



北航计算机学院
School of Computer Science & Engineering

如何采用UML 进行软件体系结构设计

张莉 教授
北京航空航天大学
软件工程研究所
lily@buaa.edu.cn



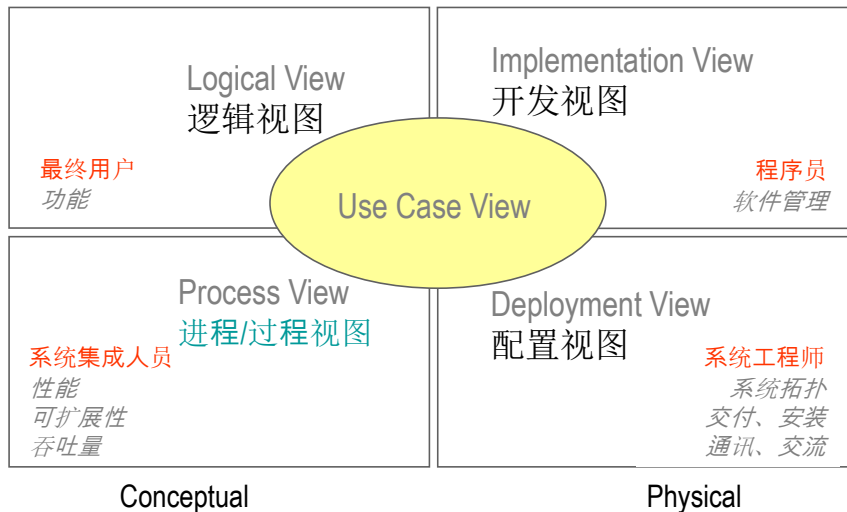
如何采用UML进行软件体系结构设计

内容

- ◆ 著名的“4+1”模型、统一软件开发过程
- ◆ 软件体系结构设计过程
 - 需求捕获
 - 逻辑视图设计
 - 并发视图设计
 - 过程视图设计
 - 构件视图和部署视图设计
- ◆ 渐进式开发过程
- ◆ 案例分析与设计

“4+1”模型

- ◆ 是Rational公司的Philippe Kruchten在1995年提出的用于描述的“4+1”模型



北京航空航天大学软件工程研究所 lily@buaa.edu.cn

“4+1”模型（续）

- ◆ **用例视图**：从系统外部执行者的角度理解、展示系统功能。
- ◆ **逻辑视图**：从系统的静态结构和动态行为的角度展示系统内部功能性设计。
- ◆ **构件视图**：展示代码构件的组织结构。
- ◆ **进程/并发视图**：展示系统的并发性，刻画并发系统中的通讯和同步问题。
- ◆ **配置视图**：展示系统的物理的体系结构，其中用到的计算机和各种设备称作节点。

北京航空航天大学软件工程研究所 lily@buaa.edu.cn



How many views?

- ◆ Simplified models to fit the context
- ◆ Not all systems require all views:
 - Single processor: drop deployment view
 - Single process: drop process view
 - Very Small program: drop implementation view
- ◆ Adding views:
 - Data view, security view

