软件的模型化
建筑学

鄂尔多斯市
北京交通大学

内容

- 建筑视图模型
- 4+1视图模型
- 统一建模语言（UML）
- Rational的4+1视图模型

建筑视图模型

模式的定义

- 模型是描述或类比，以帮助想象无法直接观察到的东西。
  - 模型是对现实的一种简化
  - 模型提供了一个系统的蓝图
  - 模型可能是结构性的，强调系统的组织，也可能是行为性的，强调系统的动态性

模型的功能

- 我们建立模型，以便我们能够更好地理解我们正在创建的系统
- 通过建模，我们实现了四件事。
  - 模型帮助我们想象一个系统的现状或我们希望它成为的样子
  - 模型给了我们一个模板，指导我们构建一个系统
  - 模型允许我们指定一个系统的结构或行为

模型的功能

- 模型记录了我们所做的决定

建筑视图模型

- 架构视图模型是对系统的简化描述（一种抽象），从一个特定的角度或有利位置，涵盖特定的关注点，并省略与此角度无关的实体。

4+1视图模型

关于克鲁赫特的论文

- Philippe Kruchten
  - 在Rational公司（现为IBM所有）担任RUP开发团队的负责人，拥有超过16年的经验。
  - 在工业领域（电信、空中交通管制系统）的宝贵经验，他用这些经验来确认他的模型。
- "4+1视图模型 "论文。

– 根据ACM门户网站，有60次引用

问题

- 架构文件过度强调开发的某个方面（即团队组织），或者没有解决所有利益相关者的关注问题
  - 软件系统的各种利益相关者：最终用户、开发人员、系统工程师、项目经理
- 软件工程师很难在一张蓝图上表现出更多的内容，因此架构文件中包含了复杂

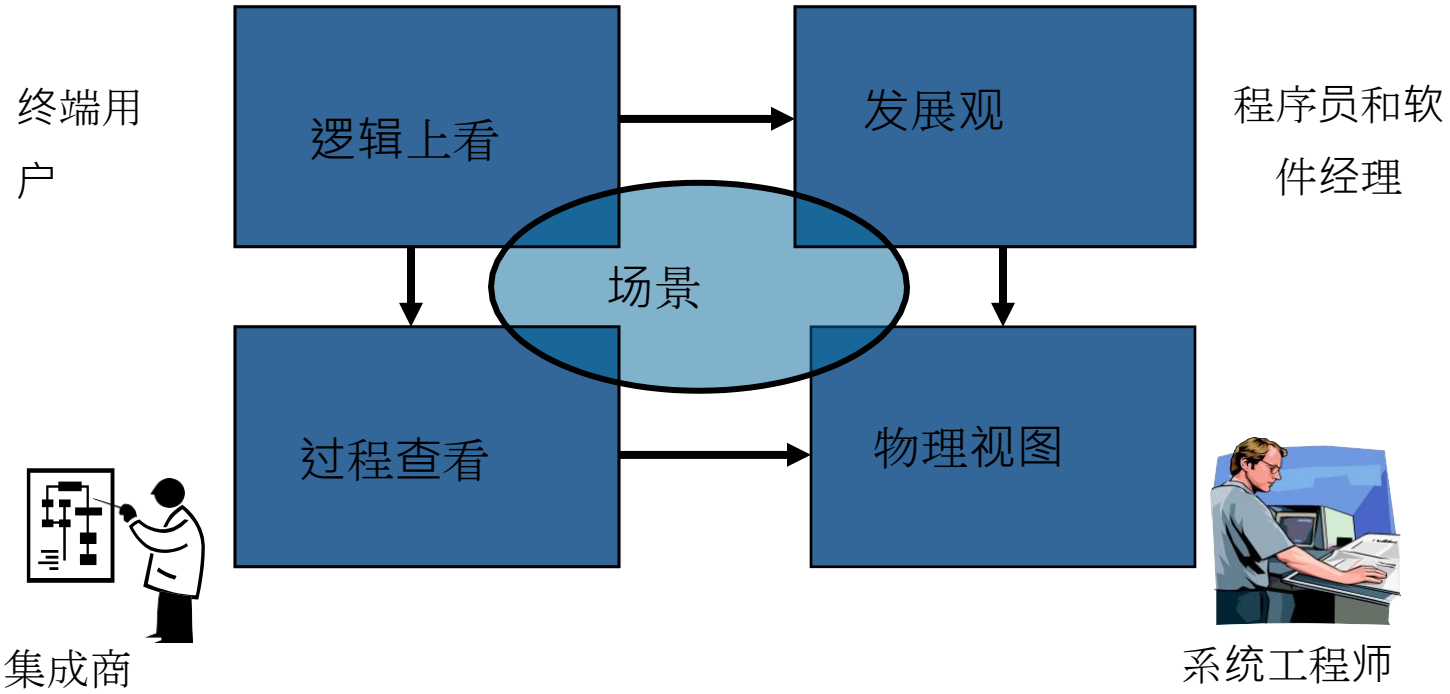
问题

的图示

解决方案

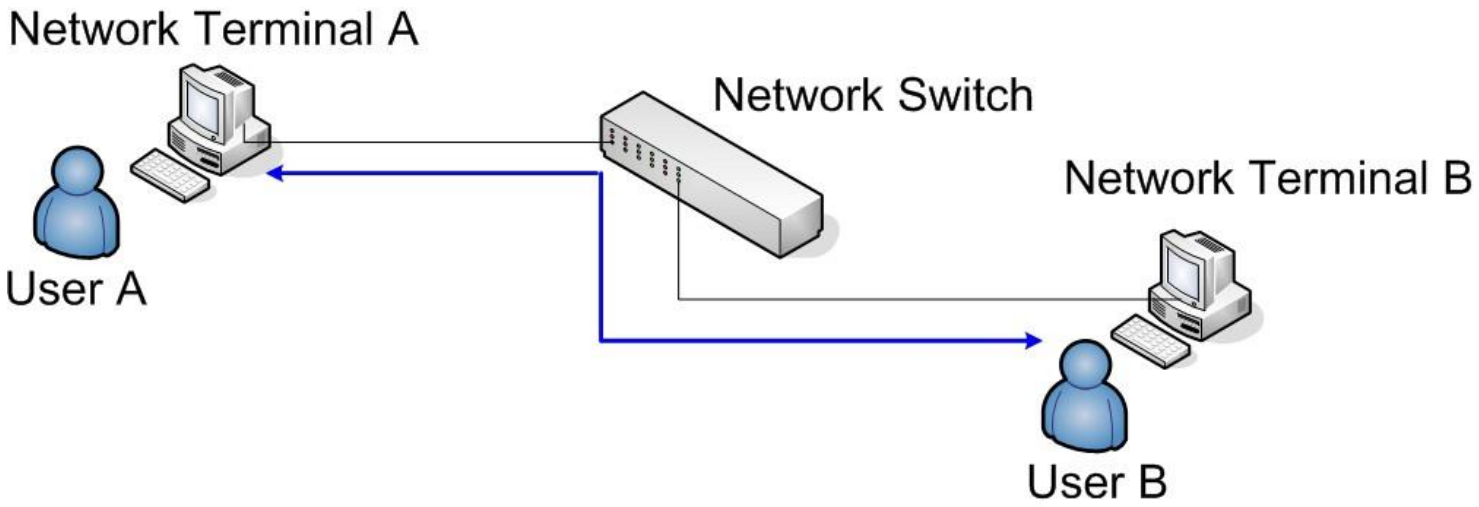
- 使用几个同时存在的观点或视角，用不同的符号，每一个都是为了解决一组具体的问题
- 提出4+1视图模型，以解决大型和具有挑战性的架构问题

