北京交通大学
BEIJING JIAOTONG UNIVERSITY

软件系统分析与设计
System Analysis & Design

M210007B

Monday, April 22, 2024

/-1 面向对象方法概述

1-1 统一建模语言UML
Unified Modeling Language

统一建模语言UML

- UML的背景
- UML的发展
- UML的内容
- UML的主要特点
- UML的功能
- UML的组成

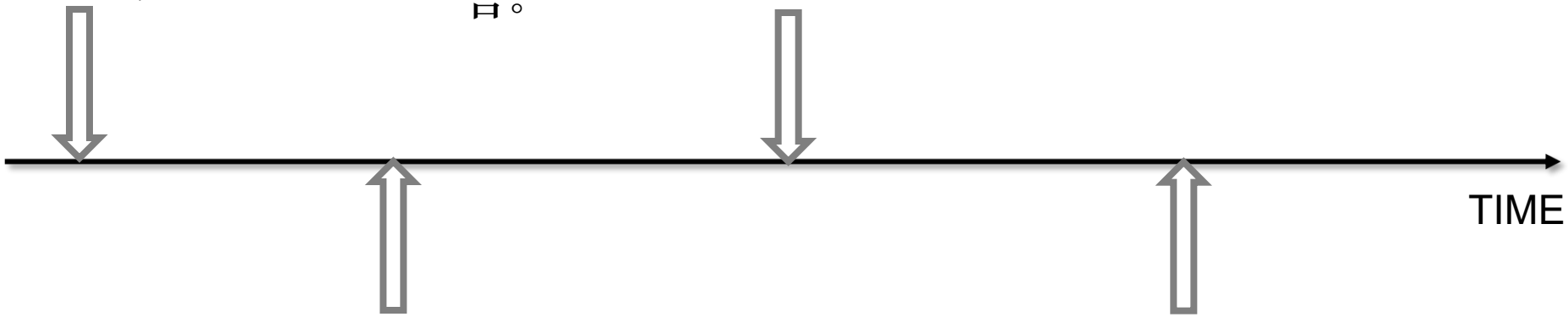
UML的背景

- 1989年到1994年，面向对象建模语言从不到10种增加到了50多种。
- 不同的建模语言具有不同的建模符号体系，妨碍了软件设计人员、开发人员和用户之间的交流。有必要建立一个标准的、统一的建模语言。
- 统一建模语言UML的诞生结束了符号方面的“方法大战”。
- UML统一了Booch方法、OMT方法、OOSE方法的符号体系，采纳了其他面向对象方法关于符号方面的许多好的概念。

UML的发展

UML的建立开始于1994年10月。

UML 1.0定义完整、富于表达、功能强大，于1997年1月被提交给OMG（Object Management Group，对象管理组织），申请成为标准建模语言。



定义UML 1.0时，DEC、HP、I-Logix、IntelliCorp、IBM、ICON计算（ICON Computing）、MCI Systemhouse、Microsoft、Oracle、Rational、Texas仪器（Texas Instrumnets）、Unisys等公司都参与了该项工作。

