第一章 软件体系结构概论

1：软件重用：

在两次或多次不同的软件开发过程中重复使用相同或相近软件元素的过程。

※2：软件体系结构的定义：

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

（软件体系结构不仅指定了系统的组织结构和拓扑结构，并且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。）

3：软件危机的表现：（1）软件成本日益增长

（2）开发进度难以控制

（3）软件质量差

（4）软件维护困难

4：软件危机的原因：（1）用户需求不明确

（2）缺乏正确的理论指导

（3）软件规模越来越大

（4）软件复杂度越来越高

※5：构件：

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；结构上，它是语义描述、通信接口和实现代码的复合体。

6：构建组装：

（1）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）

（2）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）

（3）面向对象的的组装技术（构造法和子类法）

7：软件体系结构的意义：

（1）体系结构是风险承担者进行交流的手段

（2）体系结构是早期设计决策的体现

①软件体系结构明确了对系统实现的约束条件

②软件体系结构决定了开发和维护组织的组织结构

③软件体系结构制约着系统的质量属性

④通过研究软件体系结构可能预测软件的质量

⑤软件体系结构使推理和控制更改更简单

⑥软件体系结构有助于循序渐进的原型设计

⑦软件体系结构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

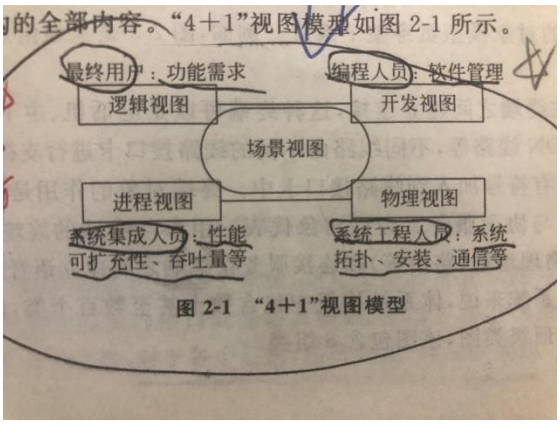
第二章 软件体系结构建模

1：软件体系结构的 5 种模型，最常用的是**结构模型**和**动态模型**

（1）结构模型。以体系结构的构件、连接件和其他概念来刻画结构，并力图通过结构来反映系统的重要语义内容，包括系统的配置、约束、隐含的条件、风格、性质等。研究结构模型的核心是体系结构描述的语言。
（2）框架模型。与结构模型类似，但不侧重描述结构的细节而更侧重于整体的结构。主要以一些特殊的问题为目标建立只针对和适应该问题的结构。
（3）动态模型。对结构或框架模型的补充，研究系统的“大颗粒”的行为性质。动态可以指系统总体结构的配置、建立或拆除通信通道或计算的过程。
（4）过程模型。研究构造系统的步骤和过程，因而结构是遵循某些过程脚本的结果。
（5）功能模型。有一组功能构件按层次组成，下层向上层提供服务。它可以看成是一种特殊的框架模型。

※2：“4+1”视图模型

4 个视图+1 个场景。该模型从 5 个不同的视角包括逻辑视图、进程视图、物理视图、开发视图和场景来描述软件体系结构。每个视图只关心系统的一个侧面，5 个视图结合在一起才能反映系统的软件体系结构的全部内容。

