1

软件体系结构概论

1.4 体系结构的应用现状

- 体系结构分析、设计与验证（1）

体系结构分析的内容可分为**结构分析**、**功能分析**和**非功能分析**。

非功能分析：定量分析方法、推断分析方法。

Kazman等人提出了一种非功能分析的体系结构分析方法SAAM，并运用场景技术，提出了基于场景的体系结构分析方法，而Barbacci等人提出了多质量属性情况下的体系结构质量模型、分析与权衡方法ATAM。

- 体系结构分析、设计与验证（2）

生成一个满足软件需求的体系结构的过程即为**体系结构设计**。

体系结构设计过程的本质在于：将系统分解成相应的组成成分（如构件、连接件），并将这些成分重新组装成一个系统。

- 体系结构分析、设计与验证（3）

体系结构设计有两大类方法：**过程驱动方法**和**问题列表驱动方法。**

基于过程驱动的体系结构设计方法适用范围广，易于裁减，具备动态特点，通用性与实践性强。

问题列表驱动法的基本思想是**枚举**设计空间，并考虑设计维的相关性，以此来选择体系结构的风格。该方法适用于特定领域，是静态的，并可以实现量化体系结构设计空间。

- 体系结构分析、设计与验证（4）

体系结构设计研究的重点内容之一就是体系结构**风格**或**模式**，体系结构模式在本质上反映了一些特定的元素、按照特定的方式组成一个特定的结构，该结构应有利于上下文环境下的特定问题的解决。

- 体系结构分析、设计与验证（5）

体系结构模式分为两个大类：**固定术语**和**参考模型**。

已知的固定术语类的体系结构模型包括管道过滤器、客户/服务器、面向对象、黑板、分层、对等模式、状态转换、一些派生的固定术语类的体系结构模式，包括 Gen Voca，C2 和 REST 等；而参考模型则相对较多，常常与特定领域相关。

- 体系结构分析、设计与验证（6）

体系结构测试着重于仿真系统模型，解决体系结构层的主要问题。由于测试的抽象层次不同，体系结构测试策略可以分为**单元/子系统/集成/验收测试**等阶段的测试策略。

在体系结构集成测试阶段，Debra等人提出了一组针对体系结构的测试覆盖标准，Paola Inveradi 提出了一种基于 CHAM 的体系结构语义验证技术。

1.4 体系结构的应用现状

- 体系结构发现、演化与重用（1）

体系结构发现解决如何从已经存在的系统中提取软件的体系结构，属于**逆向工程**范畴。

Waters等人提出了一种**迭代式**体系结构发现过程，即由不同的人员对系统进行描述，然后对这些描述进行分类并融合，发现并解除冲突，将体系结构新属性加入到已有的体系结构模型中，并重复该过程直至体系结构描述充分。

- 体系结构发现、演化与重用（2）

由于系统需求、技术、环境、分布等因素的变化而最终导致软件体系结构的变动，称之为软件体系结构**演化**。

软件系统在运行时刻的体系结构变化称为体系结构的动态性，而将体系结构的静态修改称为体系结构扩展。体系结构扩展与体系结构动态性都是体系结构适应性和演化性的研究范畴。

1.4 体系结构的应用现状

- 体系结构发现、演化与重用（3）

体系结构重用属于设计重用，比代码重用更抽象。由于软件体系结构是系统的高层抽象，反映了系统的主要组成元素及其交互关系，因而较算法更稳定，更适合于重用。

体系结构模式就是体系结构重用研究的一个成果，而体系结构参考模型则是特定域软件体系结构的重用的成熟的象征。

1.4 体系结构的应用现状

- 基于体系结构的软件开发方法（1）

  在引入了体系结构的软件开发之后，应用系统的构造过程变为：
  “问题定义—>软件需求—>软件体系结构—>软件设计—>软件实现”
  可以认为软件体系结构架起了软件需求与软件设计之间的一座桥梁。

- 基于体系结构的软件开发方法（2）

  软件开发模型是跨越整个软件生存周期的系统开发、运行、维护所实施的全部工作和任务的结构框架，给出了软件开发活动各阶段之间的关系。

  目前，常见的软件开发模型大致可分为三种类型:

  - 以软件需求完全确定为前提的瀑布模型。
  - 在软件开发初始阶段只能提供基本需求时采用的渐进式开发模型，如螺旋模型等。
  - 以形式化开发方法为基础的变换模型。

- 基于体系结构的软件开发方法（3）

所有开发方法都是要解决需求与实现之间的差距。但是，这三种类型的软件开发模型都存在这样或那样的缺陷，不能很好地支持基于软件体系结构的开发过程。在基于构件和基于体系结构的软件开发逐渐成为主流情况下，已经出现了**基于构件的软件工程。**

但是对体系结构的描述、表示、设计和分析以及验证等内容的研究还相对不足，随着需求复杂化及其演化，切实可行的体系结构设计规则与方法将更为重要。

1.4 体系结构的应用现状

- 特定领域的体系结构框架

特定领域的体系结构是将体系结构理论应用到具体领域的过程，常见的DSSA有： CASE体系结构、CAD软件的参考模型、信息系统的参考体系结构、网络体系结构DSSA、机场信息系统的体系结构和信息处理DSSA等。国内学者提出的DSSA有：北京邮电大学周莹新博士提出的电信软件的体系结构，北京航空航天大学金茂忠教授等人提出的测试环境的体系结构等。

