软件体系结构原理、方法与实践

新的开始，新的起点，让我们一起为梦想而努力。

模型-理解

文字描述

图形描述

- ✓ 交通图
- ✓ 气象图
- ✓ 行政区划图

静态模型
动态模型

新的开始，新的起点，让我们一起为梦想而努力。

模型-理解

对于软件体系结构的建模按照侧重点不同分为5种，在这五种模型中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，连接件及概念来刻画结构，通过结果反映系统的重要予以内容，包括系统的配置、约束、假设条件等。

② 框架模型：与结构模型类似，但是他不侧重于结构细节的描述，更侧重于整体结构的描述。

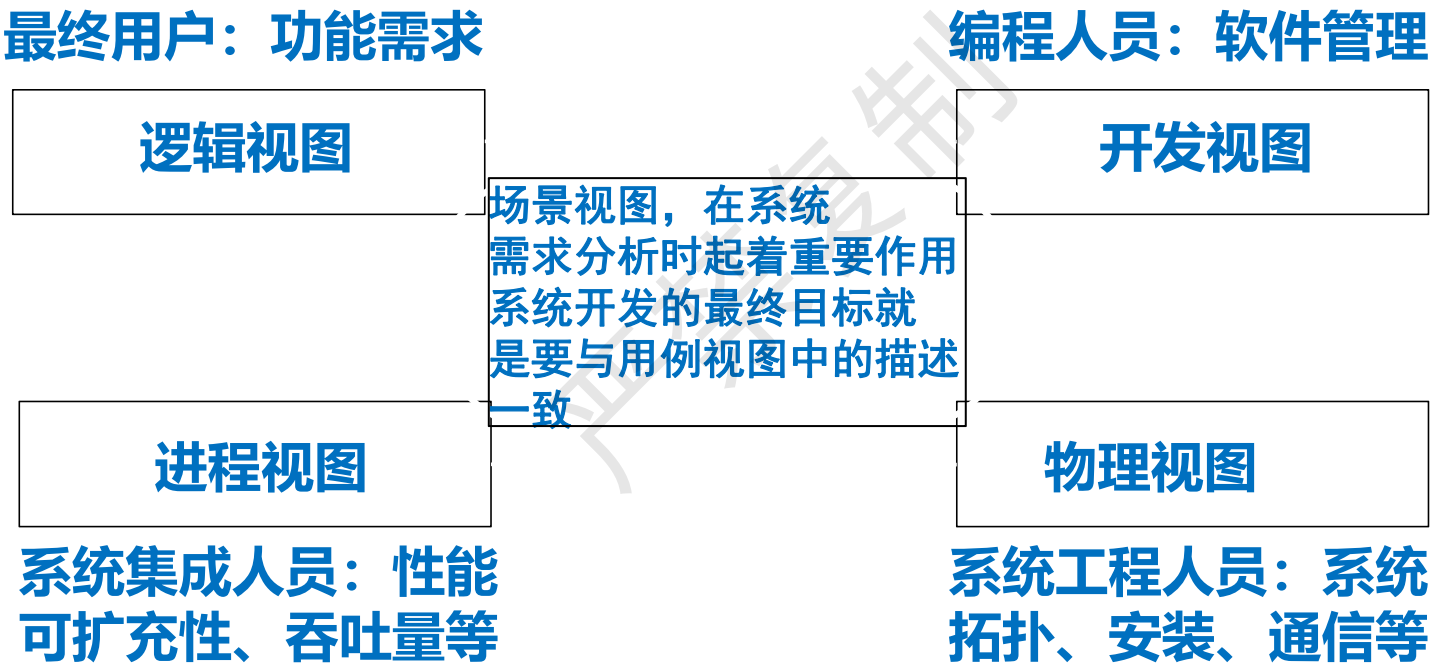
③ 动态模型：是对结构模型和框架模型的补充，研究系统大颗粒行为性质，如系统的重新配置或演化。

