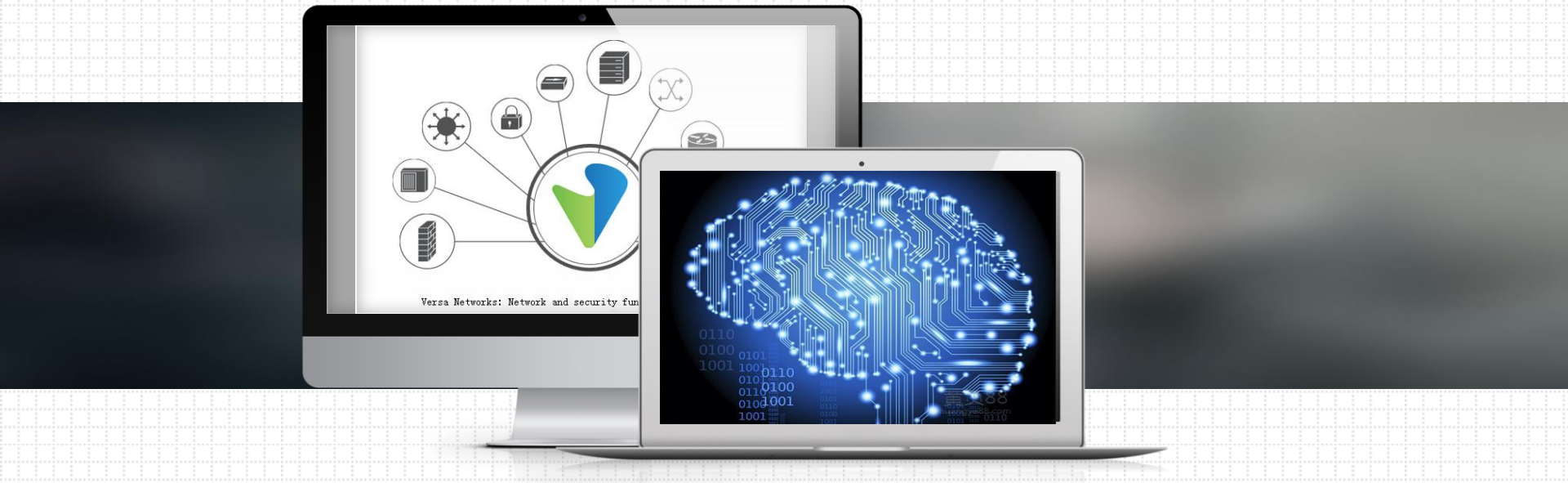
# 软件架构与设计模式

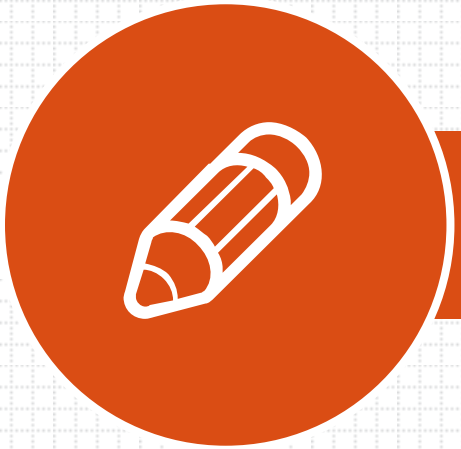
Software Architecture and Design Patterns




吴映波 副教授

Email：wyb@cqu.edu.cn

Office: 虎溪软件学院楼410



# 3. 软件体系结构建模

“ PASSION LEADS TO DESIGN, DESIGN LEADS TO PERFORMANCE, PERFORMANCE LEADS TO **SUCCESS**! ”

# 软件体系结构建模

- **建模的目的**
  - 对软件体系结构进行设计时，通常通过各种模型来建立系统体系结构的认识。通过建立的模型，人们可以分析与验证设计的正确性与有效性。
  - 但不同人在表达自己对软件的体系结构认识时，往往会采用不同的表达方法；即使同一个系统由于采用的表达方法不同，也会造成相互的不一致。
  - 于是，为了形成一种较为规范与一致性的软件体系结构的表达方式，方便设计人员之间的相互交流，需要采用一种大家均能接受的体系结构模型表达方法。

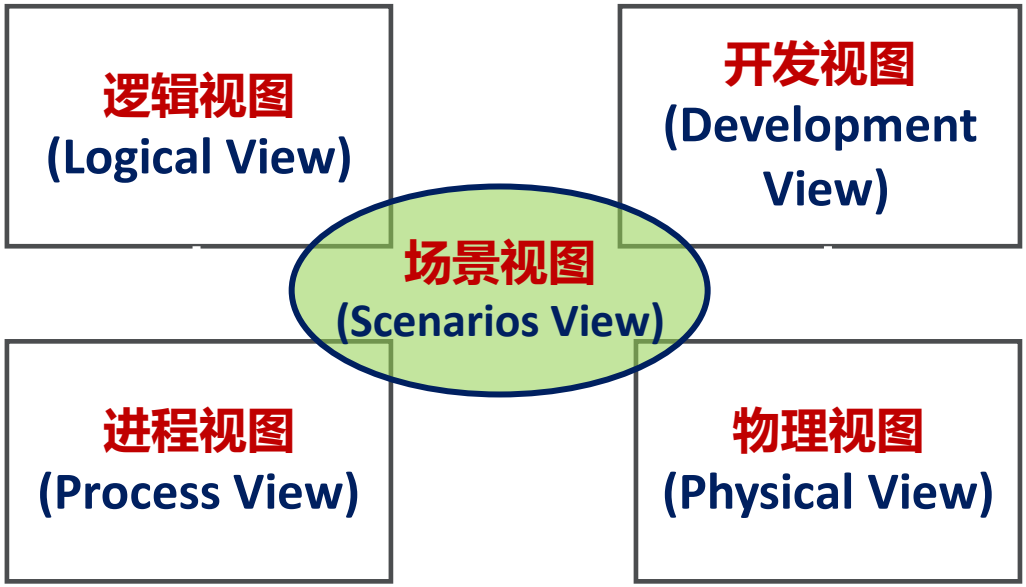
# 软件体系结构建模

- **体系结构的模型**
  - 从不同的视角（view-point），可以刻画系统的体系结构不同侧面
  - 每一个不同视角的刻画均是对系统体系结构的一个描述，我们往往需要从不同视角来描述一个系统的体系结构。

正如建筑师在设计建筑一样，需要通过建筑的正立面设计图、平面设计图和剖面图来立体的表达自己的设计一样。

# “4+1 Views” Model（“4+1”视图模型）

- Philippe Kruchten在1995年提出：使用多个并发的视图来组织软件架构的描述，每个视图仅用来描述一个特定所关注方面的问题集合。
- 每一个视图只关心软件体系结构的一个侧面，5个视图结合在一起才能完整反映软件体系结构的全部内容