- 软件体系结构支持工具

几乎每种体系结构都有相应的支持工具，如Unicon，Aesop等体系结构支持环境，C2的支持环境ArchStudio，支持主动连接件的Tracer工具等。

支持体系结构分析的工具，如支持静态分析的工具、支持类型检查的工具、支持体系结构层次依赖分析的工具、支持体系结构动态特性仿真工具、体系结构性能仿真工具等。

1.4 体系结构的应用现状

- 软件产品线体系结构（1）

产品线代表着一组具有公共的系统需求集的软件系统，它们都是根据基本的用户需求对标准的产品线构架进行定制，将可重用构件与系统独有的部分集成而得到的。

软件产品线是一个十分适合专业的软件开发组织的软件开发方法，能有效地提高软件生产率和质量、缩短开发时间、降低总开发成本。

- 软件产品线体系结构（2）

软件体系结构有利于形成完整的软件产品线。

体系结构在软件产品线的开发中具有至关重要的作用，在这种开发生产中，基于同一个软件体系结构，可以创建具有不同功能的多个系统。

1.4 体系结构的应用现状

- 建立评价软件体系结构的方法

  目前，常用的三个软件体系结构评估方法是：
  - 体系结构权衡分析方法 （ATAM方法）
  - 软件体系结构分析方法 （SAAM方法）
  - 中间设计的积极评审 （ARID方法）

目前，软件体系结构尚处在迅速发展之中，越来越多的研究人员正在把注意力投向软件体系结构的研究。关于软件体系结构的研究工作主要在国外展开的，国内到目前为止对于软件体系结构的研究尚处在起步阶段。软件体系结构在国内未引起人们广泛注意的原因主要有两点：

（1）软件体系结构从表面上看起来是一个老话题，似乎没有新东西。

（2）与国外相比，国内对大型和超大型复杂软件系统开发的经历相对较少，对软件危机的灾难性体会没有国外深刻，因而对软件体系结构研究的重要性和必要性的认识还不很充分。

1.4 体系结构的应用现状

- 本章作业与思考题
  1. 根据自己的经验，谈谈对软件危机的看法。
  2. 就项目管理方面而言，软件重用项目与非重用项目有哪些不同之处。
  3. 实际参与/组织一个软件重用项目的开发，然后总结你是如何组织该项目的开发的。
  4. 为什么要研究软件体系结构?
  5. 根据软件体系结构的定义，你认为软件体系结构的模型应该由哪些部分组成?
  6. 在软件体系结构的研究和应用中，你认为还有哪些不足之处?

2

软件体系结构建模

2.1 软件体系结构建模概述

- 软件体系结构建模的种类
  - 结构模型
  - 框架模型
  - 动态模型
  - 过程模型
  - 功能模型

- 结构模型

这是一个最直观、最普遍的建模方法。这种方法以体系结构的构件、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、隐含的假设条件、风格、性质等。

研究结构模型的核心是体系结构描述语言。

- 框架模型

框架模型与结构模型类似，但它不太侧重描述结构的细节而更侧重于整体的结构。框架模型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。

- 动态模型

动态模型是对结构或框架模型的补充，研究系统的“大颗粒”的行为性质。例如，描述系统的重新配置或演化。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。