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：功能构件按层次组成，下层向上提供服务。

4+1模型 – 概述

- Kruchten在1995年提出了4+1的视图模型。

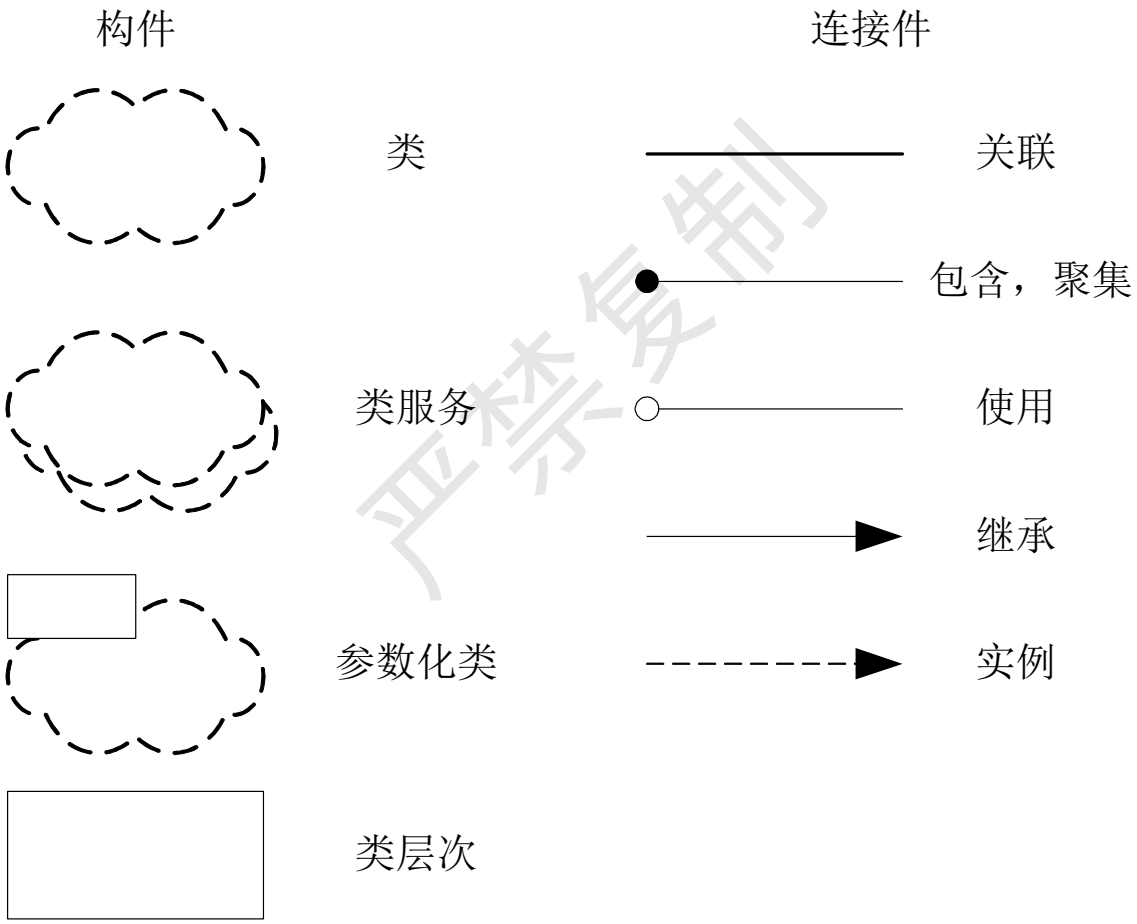


新的开始，新的起点，让我们一起为梦想而努力。

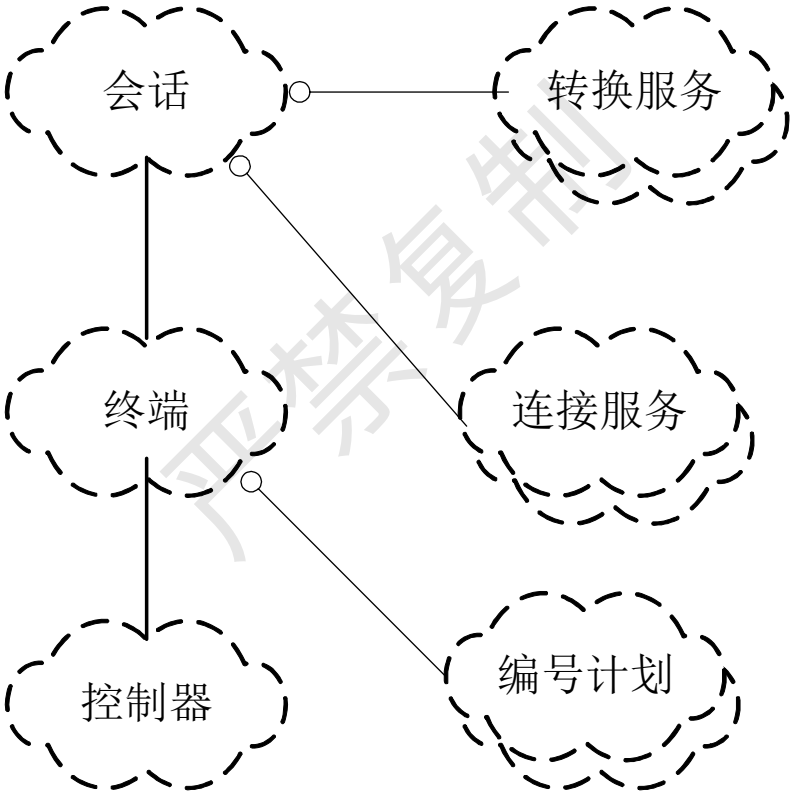