4+1视图模型



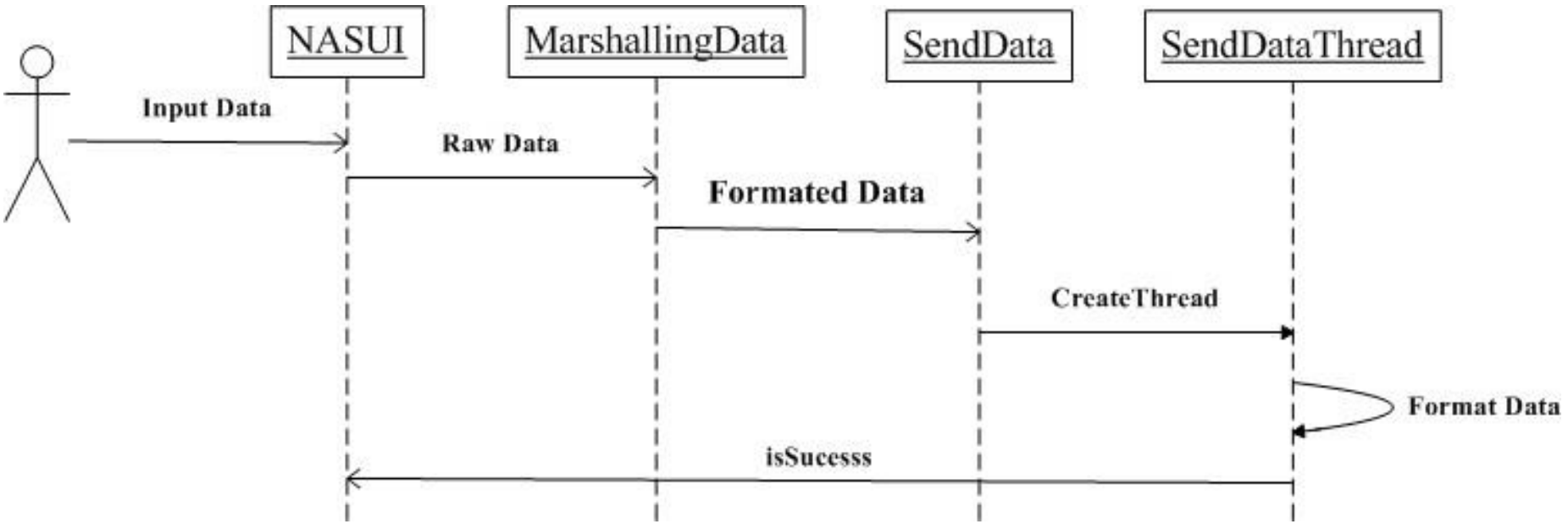
4+1视图模型。例子

- 网络应用系统（NAS）的软件架构
  - 终端接收来自用户的输入数据
  - 终端A对输入数据进行格式化，并通过网络将格式化的数据发送给终端B
  - 终端B对格式化的数据进行解析，并在屏幕上向用户展示它们。