- 过程模型

过程模型研究构造系统的步骤和过程。

结构是遵循某些过程脚本的结果。

- 功能模型

  功能模型认为体系结构是由一组功能构件按层次组成，下层向上层提供服务。功能模型可以看作是一种特殊的框架模型。

2.2 “4+1”视图模型

- “4+1”模型概述

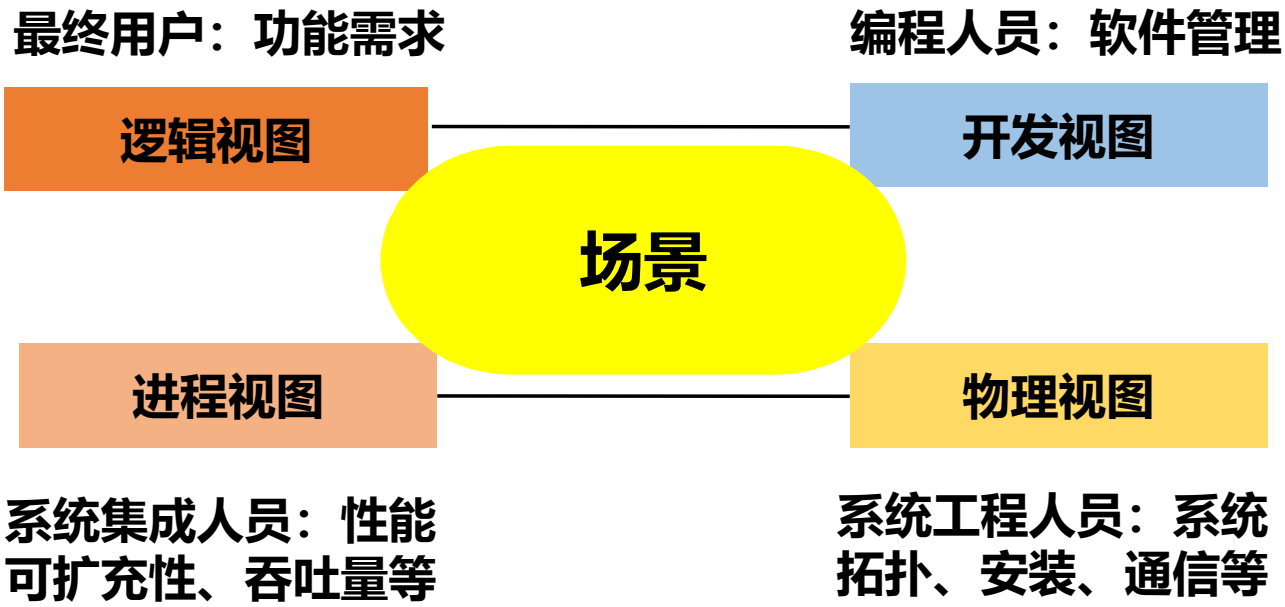
Kruchten在1995年提出了“4+1”的视图模型。

“4+1”视图模型从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**场景视图**来描述软件体系结构。

每一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。

2.2 “4+1”视图模型

- “4+1”模型概述



2.2 “4+1”视图模型

- 逻辑视图

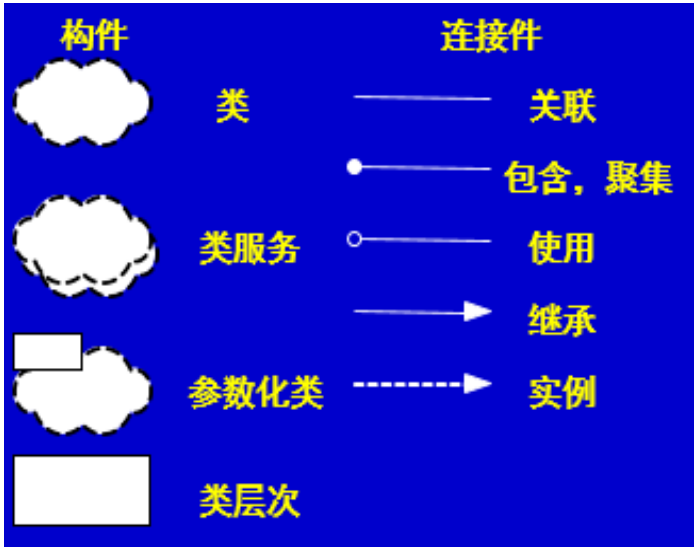
逻辑视图主要支持系统的功能需求，即系统提供给最终用户的服务。在逻辑视图中，系统分解成一系列的功能抽象，这些抽象主要来自问题领域。这种分解不但可以用来进行功能分析，而且可用作标识在整个系统的各个不同部分的通用机制和设计元素。

在面向对象技术中，通过抽象、封装和继承，可以用对象模型来代表逻辑视图，用类图来描述逻辑视图。

2.2 “4+1”视图模型

- 逻辑视图

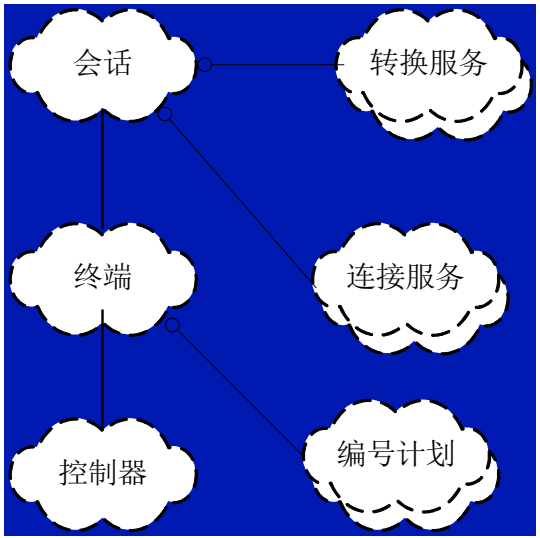
可以从 Booch 标记法中导出逻辑视图的标记法，只是从体系结构级的范畴来考虑这些符号，用 Rational Rose 进行体系结构设计。



2.2 “4+1”视图模型

- 逻辑视图

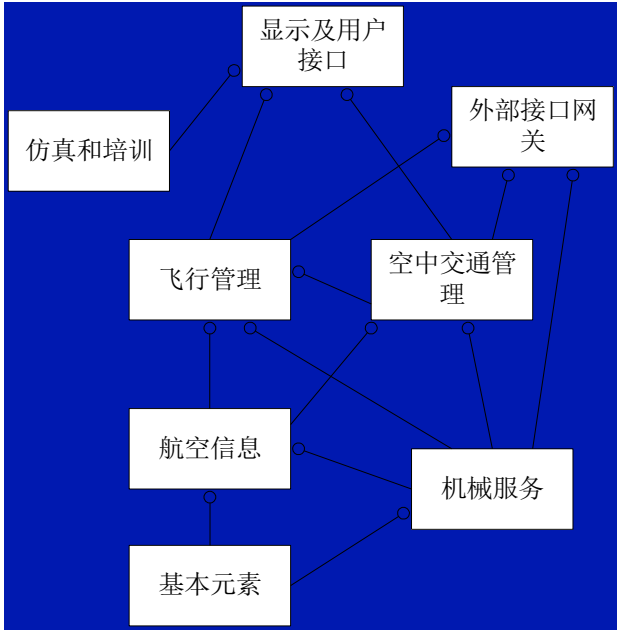
逻辑视图中使用的风格为面向对象的风格，逻辑视图设计中要注意的主要问题是要保持一个单一的、内聚的对象模型贯穿整个系统。