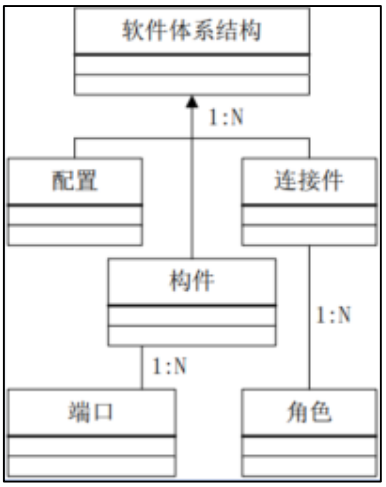


（P30 图）

逻辑视图（logic view）：主要支持系统的功能需求，即系统提供给最终用户的服务。主要图包括类图、序列图、协作图。（最终用户：功能需求）
开发视图（development view）：也称模块视图(module view)，主要侧重于软件模块的组织和管理。主要图包括构件图。（编程人员：软件管理）
进程视图（process view）:也称为并发视图，侧重于系统的运行特性，主要关注一些非功能性的需求，例如系统的性能和可用性。主要图包括活动图、交互图。（系统集成人员：性能可扩充性、吞吐量等）
物理视图（physical view）：主要考虑如何把软件映射到硬件上，它通常要考虑到系统性能、规模、可靠性等。解决系统拓扑结构、系统安装、通信等问题。主要图包括部署图。（系统工程人员：系统拓扑、安装、通信等）
场景（scenarios）：可以看作那些重要系统活动的抽象，它使 4 个视图有机联系起来，从某种意义上说场景是最重要的需求抽象。用例图。
注：逻辑视图和开发视图描述系统的静态结构，而进程视图和物理视图描述系统的动态结构。

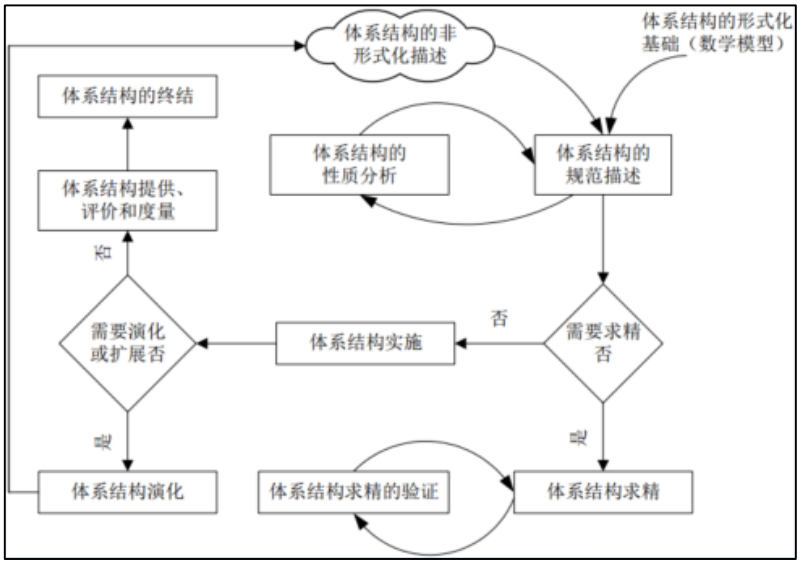
3：体系结构的核心模型由五种元素组成：构件、连接件、配置（configuration）、端口（port）和角色(role)。其中，构件、连接件和配置是最基本的元素。P36

（1）构件：具有某种功能的可重用的软件模块单元，表示了系统中主要的计算元素和数据存储。构件有两种：复合构件和原子构件，复合构件由其他复合构件和原子构件通过连接而成；原子构件是不可再分的构件，底层由实现该构件的类组成，这种构件的划分提供了体系结构的分层表示能力，有助于简化结构的设计。
（2）连接件：表示了构件之间的交互，简单的连接件如管道（pipe）、过程调用（procedure call）、事件广播（event broadcast）等，更为复杂的交互如客户-服务器（client-server）通信协议，数据库和应用之间的 SQL 连接等。
（3）配置：表示了构件和连接件的拓扑逻辑和约束。



构件作为一个封装的实体，只能通过其接口与外部环境的交互，**构件的接口由一组端口组成**，每个端口表示了构件和外部环境的交互点。
连接件作为建模软件体系结构的主要实体，同样也有接口，**连接件的接口由一组角色组成**。

4：软件体系结构的生命周期



※5：连接的特性

（1） 方向性

- 一般的连接中，通常伴有双向的信息交换

- 主控方和被控方
- 信息的发送方与接收方

（2） 角色：连接的双方所处的不同地位的表达
- 过程调用：调用方和被调用方
- 管道：读取方和写入方
- 消息传递：发送者和接受者

（3） 激发性：引起连接行为的方式，分为主动方和被动方的激发性
- 主动方：操作调用、事件触发
- 被动方：状态查询、事件触发
- 连接的发出方式：1:1 和 1:n