4+1模型 – 逻辑视图

- 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，逻辑视图使用面向对象的风格。

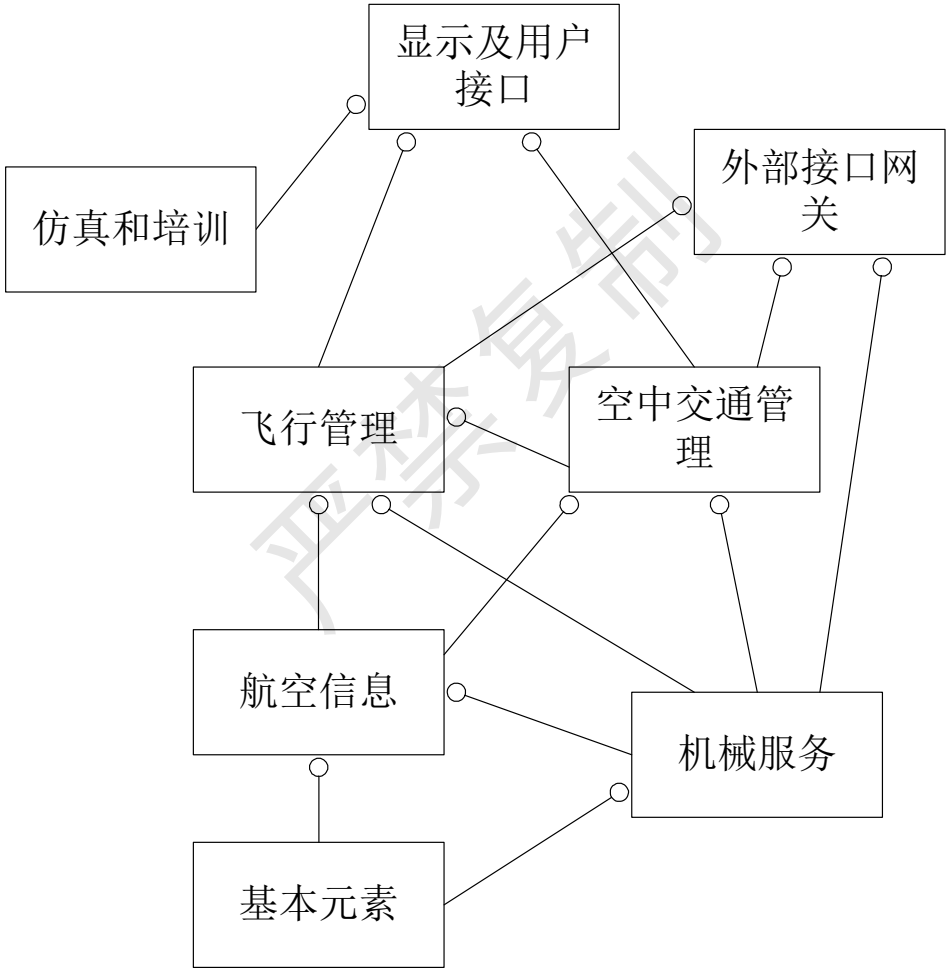
4+1模型 – 逻辑视图



4+1模型 – 逻辑视图



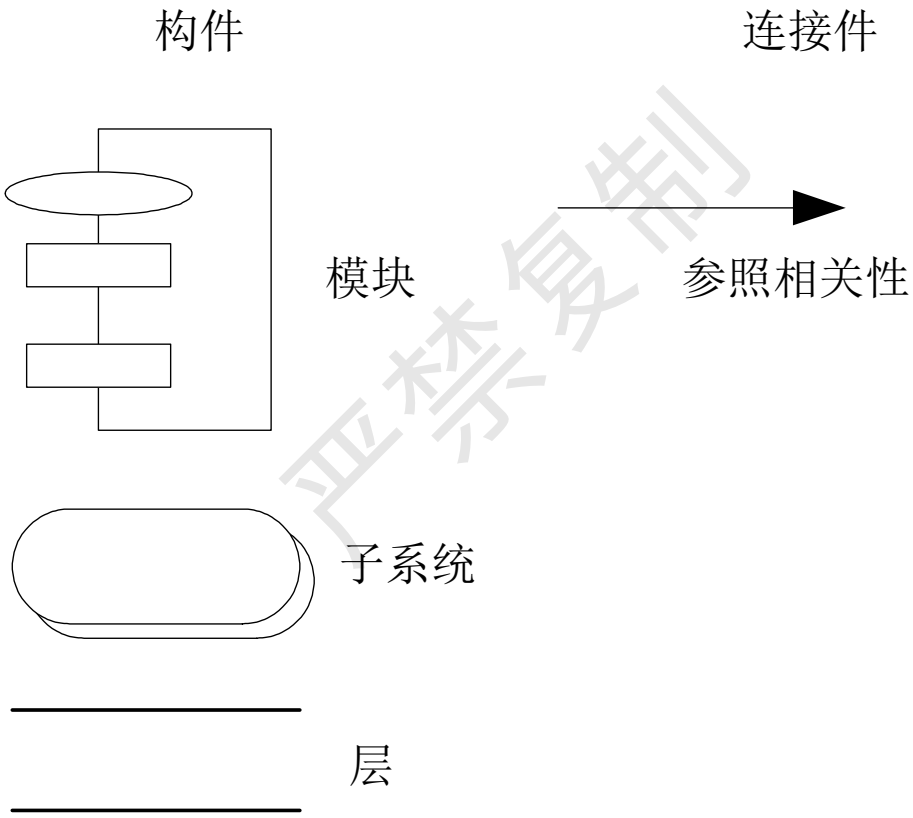