2.2 “4+1”视图模型

- 逻辑视图

对于规模更大的系统来说，体系结构级中包含数十甚至数百个类 。



2.2 “4+1”视图模型

- 开发视图

开发视图也称模块视图，主要侧重于软件模块的组织和管理。

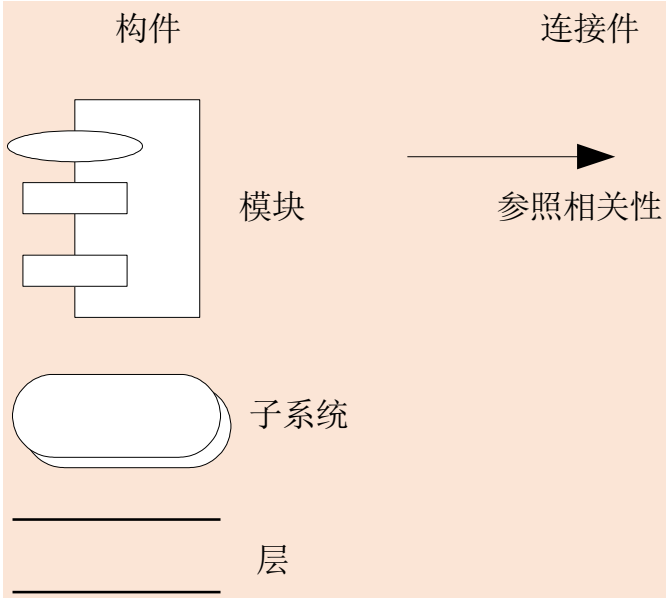
开发视图要考虑软件内部的需求，如软件开发的容易性、软件的重用和软件的通用性，要充分考虑由于具体开发工具的不同而带来的局限性。