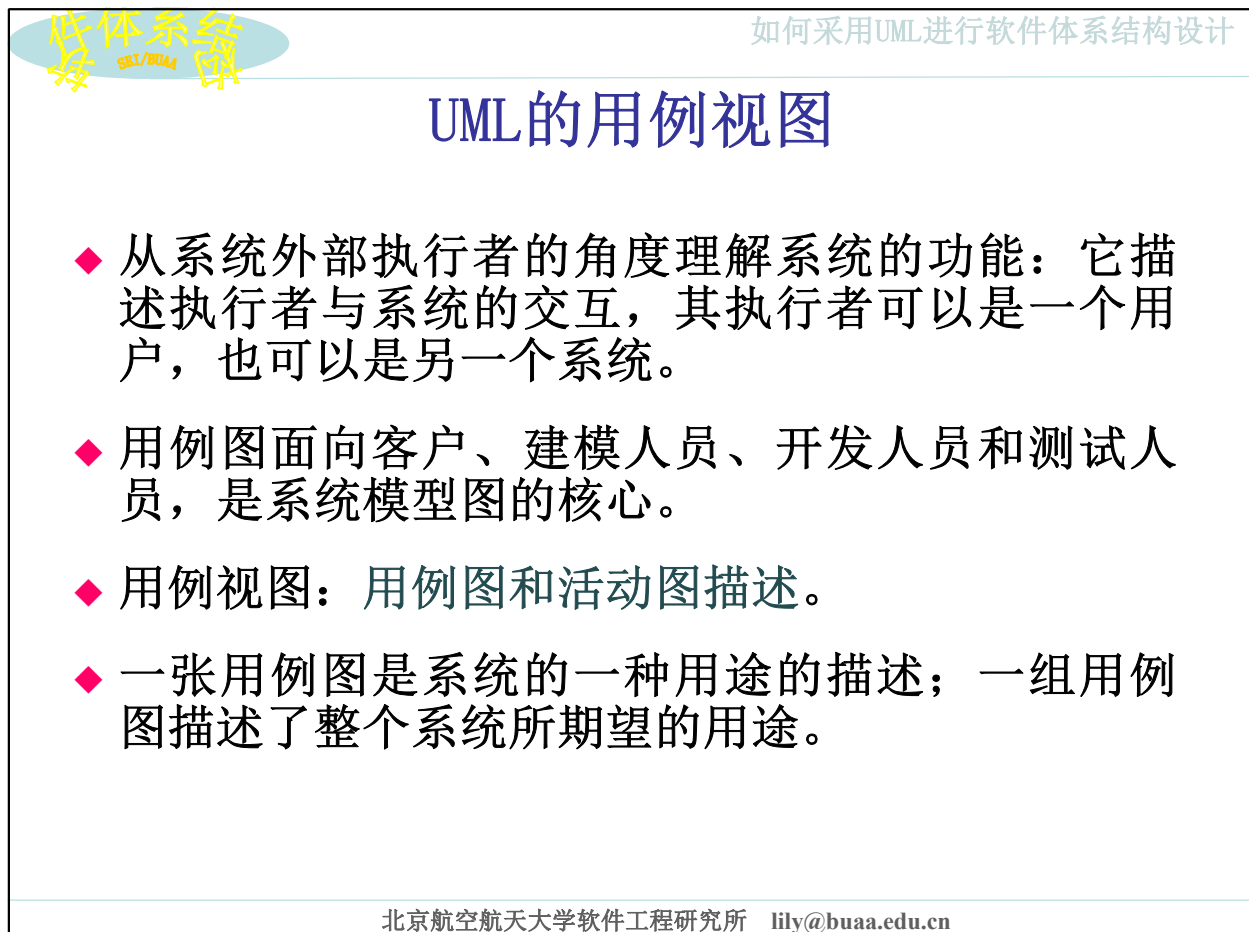
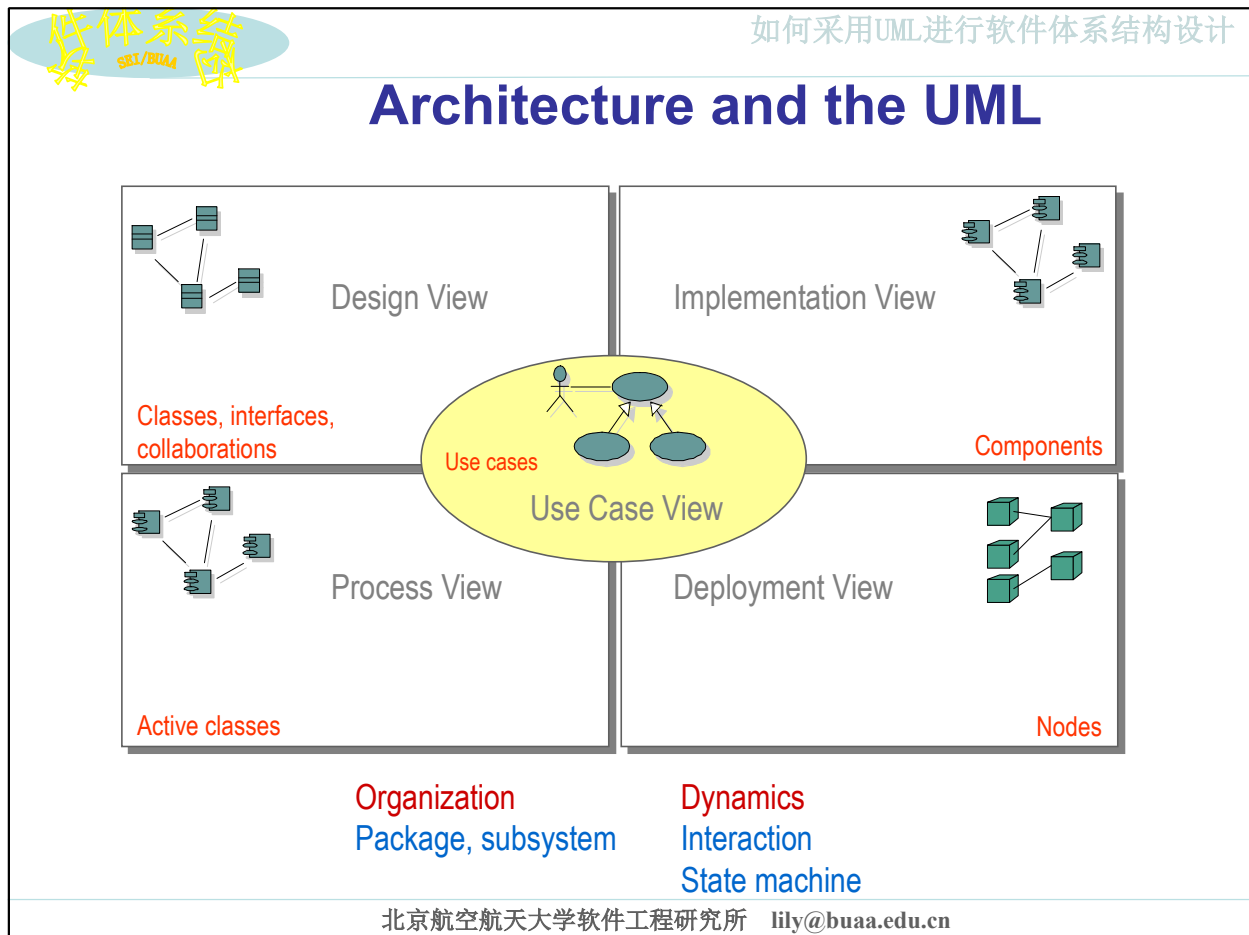
北京航空航天大学软件工程研究所 lily@buaa.edu.cn



The Value of the UML

- ◆ Is an open standard
- ◆ Supports the entire software development lifecycle
- ◆ Supports diverse applications areas
- ◆ Is based on experience and needs of the user community
- ◆ Supported by many tools

北京航空航天大学软件工程研究所 lily@buaa.edu.cn





UML的逻辑视图

- ◆ 描述系统如何提供要求的功能：
 - 静态结构：
 - 类(界面，内部结构)；对象；关系。
 - 用类图 and 对象图描述。
 - 动态行为：
 - 对象之间相互发送消息而引发的动态合作关系。
 - 用状态图、顺序图、合作图 and 活动图描述。
 - 各种特性：永久性，并发性。
- ◆ 主要是面向设计人员和开发人员。



UML的构件视图

- ◆ 描述实现的各个模块及其相互之间的关系：
 - 模块的内部结构；
 - 模块间的依赖关系；
 - 关于各构件的一些附加信息：
 - 资源分配(构件的责任)；
 - 其它管理信息，如开发工作的进展报告等。
- ◆ 主要面向开发人员。
- ◆ 构件视图：由构件图组成，其中构件表示不同类型的代码模块。



UML的并发视图

- ◆ 系统中进程和处理器的划分。
- ◆ 属于非功能性特性：
 - 着眼于资源的有效利用；
 - 并行执行，并发地执行多线程控制，处理线程 之间的通信和同步；
 - 处理来自外部环境的同步事件。
- ◆ 面向开发人员和系统集成人员。
- ◆ 描述并发视图的主要模型图：
 - 动态图：状态图，合作图，活动图。
 - 实现图：构件图，配置图。

北京航空航天大学软件工程研究所 lily@buaa.edu.cn



UML的配置视图

- ◆ 展示系统的物理配置，如计算机和其它设备(统称结点)及其相互之间的连接关系，包括在物理的体系结构中如何配置构件的映像关系。
- ◆ 面向开发、集成和测试人员。
- ◆ 配置视图用配置图描述。