4+1模型 – 逻辑视图



4+1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。

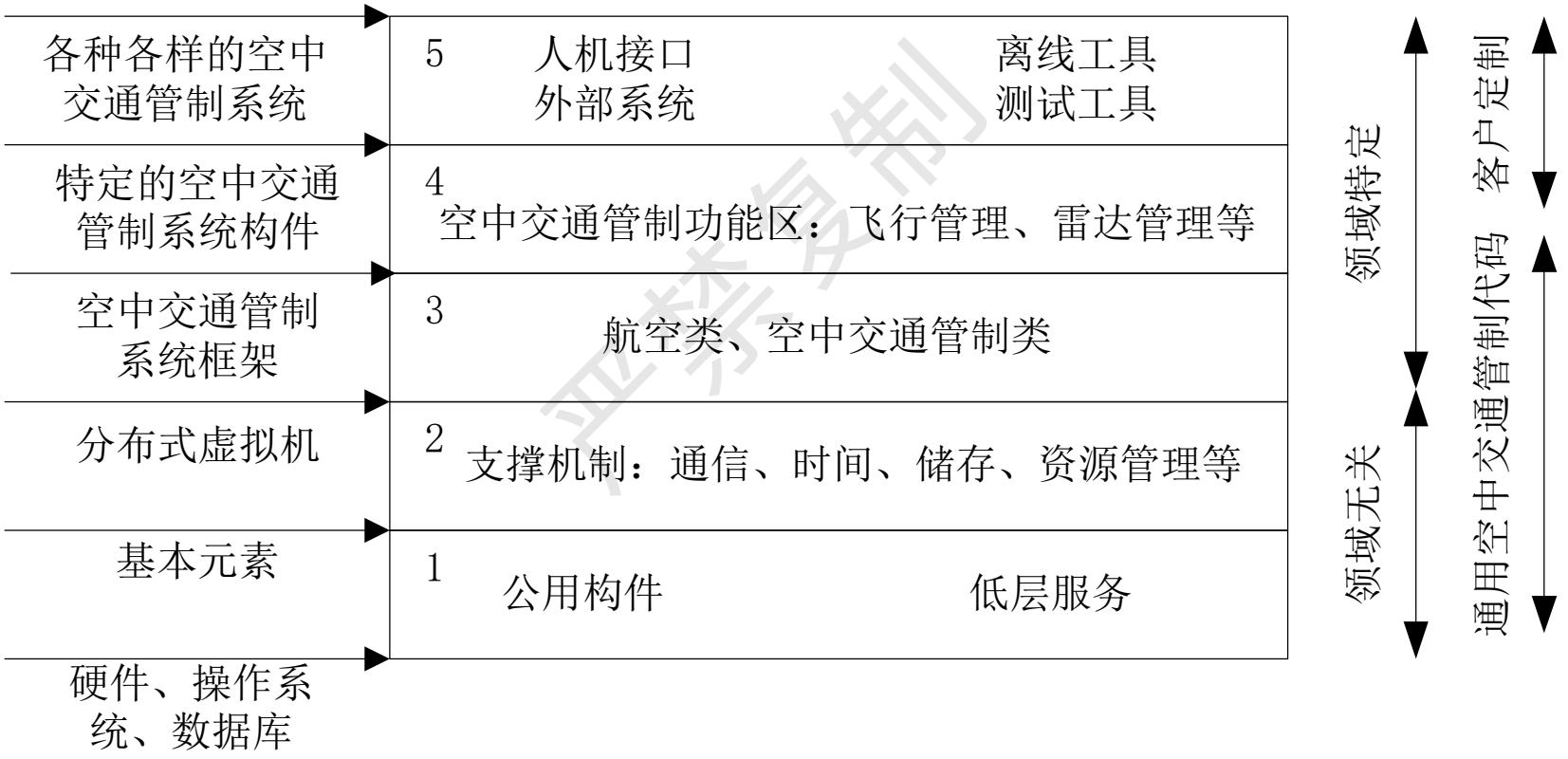
4+1模型 – 开发视图



4+1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。

4+1模型 – 开发视图