开发视图通过系统输入输出关系的模型图和子系统图来描述。

2.2 “4+1”视图模型

- 开发视图

  与逻辑视图一样，可以使用 Booch 标记法中某些符号来表示开发视图。



2.2 “4+1”视图模型

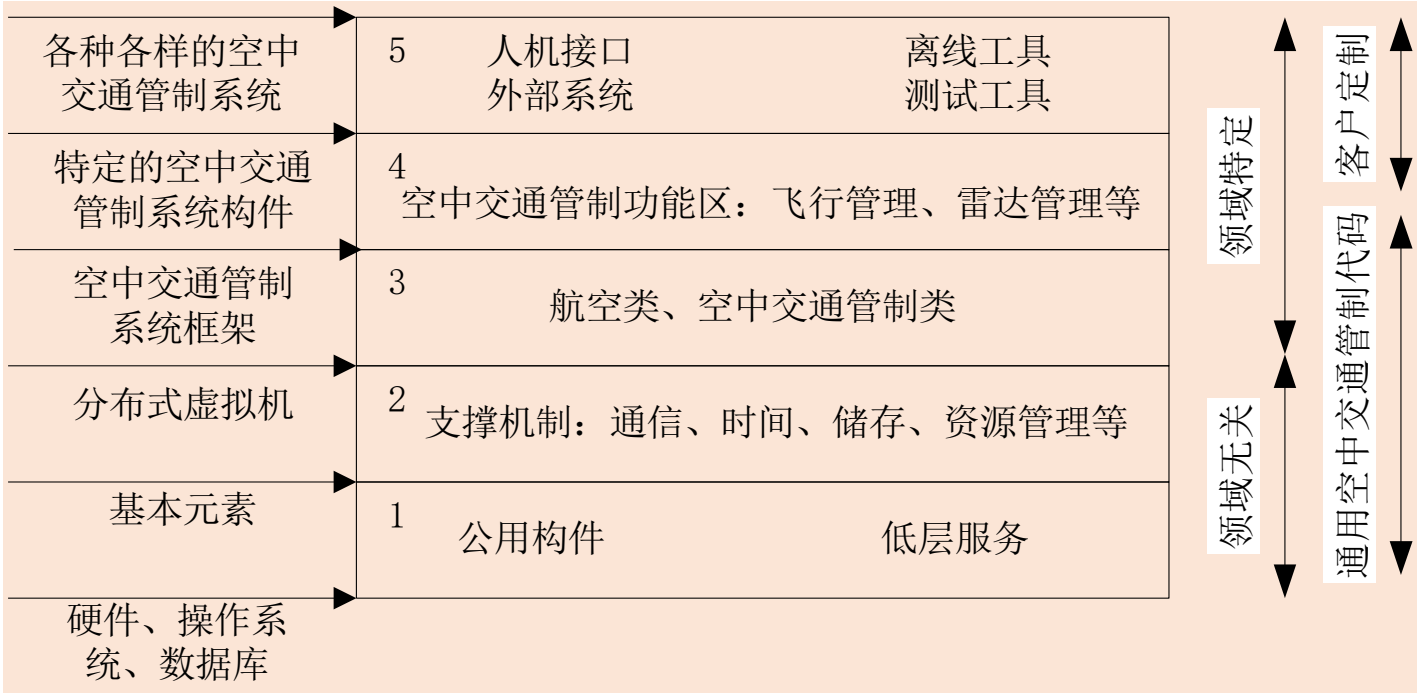
- 开发视图

在开发视图中，最好采用4-6层子系统，而且每个子系统仅仅能与同层或更低层的子系统通讯，这样可以使每个层次的接口既完备又精练，避免了各个模块之间很复杂的依赖关系。

设计时要充分考虑，对于各个层次，层次越低，通用性越强，这样，可以保证应用程序的需求发生改变时，所做的改动最小。开发视图所用的风格通常是层次结构风格。

2.2 “4+1”视图模型

- 开发视图



2.2 “4+1”视图模型

- 进程视图

进程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。

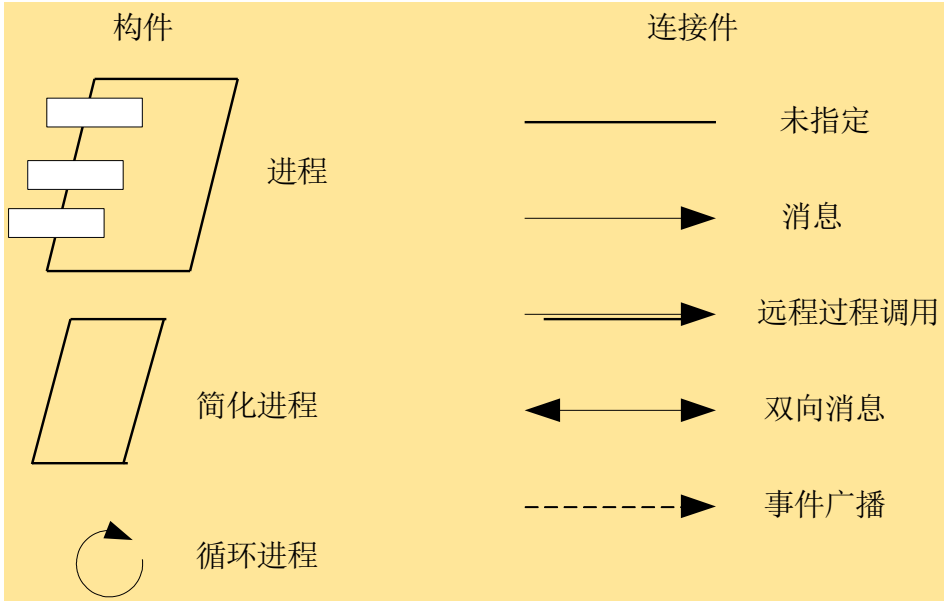
进程视图强调并发性、分布性、系统集成性和容错能力，以及从逻辑视图中的主要抽象如何适合进程结构。它也定义逻辑视图中的各个类的操作具体是在哪一个线程中被执行的。

进程视图可以描述成多层抽象，每个级别分别关注不同的方面。在最高层抽象中，进程结构可以看作是构成一个执行单元的一组任务。它可看成一系列独立的，通过逻辑网络相互通信的程序。它们是分布的，通过总线或局域网、广域网等硬件资源连接起来。