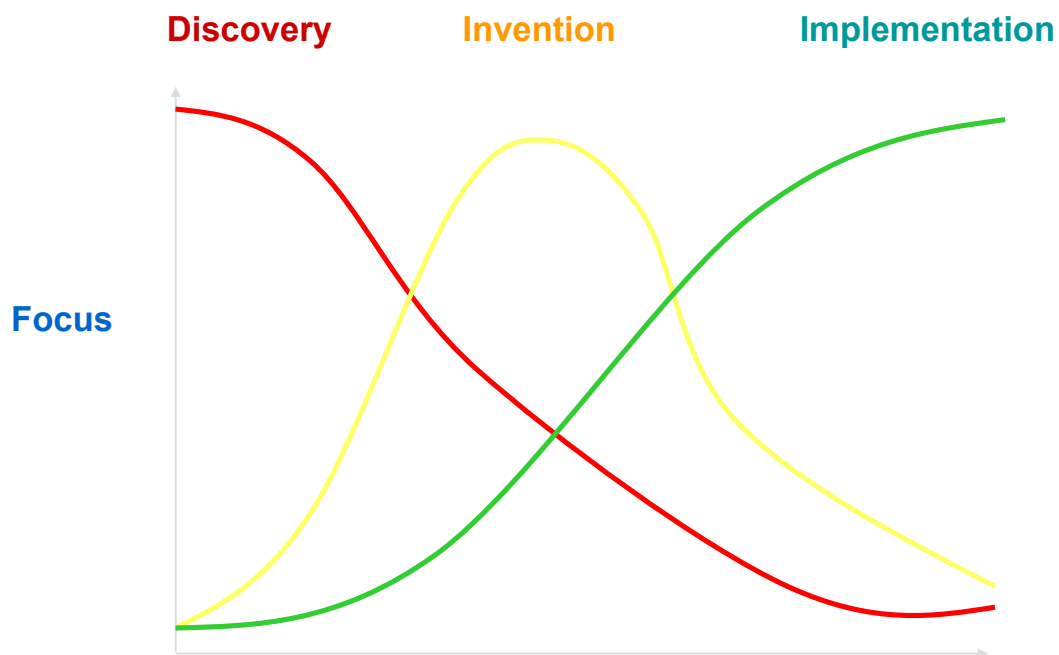
北京航空航天大学软件工程研究所 lily@buaa.edu.cn

统一软件开发过程：USDP

- ◆ 基于4+1视图
- ◆ Iterative
- ◆ Architecture-centric
- ◆ Use-case driven
- ◆ Risk confronting
- ◆ 增量式的迭代过程
- ◆ 以体系结构为中心的过程
- ◆ 用例驱动的过程
- ◆ 抵御风险的过程