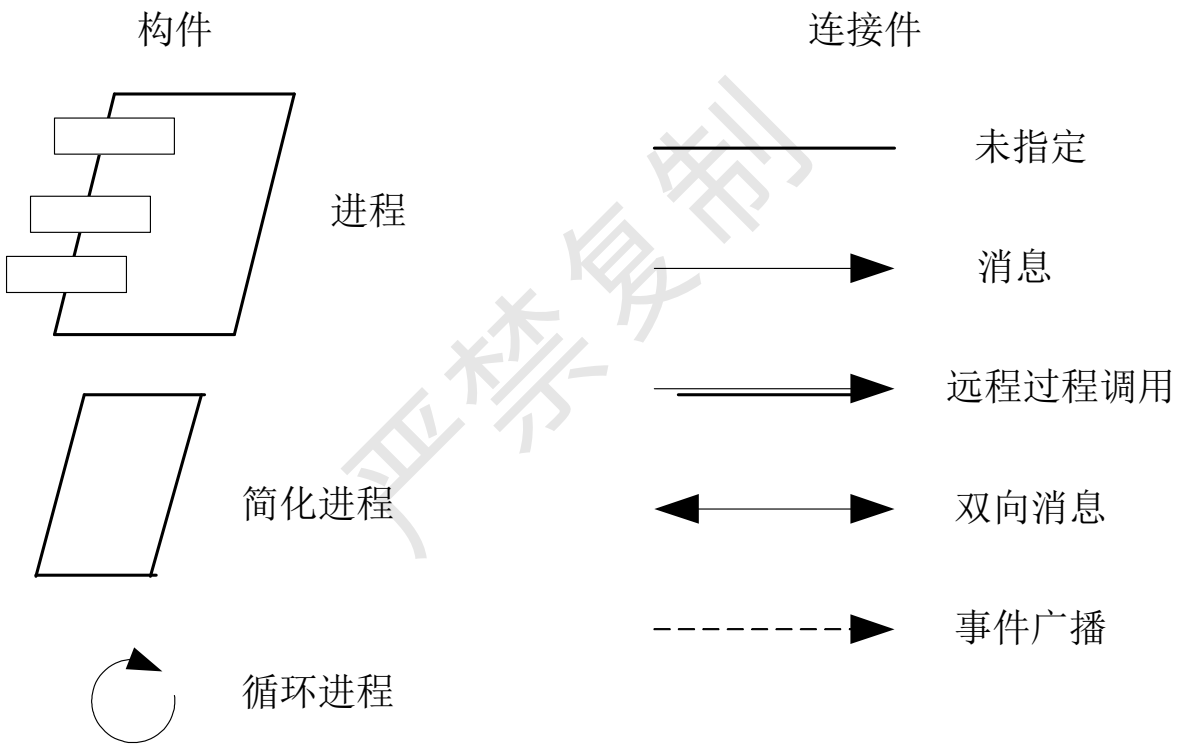


新的开始，新的起点，让我们一起为梦想而努力。

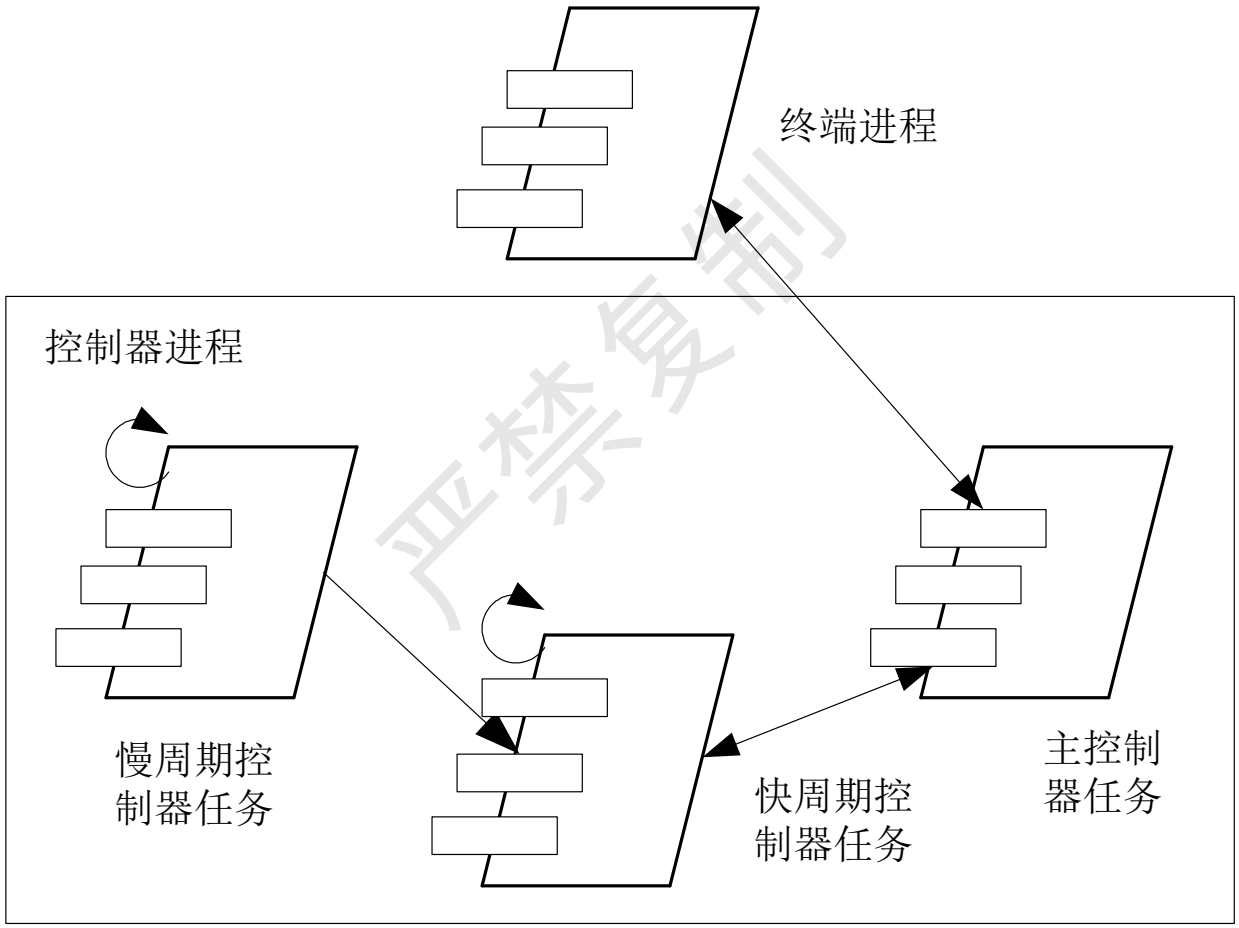
4+1模型 – 进程视图

- 进程视图也称为并发视图，侧重于系统的运行特性，主要关注一些非功能性的需求。
- 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。