2.2 “4+1”视图模型

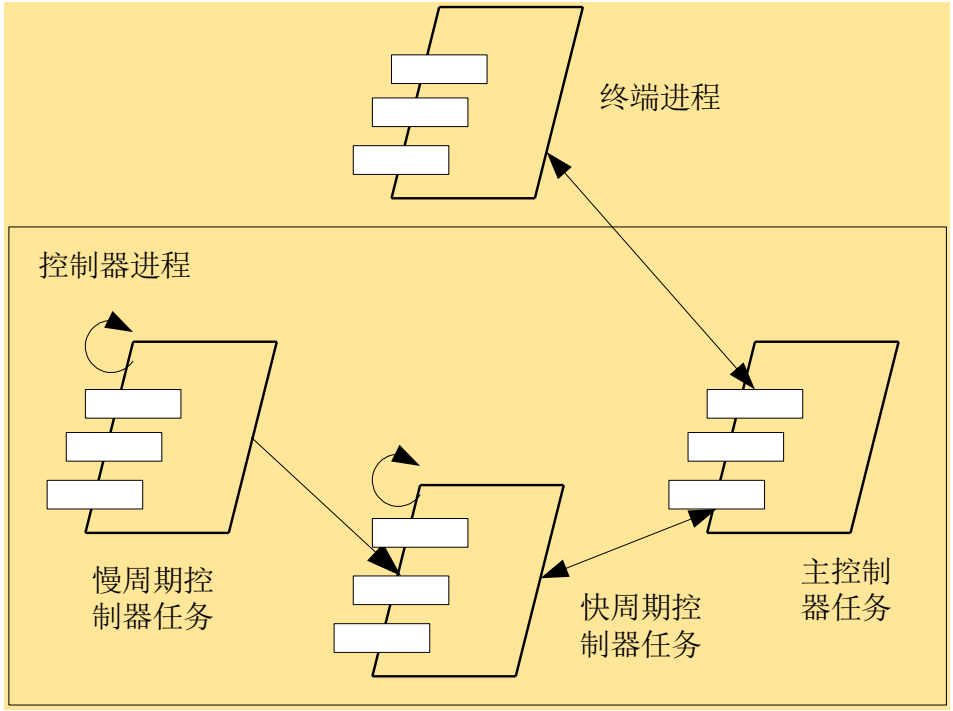
- 进程视图

  通过扩展 Booch 对 Ada 任务的表示法，来表示进程视图。



2.2 “4+1”视图模型

- 进程视图



2.2 “4+1”视图模型

- 物理视图

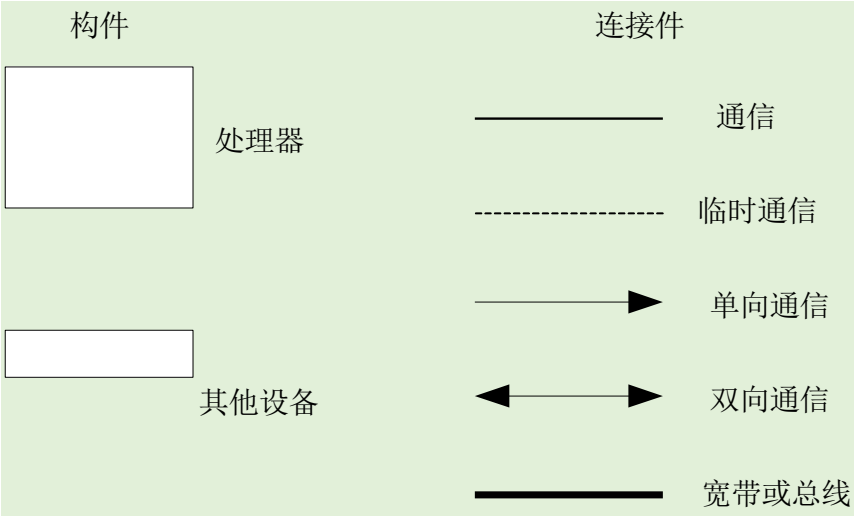
物理视图主要考虑如何把软件映射到硬件上，它通常要考虑到系统性能、规模、可靠性等。解决系统拓扑结构、系统安装、通讯等问题。

当软件运行于不同的节点上时，各视图中的构件都直接或间接地对应于系统的不同节点上。因此，从软件到节点的映射要有较高的灵活性，当环境改变时，对系统其他视图的影响最小。

2.2 “4+1”视图模型

- 物理视图

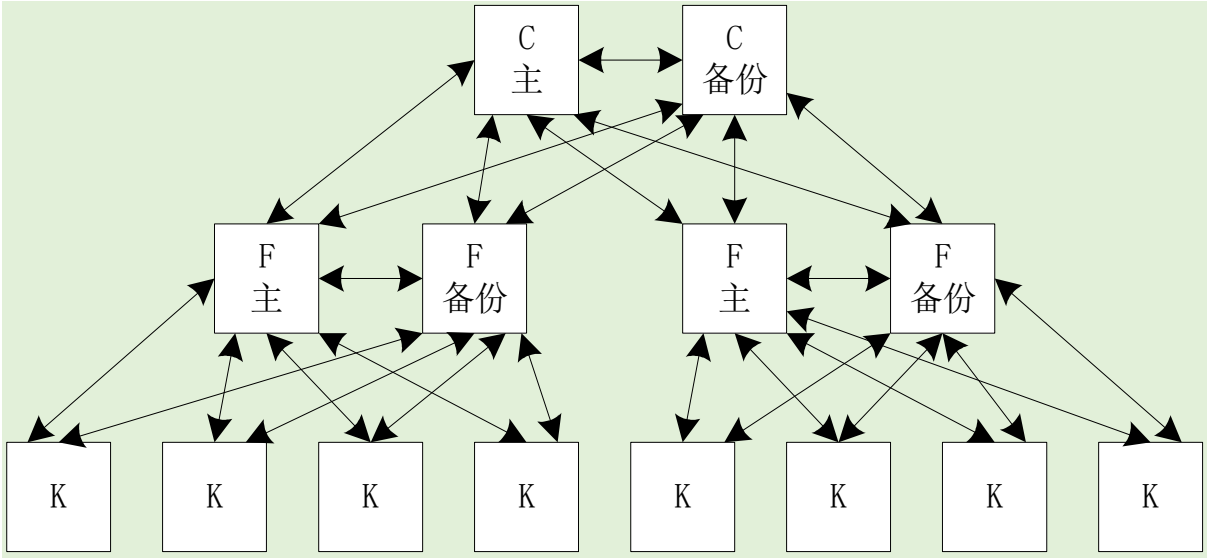
大型系统的物理视图可能会变得十分混乱，因此可以与进程视图的映射一道，以多种形式出现，也可单独出现。