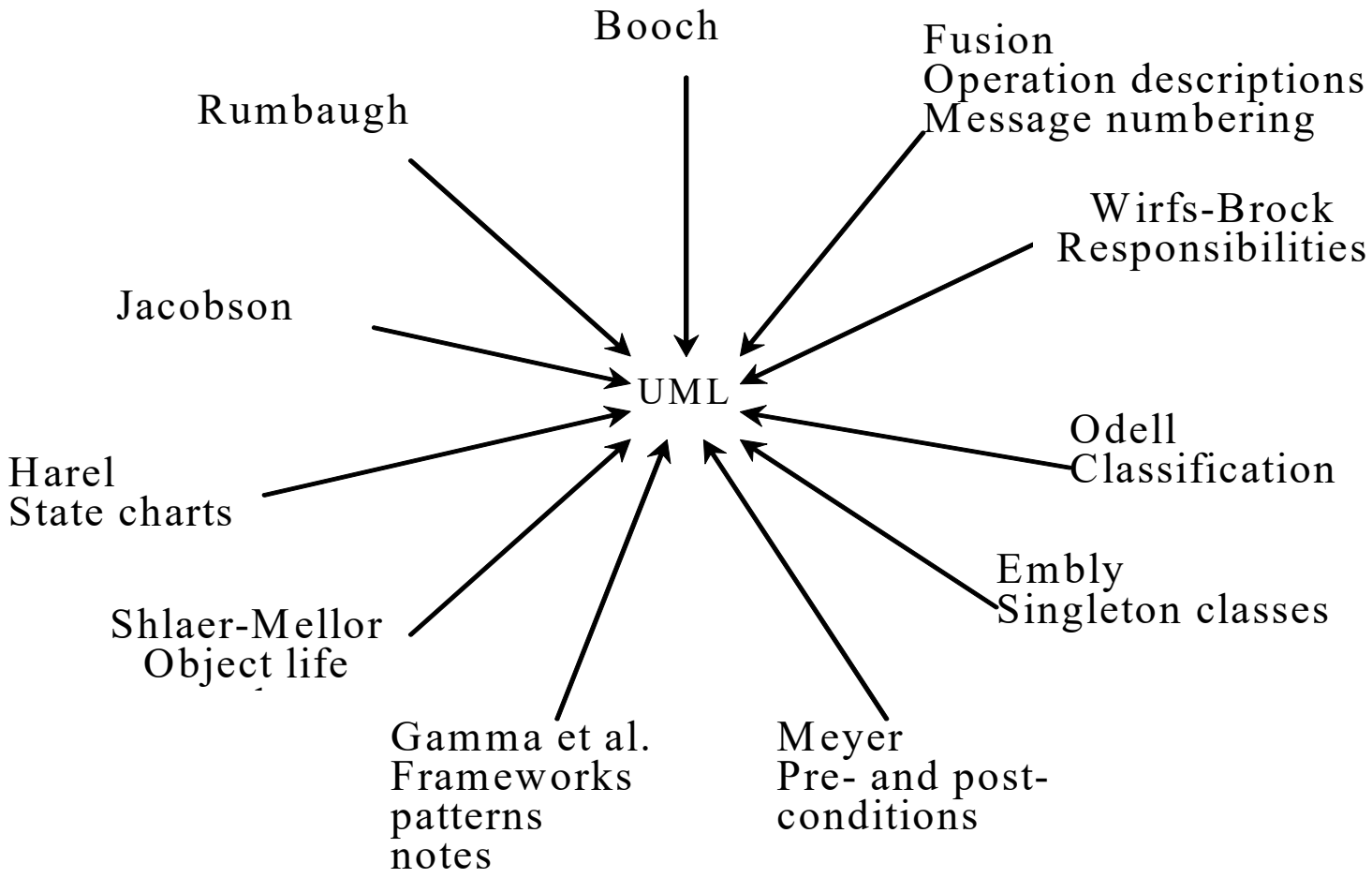
2005年，UML 2.0被OMG采纳，UML2.0对UML1.x进行了很多重大修改。

UML的内容

- UML定义包括：
  - UML语义
    - 描述了基于UML的精确元模型定义。
  - UML表示法
    - 定义了UML符号的表示方法，为开发者或开发工具使用这些图形符号和文本语法为系统建模提供了标准。

UML的主要特点

- UML统一了Booch、OMT、OOSE和其他面向对象方法的基本概念和符号。
- UML是一种建模语言，而不是一种方法。



UML的功能

- 为软件系统的产物建立可视化模型。
  - UML是一个标准的、被广泛采用的建模语言，用UML建模有利于交流。
  - UML为系统建立了图形化的可视模型，使系统的结构变得直观，易于理解。
  - UML为软件系统建立模型不但有利于交流，还有利于对软件的维护。
- 规约软件系统的产物。
  - 规约（Specifying）意味着建立的模型是准确的、无歧义的、完整的。
  - UML定义了在开发软件系统过程中所做的所有重要的分析、设计和实现决策的规格说明。

UML的功能

- 构造软件系统的产物。
  - **UML**不是可视化的编程语言，但它的模型可以直接对应到各种各样的编程语言。
  - 前向工程:从**UML**模型生成编程语言代码的过程。
  - 逆向工程:从代码实现生成**UML**模型的过程。
- 为软件系统的产物建立文档。
  - **UML**可以为系统的体系结构及其所有细节建立文档。
  - **UML**还可以为需求、测试、项目规划活动和软件发布管理活动建模。

