回）

2、激活

3、生命线

### 5.6 状态图 A

1、开始结束图符

2、状态

3、迁移线

### 5.7 活动图 A

1、活动

2、同步棒（分叉、合并）

3、泳道（不同部门的划分）

## 可扩展标记语言

XML（可扩展标记语言）A

XML可以人信息提供者根据需要，自行定义标记及属性名，结构化地描述信息内容，因此赋予了应用软件强大的灵活性，为开发者和用户带来了许多好处。

## 动态软件体系结构

### 7.1 动态软件体系结构概述 A

允许在系统运行时发生更新的软件体系结构成为动态软件体系结构，动态体系结构在系统创建后可以动态地更新。

## 基于服务的体系结构

### 8.1 SOA概述 B

1、SOA（面向服务的体系结构）

2、什么是SOA

广义：软件开发方法

狭义：体系结构风格

3、SOA的构件：服务提供者、服务使用者、服务中介

4、特征

（1）松散耦合

（2）粗粒度服务：能够提供少量业务流程可用性服务

（3）标准化接口

### 8.3 SOA的关键技术 B

|  |  |
| --- | --- |
| 服务栈 | 主要技术 |
| 发现服务层 | UDDI、DISCO |
| 描述服务层 | WSDL、XML、Schema |
| 消息格式层 | SOAP、REST |
| 编码格式层 | XML |
| 传输协议层 | HTTP、TCP/IP、SMTP等 |

### 8.4 SOA的实现方法 B

1、Web Service

web服务体系结构模型包括：服务的提供者，服务的注册中心，服务客户端。

Web服务技术核心基于可扩展标记语言XML的标准，包括简单对象访问协议，Web服务描述语言和统一描述，发现和集成协议。

### 8.5 服务描述语言 A

WSDL是对服务描述的语言，有一套基于XML的语法定义，是发布到注册中心时用的描述语言

### 8.6 统一描述、发现和集成协议 A

UDDI是一种用于描述、发现、集成Web服务的技术，它是Web服务协议栈的一个重要部分。

### 8.7 消息封装协议 A

SOAP以XML形式提供一个简单、轻量的用于在分散或分布环境中交换结构化和类型信息的机制。

## 富互联网应用体系结构

### 9.1 RIA的概念 A

RIA（Rich Internet Application）富互联网应用

RIA是Web开发和部署模式的一种演变。“富”的含义分别是丰富的数据模型和丰富的用户界面

### 9.4 AJAX技术 A

包含基于XHTML和CSS标准的表示；使用ＤＯＭ进行动态显示和交互；使用XMLHttpRequest与服务器进行异步通信；使用javaScript绑定一切

### 9.5 Mashup技术 A

Mashup就是糅合的意思，是当今网络上的一种技术，将两种以上的公共资源或者私有数据库的Web应用加在要求，形成一个整合应用。

## 软件体系结构的分析与测试

### 10.1 体系结构的可靠性建模 B

1、基于的思想：概率统计

2、1983年美国IEEE计算机学会对“软件可靠性”一词正式作出了如下的定义：

（1）在规定的条件下，在规定的时间内，软件不引起系统失效的概率，该概率是系统输入和系统使用的函数，也是软件中存在的错误的函数；系统输入将确定是否会遇到已存在的错误（如果错误存在的话）；