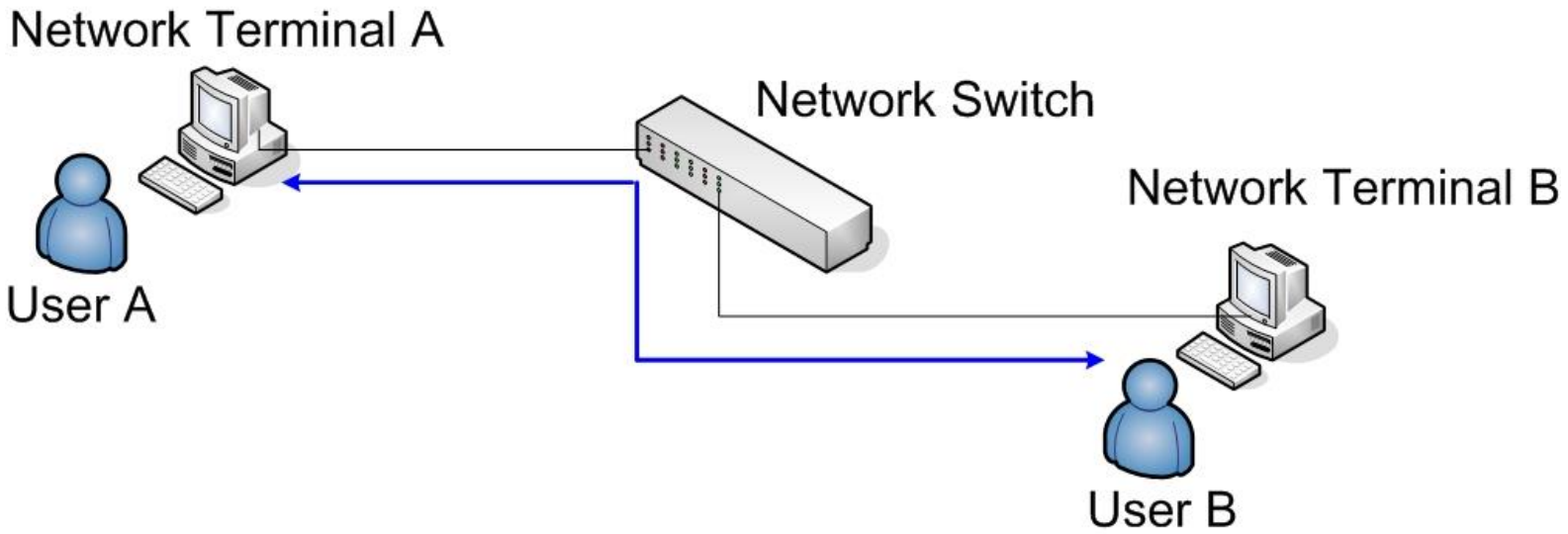
# “4+1 Views” Model

- **理解“4+1”视图：刻画系统架构的不同侧面**
  - **Logic View（逻辑视图）：**基于功能需求抽象，刻画系统的静态结构模型
  - **Process View（进程视图）**：刻画系统的运行时的结构模型

    **Development View（开发视图）**：考虑开发技术、过程与组织，刻画系统的开发管理结构模型；
  - **Physical View（物理视图）**：逻辑视图中的各功能构件在安装部署环境中的映射，刻画系统的安装部署结构模型。
  - **Scenarios View（场景视图）**：从系统使用的角度对系统结构的描述。它反映的是在完成一个系统功能时，系统各功能构件间的交互关系。

# 软件体系结构建模

## 案例分析: NAS--网络终端通讯服务系统




需求描述:

- NAS可运行于网络中任一台终端上，可为终端用户之间提供消息与文件传输等通讯服务；
- NAS在传输信息之前要求先进行加密处理，然后再以密文的形式发送；
- NAS接收到传输的密文后，要求再以明文方式显示给终端用户；

# “4+1 Views” Model

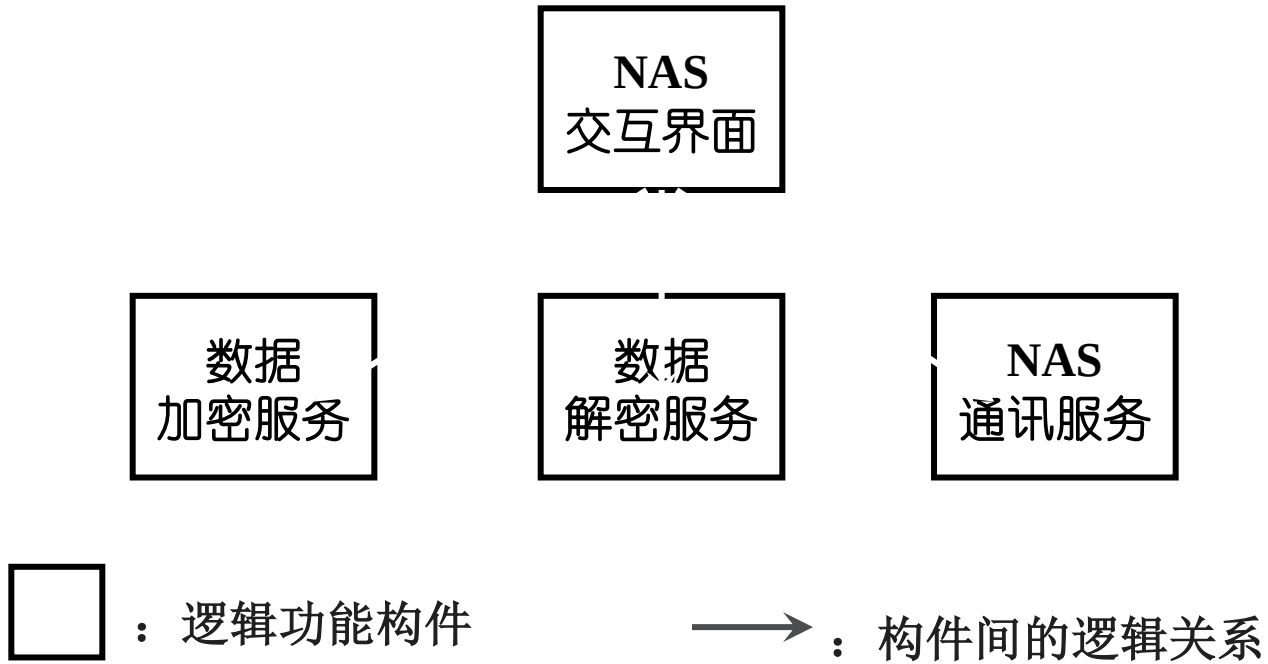
- **逻辑视图(Logical View)：也称为概念视图，主要关注系统的功能需求，即系统提供给最终用户的功能。**

  在逻辑视图中，系统分解成一系列的功能抽象，这些抽象主要来自对软件功能需求的理解抽象（问题领域）

  - **视角**：功能需求的分析理解与抽象
  - **关注点**：基于软件的功能性需求，是系统功能的抽象结构表述，关注系统提供给最终用户的功能。
  - **表示法**：线框图、UML（类、模块、包、子系统…）