UML的图

- 统一建模语言UML是用来对软件系统的产物进行可视化、规范定义、构造并为之建立文档的建模语言。
- UML的13种图如下：
  （1）类图（Class Diagram）
  （2）对象图（Object Diagram）
  （3）组件图（Component Diagram）
  （4）组合结构图（Composite Structure Diagram）
  （5）用例图（Use Case Diagram）
  （6）顺序图（Sequence Diagram）
  （7）通信图（Communication Diagram）
  （8）状态机图（State Machine Diagram）
  （9）活动图（Activity Diagram）
  （10）部署图（Deployment Diagram）
  （11）包图（Package Diagrams）
  （12）定时图（Timing Diagram）
  （13）交互概览图（Interaction Overview Diagram）

UML的组成

- 元素
  - 结构元素
  - 行为元素
  - 分组元素
  - 注释元素
- 关系
  - 依赖关系
  - 关联关系
  - 类属关系
  - 实现关系

- 图
  - 结构建模图
    - 类图、对象图、组件图、组合结构图、包图和部署图
  - 行为建模图
    - 用例图、活动图、状态机图、
  - 交互建模图
    - 顺序图、通信图、定时图和交互概览图

UML2.0

- UML 2.0 defines thirteen types of diagrams, divided into three categories: Six diagram types represent static application structure; three represent general types of behavior; and four represent different aspects of interactions:
- **Structure Diagrams**
  - Class Diagram, Object Diagram, Component Diagram, Composite Structure Diagram, Package Diagram, and Deployment Diagram.
- **Behavior Diagrams**
  - Use Case Diagram; Activity Diagram, and State Machine Diagram.
- **Interaction Diagrams**
  - Sequence Diagram, Communication Diagram, Timing Diagram, and Interaction Overview Diagram.

UML

- UML是定义良好、易于表达、功能强大的语言
- 不仅支持面向对象的分析与设计，而且支持从需求分析开始的软件开发的全过程。

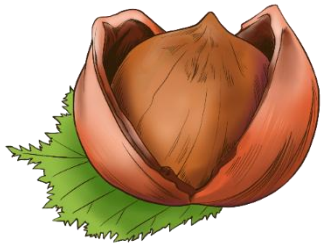
1-3 工具
Tools

工具

商业的或开源的UML计算机辅助软件工程工具：

- **Rational Software Modeler**
- **StarUML**
- **ProcessOn**
- **Visual Paradigm for UML**
- **Prosa UML**
- **Visio**
- **Together**
- **Visual UML**
- **Object Domain UML**
- **Magic Draw UML等，**

举个例子
Examples



大部分CASE工具都给软件开发者提供了一整套的可视化建模工具，包括系统建模、模型集成、软件系统测试、软件文档的生成、从模型生成代码的前向工程、从代码生成模型的逆向工程、软件开发的项目管理、团队开发管理等，为关于客户＼服务器、分布式、实时系统环境等的真正的商业需求，提供了稳健的、有效的解决方案。