（2）在规定的时间周期内，在所述条件下程序执行所要求的功能的能力。

3、软件可靠性

（1）可靠性是软件的13个质量因素中最关键、最重要的◼

（2）软件可靠性是指在规定时间和条件下软件无故障运行的概率，是系统功能或软件产品中存在的缺陷的函数◼

（3）软件故障产生的原因是软件缺陷，但缺陷并不一定导致故障的产生，高缺陷率的软件的可靠性不一定就差◼

（4）软件失效意味着软件运行中断或者无法完成所规定的

4、软件不可靠的重要原因

（1）不完善的需求定义

（2）客户与开发人员缺乏交流

（3）偏离软件需求

（4）逻辑设计错误

（5）编码错误

（6）编码与文档不一致

（7）缺少测试过程

（8）接口定义错误

（9）不受控的变更

### 10.2 软件体系结构的风险分析 A

风险分析方法是什么：➢动态方法➢构件依赖图

### 10.3 基于体系结构描述的软件测试 A

测试方法

➢测试内容

➢测试准则

➢测试需求和测试用例的生成

## 软件体系结构评估

### 11.1 软件体系结构评估概述

#### 11.1.1 软件质量属性 B

1、软件的质量属性— 概述

➢ 商业目的决定了SA必须满足的一些质量性质

➢ 重新设计系统往往不需要改变功能，而是为了满足某些质量属性

➢ 没有任何一个质量属性完全依赖于设计、实现和部署的

➢ 软件结构确定了SA对质量属性的支持，但不是惟一的。如：即使理想的SA，但可读性很差的代码也会使系统难于修改

➢ 系统的性能：是一个既依赖于、又不完全依赖于SA的质量属性。如：性能受构件间通讯的数据量、为每个构件所分配的功能、分配共享资源的方式等这些SA相关方面的影响；也受所选择实现功能的算法、算法的编码方式等非SA因素的影响

➢ 质量属性的实现不能以孤立的方式实现。任何一个质量属性的实现都会对其它的质量属性带来积极或消极的影响

1. 软件质量属性
2. 性能
3. 可靠性 a、容错； b、健壮性
4. 可用性
5. 安全性
6. 可修改性 a、可维护性； b、可扩展性； c、结构重组； d、可移植性
7. 功能性
8. 可变性
9. 集成性
10. 互操作性

#### 11.1.2 几个基本概念 B

1、敏感点和权衡点

敏感点是一个或多个构件的特性

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

2、风险承担者（项目干系人/涉众）

3、场景：在进行体系结构评估时，一般首先要精确地得出具体的质量目标，并以之作为判定该体系结构优劣的标准。为得出这些目标而采用的机制叫场景，是从风险承担者的角度对与系统的交互的简短描述。

#### 11.1.3 评估方式 A

1、基于调查问卷或检查表的评估方式

2、基于场景的评估方式

3、基于度量的评估方式

4、评估方式比较

### 11.2 ATAM评估方法 A

### 11.3 SAAM评估方法 A

### 11.4 ARID评估方法 A

## 第十二章 基于体系结构的软件开发

### 12.1 设计模式

设计模式时前人经验的总结，它使人民可以方便地复用成功的设计和体系结构。

#### 12.1.8 MVC模式的设计与实现 B

MVC模式可以包括三个抽象类，一个是View抽象类,它从模型获得显示信息,并以特定的形式展示给用户，对于相同的信息可以有多个不同的显示形式或视图:一个是Controller抽象类，处理用户与软件的交互操作，其职责是决定软件的控制流程,确保用户界面与模型间的对应联系，它接收用户的输人，将输人反馈给模型，进而实现对模型的计算控制,再根据用户的需求,创建合适的视图返回到用户界面,它是使模型和视图协调工作的部件;另外一个是Model抽象类，它向视图(View抽象类)和控制器(Controller抽象类)提供业务逻辑服务。这三个抽象类之间的关系可以描述如下:控制器把接收到的请求或者数据传送到模型去处理,再根据用户的要求,创建一个合适的视图，该视图从模型中读取处理后的结果，以特定的形式显示出来。

### 12.2 中间件技术

#### 12.2.1中间件概述 B

是处于系统软件和应用软件之间的一类软件，它使设计师集中设计与应用有关的部分，大大简化了设计和维护工作。

#### 12.2.2主要的中间件 A

1. 远程过程调用 RPC
2. 对象请求代理 ORB
3. 远程方法调用 RMI
4. 面向消息的中间件 MOM
5. 事物处理监控器 TPM

### 12.3 基于体系结构的设计方法 A

ABSD（architecture-Based software design）

### 12.4 体系结构的设计与演化 A

1、实验原型阶段

2、演化开发阶段

### 12.5 基于体系结构的软件开发模型 A

ABSDM

1. 体系结构需求
2. 体系结构设计
3. 体系结构文档化
4. 体系结构复审
5. 体系结构实现
6. 体系结构演化

## 第十三章 软件产品线体系结构

### 13.1 软件产品线的出现和发展

1、软件产品线方法是软件工程领域中软件体系结构个软件重用技术的发展结果。

2、软件重用的基本实现是尽最大可能重用已有的软件资源

### 13.2 软件产品线概述

“软件产品线” 是CMU(Carnegie Mellon University) 的SEI (Software Engineering Institute) 提出的。

一个产品线(Product Line)：是共享一组公共的、可管理的特性，并且满足特定市场需求的产品集合。

### 13.3 框架和应用框架技术

.net framework 软件-->中间件-->产品线思想

### 13.4 软件产品线的基本活动

核心资源开发 产品开发 管理