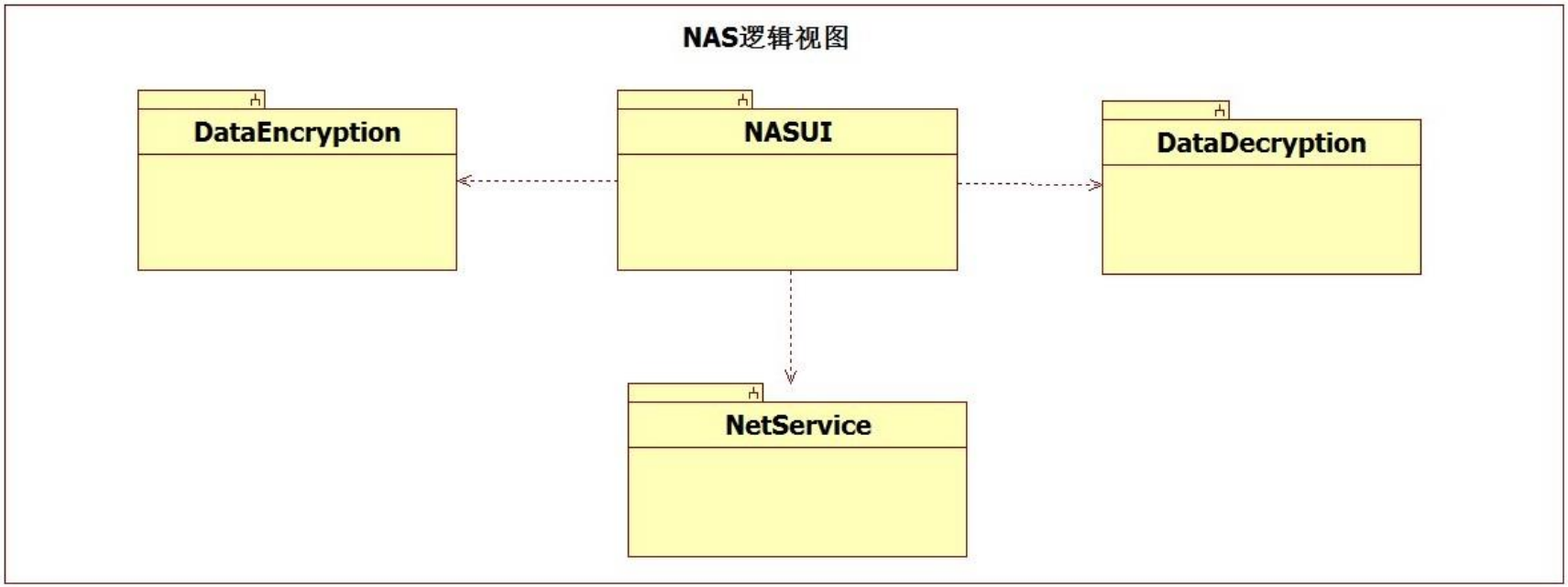
# “4+1 Views” Model

- **逻辑视图（Logic View）**
  - **线框图表示法**



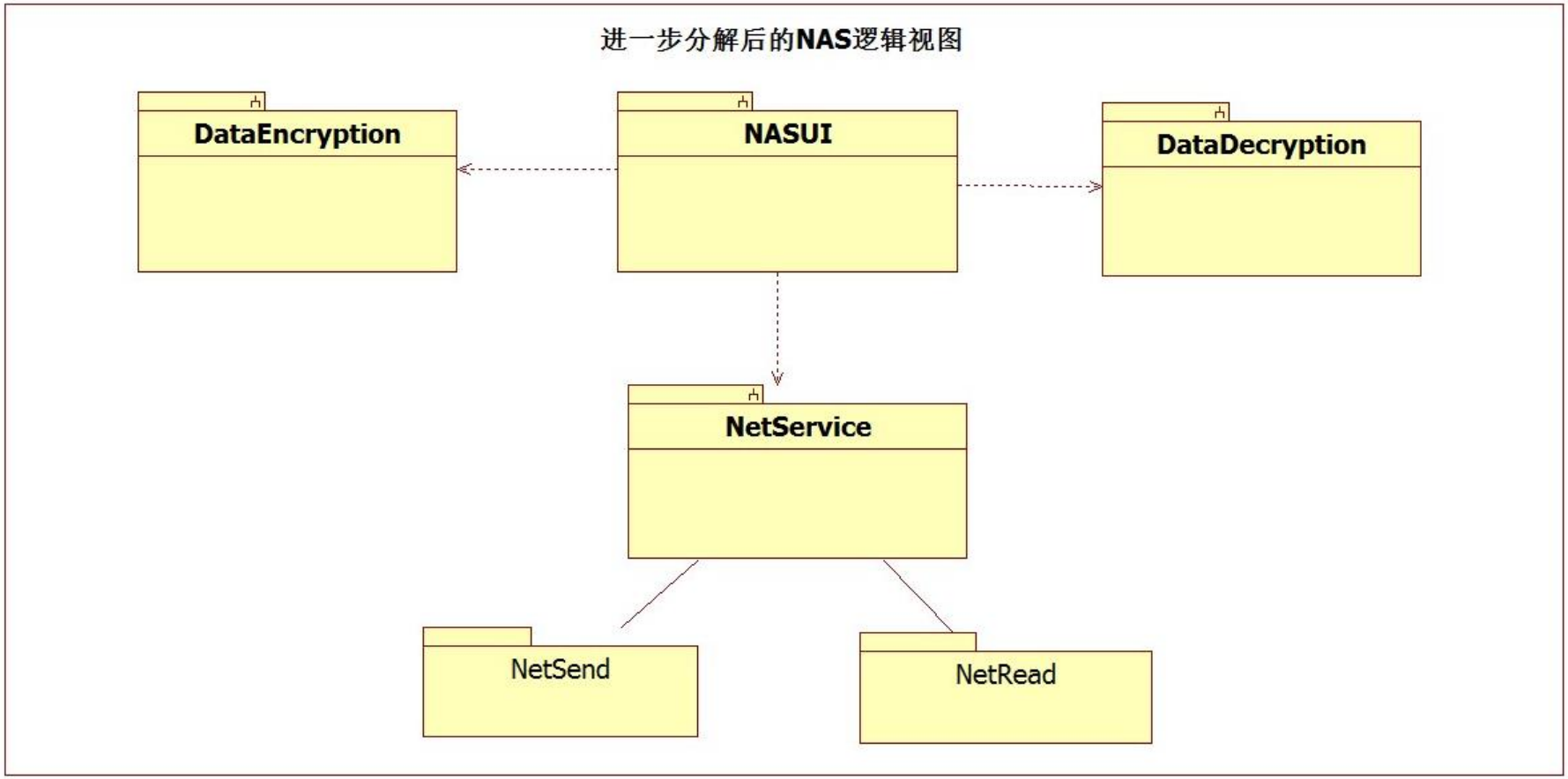
# “4+1 Views” Model

- 逻辑视图（Logic View）
  - UML建模的NAS系统逻辑视图



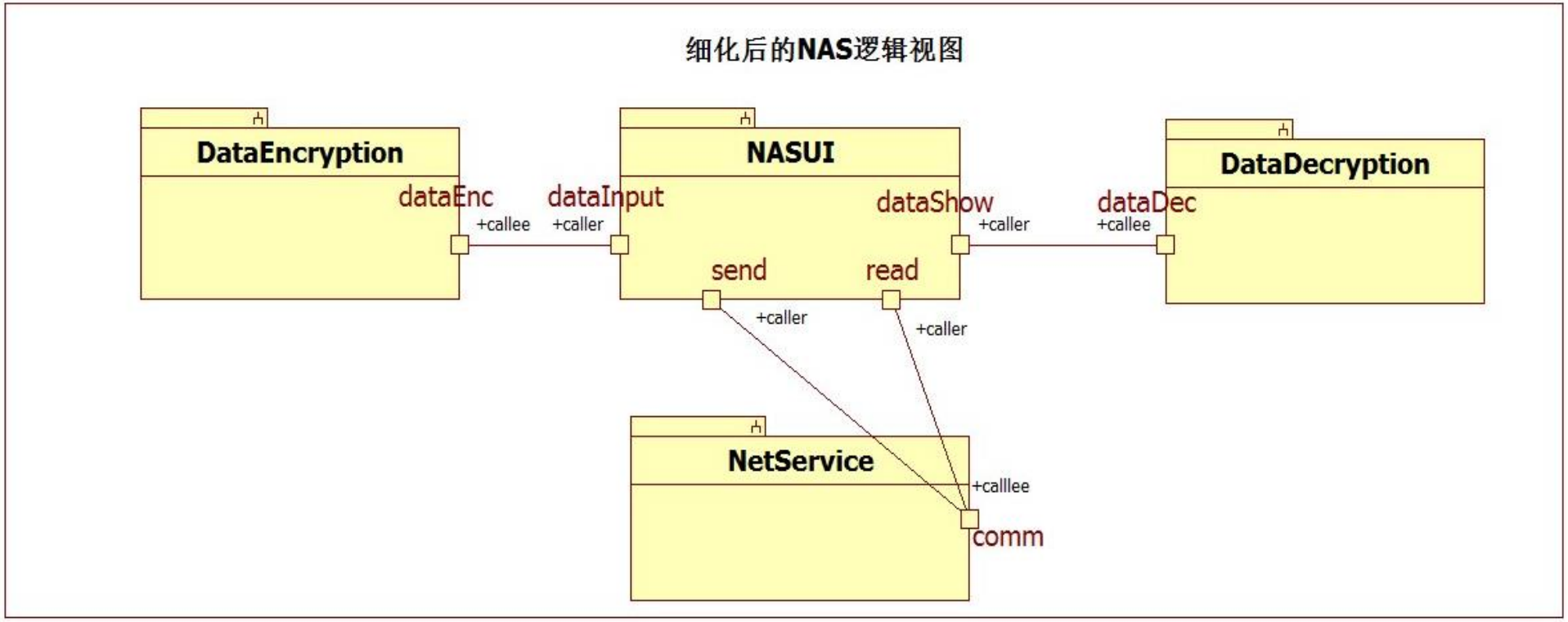
# “4+1 Views” Model

- 逻辑视图（Logic View）--- 进一步分解后的逻辑视图



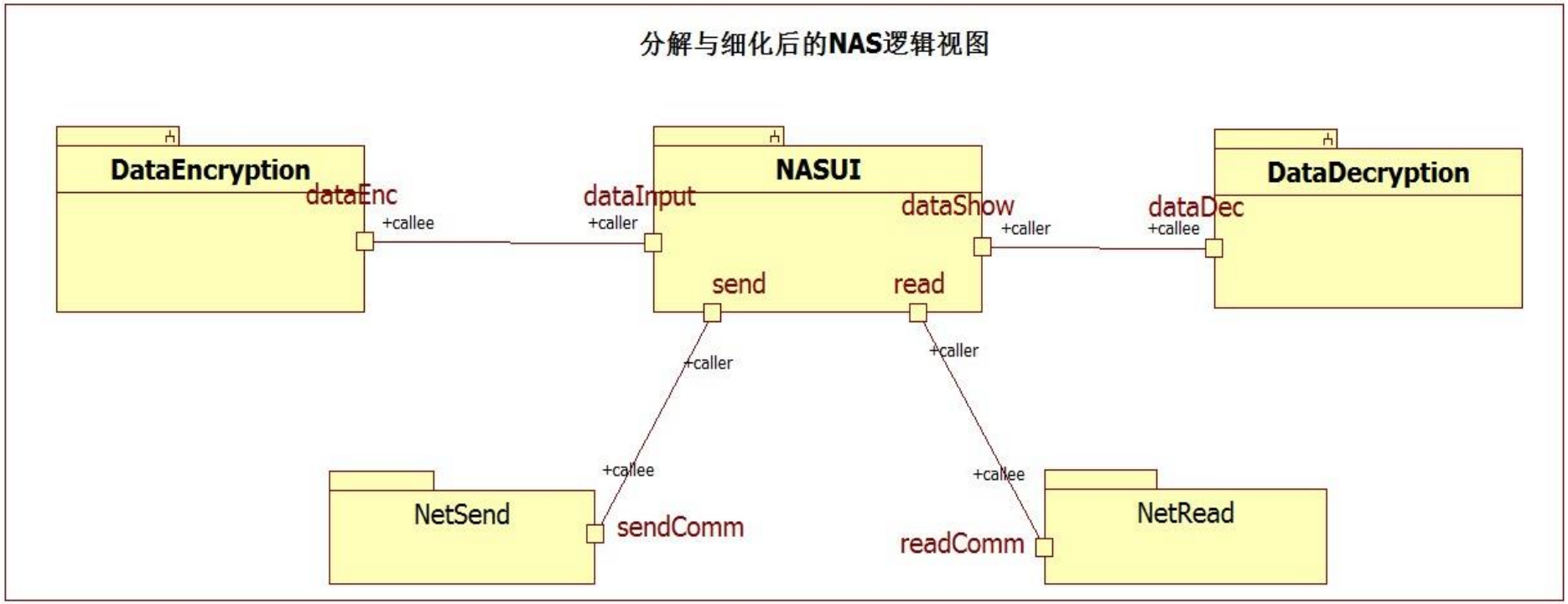
# “4+1 Views” Model

- 逻辑视图（Logic View）--- 细化后的逻辑视图



