软件架构

# 软件体系结构定义和意义

**定义：**软件体系结构为软件系统提供了一个结构、行为和属性的高级抽象，由构成系统的元素的描述、这些元素的相互作用、知道元素集成的模式以及这些模式的约束组成。软件体系结构不仅指定了系统的组织结构和拓扑结构，还显示了系统需求和构成系统的元素之间的对应关系，提供了一些设计决策的基本原理。

**意义：**1体系结构是风险承担着进行交流的手段。2体系结构是早期设计决策的体现。（明确了约束、决定了结构、制约着质量属性、预测质量、使得控制更简单、有助于循序渐进的原型设计、可以作为培训的基础）3软件体系结构是可传递和可重用的模型。

# 软件体系结构的产生、发展、兴起的原因以及发展现状

**产生：**20世纪60年代的软件危机（成本日益增长、进度难控制、质量差、维护困难）使得人们开始重视软件工程。起初重心都放在数据结构和算法上，但随着规模越来越大越复杂、系统结构设计已经比数据结构和算法重要。因此在此背景下，软件体系结构认识到重要所在。对软件体系结构的系统、深入的研究将会成为提高软件生产率和解决软件维护问题的新的有希望的途径。软件体系结构产生于软件工程，结合了计算机体系结构和网络体系结构的思想，成为一门新的研究方向和独立学科。

**发展：**从最初的“无结构”设计到现在的基于体系结构的软件开发，软件体系结构技术的发展经历了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为特征。

（2）萌芽阶段。出现了程序结构设计主题，以控制流图和数据流图构成软件结构为特征。

（3）初期阶段。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结构为中心，不关心具体的建模细节，划分了体系结构模型与传统软件结构的界限，该阶段以4+1模型为标志。

现阶段是20世纪90年代后的基于构件的软件的开发阶段，以过程为中心，强调软件开发采用构建化技术和体系结构技术，要求开发出的软件具备很强大的自适应性、互操作性、可扩展性和可重用性。此阶段软件体系结构已经作为一个明确的文档和中间产品存在于软件开发过程中。

**目前体系结构的研究活动主要包括有：**

1软件体系结构描述语言

2体系结构描述构造与表示

3体系结构分析、设计与验证

4体系结构发现、演化与重用

5基于体系结构的软件开发方法

6特定领域的体系结构框架

7软件体系结构的支持工具

8软件产品线体系结构

9建立评价软件体系结构的方法

# 典型的构建实现模型

（软件重用是指在两次或者多次不同的软件开发过程中重复使用相同或者相似软件元素的过程。）

构件是指语义完整、语法正确和有可重用价值的单位软件，是软件重用过程中可以明确辨识的系统。

构件模型是对构件本质特征的抽象描述。目前国际上的模型目标和作用不同，已经有3个流派，分别是：OMG的CORBA（Common Object Request Broker Architecture）；sun的EJB；Microsoft的DCOM（Distributed Component Object Model）。