（4） 响应特性：包括连接的被动方对连接请求的处理实时性、时间、方式、并发处理能力等。
- 同步/异步
- 并发/互斥

第三章 软件体系结构风格

※1：软件体系结构风格定义：

软件体系结构风格是描述某一特定应用领域中系统组织方式的惯用模式。

体系结构风格定义了一个系统家族，即一个体系结构定义一个词汇表和一组约束。词汇表中包含一些构件和连接件类型，而这组约束指出系统是如何将这些构建和连接件组合起来的。

=>体系结构风格最关键的四要素：提供一个词汇表、定义一套配置规则、定义一套语义解释原则和定义对基于这种风格的系统进行的分析。

2：管道与过滤器：

（1） 每个构件都有一组输入和输出，构件读输入的数据流，经过内部处理，然后产生输出数据流。这个过程通常通过对输入流的变换及增量计算来完成，所以在输入被完全消费之前，输出便产生了。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。

（2） 过滤器必须是独立的实体，不能与其他的过滤器共享数据，而且一个过滤器不知道它上游和下游的标识。输出的正确性不依赖于过滤器进行增量计算的顺序。

（3） 典型应用：传统的编译器、Unix shell 中用户定义和启动多个任务进程来充当过滤器、图像处理、信号处理、音视频流处理。

（4） 管道-过滤器的类型有：管线、命名管道。管线是把过滤器严格限制为单输入、单输出的类型；命名管道是过滤器之间通过有名称的管道进行数据传送的系统。

（5） 不适合处理交互的应用。

※3：基于事件的隐式调用：

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。系统中其他构件中的过程在一个或多个事件中注册，当一个事件被出发，系统自动调用在这个事件中注册的所有过程，这样，一个事件的触发就导致了另一模块中的过程的调用。

该风格的主要特点是事件的触发者不知道哪些构件会被这些事件影响。许多隐式调用的系统也包含显式调用作为构件交互的补充形式。

- **※优点：**

（1） 为软件重用提供了强大的支持。当需要将一个构件加入现存系统时，只需将它注册到系统的事件中。

（2） 为改进系统带来了方便。当用一个构件代替另一个构件时，不会影响到其他构件的接口。

- **※缺点**

（1） 构件放弃了对系统计算的控制。

（2） 数据交换的问题。有些情况下必须依靠一个共享的仓库进行交互，此时全局性能和资源管理成了问题。

（3） 由于过程的语义必须依赖于被触发事件的上下文约束，因此，关于正确性的推理存在问题。

4：显式调用与隐式调用

（1） 显式调用：各个构件之间的互动是由显性调用函数或程序完成的，调用过程与次序是固定的、预先设定的。

（2） 隐式调用：软件更多地变成被动性系统，构件持续地与其所处的环境打交道，但并不知道确切的交互次序。

※5：黑板系统：P55

（1） 概念：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，这种知识库成为黑板知识库。

（2） 组成部分

- 知识源：包含参与问题求解的条件和执行的操作行为。条件对写入黑板的信息和求解进程的状态做出评价，条件满足时执行相应操作：产生新的假设或对黑板上数据结构的变换处理。
- 黑板数据结构：保存系统输入、问题求解的局部和中间结果，反应问题求解进程的状态的集合。
- 控制：负责监视黑板上信息和状态的变化，并根据变化决定采取的行动。