# “4+1 Views” Model

- 逻辑视图（Logic View）--- 进一步分解与细化后的逻辑视图



# “4+1 Views” Model

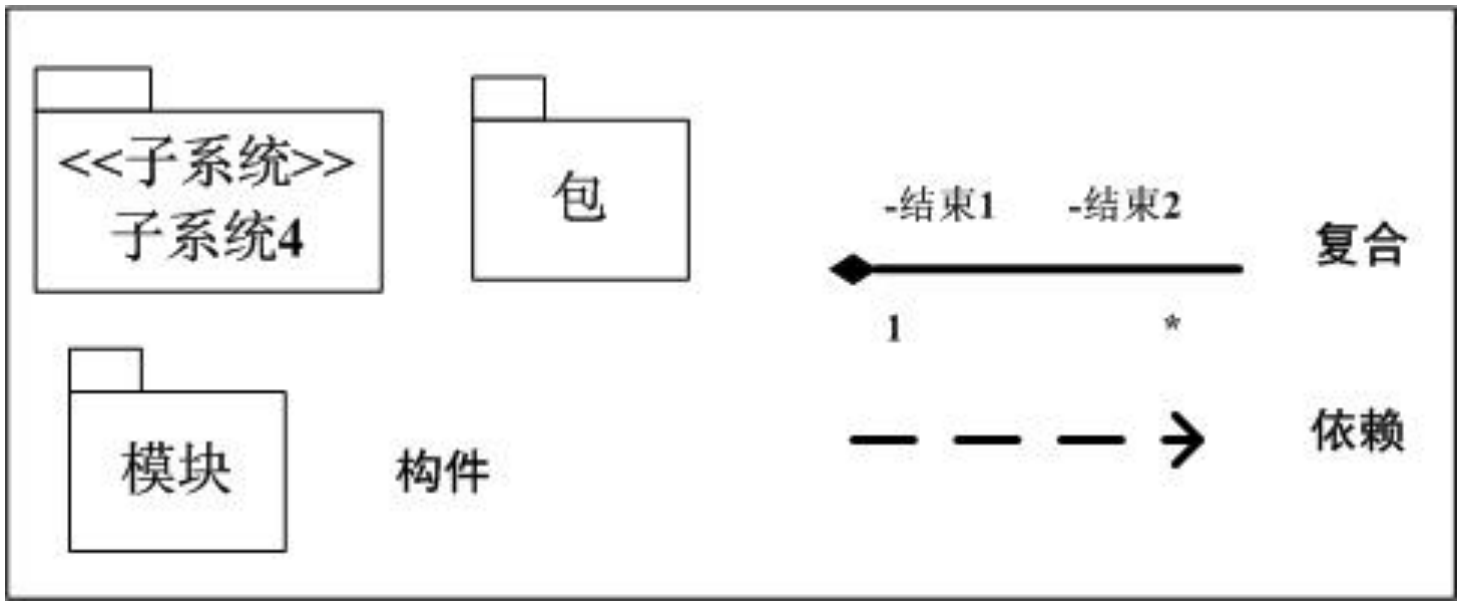
- **开发视图(Development View)：也称模块视图，主要侧重于软件模块的结构化组织和管理，体现为软件模块、库、子系统和开发单元的结构化组织。**

  开发视图要充分考虑软件实现的要求与约束，如软件开发技术要求、开发过程与组织形式、软件的复用性，以及技术与管理风险等因素。

  - **视角**：软件的开发实现
  - **关注点**：是软件体系结构的逻辑视图在具体实现阶段的表示，关注软件实现的技术与组织管理要求及约束。
  - **表示法**：线框图、UML（模块、包、子系统…）