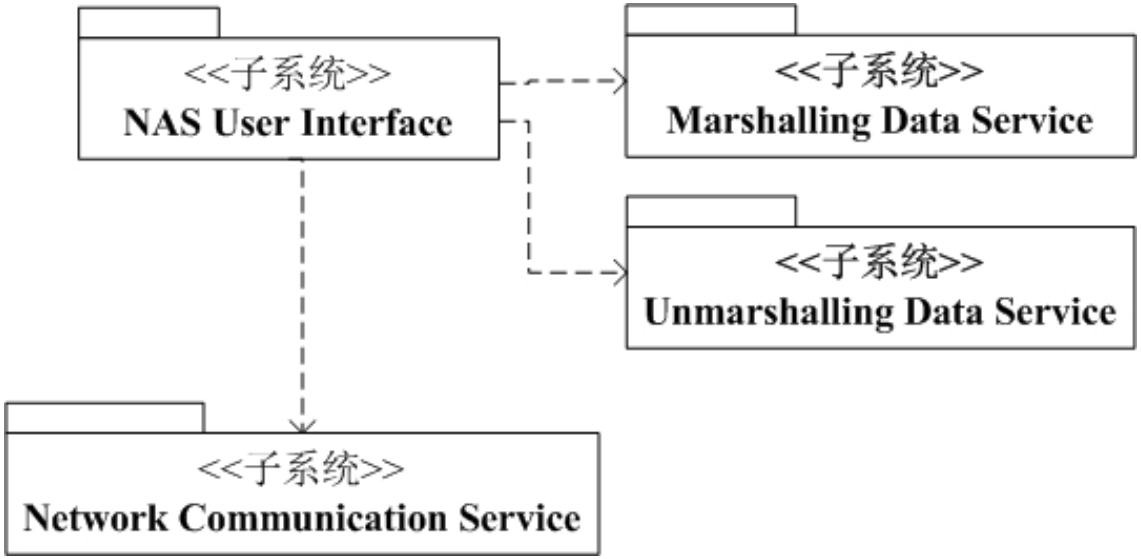
4+1视图模型。例子

- 情景视图
  - 它是用来描述重要的系统用例的



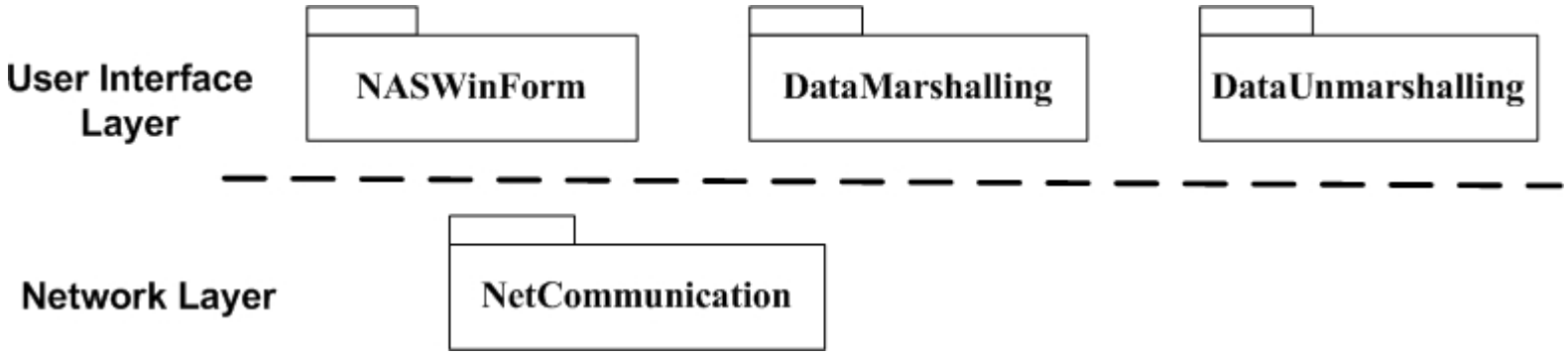
4+1视图模型。例子

- 逻辑视图
  - 系统的功能抽象化。它主要集中在将系统划分为几个功能部件，并描述它们的功能关系。



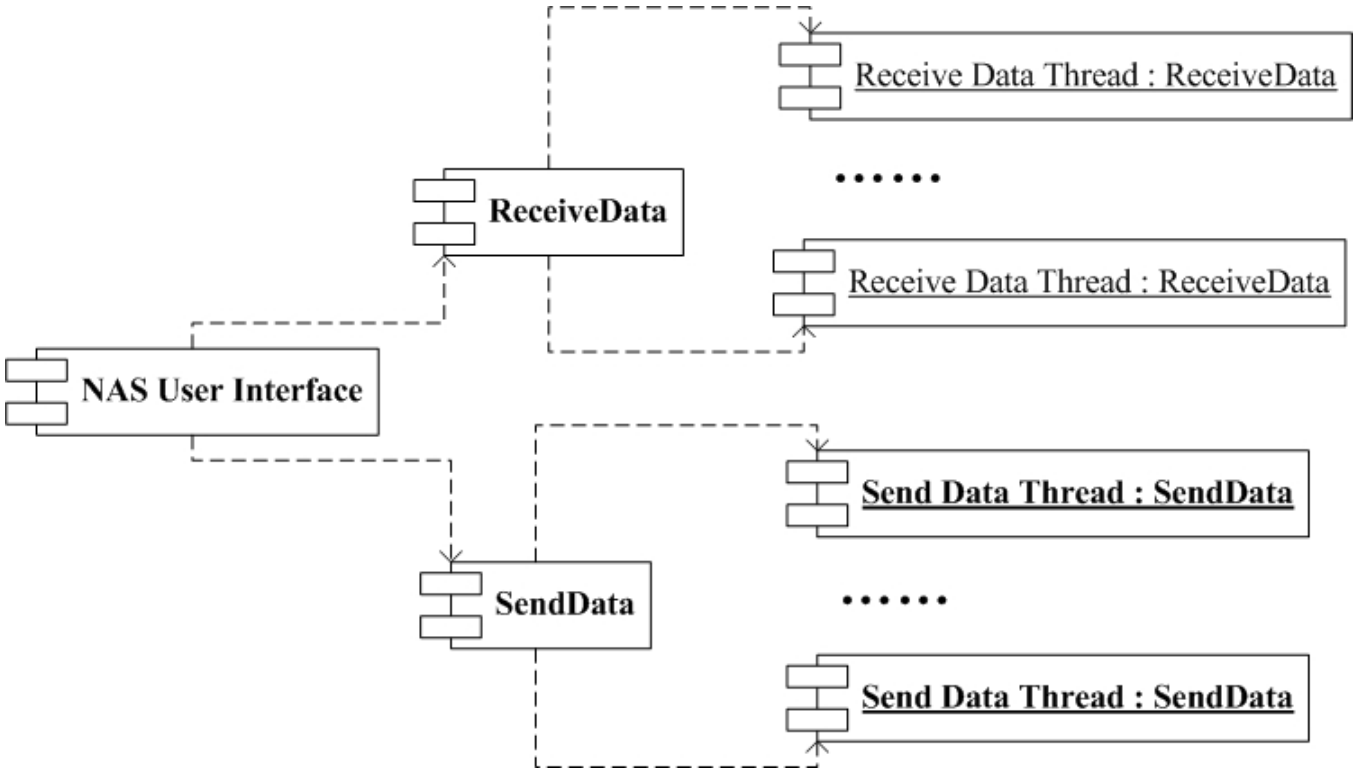
4+1视图模型。例子

- 发展观
  - 系统的详细设计和建设的抽象化。它主要为详细设计和建设提供系统的总体结构。



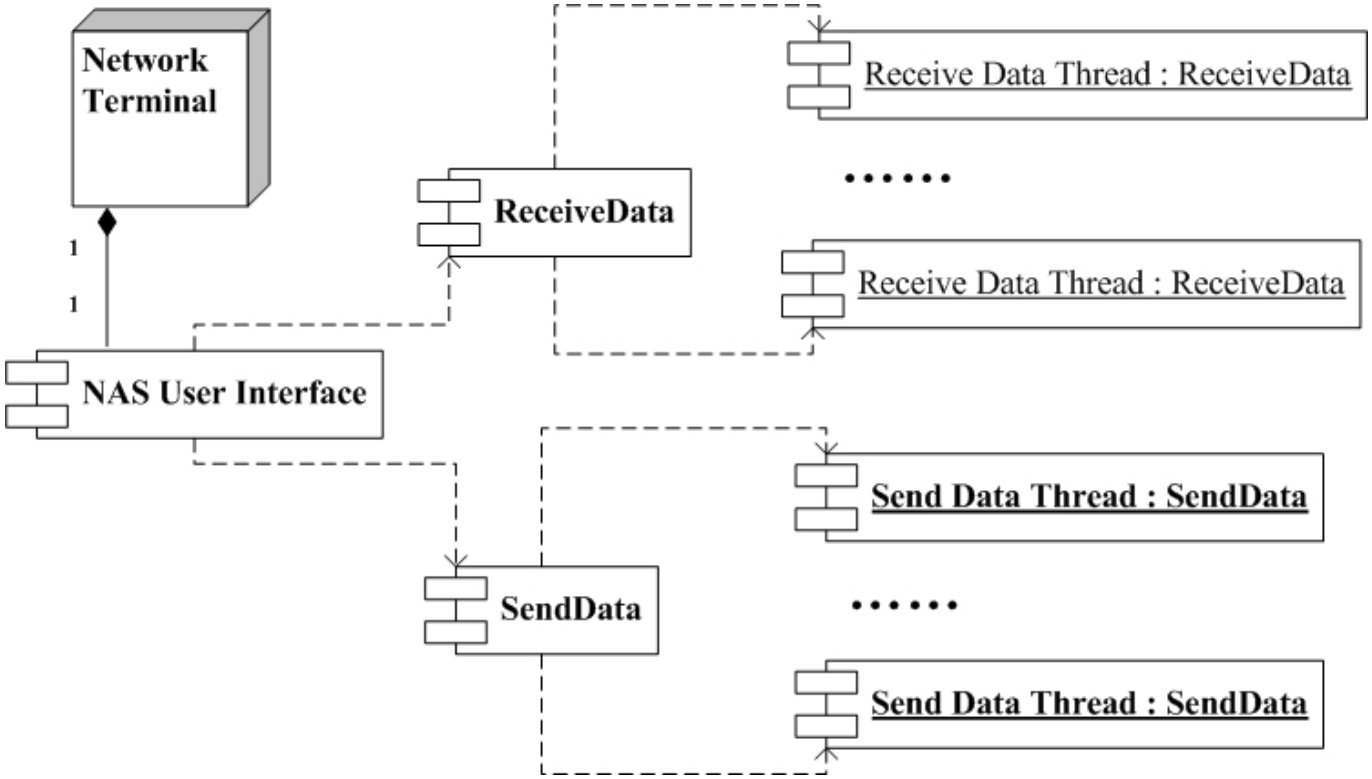
4+1视图模型。例子

- 过程视图
  - 它主要是针对系统的非功能特性和运行时特性。



4+1视图模型。例子

- 物理视图
  - 与系统的物理部署环境的映射关系



统一建模语言
(UML)

面向对象的建模语言

- 面向对象的建模语言开始出现在70年代中期到80年代后期的某个时候，因为方法论者
- 在1989年至1994年期间，面向对象的建模方法的数量从不到10个增加到50多个。
  - Grady Booch的Booch方法，Rational软件公司：在项目的设计和施工阶段特别具有表现力
  - Ivar Jacobson的《面向对象的软件工程》（OOSE），Objectory公司：对商业工程和需求的出色支持
  - James Rumbaugh的对象建模技术（OMT），通用电气公司：用于分析数据密集型信息系统的表现力。

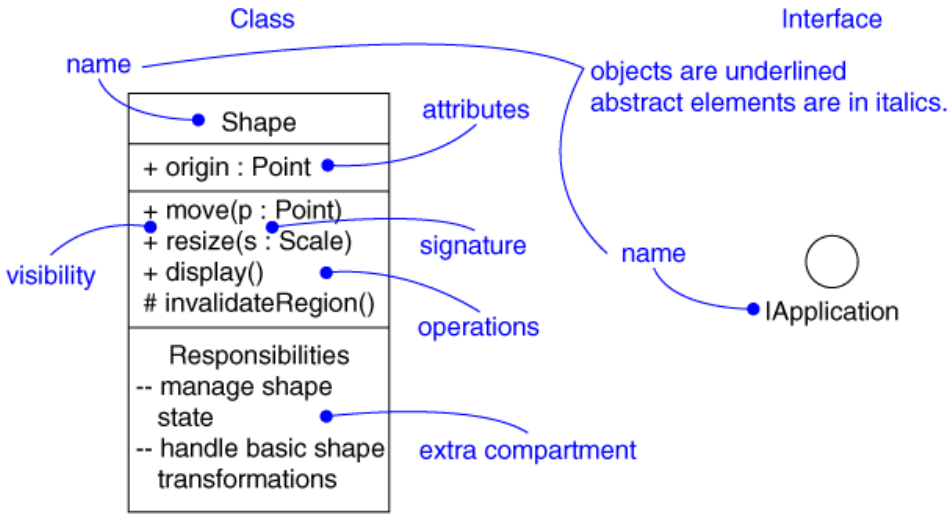
统一建模语言（UML）

- 统一建模语言（UML；来自Rational公司）是一种用于
  - 视觉化
  - 指定
  - 构建
  - 记录

软件密集型系统的人工制品

UML。建模元素

- 结构要素
  - 类，接口，协作，用例，活动类，组件，节点
- 行为要素
  - 交互，状态机
    - 分组要素
    - 包，子系统
- 其他要素
  - 注意事项