（3） 应用场合及功能

针对那些尚不存在确定解决方案的、从原始数据向高层结构转换的应用问题。传统上应用在复杂信号处理解释上，e.g.图像识别、语言和模式识别。

※6： C2 风格

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。

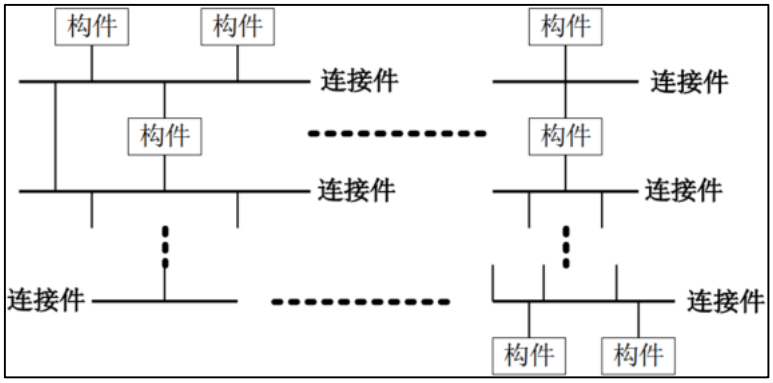
系统组织规则：

（1）系统中的构件和连接件都有一个顶部和一个底部。

（2）构件的顶部应连接到某连接件的底部，构件的底部则应连接到某连接件的顶部，而构件与构件之间的直接连接是不允许的。

（3）一个连接件可以和任意数目的其他构件和连接件连接。

（4）当两个连接件进行直接连接时，必须由其中一个的底部到另一个的顶部。



（C2 体系结构图）

C2 风格特点：

（1）系统中的构件可实现应用需求，并能将任意复杂度的功能封装在一起

（2）所有构件之间的通信是通过以连接件为中介的异步消息交换机制来实现的。

（3）构件相对独立，构件之间依赖性较少。系统中不存在某些构件将在同一地址空间内执行，或某些构件共享特定控制线程之类的相关性假设。

※7：C/S 体系结构（论述 C/S，三层 C/S，B/S 优缺点、实例）

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，使用繁杂，不利于推广使用。

（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

优点：

（1） C/S 体系结构具有强大的数据操作和事务处理能力，模型思想简单，易于人们理解和接受。

（2） 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，系统中每台服务器都可以适合各构件的要求，对于软硬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。

（3） 在 C/S 体系结构中，系统中的功能构件充分隔离，客户应用程序 的开发集中于数据的显示和分析，而数据库服务器的开发则集中于数据的管理，不必在每一个新的应用程序中都要对一个 DBMS 进行编码。将大的应用处理任务分布到许多通过网络连接的低成本计算机上，以节约大量费用。

实例：证券交易客户端等桌面应用程序

8：三层 C/S 结构风格

比起二层 C/S 结构，增加了一个<u>应用服务器</u>。

瘦客户机、胖服务器：整个应用逻辑驻留在应用服务器上，只有表示层在客户机上。

胖客户机、瘦服务器(2 层 C/S)：服务器负责数据管理，客户机完成与用户的交互任务。

C/S 将应用功能分为：表示层、功能层和数据层。

（1）表示层：是应用层的用户接口部分，它担负着用户与应用间的对话功能。

（2）功能层：相当于应用层的本体，它将具体的业务处理逻辑编入程序中。

（3）数据层：就是数据库管理系统，负责管理对数据库数据的读写。

9：三层 C/S 结构优缺点：

优点：

（1）允许合理的划分三层结构的功能，使逻辑上保持相对独立性，能够提高系统和软件的可维护性和可扩展性。

（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；并且这些平台和各组成部分可以具有良好的可升级性和开放性。

（3）应用各层可以并行开发，各层也可以选择各自最适合的开发语言。

（4）允许充分利用功能层有效地隔离开表示层与数据层，未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。

缺点：

（1） 三层 C/S 结构各层间的通信效率若不高,即使分配给各层的硬件能力很强，其作为整体来说也达不到所要求的性能。

（2） 设计时必须慎重考虑三层间的通信方法、通信频度及数据量。这和提高各层的独立性一样是三层 C/S 结构的关键问题。

实例：某国有特大型企业的劳动管理信息系统

10：B/S（也是三层应用结构：浏览器/Web 服务器/数据库服务器）：

优点：

（1） 基于 B/S 体系结构的软件，系统安装、修改和维护全在服务器端解决。用户在使用系统时，仅仅需要一个浏览器就可运行全部的模块，真正达到了“零客户端”的功能，很容易在运行时自动升级。