•构建获取方法

1从现有构件中获得符合要求的构件，直接使用或做适应性的修改，得到可重用的构件。

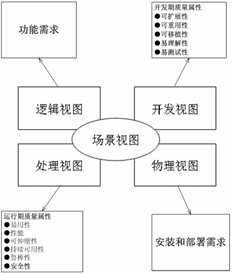
2通过遗留工程，将具有潜在重用价值的构件提取出来，得到可重用的构件。

3从市场上购买现成的商业构件，即COTS构件。

4开发新的符合要求的构件。

一个组织在进行以上决策的时候必须要考虑到不同方式获取构件的一次性成本和以后的维护成本，然后最优选择。

# 4+1视图



* 逻辑视图（Logical View）

设计的对象模型（使用面向对象的设计方法时）

* 过程视图（Process View）

捕捉设计的并发和同步特征

* 物理视图（Physical View）

描述了软件到硬件的映射，反映了分布式特性

* 开发视图（Development View）

描述了在开发环境中软件的静态组织结构

**逻辑视图**主要用来描述系统的功能需求.

• 在逻辑视图中，系统分解成一系列的功能抽象、功能分解与功能分析，主要来自问题领域（Problem Definition)。

• 在面向对象技术中，通过抽象、封装、继承,可以用对象模型代表逻辑视图，用类图（Class Diagram）描述逻辑视图。

* 构件(Components)：类、类服务、参数化类、类层次
* 连接件(Connectors)：关联、包含聚集、使用、继承、实例化

**开发视图**主要用来描述软件模块的组织与管理，服务于软件编程人员。

• 开发视图的风格通常是层次结构，每个层为上一层提供良好定义的接口，层次越低，通用性越好。

* 构件：模块、子系统、层
* 连接件：参照相关性、模块/过程调用

**进程试图**侧重系统的运行特性，关注非功能性的需求（性能，可用性）。服务于系统集成人员，方便后续性能测试。

• 强调并发性、分布性、集成性、鲁棒性（容错）、可扩充性、吞吐量等。定义逻辑视图中的各个类的具体操作是在哪一个线程（Thread）中被执行。

* 构件：进程、简化进程、循环进程
* 连接件：消息、远程过程调用（RPC）、双向消息、事件广播

**物理试图**主要描述硬件配置。服务于系统工程人员，解决系统的拓扑结构、系统安装、通信等问题。主要考虑如何把软件映射到硬件上，也要考虑系统性能、规模、可靠性等。可以与进程视图一起映射。

* 构件：处理器、计算机、其它设备
* 连接件：通信协议等

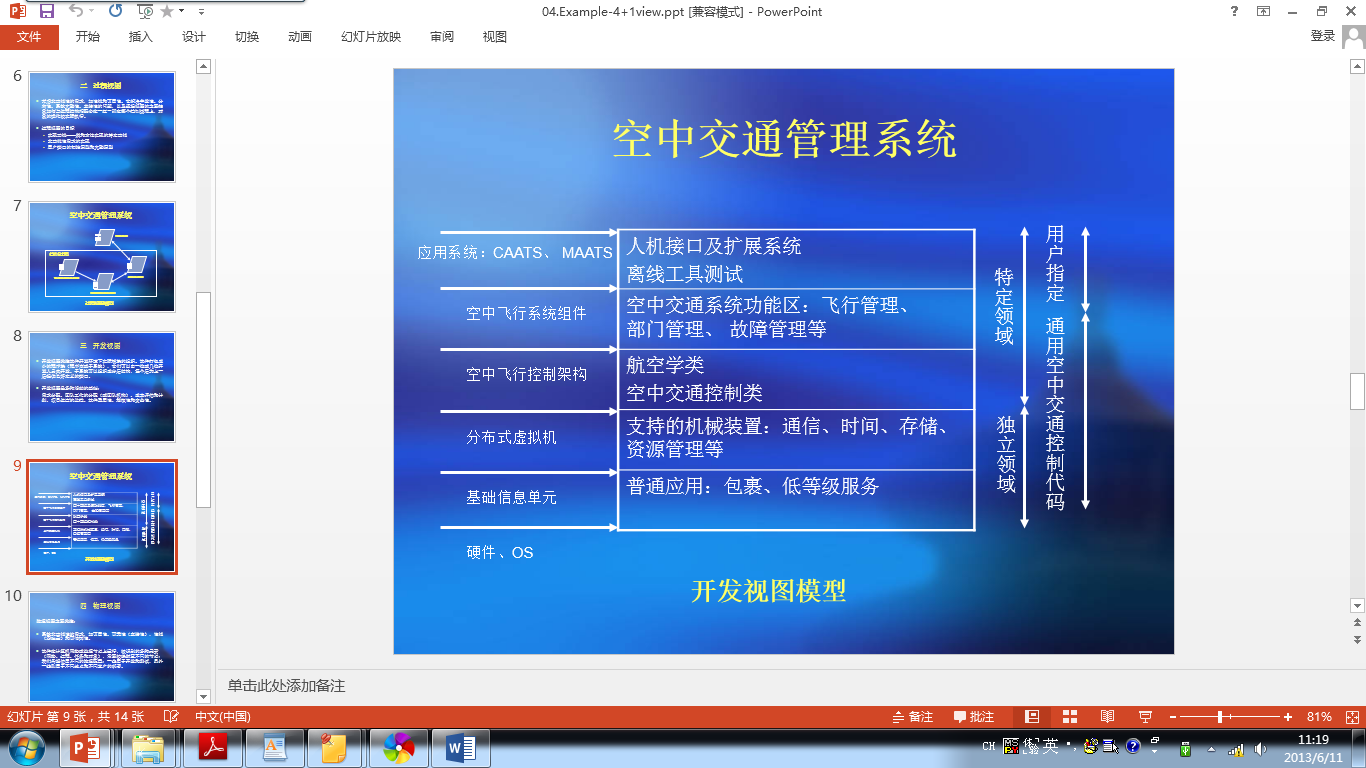
**场景**用于刻画构件之间的相互关系，将四个视图有机地联系起来。可以描述一个特定的视图内的构件关系，也可以描述不同视图间的构件关系。文本、图形表示皆可。

场景的表示法:

场景表示法与组件逻辑视图非常相似，但它使用过程视图的连接符来表示对象之间的交互，对象实例使用实线来表达。

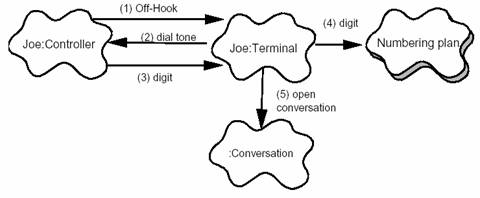
# “4+1”视图模型分析实例———空中交通管理系统







空中交通管理系统  
场景（本地呼叫初期）示例片段

****

* 1. Joe的电话控制器检测和校验摘机状态的变换,并发送消息唤醒相应的终端对象。
* 2. 终端分配一些资源，并要求控制器发出拨号音。
* 3. 控制器接受拨号并传递给终端。
* 4. 终端使用拨号方案来分析数字流。
* 5. 有效的数字序列被键入,终端开始会话。

# 核心模型

体系结构的核心模型由5种元素组成：构件、连接件、配置、端口、和角色。其中构件和连接件是最基本的元素。

模型组成元素

►构件 （Component）：原子构件、复合构件

具有某种功能的可重用软件模板单元

►连接件 （Connector）：表示构件之间的交互关系

♦ 管道（pipe）

♦ 过程调用 （procedure call）

♦ 事件广播 （Event Broadcast）

♦ 通信协议 （protocol）

►配置 （Configuration）：CC逻辑拓扑和约束

►端口 （Port）：构件外部接口

►角色 （Role）：连接件的接口

♦RPC角色：caller 和callee

♦Pipe角色：reading 和 writing



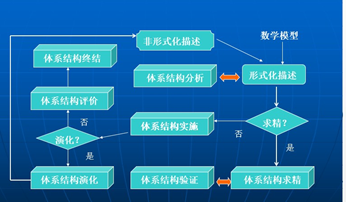
# 生命周期模型

* **软件系统开发的生命周期**

需求分析、体系结构建立、软件设计、软件实现、软件测试、系统维护

* **软件体系结构的生命周期模型**

软件体系结构的非形式化描述、软件体系结构的形式化描述与分析、软件体系结构的求精与验证、软件体系结构的实施、软件体系结构的演化与扩展、软件体系结构的评价与度量、软件体系结构的终结



# 软件体系结构风格分类

* **以数据为中心的构架**

数据集成——一个集中式的数据集与多个客户端进行通信

* **数据流构架**

成批数据流；管道和过滤器

* **虚拟组织架**

目标：可移植性

模拟对象：模拟硬件不具备的功能或软件环境

实例：java虚拟机(java的平台独立性)