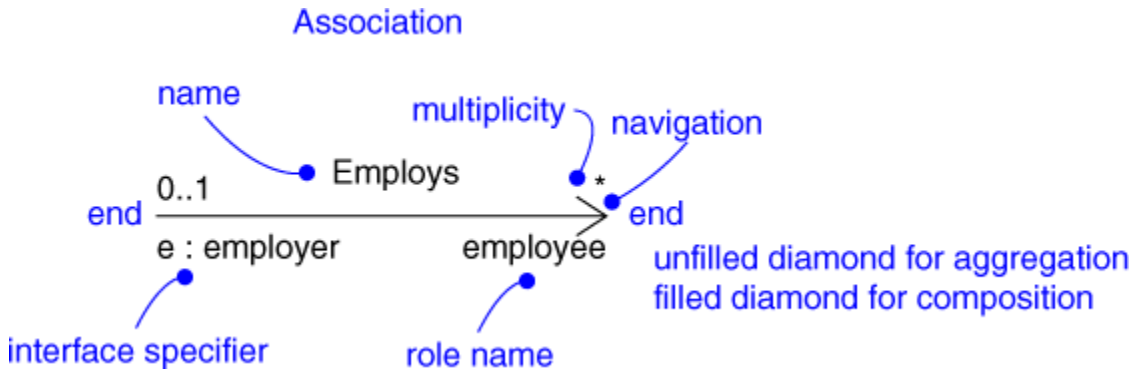


UML。关系

演示者
2017-09-18 14:18:28

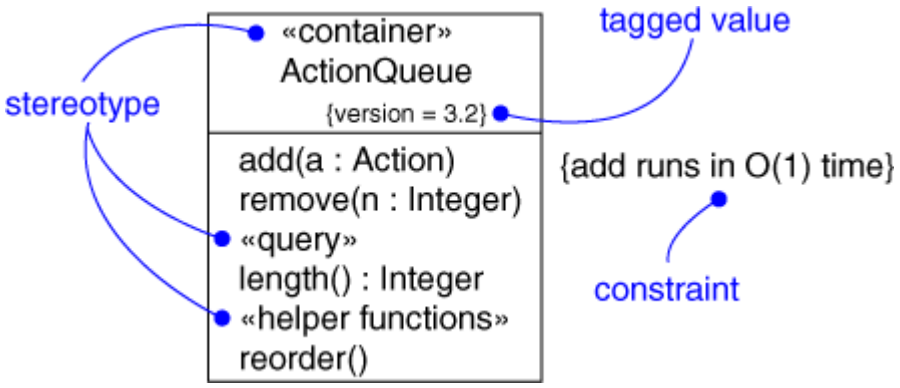
协会。A有B
聚合。A有B的一部分，但B可以在没有A的情况下工作。A有B的一部分， 而B没有A不能工作 依赖性。A需要B

- 归纳
- 实现
- 协会
  - 集合
  - 构成
- 依赖性



UML: 可扩展性机制

- 陈规定型
- 标签 价值
- 拘束



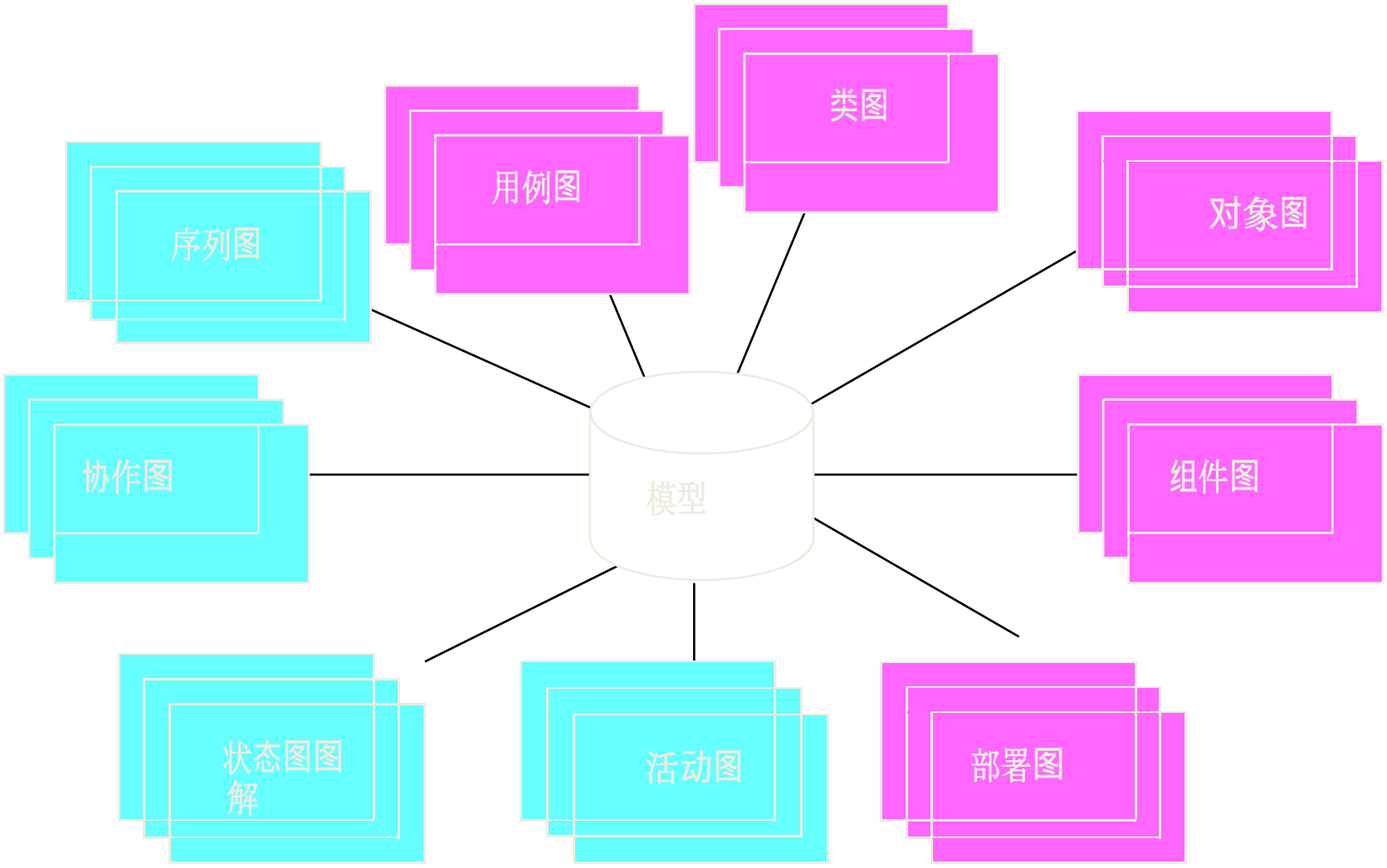
图示

- 一张图就是一个视图模型
  - 从一个特定的利益相关者方面提出来的
  - 提供系统的部分表述
  - 在语义上与其他观点一致
- 在UML中，有13个标准图
  - 静态视图：用例、类、对象、组件、部署、包、复合结构