（2） B/S 体系结构还提供了异种机、异种网、异种应用服务的联机、联网、统一服务的最现实的开放性基础。

缺点：

（1）缺乏对动态页面的支持能力，没有集成有效的数据库处理功能。

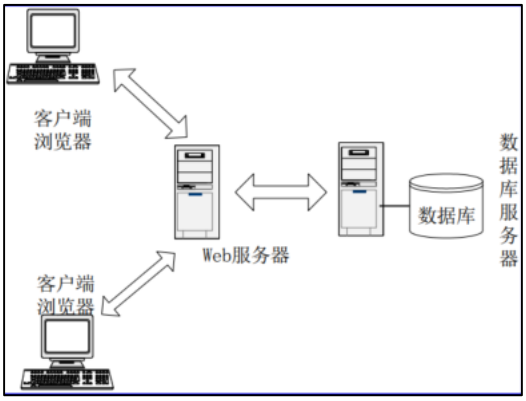
（2）系统扩展能力差，安全性难以控制。

（3）在数据查询等响应速度上，要远远低于 C/S 体系结构。

（4）数据提交一般以页面为单位，数据的动态交互性不强，不利于在线事务处理（OnLine Transaction Processing,OLTP）应用。

实例：淘宝网等网站

(C/S 结构的计算机应用系统网络负载较小)



（B/S 模式结构）

※11：正交软件体系结构

（1）概念

由组织层和线索的构件构成。层是由一组具有相同抽象级别的构件构成，线索由完成不同层次功能的构件组成。　（即：线索的纵向的，组织层是横向的）

如果线索相互独立（不同线索的构件不相互调用），则是完全正交结构。

（2）特征

- 由完成不同功能的 n 个线索组成；
- 系统具有 m 个不同抽象级别的层；
- 线索之间相互独立；
- 系统有一个公共驱动层(最高层)和公共数据结构(最低层)

（2）特征

- 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

※※12：异构结构风格（C/S 与 B/S 混合软件体系结构的两种模型，两种模型二选一画图并解释）

（1）※为什么要使用异构结构（可能简答）

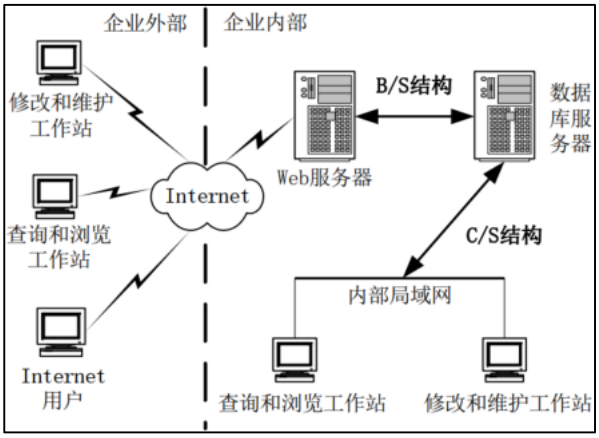
- 不同的结构有不同的处理能力的强项和弱点，一个系统的体系结构应该根据实际需要进行选择，以解决实际问题；
- 变动是绝对的；
- 遗留的代码不再重写它们；
- 即使规定了共享共同的软件包或相互关系的一些标准，仍会存在解释或表示习惯上的不同

（2）内外有别模型

内部的局域网通过客户端(C/S 结构)访问数据库服务器，外部通过浏览器(B/S 结构)访问 Web 服务器间接访问数据库服务器。内部用户交互性强，外部用户无法直接访问内部数据库服务器。