北京航空航天大学软件工程研究所 lily@buaa.edu.cn

Focus over time



北京航空航天大学软件工程研究所 lily@buaa.edu.cn



标准过程的生命周期阶段视图



时间 →

初始阶段：项目范围（Scope）、效益，开发事务实例（Business Case）

细化阶段：建立项目计划，描述系统特征，捕获详细需求，建立基线体系结构

构造阶段：在基线体系结构的基础上建立系统

移交阶段：确认测试，用户测试，向用户移交最终的软件

北京航空航天大学软件工程研究所 lily@buaa.edu.cn



Major Milestones



time →



Vision



Baseline
Architecture

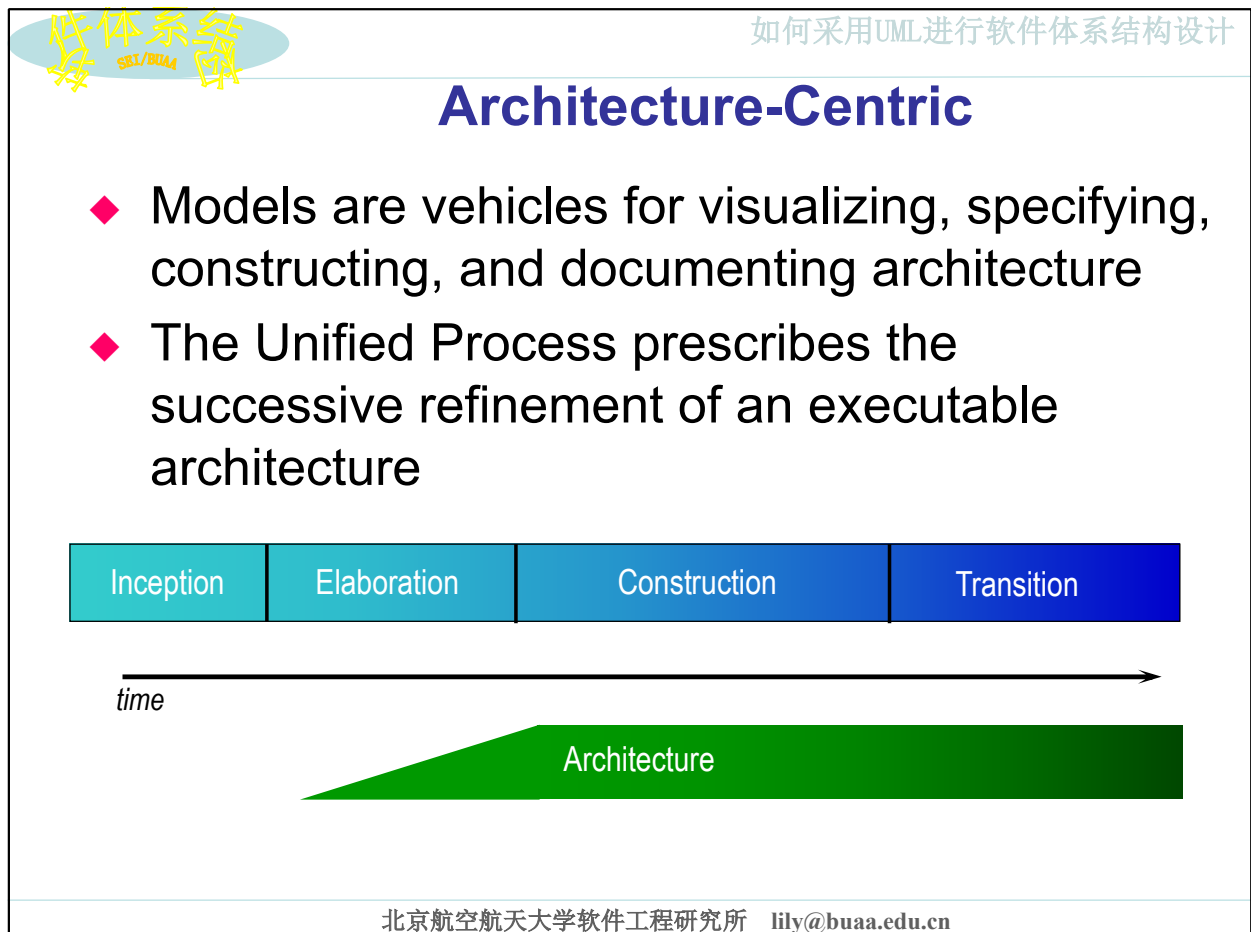
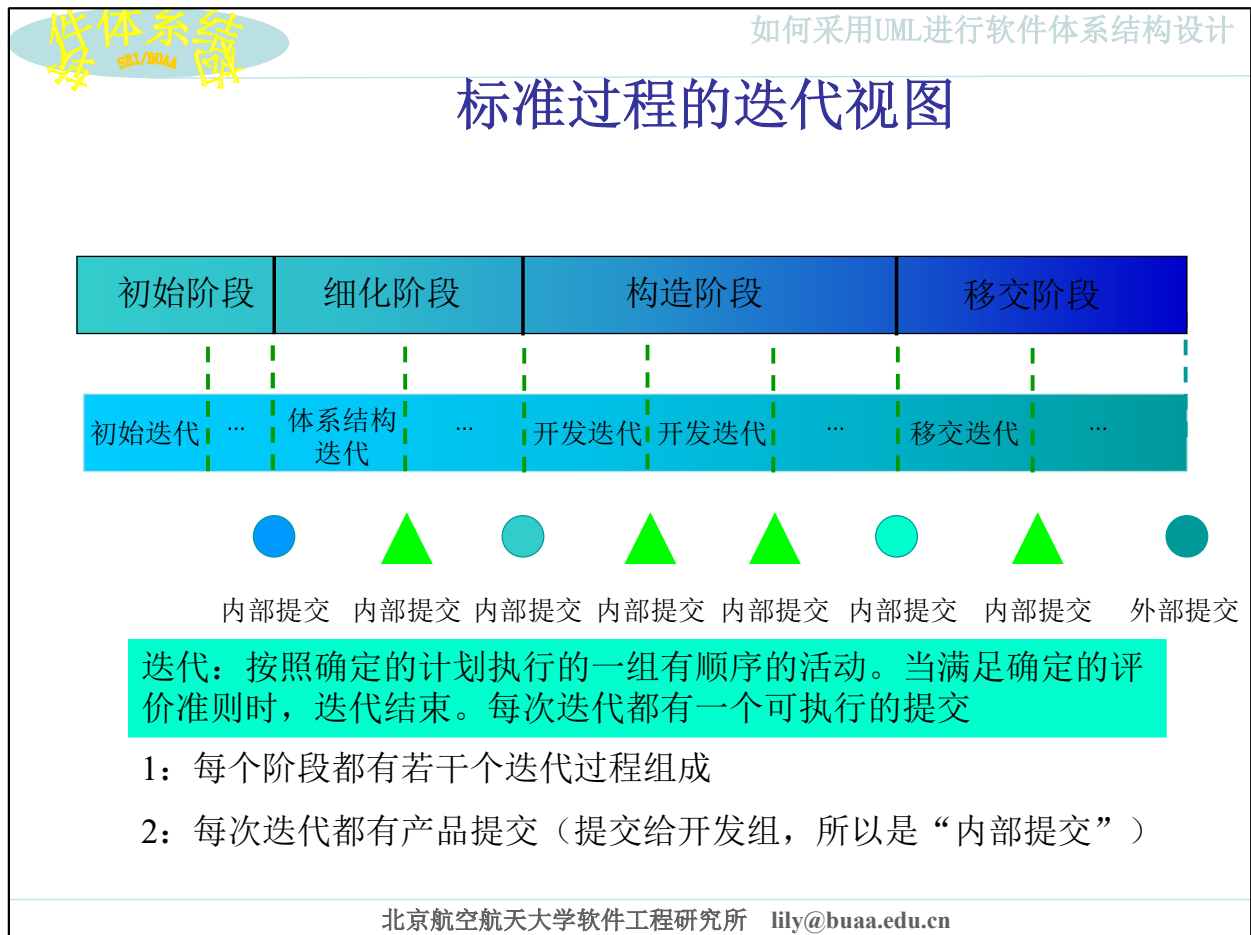