2.2 “4+1”视图模型

- 物理视图

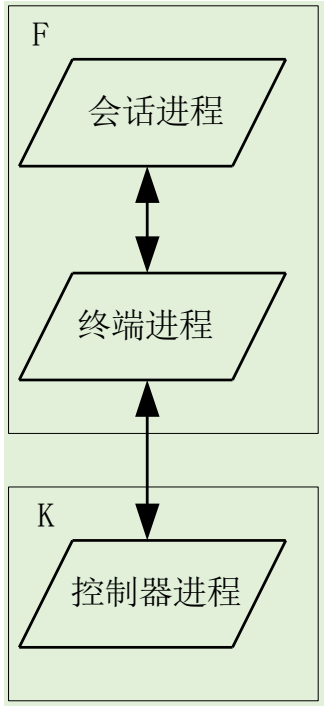
ACS系统的物理视图



2.2 “4+1”视图模型

- 物理视图

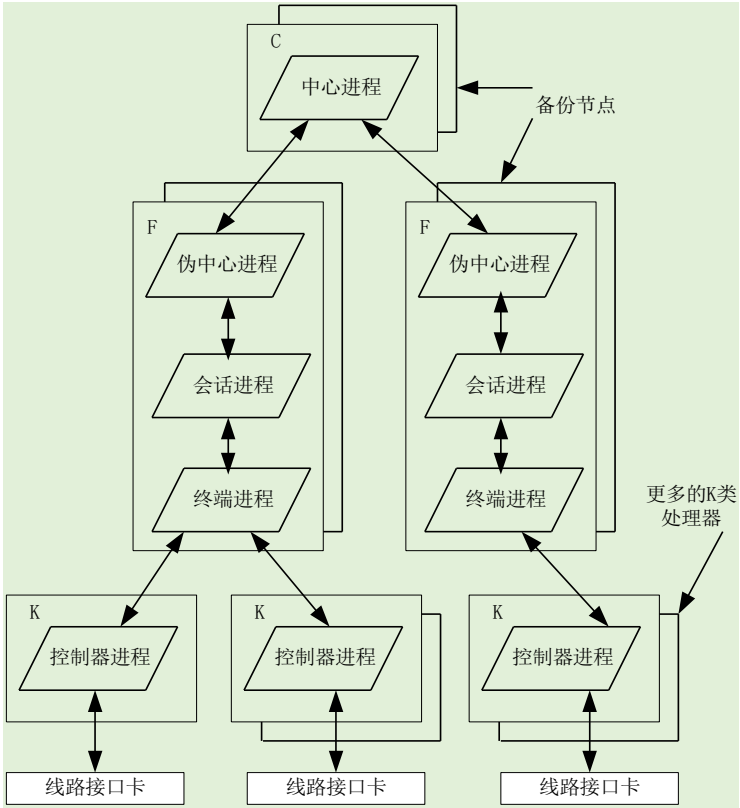
具有进程分配的小型ACS系统的物理视图



2.2 “4+1”视图模型

- 物理视图

具有进程分配的大型ACS系统的物理视图



2.2 “4+1”视图模型

- 场景

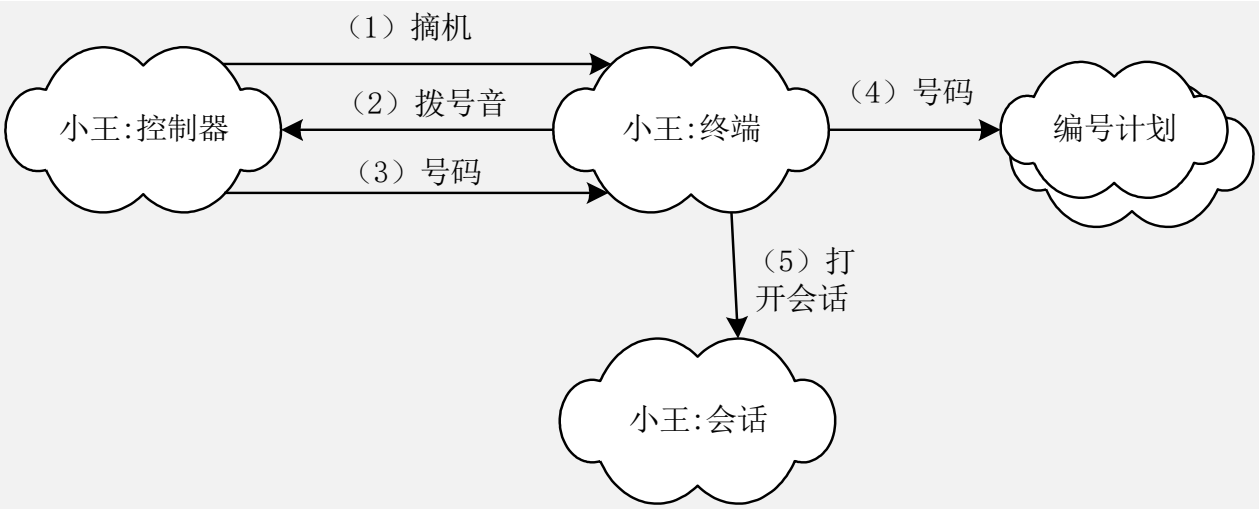
场景可以看作是那些重要系统活动的抽象，它使四个视图有机联系起来，从某种意义上说场景是最重要的需求抽象。在开发体系结构时，它可以帮助设计者找到体系结构的构件和它们之间的作用关系。同时，也可以用场景来分析一个特定的视图，或描述不同视图构件间是如何相互作用的。