优点：外部用户不直接访问数据库服务器，能保证企业数据库的相对安全。企业内部用户的交互性较强，数据查询和修改的响应速度较快。

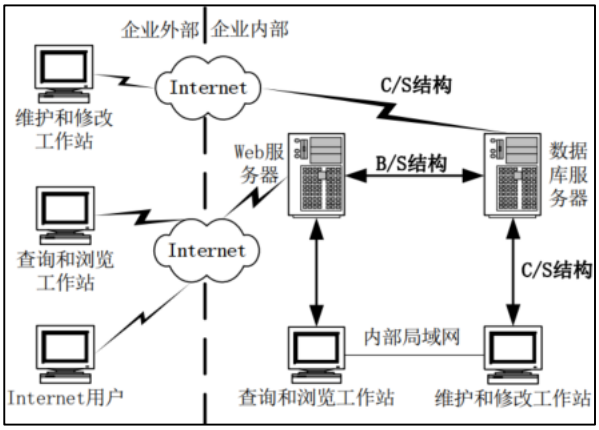
缺点：企业外部用户修改和维护数据时，速度较慢，较烦琐，数据的动态交互性不强。



（2）查改有别模型

内部和外部要对数据进行维护和修改都是通过 C/S 结构，要查询和浏览都是通过 B/S 结构。

该模型体现了 B/S 体系结构和 C/S 体系结构的共同优点。但因为外部用户直接通过 Internet 连接到数据库服务器，企业数据容易暴露给外部用户，给数据安全造成一定的威胁。



※13：特定领域软件体系结构 DSSA

（1）定义

DSSA 是在一个特定应用领域中为一组应用提供组织结构参考的标准软件体系结构。

※（2）两种角度理解 DSSA 中领域的含义：

- 垂直域：定义了一个特定的系统族，包含整个系统族内的多个系统，结果是在该领域中可作为系统的可行解决方案的一个通用软件体系结构。
- 水平域：定义了在多个系统和多个系统族只不过功能区域的共有部分，在子系统级上涵盖多个系统族的特定部分功能，无法为系统提供完整的通用体系结构。

（3）基本活动

- 领域分析=>获得领域模型
- 领域设计=>获得 DSSA
- 领域实现=>依据领域模型和 DSSA 开发和组织可重用信息

※（4）DSSA 与体系结构风格的比较

- DSSA 以问题域为出发点，体系结构风格以解决域为出发点；

- DSSA 只对某一个领域进行设计专家知识的提取、存储和组织，但可以同时使用多种体系结构风格；而在某个体系结构风格中进行体系结构设计专家知识的组织时，可以将提取的公共结构和设计方法扩展到多个应用领域。
- DSSA 通常选用一个或多个适合研究领域的体系结构的风格，并设计一个该领域专用的体系结构分析设计工具；
- 体系结构风格的定义和该风格的领域是直交的，提取的设计知识比用 DSSA 提取的设计专家知识的应用范围广泛；
- DSSA 和体系结构风格是互为补充的两种技术

14：进程通信（软件运行调度层）

（1）信号机制

信号是一种不携带任何数据信息的事件，该机制以传送简单信号的方式达到进程对相关进程的控制。

（2）信号检测

OS 定点定时检测

15：小结

主程序和子程序、面向对象系统是一切软件部件、连接设计和实现的基础，层次结构几乎是一切系统的基本结构方法，事件系统是一种受操作系统管理控制的部件连接方式，通信进程是部件的设计结构，管道/过滤器是特殊的系统结构方法。

第四章 软件体系结构描述

※1：软件体系结构描述语言 ADL

ADL 是在底层语义模型的支持下，为软件系统的概念体系结构建模提供了具体语法的概念框架。基于底层语义的工具为体系结构的表示、分析、演化、细化、设计过程等提供支持。其三个基本元素是：构件、连接件、体系结构配置。

※2：典型的软件体系结构描述语言-C2

在 C2 中，连接件负责构件之间消息的传递，而构件维持状态、执行操作并通过两个名字分别为“top”和“bottom”的端口和其他的构件交换信息。每个接口包含一种可发送的消息和一组可接收的消息。构件之间的消息交换不能直接进行，只能通过连接件来完成。连接件：构件=1：n。请求消息只能向上传送，通知消息只能向下传送。通知消息的传递只对应于构件内部的操作，而和接收消息的构件的需求无关。(重点：会议安排系统的 C2 风格描述中 P112,P113 中 C2 对 ImportantAttendee 构件的描述和对体系结构的描述)