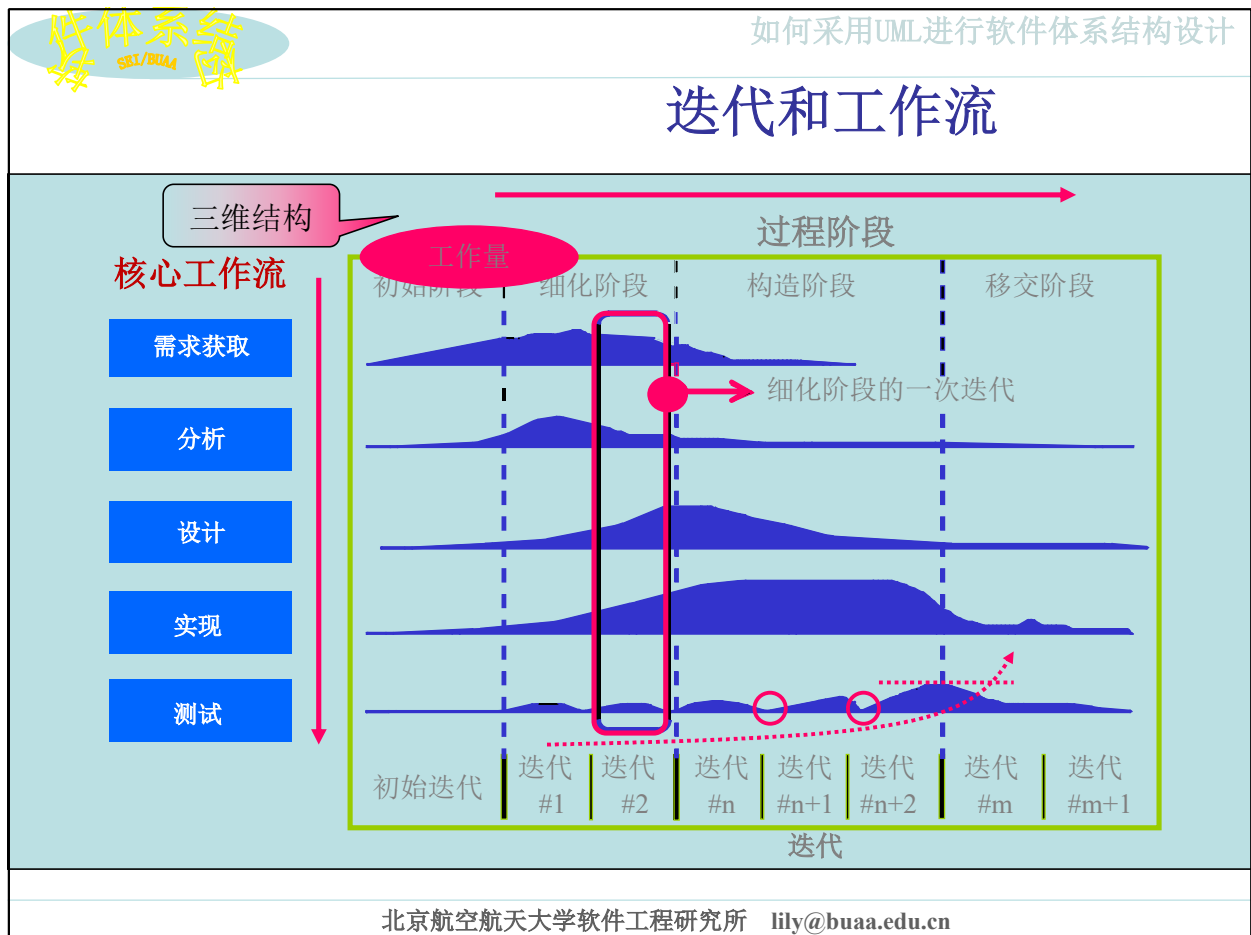
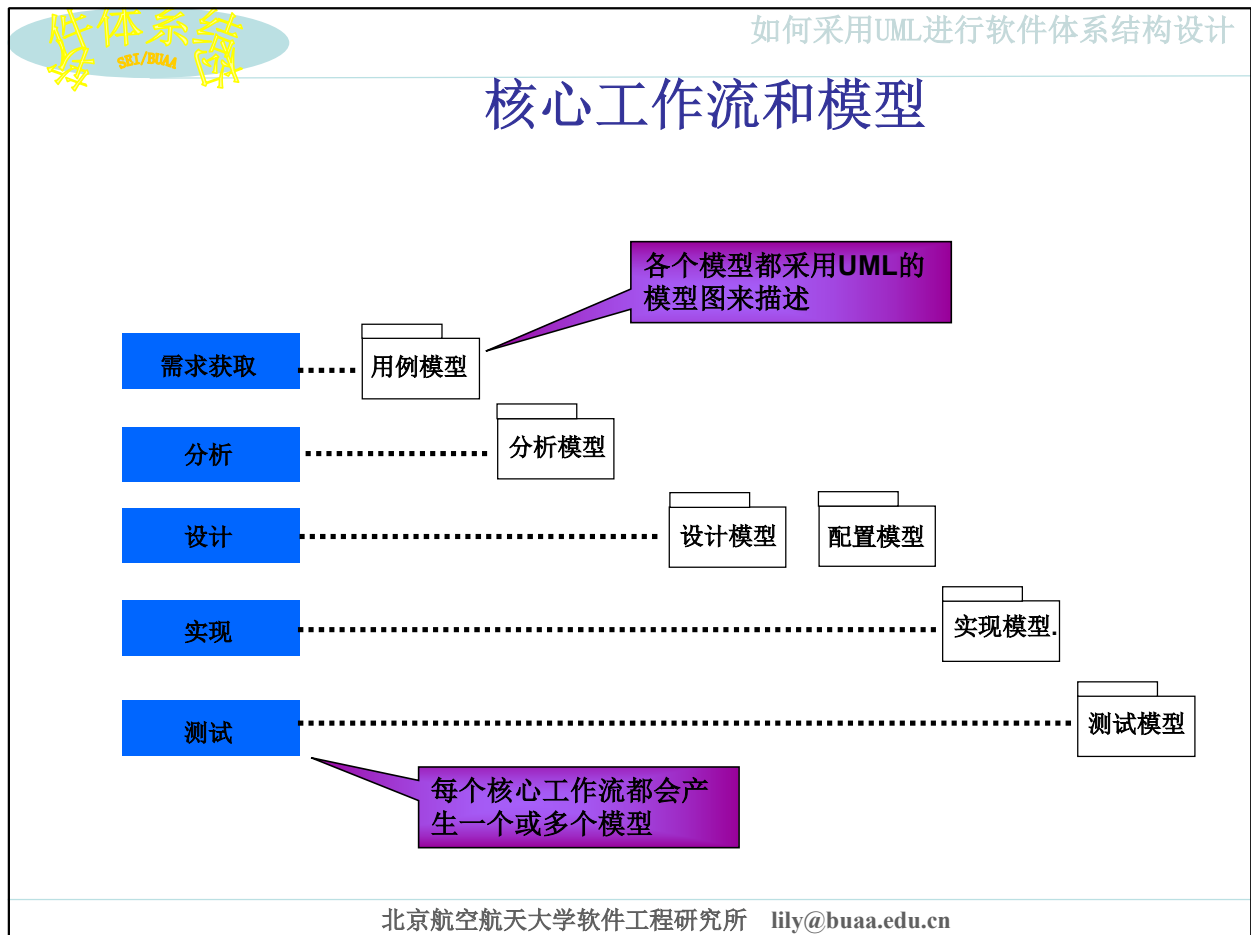
Initial
Capability