场景可以用文本表示，也可以用图形表示。

2.2 “4+1”视图模型

- 场景

本地呼叫场景的一个原型



2.2 “4+1”视图模型

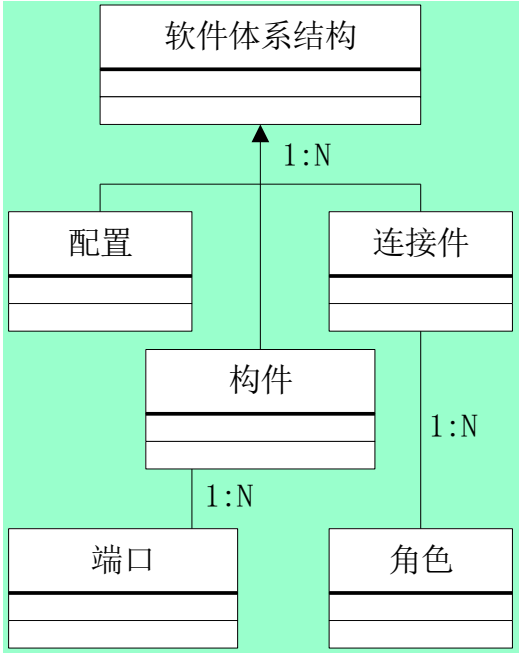
- 小结

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。

对于不同的软件系统来说，侧重的角度也有所不同。例如，对于管理信息系统来说，比较侧重于从逻辑视图和开发视图来描述系统，而对于实时控制系统来说，则比较注重于从进程视图和物理视图来描述系统。

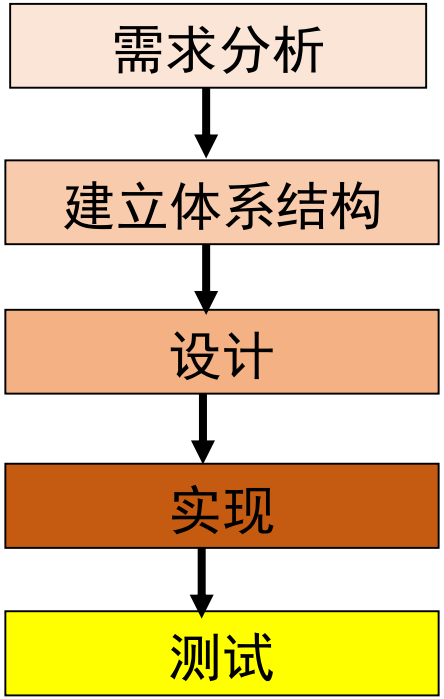
2.3 体系结构的核心模型

- 体系结构的核心模型



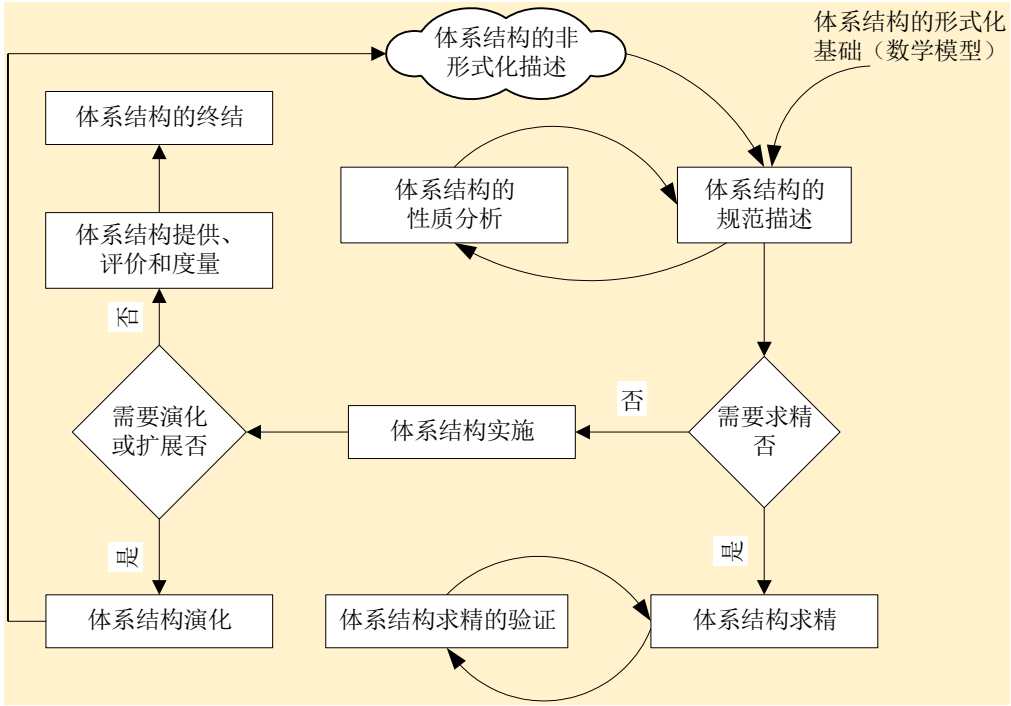
2.3 体系结构的核心模型

- 软件过程



2.3 体系结构的核心模型

- 生命周期模型



2.3 体系结构的核心模型

- 本章作业与思考题

1. 选择一个规模合适的系统，为其建立“4+1”模型。
2. 引入了软件体系结构以后，传统软件过程发生了哪些变化？这种变化有什么好处？
3. 软件体系结构的生命周期模型与软件生命周期模型有什么关系？

End of Session
Thank you!