4+1模型 – 进程视图



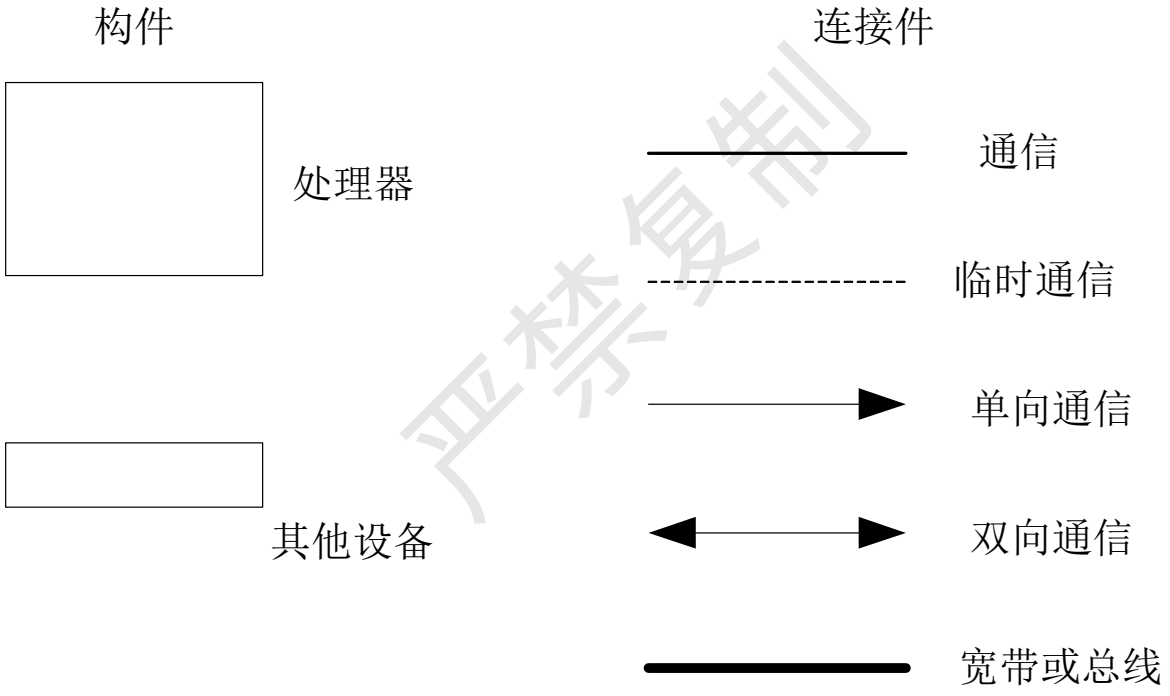
4+1模型 – 进程视图



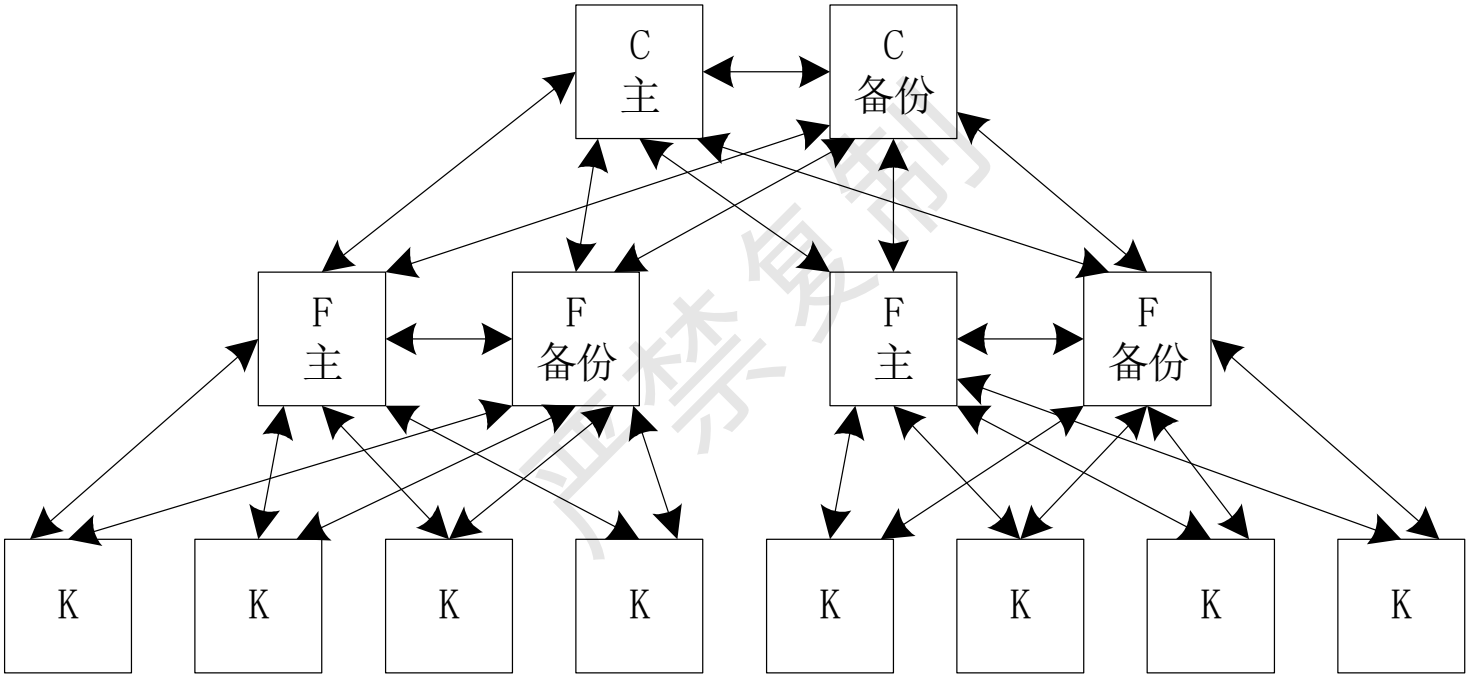
4+1模型 – 物理视图

- 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。

4+1模型 – 物理视图

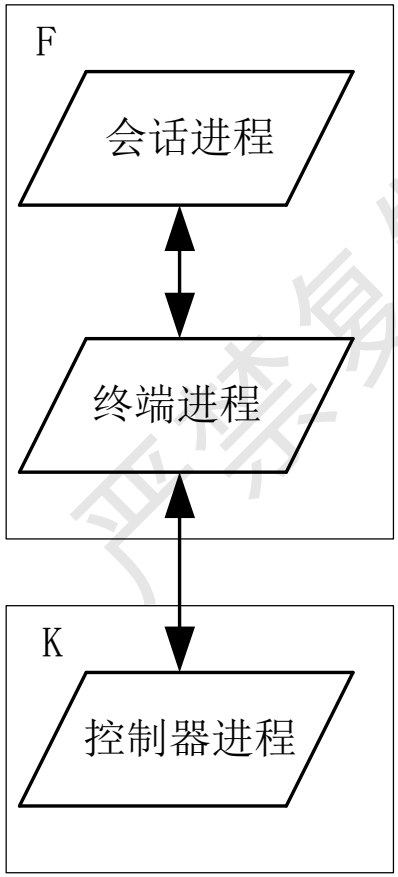


4+1模型 – 物理视图



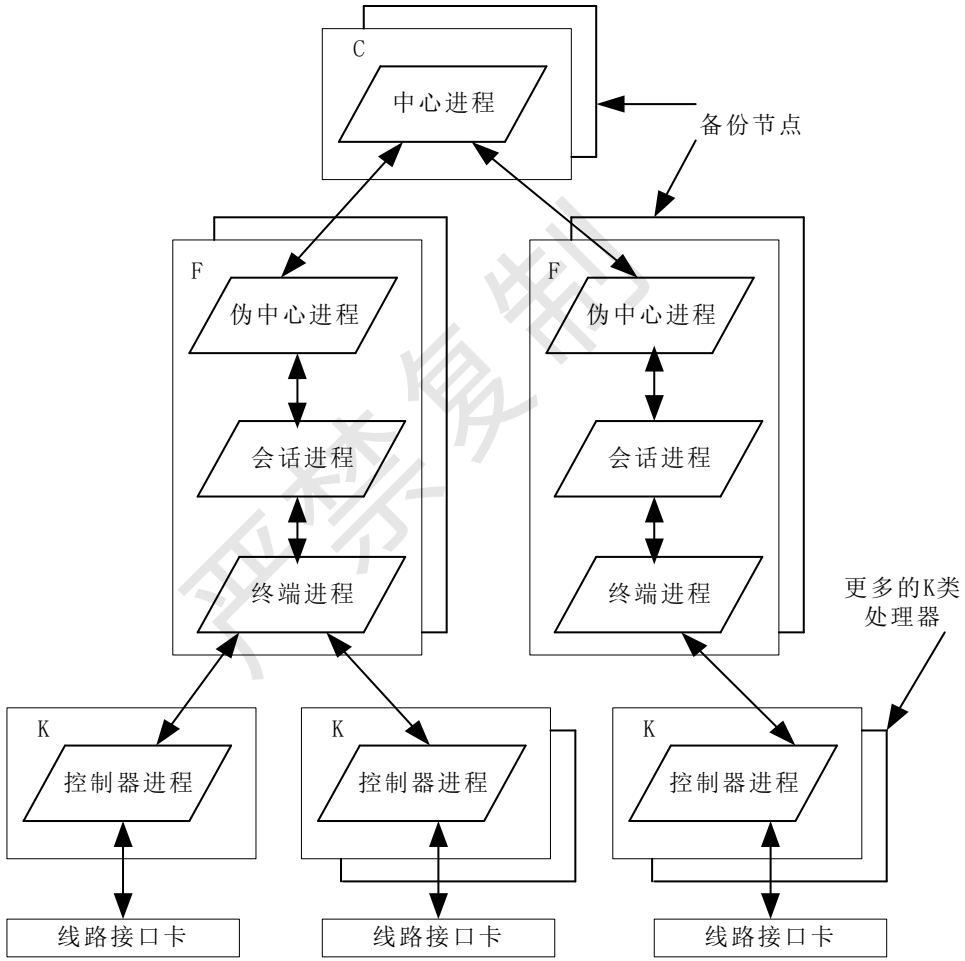
控制系统物理视图:可能的硬件配置

4+1模型 – 物理视图



控制系统物理视图:进程视图分配的物理视图

4+1模型 – 物理视图

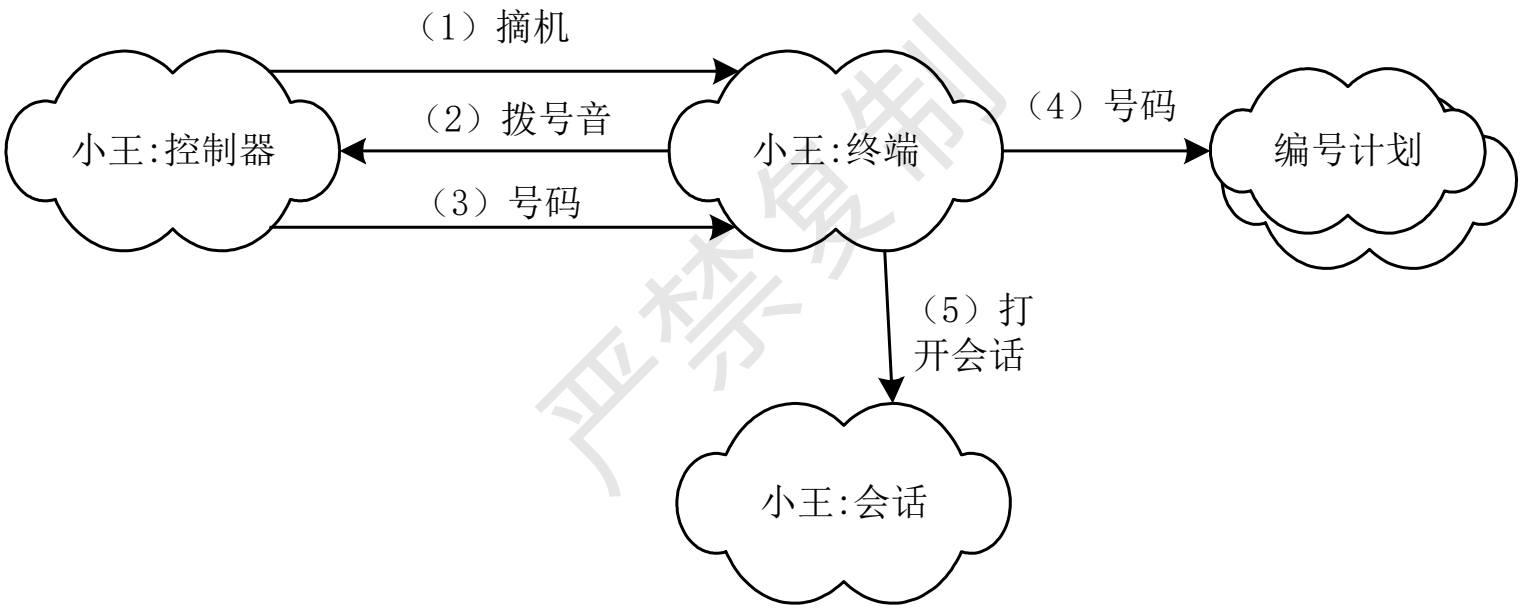


控制系统物理视图:进程视图分配的物理视图

4+1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。

4+1模型 – 场景