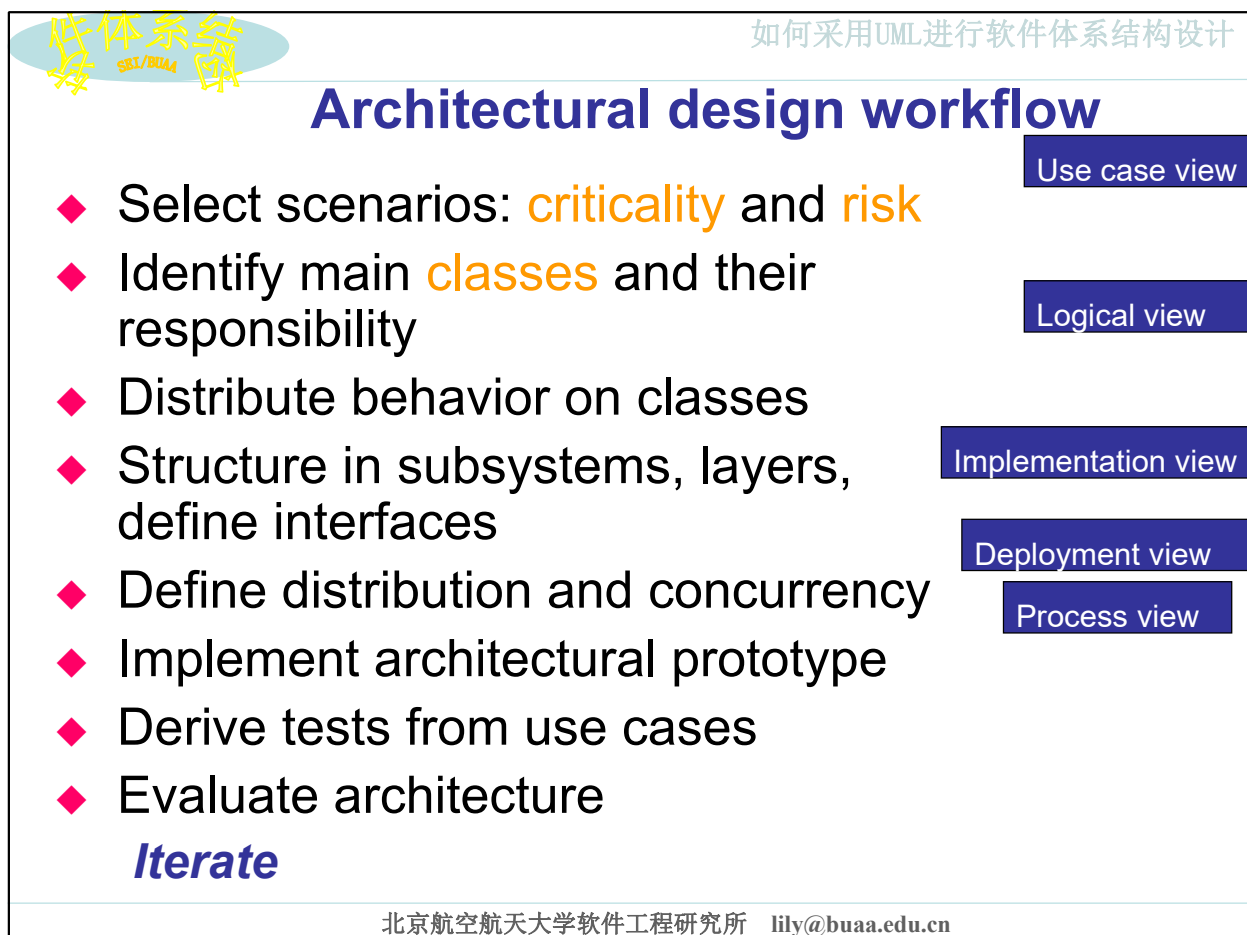
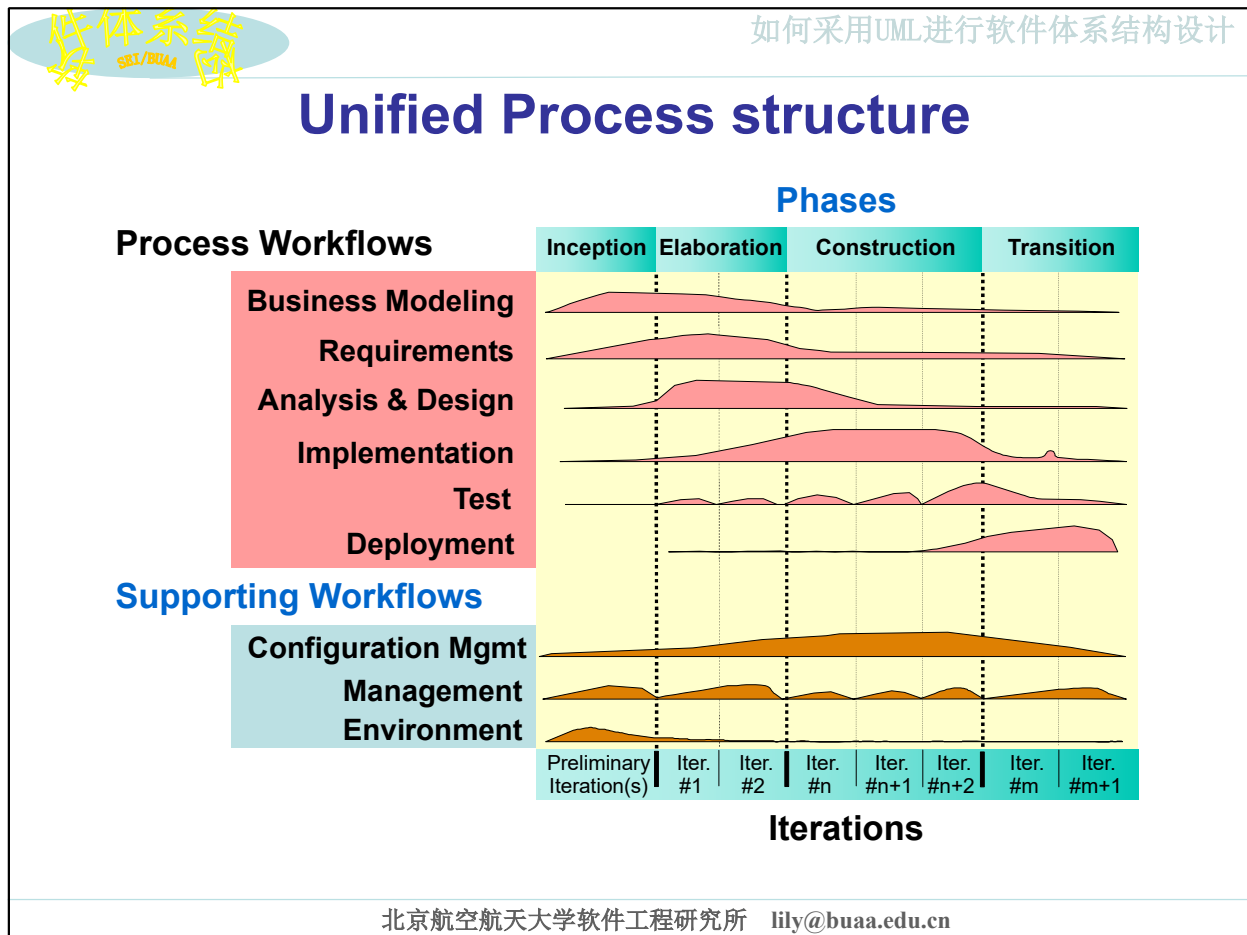