* **调用——返回构架**

目标：可更改性和可扩展性

主——子程序（传统）；面向对象（类派生）；层次样式

* **独立组件构架**

由独立进程或对象组成，通过消息进行通信

例：C/S模式

* **异质构架**

多种构架样式的综合

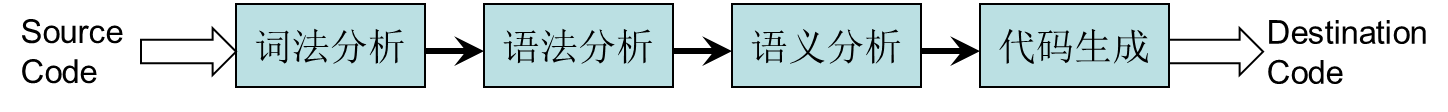
局部异质/层次异质/并行异质（同时符合几种样式）

# 管道过滤器

Component: Filter----Data Handling

Connector: Pipe----Data Translation and Transportation

例子：编译器系统



Advantages：

高内聚和低耦合

通过过滤器增加了灵活性。

过滤器组件的重用。

有利于系统的维护与更新－Evolution。

支持并行处理以提高效率。

Disadvantages：

增量式处理数据时，存在效率问题。

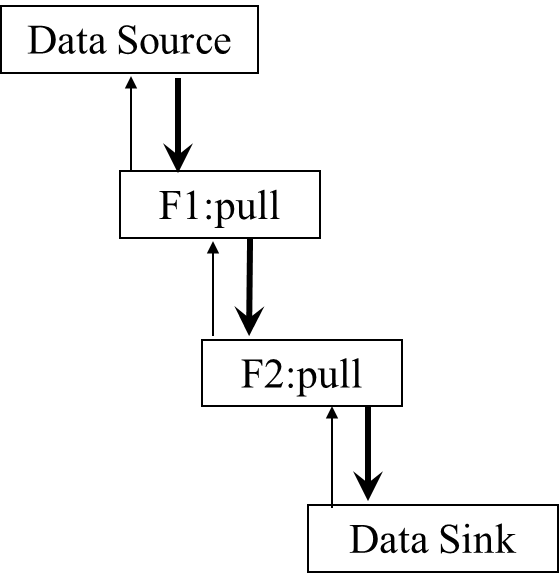
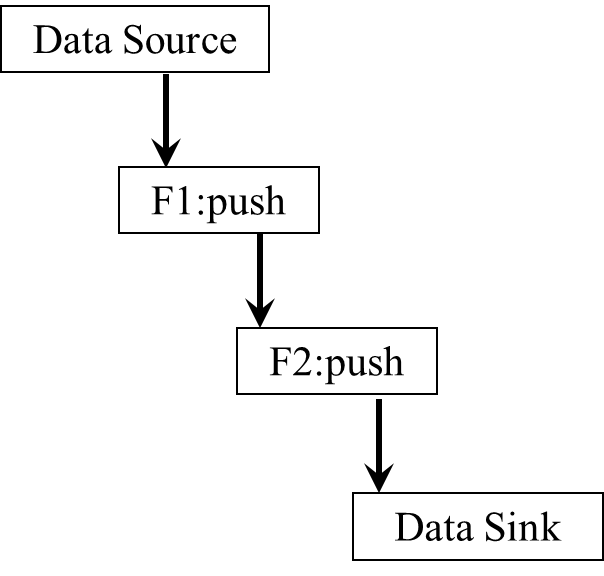
数据格式问题：数据转换额外开销。

不适合交互式应用系统

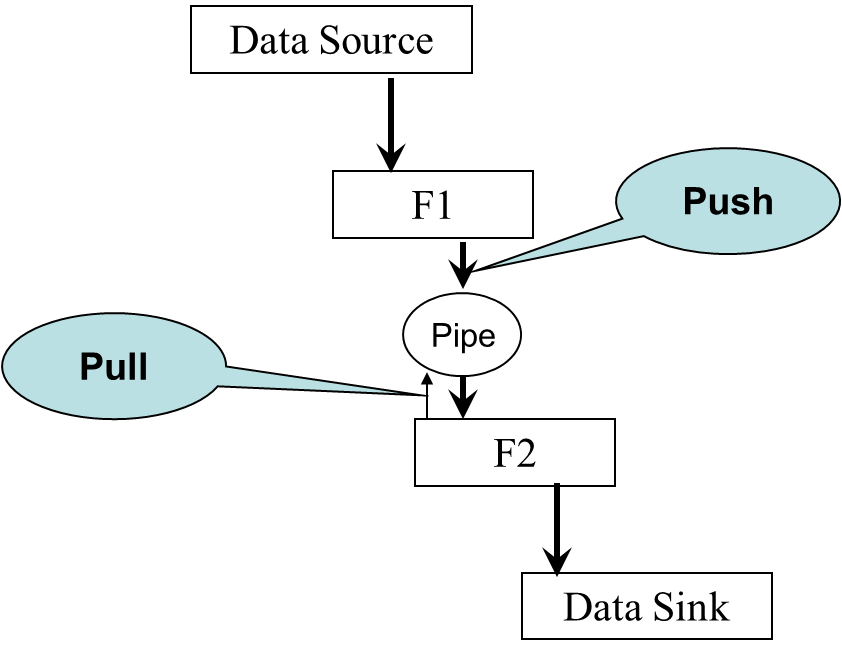
**Strus**

过滤器，是在java web中，你传入的request,response提前过滤掉一些信息，或者提前设置一些参数，然后再传入servlet或者struts的 action进行业务逻辑，比如过滤掉非法url（不是login.do的地址请求，如果用户没有登陆都过滤掉）,或者在传入servlet或者 struts的action前统一设置字符集，或者去除掉一些非法字符。

**Passive Filter**



**Active Filter**

****

# 层次结构（3层CS）

三层C/S结构是将应用功能分成表示层、功能层和数据层三个部分

**表示层**

用户接口部分，它担负着用户与应用间的对话功能。在变更用户接口时，只需改写显示控制和数据检查程序，而不影响其他两层。

**功能层**

应用的本体，它是将具体的业务处理逻辑编入程序中。处理所需的数据则要从表示层或数据层取得。通常，在功能层中包含有确认用户对应用和数据库存取权限的功能以及记录系统处理日志的功能。功能层的程序多半是用可视化编程工具开发的，也有使用COBOL和C语言的。