4+1模型 – 小结

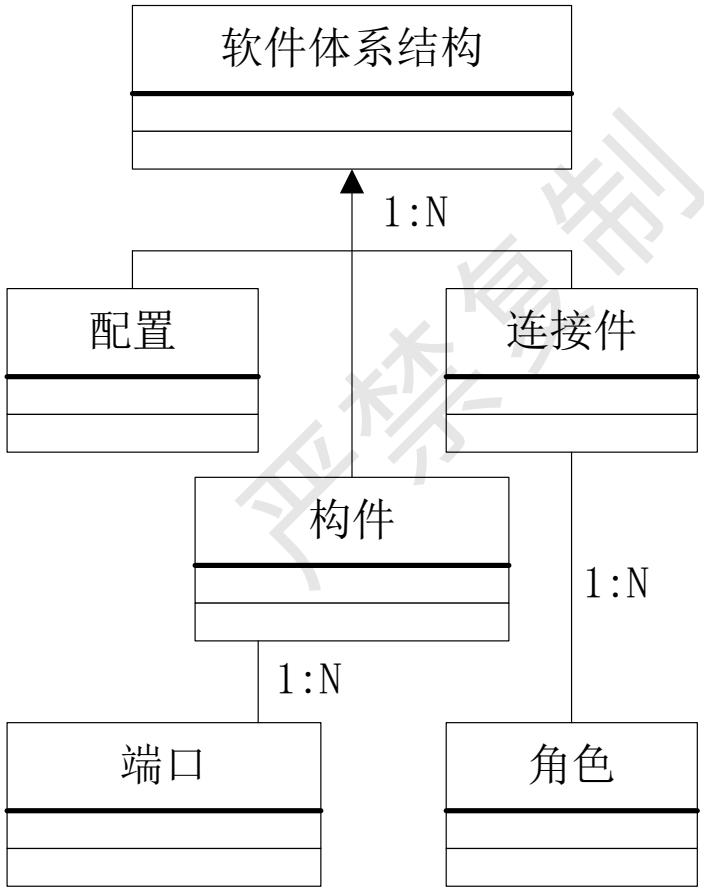
- “4+1”模型是综合了结构模型、框架模型、动态模型、过程模型和功能模型。“4”是指：逻辑视图、开发视图、进程视图、物理视图，“1”是指：场景视图。每一个视图只关心并展示一个单独的方面，可以这样认为，每一个视图是一个面，五个视图结合形成了软件体系结构这个多维的立体结构。
- 逻辑视图和开发视图描述系统的**静态结构**，而进程视图和物理视图描述系统的**动态结构**。

4+1模型 – 小结

逻辑视图主要是支持系统的**功能需求**，即系统提供给最终用户的服务，使用面向对象的风格。开发视图：也称模块视图，侧重于模块的**组织和管理**，考虑内部的需求和开发工具的不同所带来的局限性。进程视图：也称并发视图，侧重于系统的**运行特性**，使用的风格有管道和过滤器风格、客户–服务器等。物理视图：考虑软件**映射到硬件的方法**，考虑到系统的性能、规模、可靠性等。解决系统的拓扑结构、系统安装、通信等问题。场景视图：重要活动的抽象，场景是最重要的需求抽象，场景可以用文本表示，也可以用图像表示。

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。

软件体系结构的核心模型



软件体系结构的核心模型

软件体系结构的核心模型由：构件、连接件、配置、端口和角色组成

- 构件:具有某种功能的可重用的软件模板单元，表示了系统中主要的计算元素和数据存储。构件分为复合构件和原子构件。
- 连接件：表示了构件之间的交互，简单的连接件如管道、过程调用、事件广播等；复杂的如C/S通信协议，数据库和应用之间的SQL连接等。
- 配置表示了构件和连接件之间的拓扑逻辑和约束。

软件体系结构的核心模型

构件只能通过接口与外部环境交互，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。一个构件可以提供多个接口。

连接件的接口由一组角色组成，连接件的每一个角色定义了该连接件表示的交互的参与者，二元连接件有两个角色，如：远程过程调用的角色为caller和callee，管道的角色为reading和writting，消息传递连接件的角色是sender和receiver。有的连接件有多于两个的角色，如事件广播有一个事件发布者和多个时间接收者角色。