Product
Release

北京航空航天大学软件工程研究所 lily@buaa.edu.cn

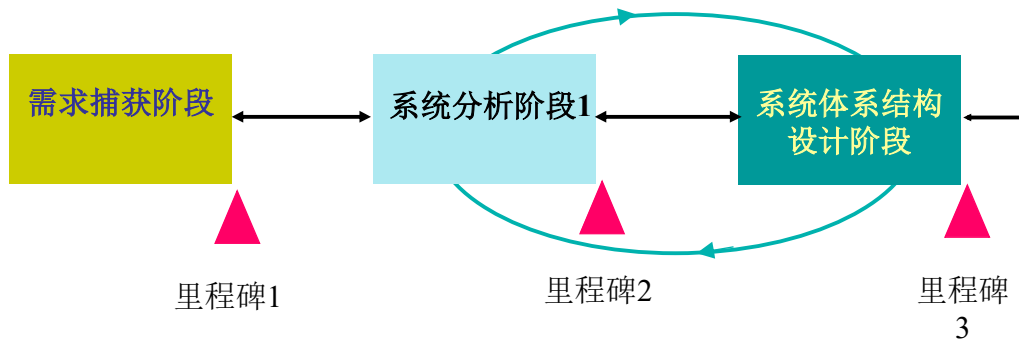






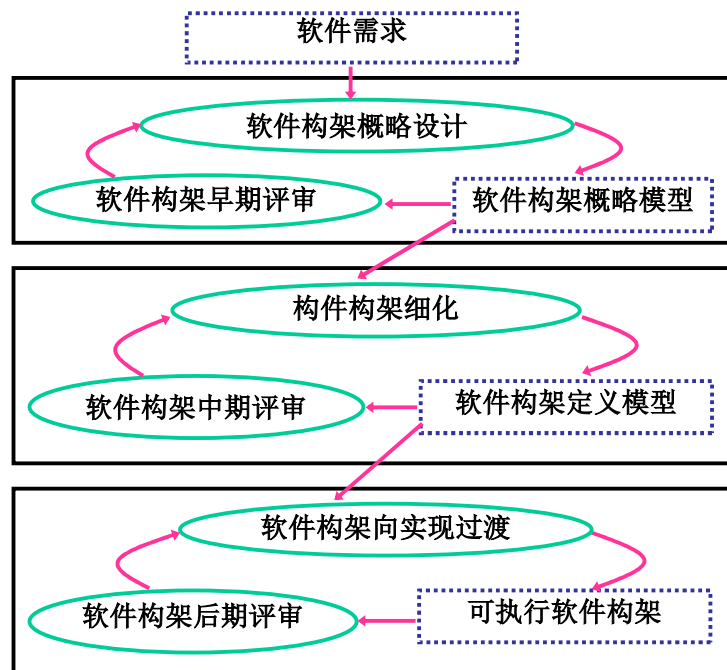
软件体系结构设计过程

- ◆ 从需求分析到概要设计：三个阶段、三个里程碑



北京航空航天大学软件工程研究所 lily@buaa.edu.cn

软件体系结构渐进设计过程



北京航空航天大学软件工程研究所 lily@buaa.edu.cn



软件体系结构渐进设计的基本思想

- ◆ 将整个软件构架设计过程划分为若干阶段
 - 保证软件构架设计可以由粗略到详细逐步推进
 - 减小设计的难度、复杂度以及风险
- ◆ 明确定义每一阶段的实施步骤
 - 整个设计过程更为透明化
 - 更容易被理解和实施
- ◆ 为每个阶段设置明确的结束准则
 - 组织对软件构架的阶段性的评审，以有利于问题的及早发现和解决
 - 而不仅仅是在整个设计完成后才一次性进行分析和验证
- ◆ 每一阶段对软件构架模型描述采用相似的核心概念
 - 减少各阶段设计结果之间的语义隔阂
 - 保证各阶段之间可以进行平滑的过渡

北京航空航天大学软件工程研究所 lily@buaa.edu.cn

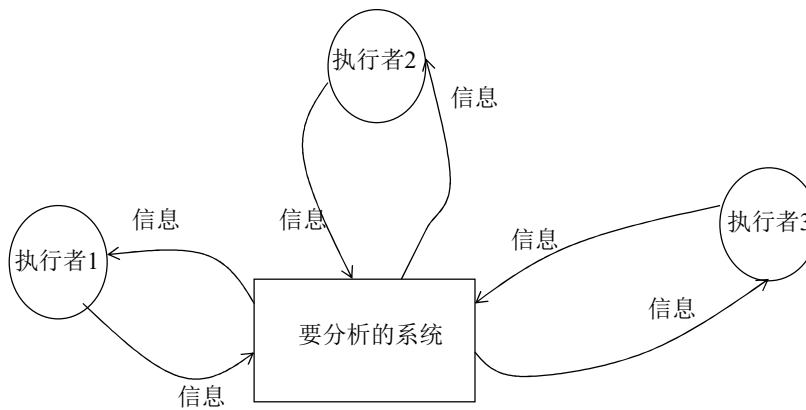


需求捕获阶段

- ◆ 主要任务：
 - 确定系统的边界、捕获系统的各种需求，包括功能性的和非功能性的需求。其中，功能性需求以用例的形式来描述，与特定用例相关的非功能性需求将作为用例描述内容中的一项进行说明；不与任何特定用例相关的非功能性需求需要单独进行描述。
- ◆ 需求捕获阶段主要由6个活动步骤组成：
 - 创建系统语境图
 - 创建用例图
 - 详细描述用例
 - 捕获其它需求
 - 编写需求规格说明
 - 阶段评审。