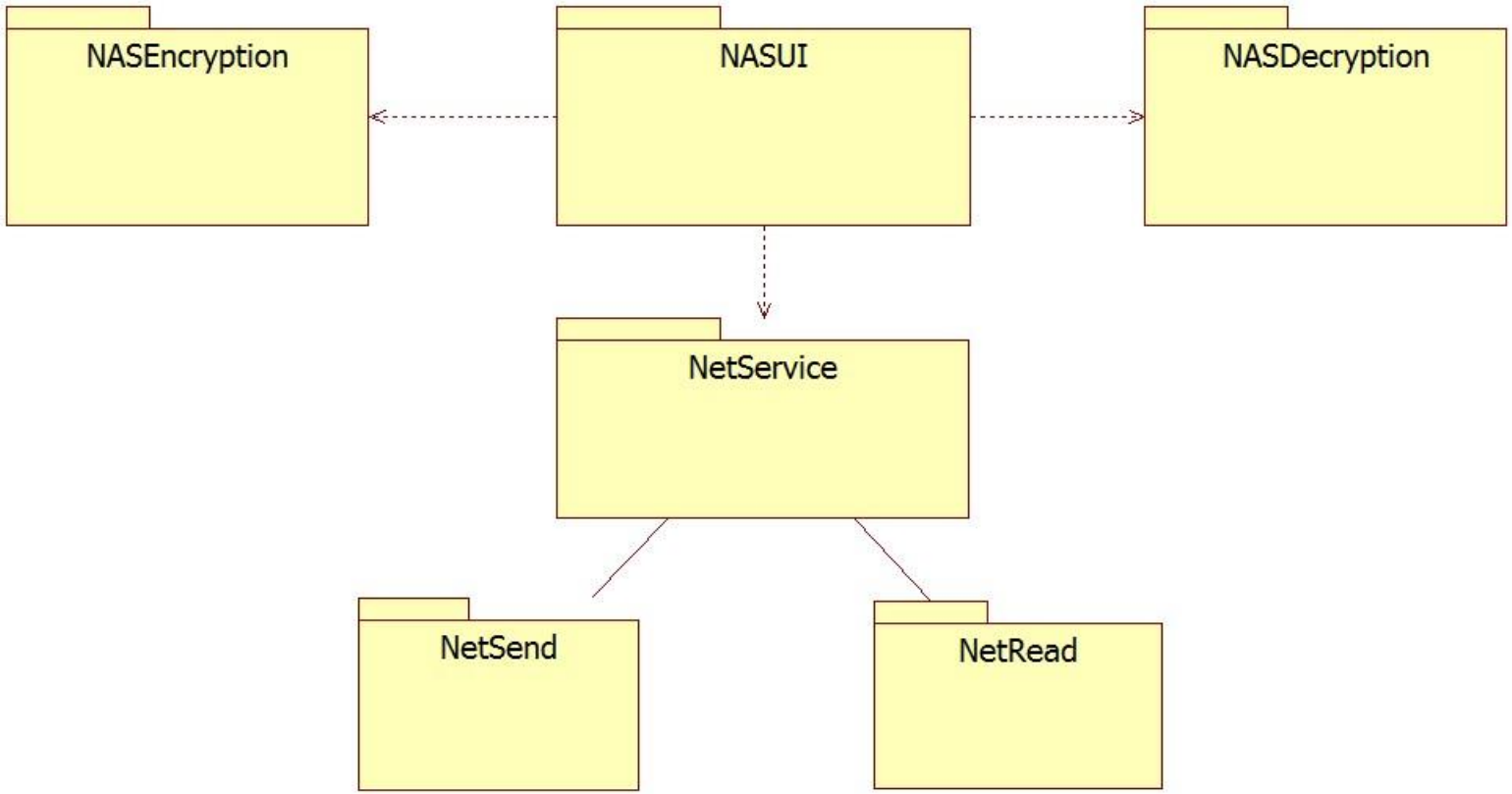
# 2.2 “4+1 Views” Model

- 开发视图
  - 开发视图的UML表示法



# “4+1 Views” Model

- **开发视图**
  - **NAS系统的开发视图**



# “4+1 Views” Model

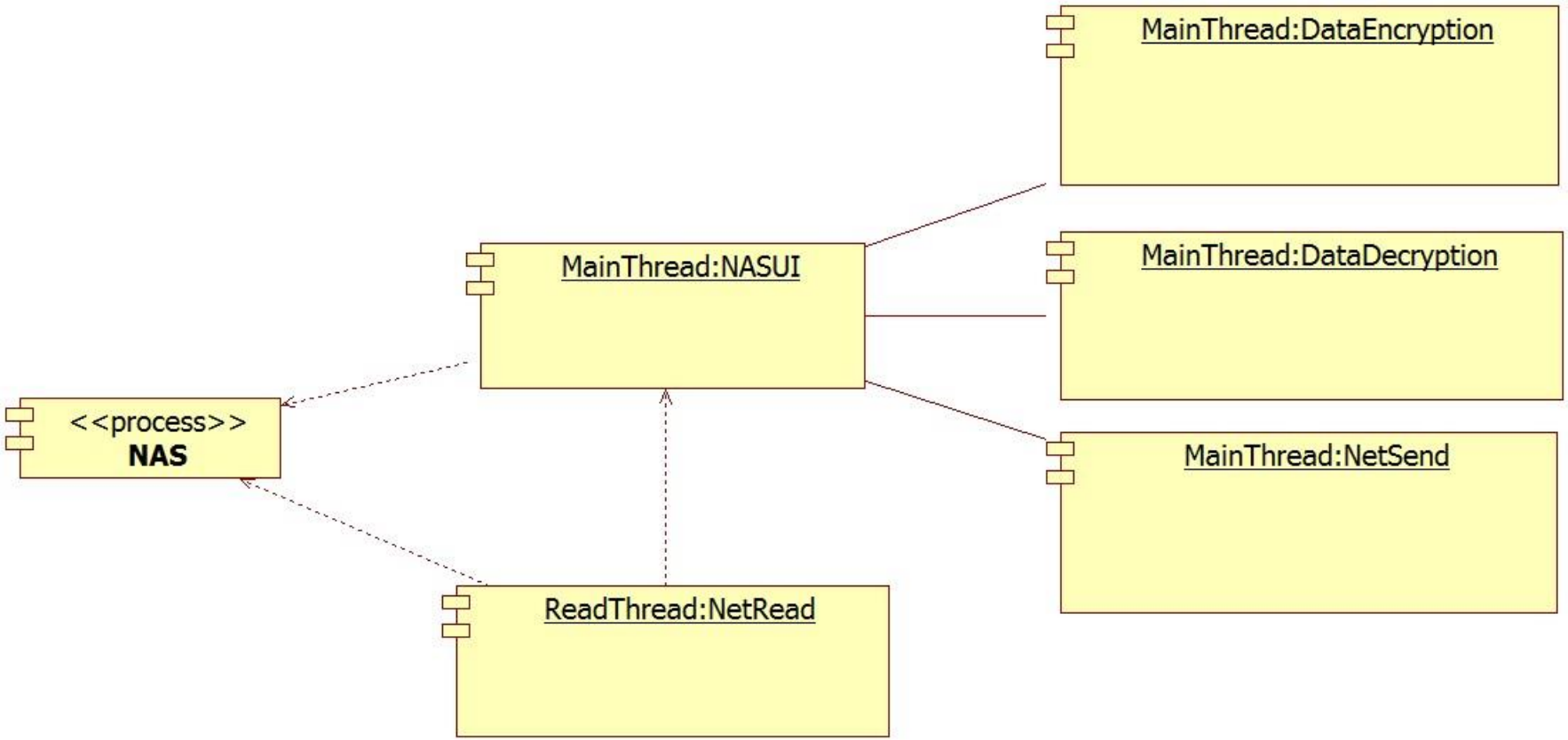
- **进程视图(Process View)：侧重于系统的运行特性，主要关注系统的非功能性的需求的满足。**

  进程视图强调并发性、分布性、系统集成性和容错能力，为逻辑视图中的构件设计适合的进程/线程结构。它定义逻辑视图中的各个构件具体在进程/线程中的映射结构。

  - **视角**：软件运行时(Run-time)的结构形态
  - **关注点**：基于软件的非功能性需求，是软件系统运行时的动态结构，关注的是系统非功能性需求的满足。
  - **表示法**：线框图、UML（类图、交互图、顺序图、状态图）...