第5章 视与图

5-1 视

架构视图（Architectural view）

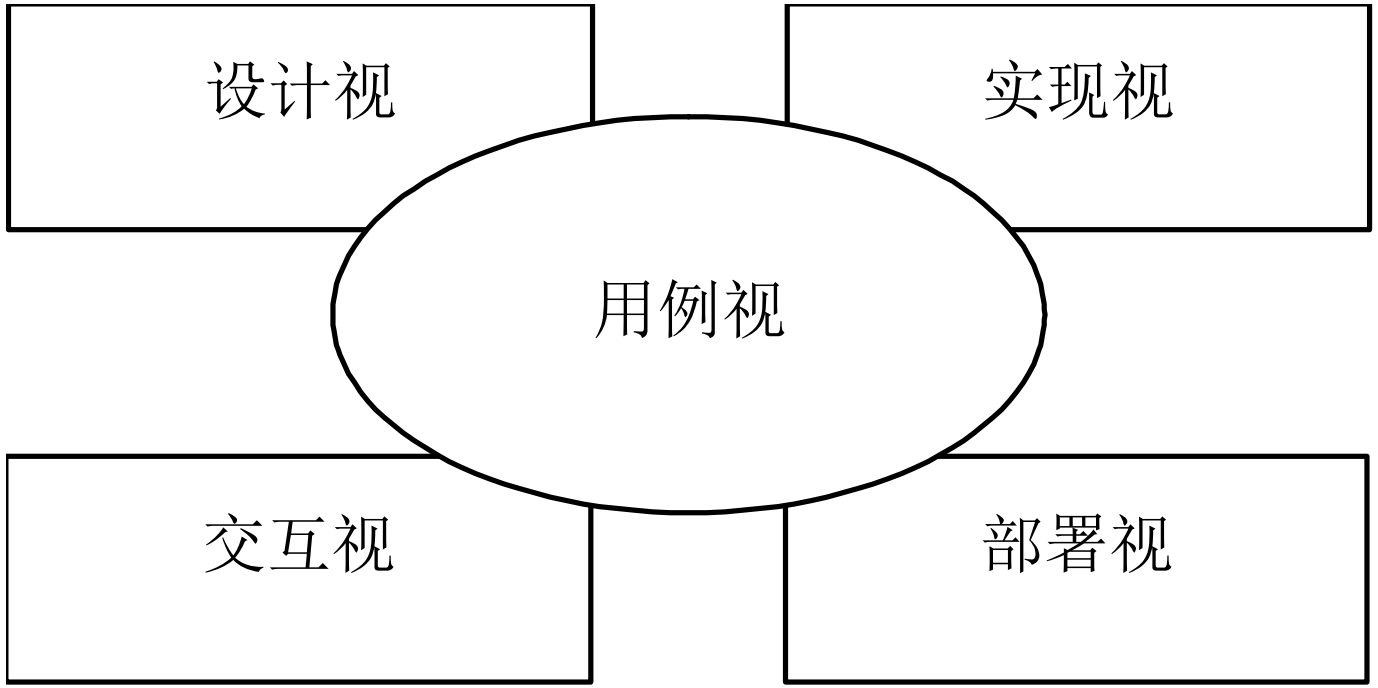
- 架构视图是从特定角度或有利位置对系统进行简化描述（抽象），涵盖特定关注点，并省略与此透视无关的实体
- 架构视图文档过分强调开发的某个方面（即团队组织）或未解决所有利益相关者的关注点
- 软件系统的利益相关者：最终用户、开发人员、系统工程师、项目经理等
- 软件工程师很难在一个蓝图上表示更多内容，因此架构视图文档包含复杂的图表。

解决方案

- 使用多个视图，每个视图用不同的表示解决一个特定的关注点集
- 提供"4+1"视图模型，以解决大型和具有挑战性的架构

视

- 软件系统的体系结构可以用5个视图来描述，每个视图都侧重描述系统的一个方面。



视

- 用例视（Use Case View）
  - 系统的用例视通过用例描述了最终用户、分析人员和测试人员可以看到的系统行为。
- 设计视（Design View）
  - 系统的设计视包括类、接口、和协作，这些类、接口和协作组成了问题域词汇表和解决方案，支持系统的功能需求，也即系统应该提供给最终用户的服务。
- 交互视（Interaction View）
  - 系统的交互视描述了系统不同部分之间的控制流，包括可能的并发和同步机制。它体现了系统的性能、可扩展性、和总处理能力。