图示

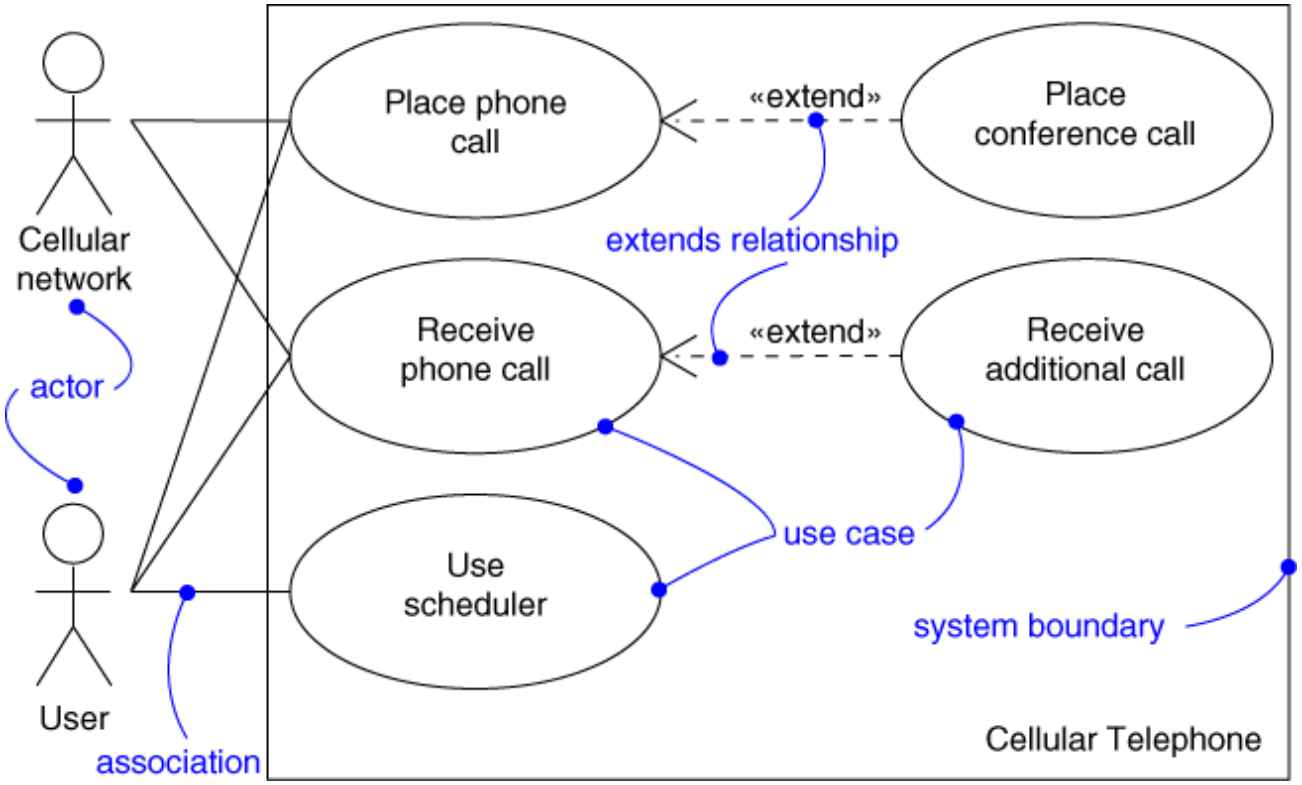
- 动态视图：序列、协作、状态图、活动、计时、互动

图示



用例图

- 捕获用户看到的系统功能



用例图

- 捕获用户看到的系统功能
- 建立在发展的早期阶段
- 宗旨
  - 指定一个系统的背景
  - 捕捉系统的需求
  - 验证一个系统的架构

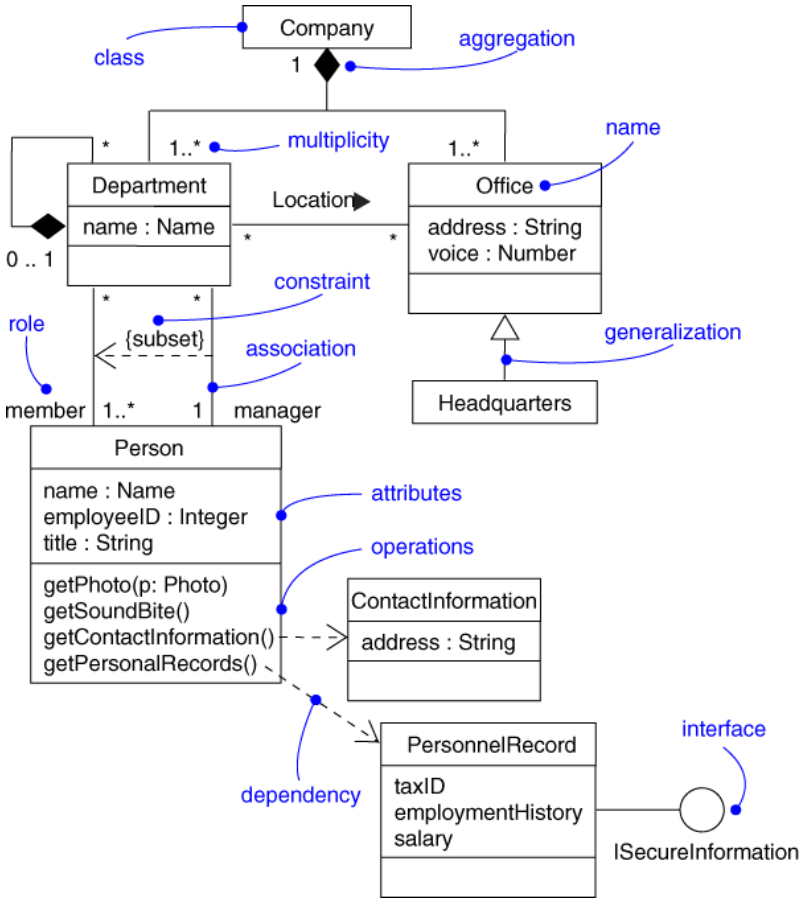
用例图

- 推动实施并生成测试案例

- 由分析员和领域专家开发

类图

类图



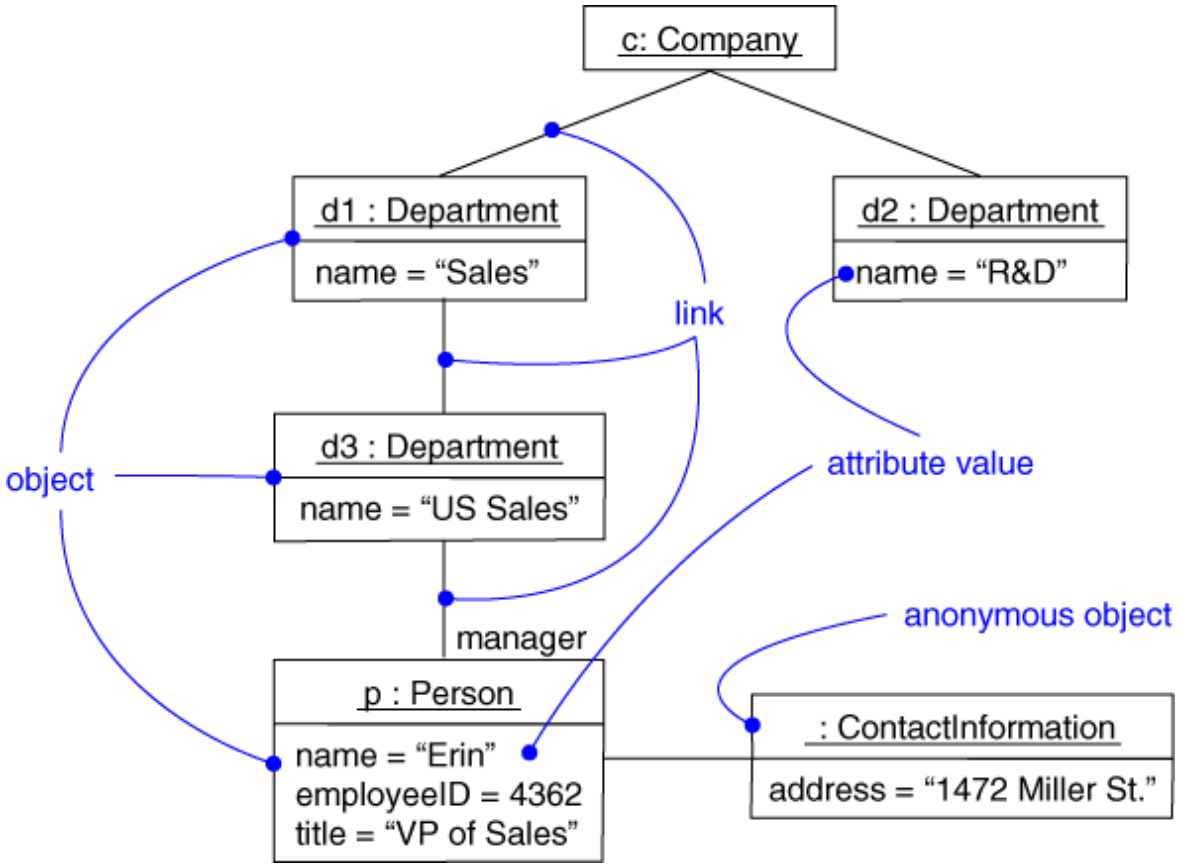
- 捕捉一个系统的词汇

类图

- 捕捉一个系统的词汇
- 在整个开发过程中建立和完善
- 宗旨
  - 对系统中的概念进行命名和建模
  - 明确合作内容
  - 指定逻辑数据库模式
- 由分析员、设计师和实施者开发

对象图

- 捕获实例和链接


c: Company
d1 : Department
name = “Sales”
d2 : Department