第五章 动态软件体系结构

1：什么是动态软件体系结构

允许在系统运行时发生更新的软件体系结构称为动态软件体系结构，动态体系结构在系统被创建后可以被动态地更新。

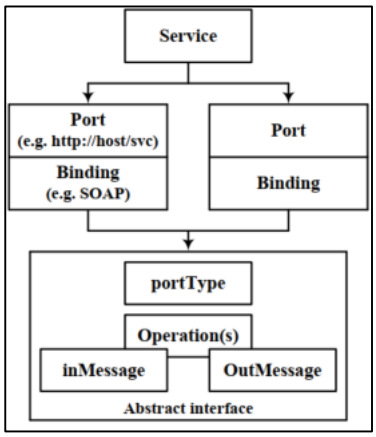
※2：动态体系特征及区别

（1）可构造性动态特征。可构造性动态特征通常可以通过结合动态描述语言、动态修改语言和一个动态更新系统来实现。
（2）适应性动态特征。适应性动态特征是基于一系列预定义配置而且所有事件在开发期间已经进行了评估，执行期间，动态改变是通过一些预定义的事件集触发，体系结构选择一个可选的配置来执行系统的重配置。
（3）智能性动态特征。智能性动态特征是用一个有限的预配置集来移除约束。
对比适应性体系结构特征，智能性动态特征改善了选择转变的功能，适应性特征是从一系列固定的配置中选择一个适应体系结构的配置，而智能性特征是包括了动态构造候选配置的功能。

第六章 Web 服务体系结构

※1：Web 服务描述语言 WSDL

WSDL 是一种 XML 格式，用于将网络服务描述为一组端点，这些端点对包含面向文档或面向过程信息的消息进行操作。



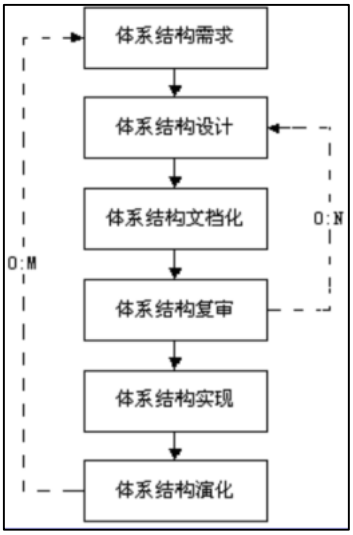
（WSDL 模型）

第七章 基于体系结构的软件开发

※1：设计模式

模式指从某个具体的形式中得到的一种抽象，在特殊的非任意性环境中，该形式不断地重复出现。一个软件体系结构的模式描述了一个出现在特定设计语境中的特殊的再现设计问题，并为它的解决方案提供了一个经过充分验证的通用图示。解决方案图示通过描述其组成构件及其责任和相互关系以及它们的协作方式来具体指定。
模式组成部分：模式名称、问题、解决方案和效果。

※2：基于体系结构的软件开发模型 ABSDM



第九章 软件体系结构评估

※1：SA 评估的主要方式及区别、举例

（1）基于调查问卷或检查表的评估方式。调查问卷是一系列可以应用到各种体系结构评估的相关问题，检查表也包含一些比问卷更细节和具体的问题，更趋向于考察某些关心的质量属性。这一评估方式比较自由灵活，可以评估多种质量属性，也可以在 SA 设计的多个阶段进行，但是评估结果很大程度上来自评估人员的主观推断，因此评估人员对领域的熟悉程度、经验丰富程度都将影响评估结果的正确性。E.g.体系结构的表示用的是何种 ADL，系统的核心功能是否与界面分开。

（2）基于场景的评估方式。这种评估方式分析软件体系结构对场景也就是对系统的使用或修改活动的支持程度，从而判断该体系结构对这一场景所代表的质量需求的满足程度。这一评估方式考虑到了包括系统的开发人员、维护人员、最终用户、管理人员、测试人员等在内的所有与系统相关的人员对质量的要求。基于场景的评估方式涉及到的基本活动包括确定应用领域的功能和软件体系结构的结构之间的映射，设计用于体现待评估质量属性的场景以及分析软件体系结构对场景的支持程度；不同的应用系统对同一质量属性的理解可能不同，由于存在这种不一致性，对一个领域适合的场景设计在另一个领域内未必合适，因此基于场景的评估方式是特定于领域的。这一评估方式的实施者一方面需要有丰富的领域知识以对某以质量需求设计出合理的场景，另一方面，必须对待评估的软件体系结构有一定的了解以准确判断它是否支持场景描述的一系列活动。E.g.用一系列对软件的修改来反映易修改性方面的需求，用一系列攻击性操作来代表安全性方面的需求等。