**数据层**

数据库管理系统，负责管理对数据库数据的读写。数据库管理系统必须能迅速执行大量数据的更新和检索。因此，一般从功能层传送到数据层的要求大都使用SQL语言。

**三层C/S的解决方案是：**

对这三层进行明确分割，并在逻辑上使其独立。关键是要将表示层和功能层分离成各自独立的程序，并且还要使这两层间的接口简洁明了。

一般情况是只将表示层配置在客户机中

如果将功能层和数据层分别放在不同的服务器中，则服务器和服务器之间也要进行数据传送（如何通信？）。

由于在这种形态中三层是分别放在各自不同的硬件系统上的，所以灵活性很高，能够适应客户机数目的增加和处理负荷的变动。例如，在追加新业务处理时，可以相应增加装载功能层的服务器。因此，系统规模越大这种形态的优点就越显著。

# 正交软件体系结构

**特征：**

正交软件体系结构由完成不同的功能的n(n > 1)个线索（子系统）组成。

系统具有m(m>1)个不同抽象级别的层。

线索之间是相互独立的（正交的）。

系统由一个公共驱动层（一般为最高层）和公共数据结构（一般为最低层）。

优点：

结构清晰，易于理解。

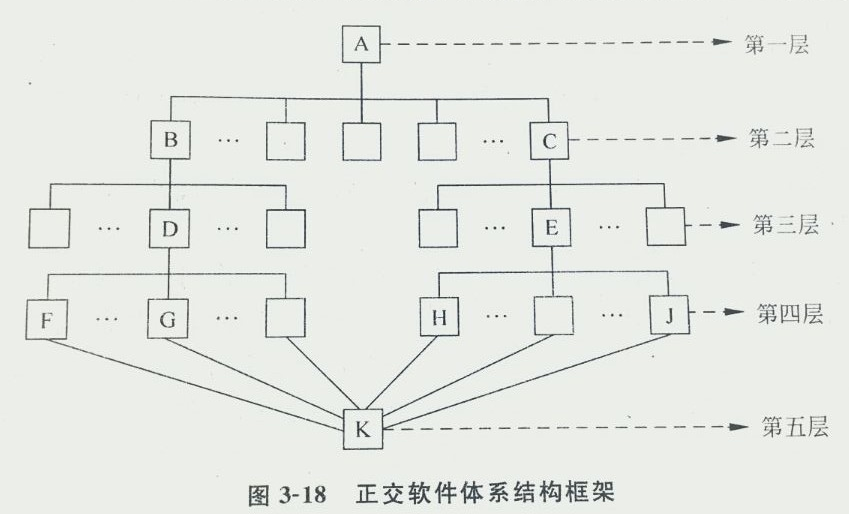
易修改，可维护性强。

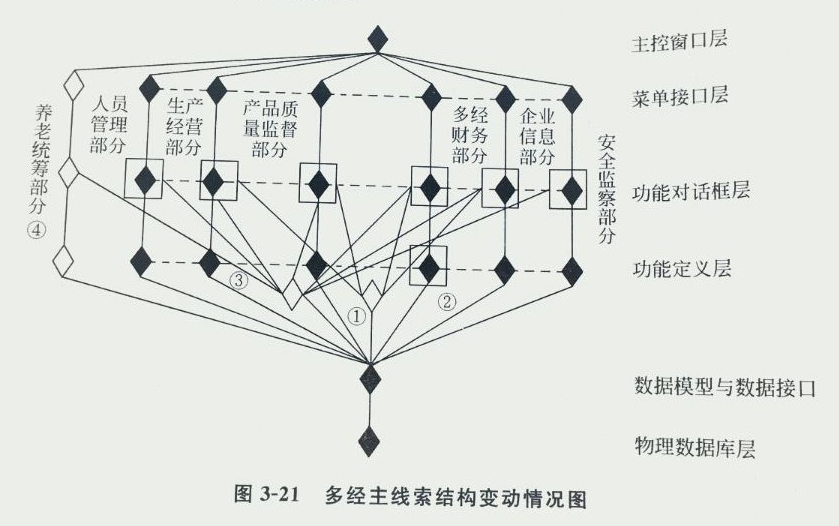
可移植性强，重用粒度大。

**什么时候使用：**

多级正交结构对于降低软件演化更新的开销是行之有效的，而且非常适合大型软件开发，特别在与MIS领域，由于其结构在一定应用领域内均有许多共同点，因此有一定的通用性。

**有什么样的结构：**

****

****

**7.5 SIMIS系统**

# ATAM评估步骤

1. 描述ATAM方法
2. 描述业务动机
3. 描述体系结构
4. 确定体系结构方法
5. 生成质量属性效用树
6. 分析体系结构方法
7. 讨论和分级场景
8. 分析体系结构方法
9. 描述评估结果