name = “R&D”
link
d3 : Department
name = “US Sales”
object
attribute value
manager
anonymous object
p : Person
name = “Erin”
employeeID = 4362
title = “VP of Sales”
: ContactInformation
address = “1472 Miller St.”

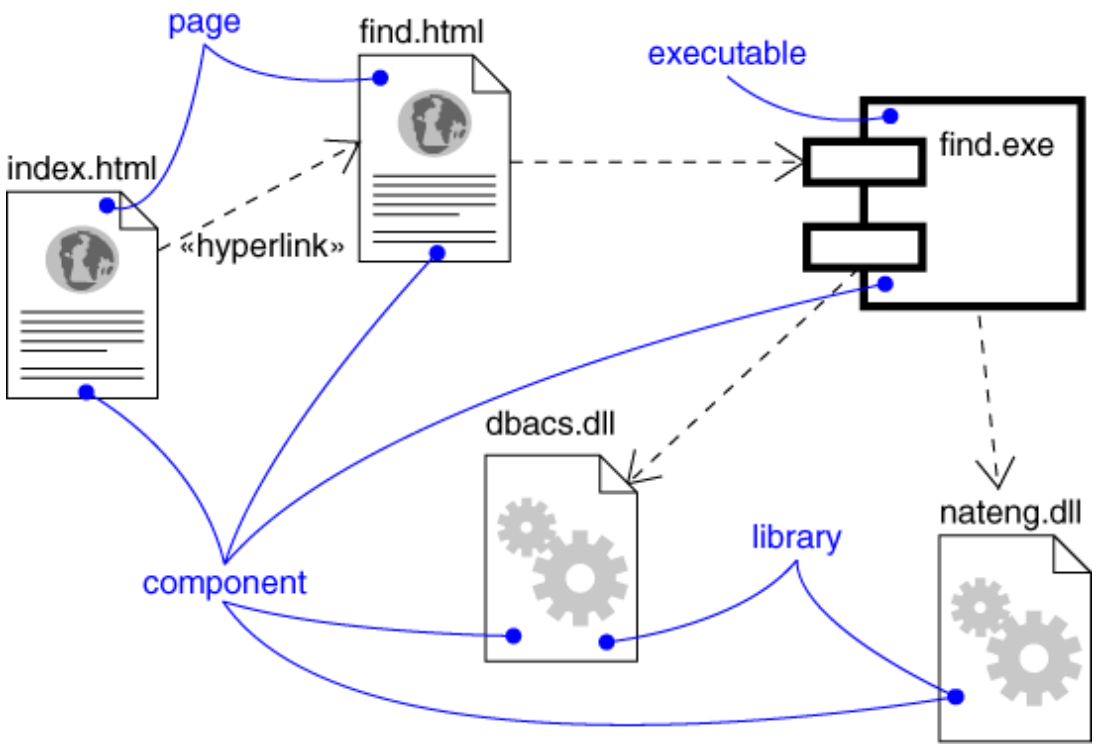
对象图

- 捕获实例和链接
- 在分析和设计期间建立
- 宗旨
  - 说明数据/对象结构
  - 指定快照
- 由分析员、设计师和实施者开发

演示者
2017-09-18 14:18:28

组件图

- 捕捉到实施的物理结构



组件图

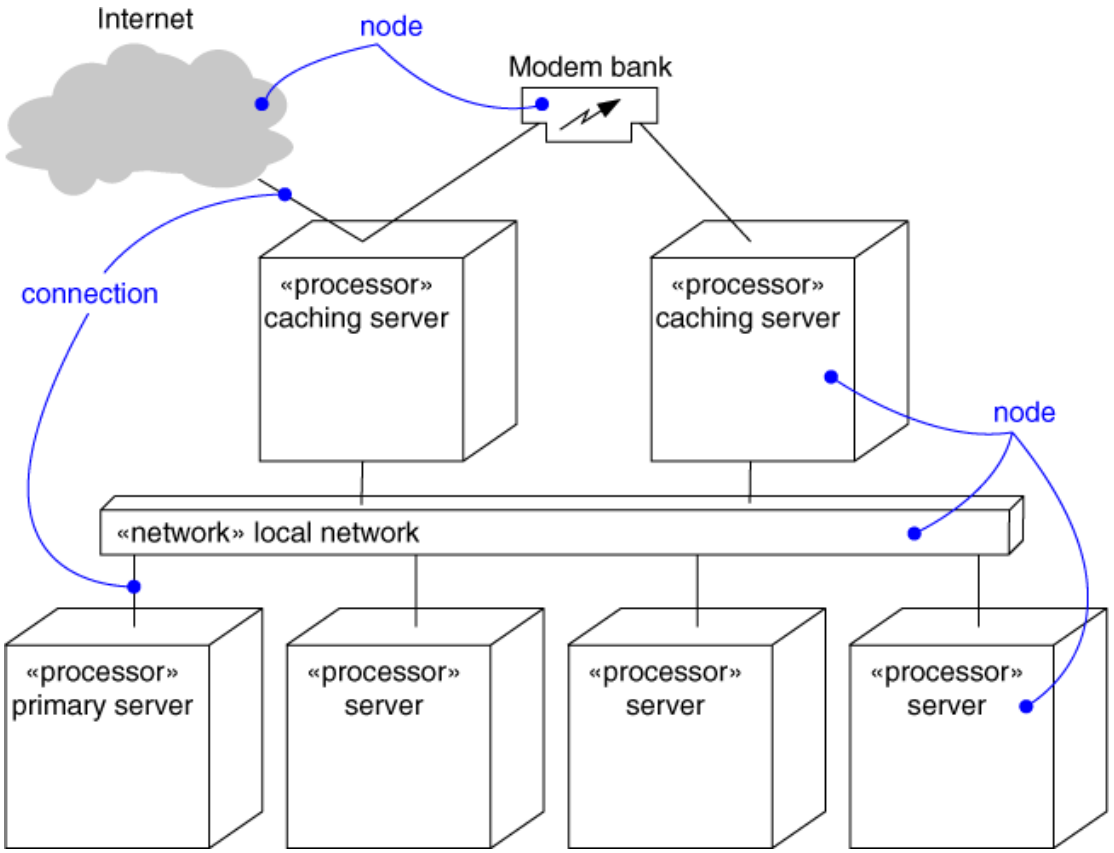
- 捕捉到实施的物理结构
- 作为建筑规格的一部分而建造
- 宗旨
  - 组织源代码
  - 构建一个可执行的版本
  - 指定一个物理数据库