视（续）

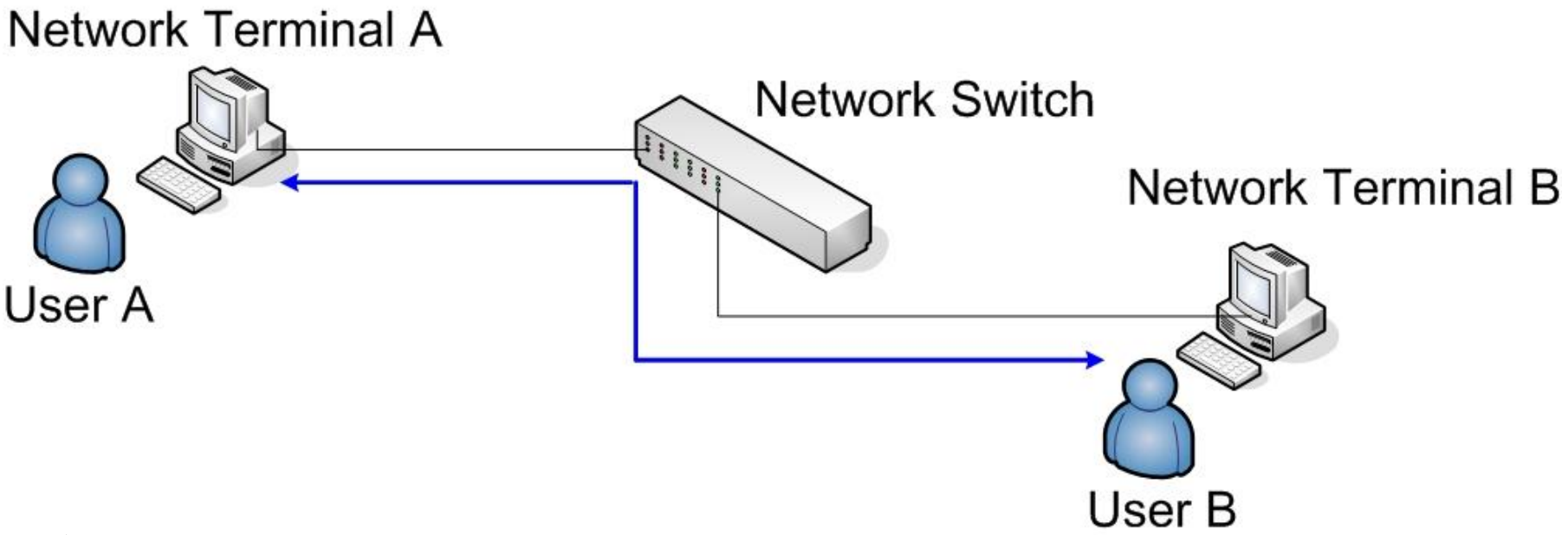
- 实现视（Implementation View）
  - ➢系统的实现视包括用于组装、发布物理软件系统所需的各种产物，主要描述了软件系统版本的配置管理。
- 部署视（Deployment View）
  - ➢部署视包括了构成用于运行软件系统的系统硬件拓扑的节点，主要描述了物理系统组成部分的分布、交付、和安装。

“4+1 视图” 模型

- 模型可以简化以适应上下文
- 不是所有的系统都需要所有的视图:
  - 单处理器: 不需要deployment view
  - 单进程: 不需要process view
  - 小程序: 不需要implementation view
- 添加视图:
  - Data view, security view

“4+1 视图” 模型

- 例子：网络应用程序系统软件架构

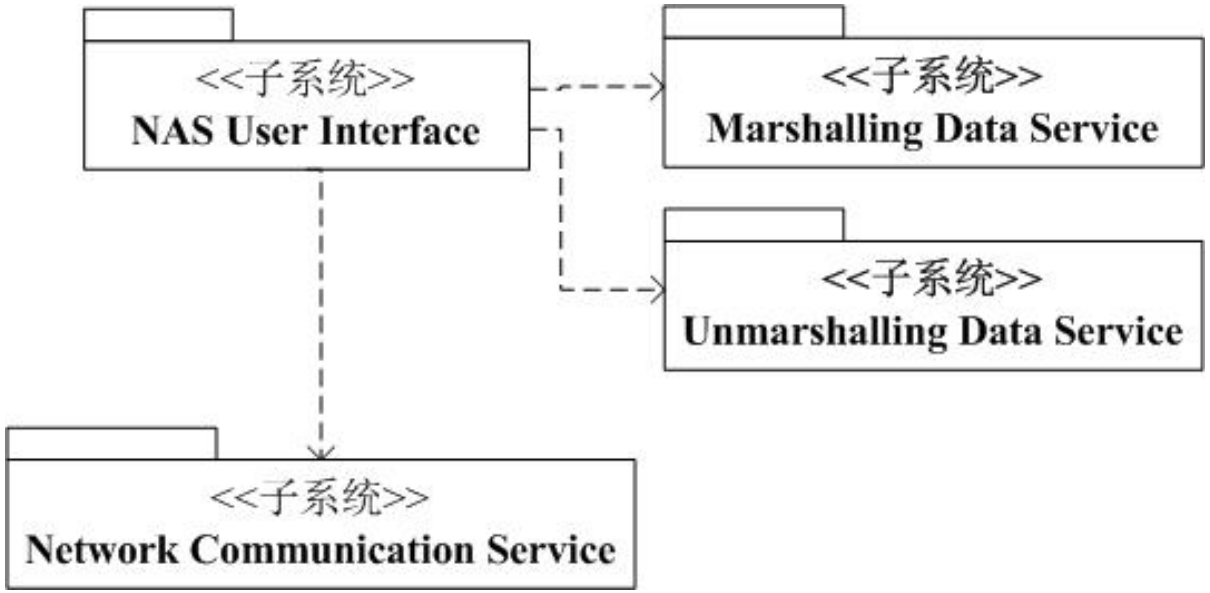


需求:

- Terminals 从用户接收数据.
- Terminal A 格式化用户输入数据，将格式化的数据传输给Terminal B.
- Terminal B解析格式化的数据，将数据展示给用户。

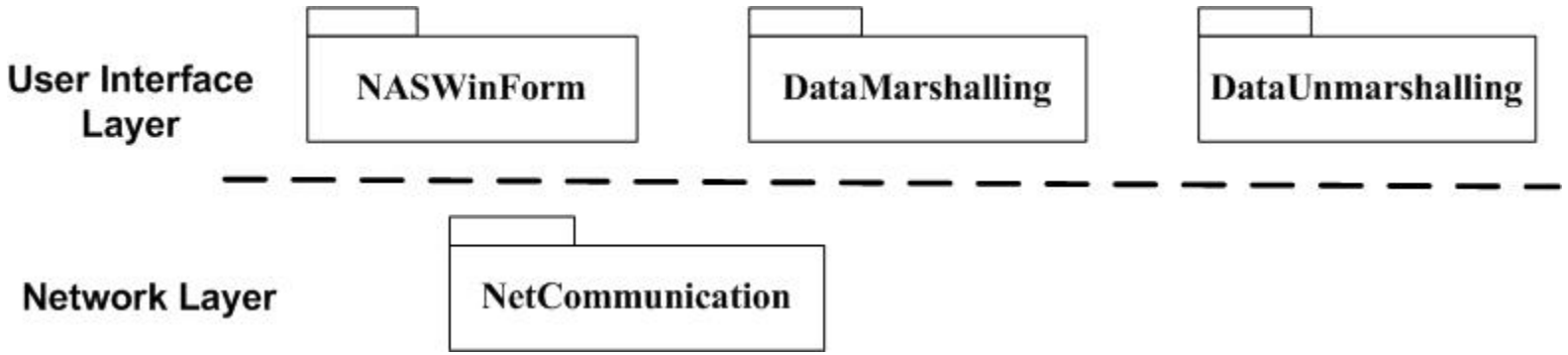
“4+1 视图”模型

- 设计视图
  - 描述系统的功能抽象。主要描述系统的功能模块划分和模块之间的关系。



“4+1 视图” 模型

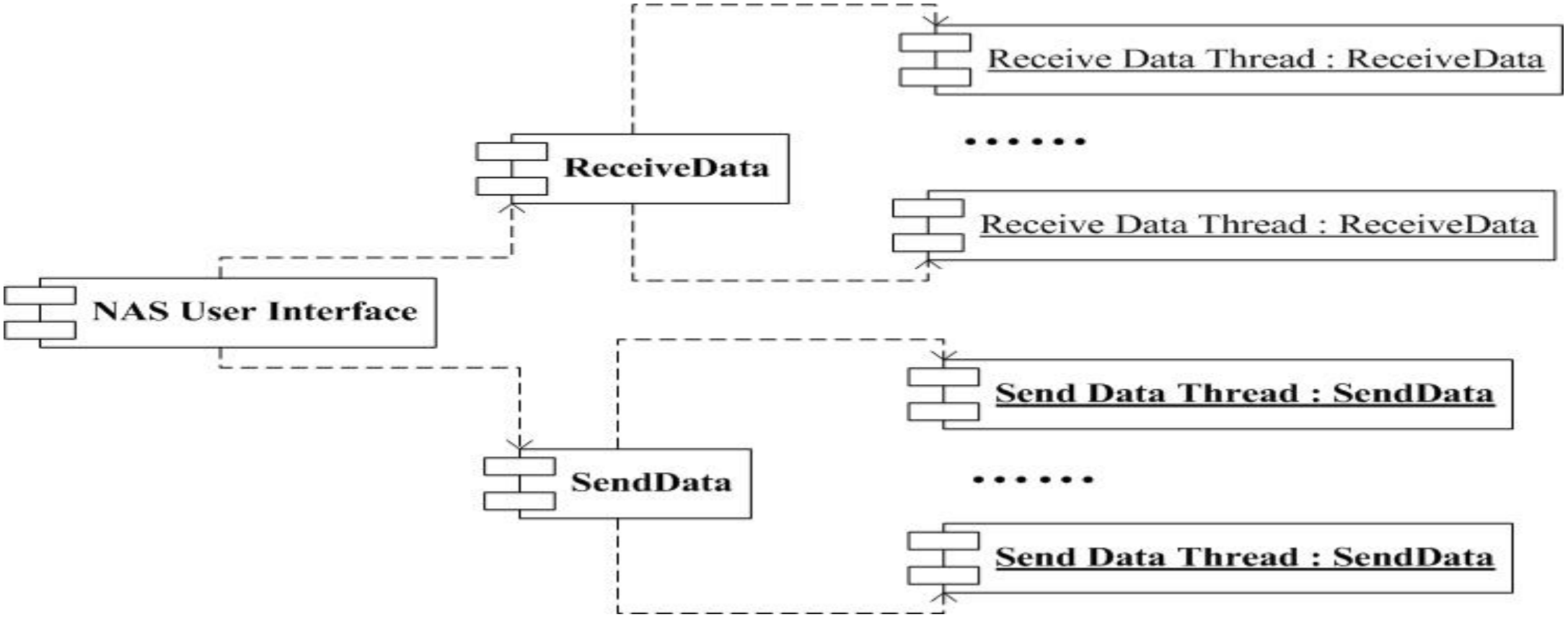
- 实现视图
  - 系统的详细设计和构造抽象。主要给出了系统详细设计和构造的结构