# “4+1 Views” Model

- **进程视图 --- UML建模表示**

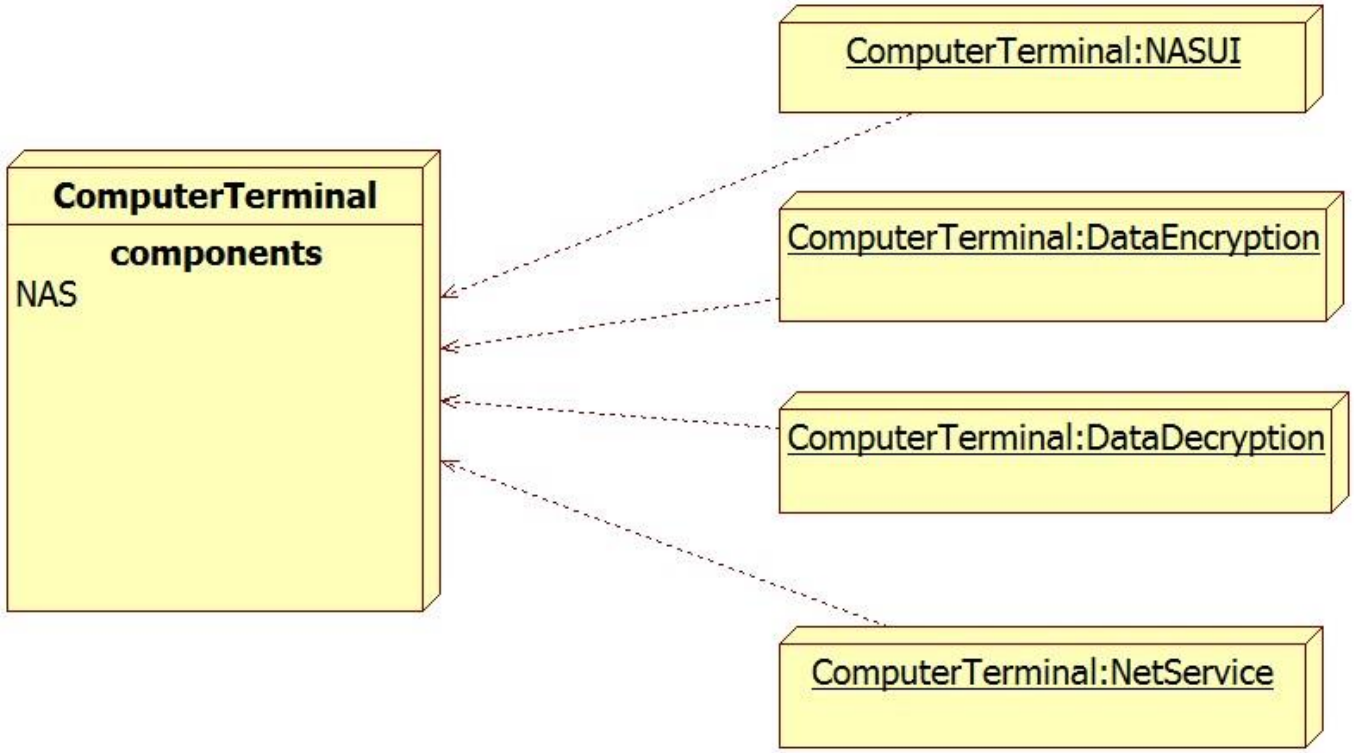


# “4+1 Views” Model

- **物理视图（Physical View）：物理视图主要考虑如何把软件映射到硬件上，它通常要考虑软件系统在在计算物理节点与网络拓扑结构上的运行部署等问题。主要关注系统性能、可扩展性、可靠性等软件非功能性需求约束。**
  - **视角**：软件在实际安装部署环境中的结构形态
  - **关注点**：基于软件的非功能性需求，是软件系统安装运行时的动态结构，关注的是系统非功能性需求的满足。
  - **表示法**：安装部署结构图（Visio画图工具）、UML…

# “4+1 Views” Model

- 物理视图 --- UML建模表示



# “4+1 Views” Model

- **场景视图（Scenarios View）：从系统使用的角度对系统结构的描述。它反映的是在完成一个系统功能时，系统各功能构件间的协作关系。**

  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。

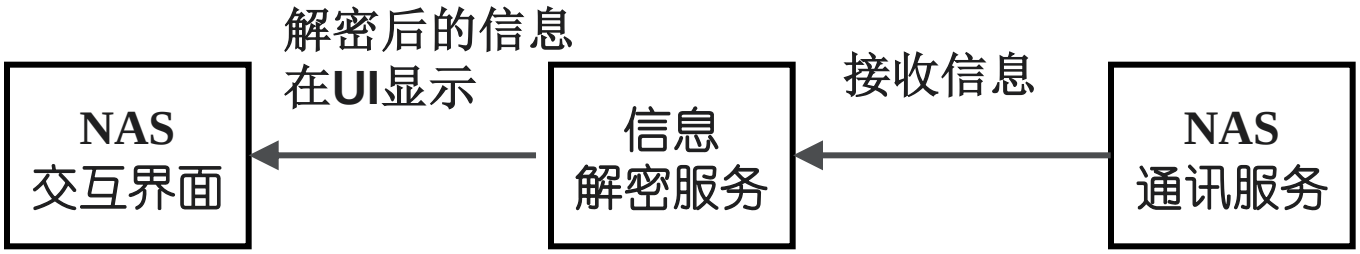
  - **视角**：用户视角
  - **关注点**：基于软件的功能性需求，关注的是在完成一个系统功能时，系统各功能构件间的协作关系，增加设计的可理解性，为其它视图的分析设计服务。
  - **表示法**：线框图、UML（用例图、序列图、活动图、状态图）...