组件图

- 由建筑师和程序员开发

部署示意图

- 捕获一个系统的硬件拓扑结构



部署示意图

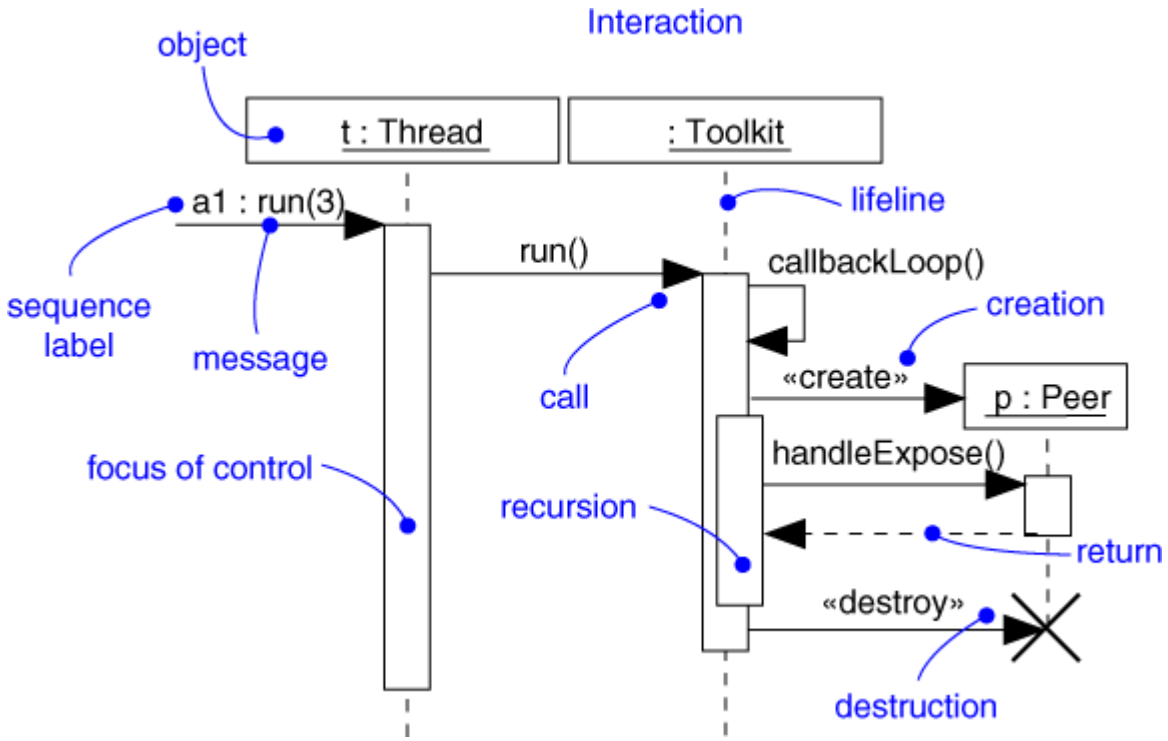
- 捕获一个系统的硬件拓扑结构
- 作为建筑规格的一部分而建造
- 宗旨
  - 指定组件的分布
  - 识别性能瓶颈
- 由建筑师、网络工程师和系统工