“4+1 视图” 模型

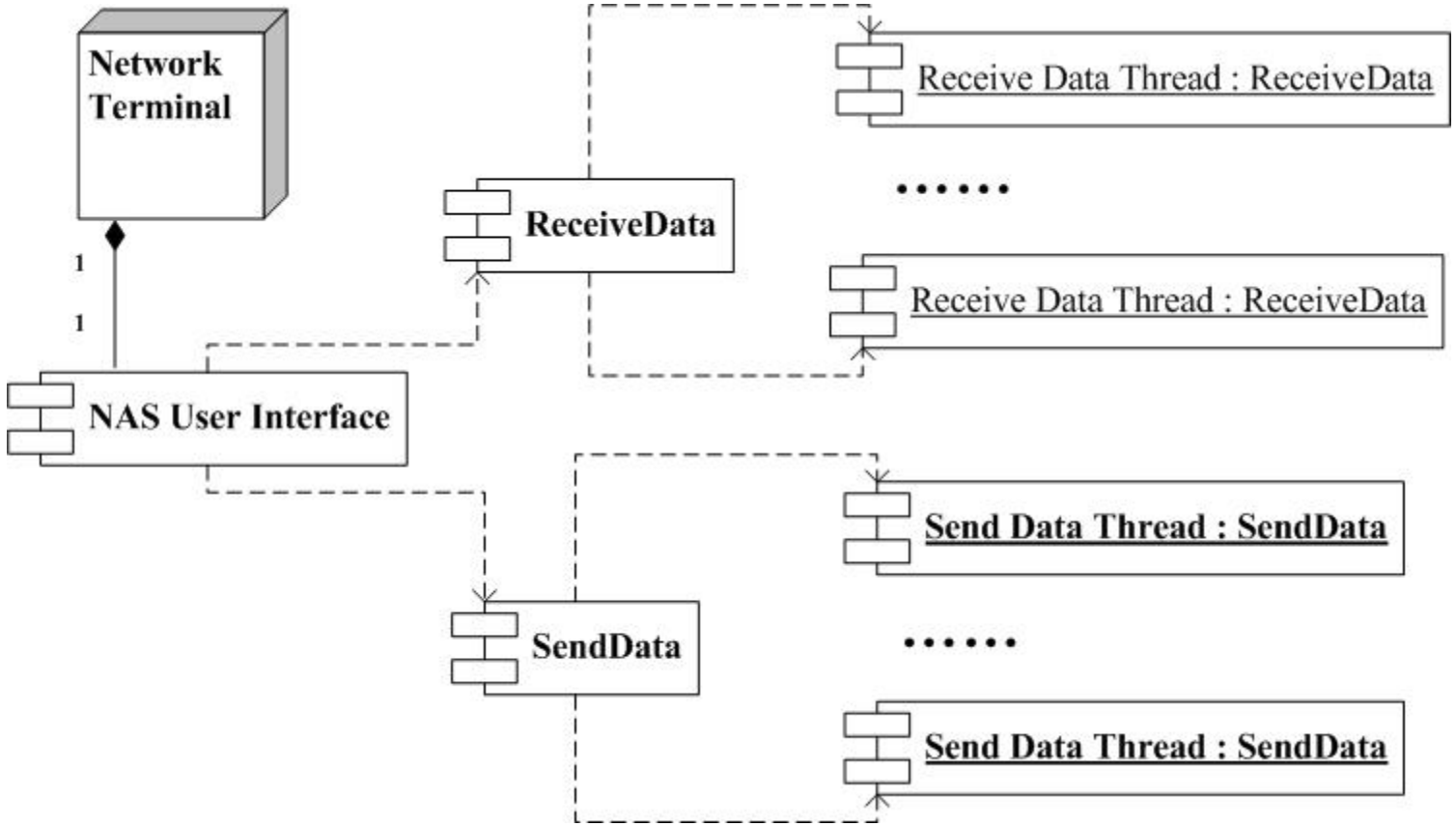
- 交互视图

➢主要描述系统运行时的非功能性属性