# “4+1 Views” Model

- **场景视图 --- 线框图表示**

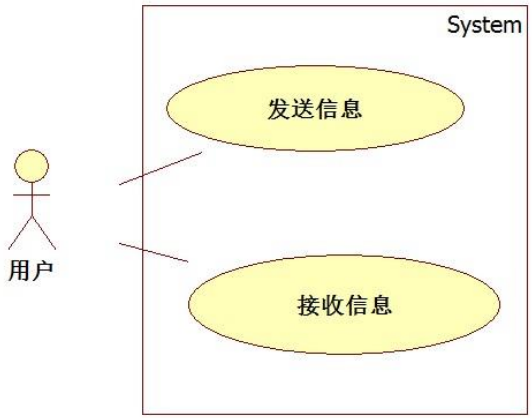
场景视图1：发送信息

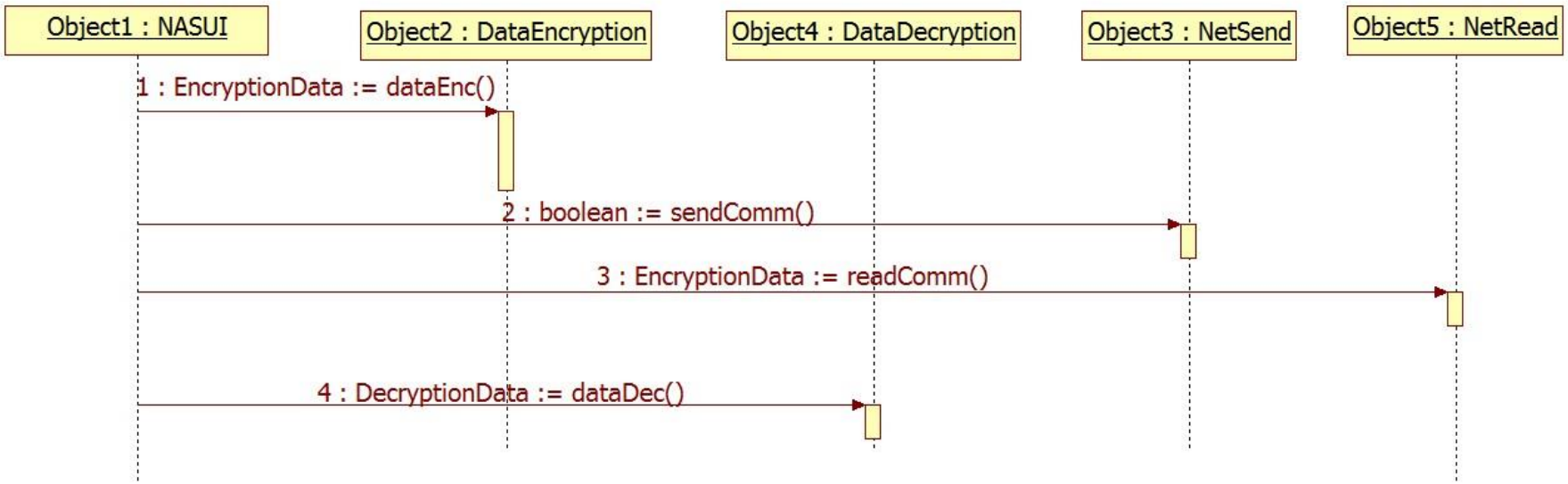

场景视图2：接收信息



# “4+1 Views” Model

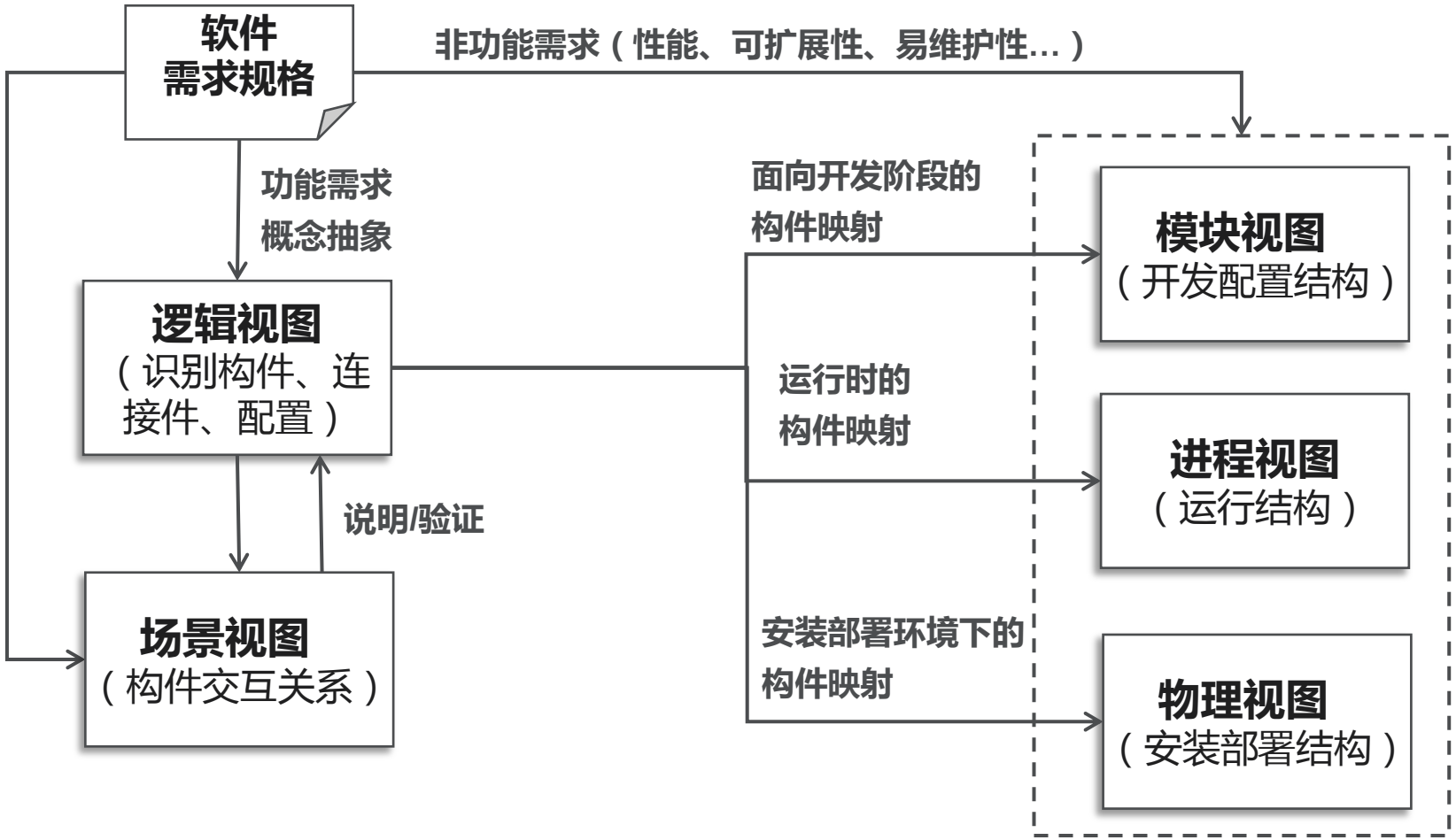
- **场景视图 --- UML建模表示**



# “4+1 Views” Model

- “4+1视图”运用过程



# “4+1 Views” Model

- **理解“4+1”视图：各视图比较**

| | 逻辑视图 | 进程视图 | 开发视图 | 物理视图 | 场景视图 |
|---|---|---|---|---|---|
| **构件** | 逻辑功能 | 进程/线程 | 模块、包、子系统 | 计算资源节点（分布式） | 步骤、脚本 |
| **连接件** | 使用、依赖、聚集… | 进程/线程间的通讯 | include、import、using | 通讯连接（分布式） | |
| **关注点** | 静态的功能结构 | 运行时的系统结构 | 实现划分 | 部署及维护环境 | 可理解性 |
| **主要涉及者** | 架构师、用户 | 架构师 | 架构师、系统开发人员 | 架构师、系统部署与维护人员 | 架构师、用户 |

# “4+1 Views” Model

- **理解“4+1”视图**
  - 对系统建立的某种视图并不一定是唯一的，例如逻辑视图，可能首先建立的是系统的高层逻辑视图，随着分析的深入，逐步建立更详细逻辑视图；
  - 对有些系统每个视图也不一定是必需的，如对于一个单进程的系统，进程视图就没必要。物理视图对于一个非分布式的系统也是没有必要的。
  - “4+1视图”只是对系统体系结构分析提出了一种分析方法，强调从不同角度和关注点来刻画系统的结构。但是其本身并没形成一套严格的视图表示法。实践工程中，体系结构设计人员往往会各自采用不同的表示法。UML目前能较好地应用于对象的分析和设计，但是对于系统宏观结构的设计仍显得有很多的不足。