部署示意图

程师开发

序列图

- 捕捉动态行为（面向时间）。

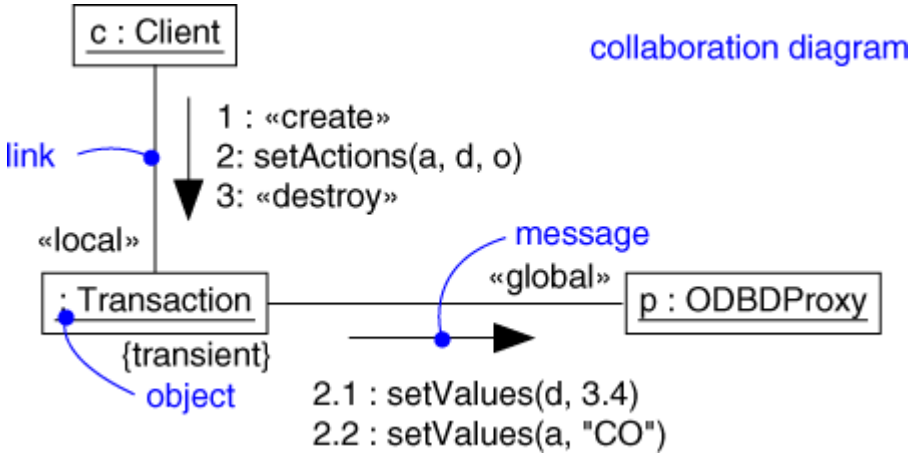


序列图

- 捕捉动态行为（面向时间）。
- 宗旨
  - 控制的模型流程
  - 说明典型情况

协作图

- 捕捉动态行为（面向消息）。

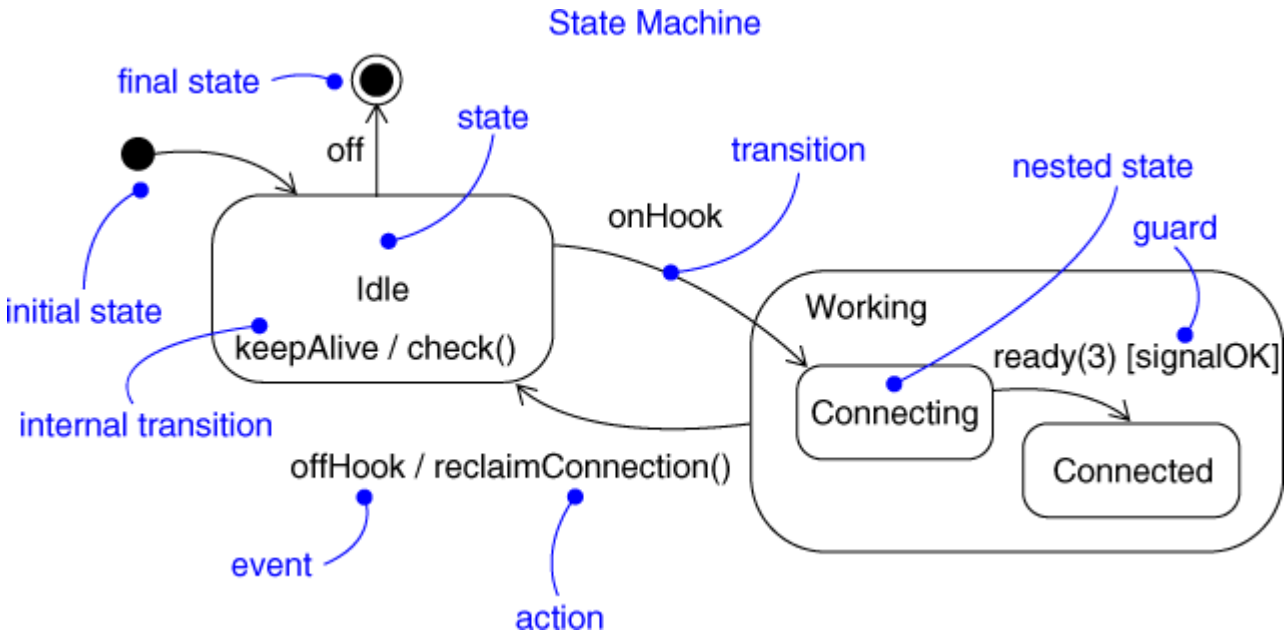


协作图

- 捕捉动态行为（面向消息）。
- 宗旨
  - 控制的模型流程
  - 说明对象结构和控制的协调

状态图

- 捕捉动态行为（面向事件）。

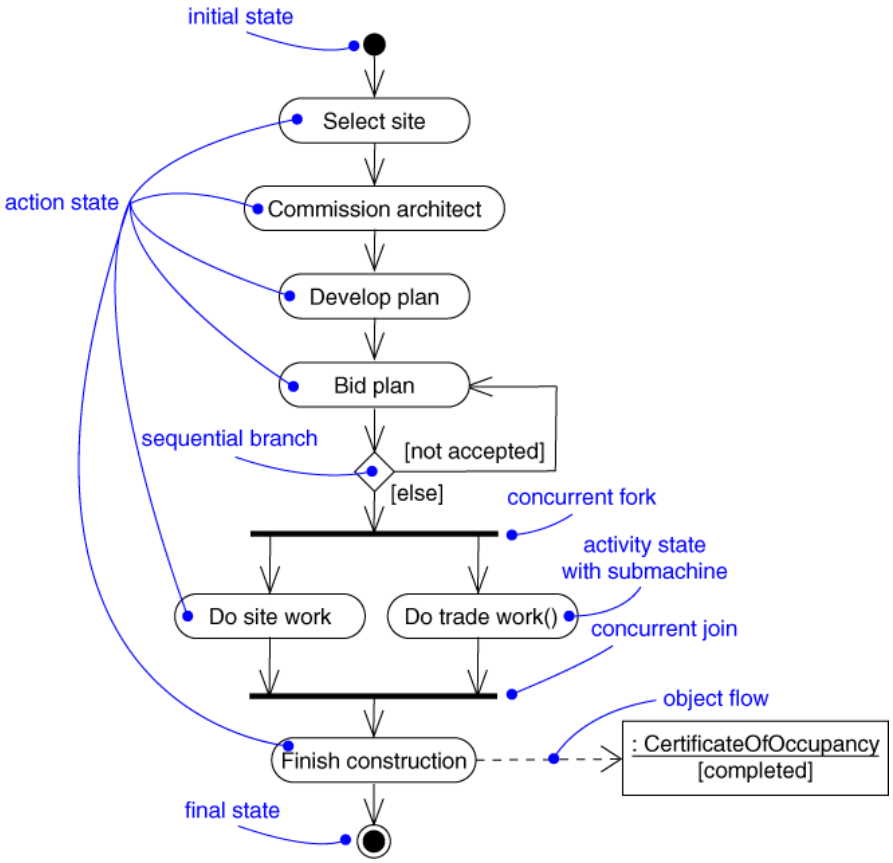


状态图

- 捕捉动态行为（面向事件）。
- 宗旨
  - 模型对象的生命周期
  - 对反应性对象（用户界面、设备等）进行建模

活动图

- 捕捉动态行为（面向活动）。

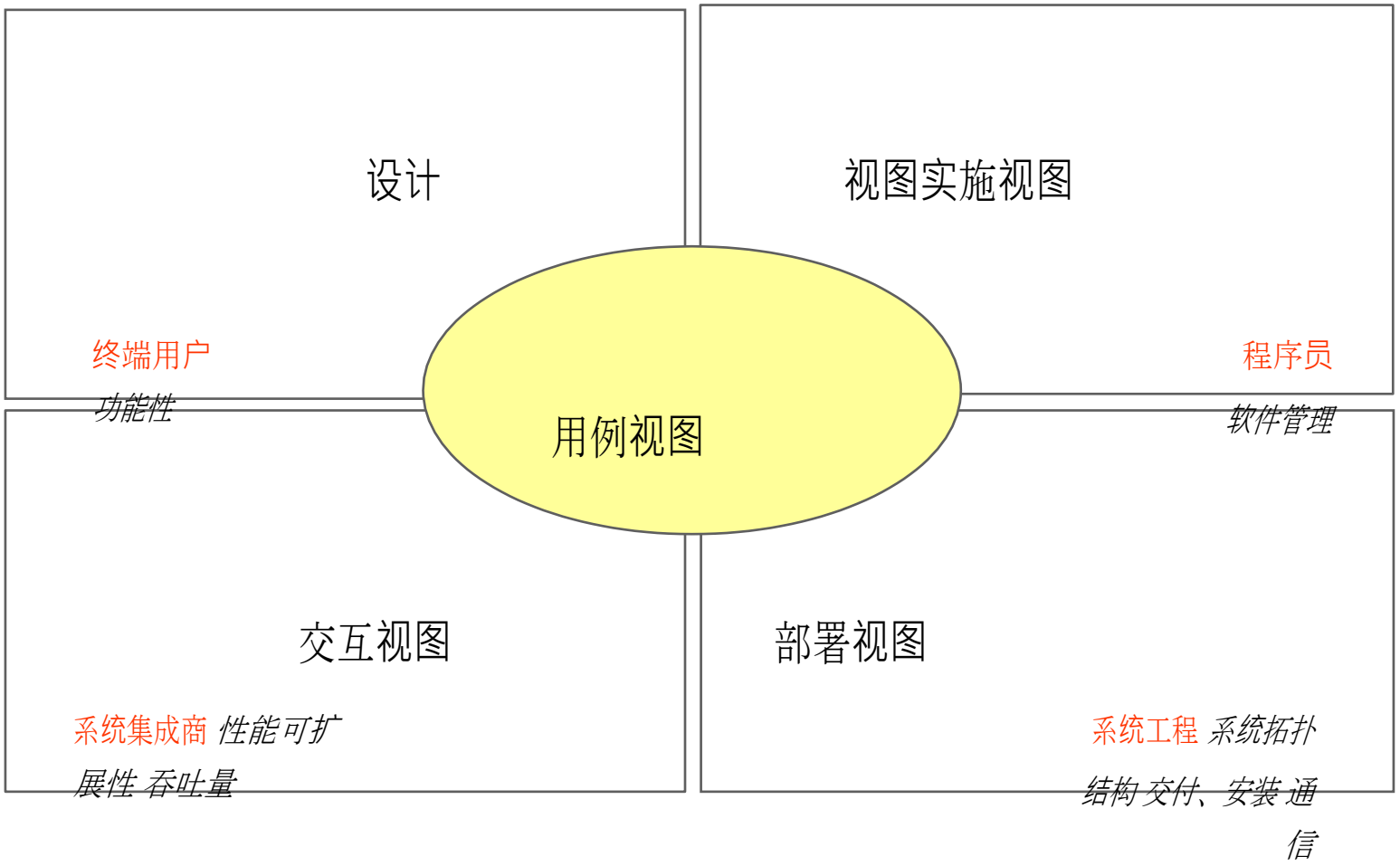


活动图

- 捕捉动态行为（面向活动）。
- 宗旨
  - 业务工作流程模型
  - 模型操作

Rational的4+1视图模型

Rational的4+1视图模型



用例视图

- 包括描述终端用户、分析员和测试员所看到的系统行为的用例。
- 存在的目的是指定形成系统结构的力量
- 静态方面在用例图中得到体现
- 这种观点的动态方面在协作图、状态图和活动图中得以体现。

设计视图

- 涵盖了构成问题及其解决方案词汇的类、接口和协作
- 主要支持系统的功能需求，即系统应向其终端用户提供的服务
- 在类图和对象图中捕获静态方面的信息。
- 动态方面在协作图、状态图和活动图中得到体现。

交互视图

- 显示其各部分之间的控制流，包括可能的并发和同步机制
- 主要解决系统的性能、可扩展性和吞吐量问题
- 在与设计视图相同的图表中捕获静态和动态方面，但重点是控制系统的活动类和它们之间的信息流动。

实施意见

- 涵盖了用于组装和发布物理系统的工件。
- 主要解决系统发布的配置管理，由一些独立的组件组成，可以通过各种方式组装，产生一个运行的系统。
- 静态方面被记录在工件图中
- 动态方面在协作图、状态图和活动图中得到体现。

部署视图

- 包括构成系统的硬件拓扑结构的节点，系统在这些节点上执行
- 主要解决构成物理系统的部件的分配、交付和安装问题
- 静态方面在部署图中得到体现
- 动态方面在协作图、状态图和活动图中得到体现。

Rational的4+1视图模型

- 不是所有的系统都需要所有的视图
  - 单一处理器：放弃部署视图
  - 单一进程：放弃交互视图
  - 非常小的程序：放弃执行视图
- 添加视图
  - 数据视图、安全视图