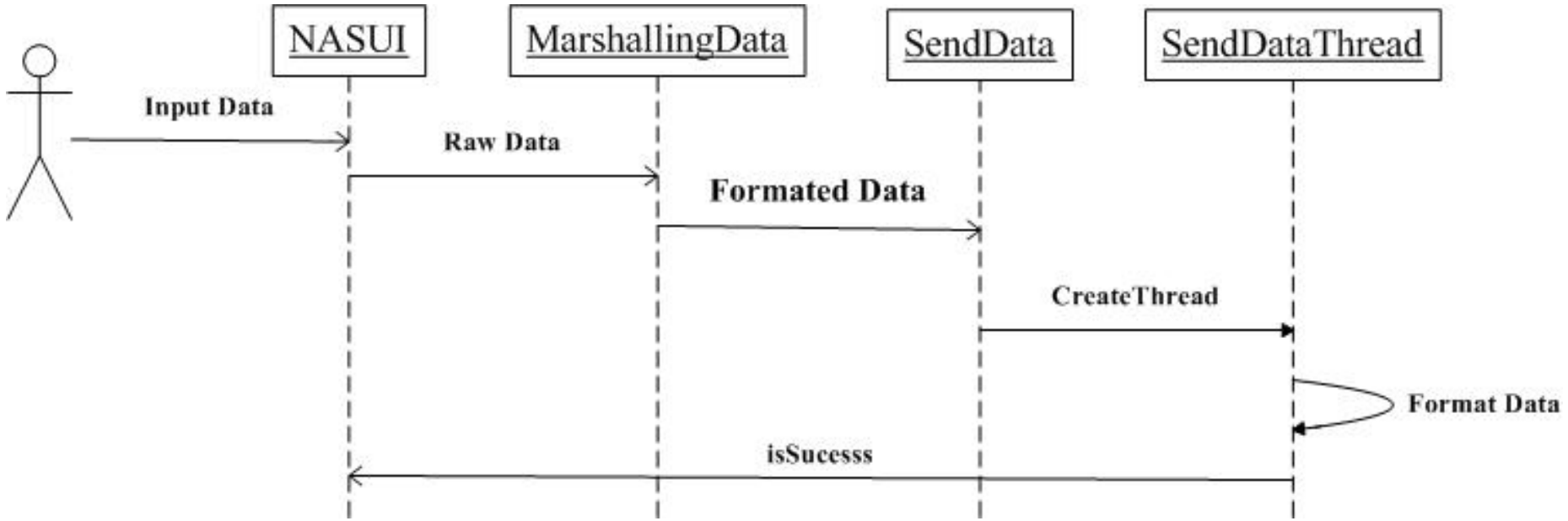
“4+1 视图”模型

- 部署视图
  - 系统物理部署环境的映射关系。



“4+1 视图”模型

- 用例视图
  - ➢描述重要的系统用例。



谢谢！