# 课程作业&实验--ATM自动存取款机软件系统的架构分析与设计

## ATM自动存取款机

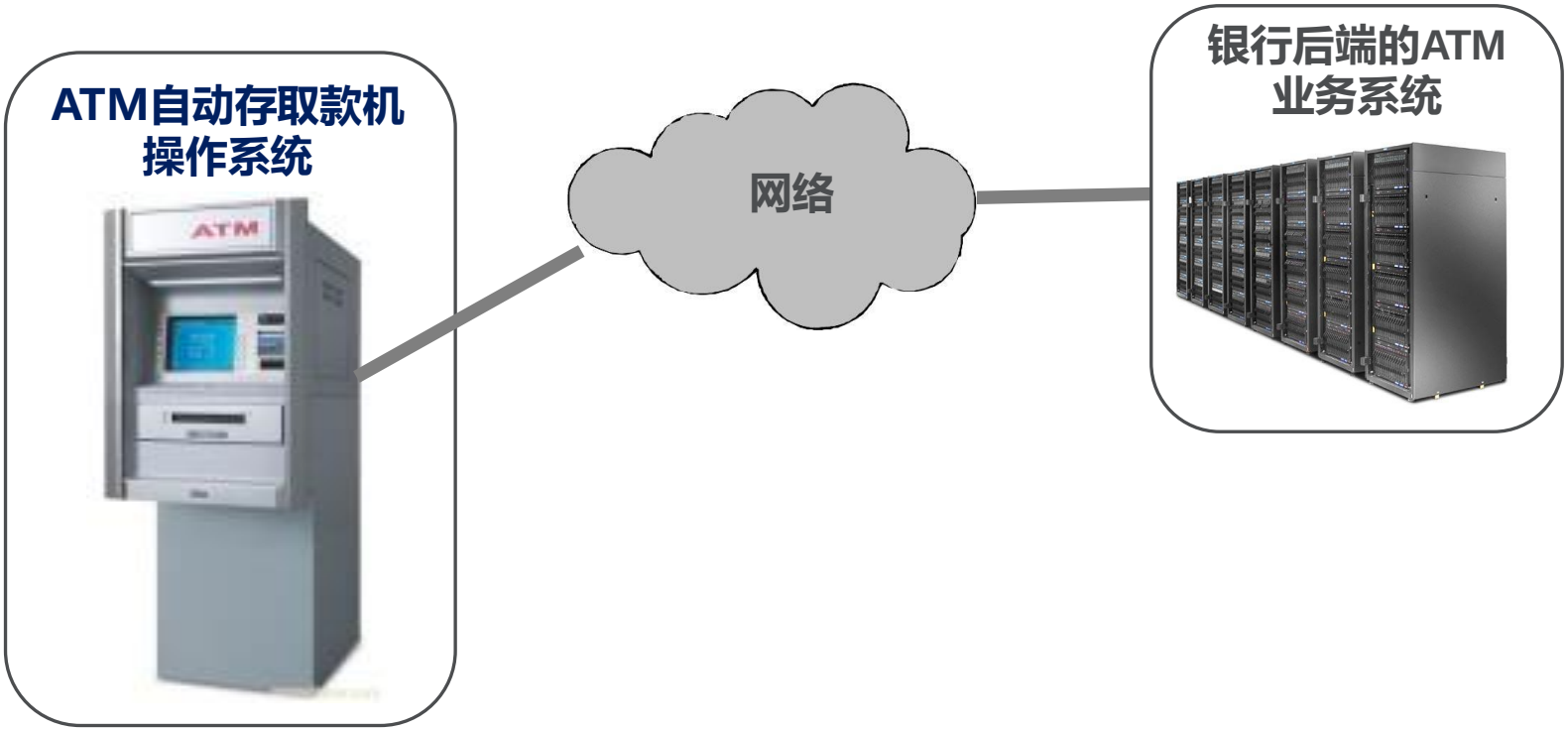
ATM自动存取款机（简称ATM）是由一套银行持卡人（即客户）自我服务型的金融专用设备。ATM是英文Automatic Teller Machine的缩写。ATM是最普遍的自助银行设备，可以提供最基本的银行服务，如取款、存款与转账等客户账户交易服务。在ATM上也可以进行客户账户的查询、存取款和转账等业务。作为自助式金融服务终端设备，除了提供金融业务功能之外，ATM还具有维护、测试、事件报告、监控和管理等多种功能。

ATM硬件系统一般主要由终端计算机（包括cpu和存储设备、键盘输入设备、显示设备等）、外围硬件（包括：设备读卡器、钞票吐纳设备、验钞设备、摄像设备、打印设备、钞箱和报警设备等）组成。

# 课程作业&实验--ATM自动存取款机软件系统的架构分析与设计

- **ATM自动存取款机软件系统**

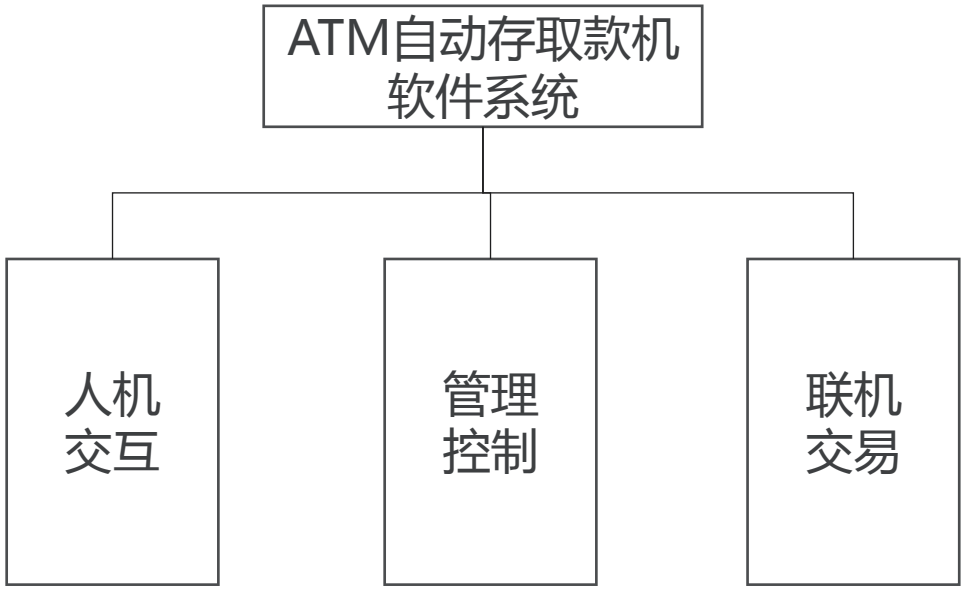
**ATM自动存取款机软件系统**是运行在ATM的终端计算机上的软件系统。它除提供在ATM上的各种人机交互功能，还管理控制ATM的各硬件设备，以及与银行后端ATM业务系统的网络通信，实现用户联机交易服务。



# 课程作业&实验--ATM自动存取款机软件系统的架构分析与设计

## ATM自动存取款机软件系统 --- 功能模块

- **人机交互**：用户在ATM上的各种功能交互UI。
- **管理控制**：ATM的各外围硬件设备的访问、驱动、控制等，如驱动读卡器读取用户账户、启动摄像设备记录保持用户操作、启动吐纳设备接收/吐出钞票、启动报警设备警示安全等。
- **联机交易**：通过Internet或银行专用网络与银行后端的ATM业务处理系统通讯，联机实现用户在ATM上的各种人机交互操作（包括：客户账户认证、登录、查询、取款、存款、转账等）。



# 课程作业&实验--ATM自动存取款机软件系统的架构分析与设计

- **作业&实验要求：**

**1. 完成ATM自动存取款机操作系统的体系结构分析与设计**

第一次作业：完成ATM自动存取款机操作系统的逻辑视图和场景视图

要求：使用StarUML建模工具；建立系统的逻辑视图（系统全部功能构件的分解设计，可暂不设计构件端口、连接件角色）和场景视图

完成时间：2016-5-8 晚 12：00 截止（11周日）

第二次作业：进一步完成细化的逻辑视图，以及其他视图

要求：细化概念视图并完善设计构件端口、连接件角色，以及完成其它4个视图设计

完成时间：2016-5-15 晚 12：00 截止（12周日）

第三次作业：应用软件体系结构风格，进一步完成优化设计

要求：使用典型的软件体系结构风格，完成系统架构的优化设计

完成时间：2016-5-22 晚 12：00 截止（14周日）

**2. 提交实验报告：按实验报告格式，提交最终的系统架构设计方案**

2016

# End