（3）基于度量的评估方式。度量是指为软件产品的某一属性所赋予的数值，传统的度量研究主要针对代码，但近年来也出现了一些针对高层设计的度量，软件体系结构度量即是其中之一。基于度量的评估方式提供更为客观和量化的质量评估。 这一评估方式需要在软件体系结构的设计基本完成以后才能进行，而且需要评估人员对待评估的体系结构十分了解，否则不能获取准确的度量。自动的软件体系结构度量获取工具能在一定程度上简化评估的难度，例如 MAISA 可从文本格式的 UML 图中抽取面向对象体系结构的度量。E.g.代码行数、方法调用层数、构件个数等。

评估方式	调查问卷或检查表		场景	度量
	调查问卷	检查表		
通用性	通用	特定领域	特定系统	通用或特定领域
评估者对体系结构的了解程度	粗略了解	无限制	中等了解	精确了解
实施阶段	早	中	中	中
客观性	主观	主观	较主观	较客观

※2：体系结构权衡分析方法 ATAM：

整个 ATAM 评估过程包括九个步骤：描述 ATAM 方法、描述商业动机、描述体系结构、确定体系结构方法、生成质量属性效用树、分析体系结构方法、讨论和分级场景、分析体系结构方法、描述评估结果。

※2：生成质量属性效用树(判断各个质量属性的重要程度**)**

（每个场景对系统成功与否的重要性，体系结构设计人员估计的实现难易程度）

判定方法：(H,H) > (H,M) > (M,H) > (M,M)

※3：为什么要评估软件体系结构？

软件体系结构在软件开发和管理中扮演着越来越重要的角色，软件体系结构设计对软件质量有着至关重要的影响，对最终确保系统的质量有重要的意义。软件体系结构评估，是对系统的某些值的关心的属性进行评估和判断。评估结果可用于确认潜在的风险，并检查设计阶段系统需求的质量，在系统被实际构造之前，预测其属性质量。

4：从哪些方面评估软件体系结构？

性能、可靠性、可用性、安全性、可修改性、功能性、可变性、可集成性、互操作性。

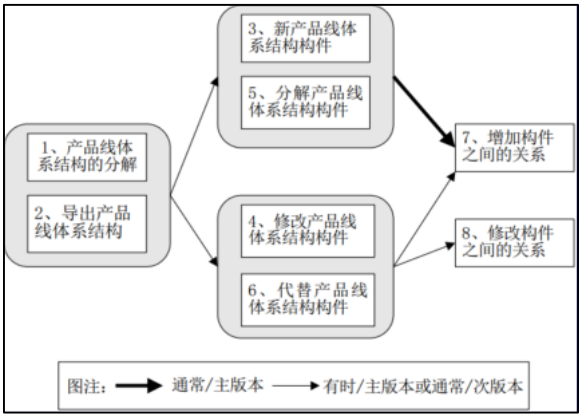
第十章 软件产品线体系结构

1：软件产品线

产品线是一个产品集合，共享一个公共的、可管理的特征集，满足选定的市场或任务领域的特定需求。这些系统遵循一个预描述的方式，在公共的核心资源基础上开发的。

软件产品线主要由两部分组成：核心资源、产品集合。核心资源是领域工程的所有结果的集合，是产品线中产品构造的基础。

2：需求和演化的分类-产品线体系结构分类之间的关系


1、产品线体系结构的分解
2、导出产品线体系结构
3、新产品线体系结构构件
5、分解产品线体系结构构件
4、修改产品线体系结构构件
6、代替产品线体系结构构件
7、增加构件之间的关系
8、修改构件之间的关系
图注：
通常/主版本
有时/主版本或通常/次版本