北京航空航天大学软件工程研究所 lily@buaa.edu.cn

1.1 创建系统语境图



系统语境图

北京航空航天大学软件工程研究所 lily@buaa.edu.cn

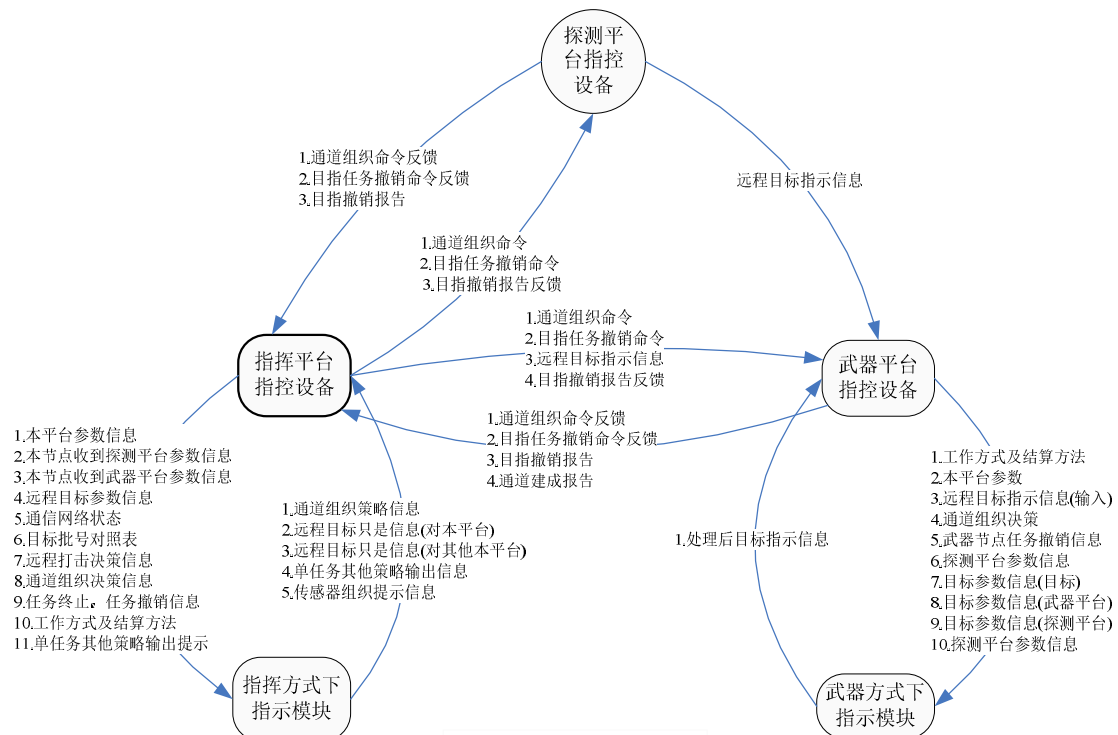


图1远程目标指示系统语境图

北京航空航天大学软件工程研究所 lily@buaa.edu.cn

- ◆ 如果进出系统的信息很多，或者信息的某些特性需要引起关注，如周期性、响应时间等，也可以做一个专门的信息/事件列表，如表1。

表1. 信息/事件列表

序号	信息/事件	系统动作	流向	到达模式	同步/异步	响应时间
1	加电自检指令	1.SDU Dsp 加电自检; 2.向PL/BC传送加电自检结果	In (来自 PL/BC)	随即性	同步	1s
2	发现爆发区	1. SDU 调整预处理时序 2.SDU通知MM清空已积累的数据	In (来自 HWT)	偶尔	异步	1s
3	申请观测条目		Out (去往 PL/BC)	周期性	异步	3minute

北京航空航天大学软件工程研究所 lily@buaa.edu.cn

- ◆ 在这一活动中，需求分析人员需要详细阅读可获得的各种文档资料，并与领域专家进行面谈，或通过专题讨论会等形式进行沟通，尽量找到系统的所有执行者。这里，系统的执行者指所有可能与系统发生交互的外部对象，可能是系统的用户，也可能是其它子系统或外部系统。识别潜在执行者的一些技巧：
 - 向系统发送信息的对象；
 - 系统控制的对象；
 - 对系统的执行结果感兴趣的对象。

北京航空航天大学软件工程研究所 lily@buaa.edu.cn

1.2 创建用例图

- ◆ 这一活动的主要目的是捕获系统的所有功能性需求，并以用例图的形式进行描述。
- ◆ 针对每一个执行者，确定系统中应为其提供哪些服务，从而识别出与其相关的潜在用例。识别潜在用例的一些技巧：
 - 针对流入系统的每一条信息，判断该信息是否会引起系统的一系列动作，这一系列动作是否可构成一个完整的功能，如果是，考虑将这一系列动作定义为一个用例；
 - 针对流出系统的每一条信息，判断该信息是如何产生的？
 - 系统最明显、最主要的功能是什么？

北京航空航天大学软件工程研究所 lily@buaa.edu.cn

- ◆ 一开始可能很难详细列出系统中的所有用例，这时可以从已发现的用户入手，详细描述这些用例的执行流程。在描述过程中，可能会发现某些步骤的实现需要添加新的用例。如此迭代，直到不再有新用例添加进来。这一步骤与第2.3步是迭代进行的。
- ◆ 如果用例数量较多，要考虑对用例进行分包处理。对用例进行分包，可以参照这样一些原则（但不是必须要遵守的）：
 - 支持同一个执行者的用例划分到同一个包中；
 - 具有使用或扩展关系的用例划分到同一个包中；
 - 支持一个具体业务过程的用户划分到同一个包中；

北京航空航天大学软件工程研究所 lily@buaa.edu.cn

1.3 详细描述用例

- ◆ 详细描述已有用例的执行流程。这一描述可采用“白盒”形式，即描述过程中可以包括用例内部对外部执行者来说不可见的处理过程，这一处理过程可以具体说明是由系统中哪一个组成部分完成的。但必须满足的条件是，这个组成部分必须包含在该系统的硬件或软件配置项列表中。
- ◆ 在对用例进行描述时最好采用文本和图形两种方式，文本方式的优点是可以描述得很详细，图形方式的优点是严格、直观。在用例描述初期要求必须提供文本形式的描述（用例描述模板参见《用例描述规范》）。图形方式的用户描述可以采用活动图、顺序图等手段。
- ◆ 使用活动图和顺序图的参考原则：
 - 侧重于理解用例的执行流程，而不关心其间具体的通信机制时用活动图；
 - 希望在理解用例执行过程的同时了解系统内各组成部分的相应职责以及各部分间的通讯关系和同步情况时用交互图（顺序图和合作图）。

北京航空航天大学软件工程研究所 lily@buaa.edu.cn

1.4 捕获非功能性需求

- ◆ 主要是列举与系统质量属性相关的需求，如性能要求（实时性、响应时间）、吞吐量、可靠性、安全性、鲁棒性、可扩展性等，以及各种技术约束条件（如特定技术要求 and 软硬件配置等）。

北京航空航天大学软件工程研究所 lily@buaa.edu.cn

1.5 编写《软件需求规格说明》文档

- ◆ 需求规格说明的编写并不是到这一步骤才开始，而是在前几个活动步骤实施过程中就已经开始了。这一步骤主要是对文档进行全面的整理，形成完整、规范的需求文档。

1.6 评审

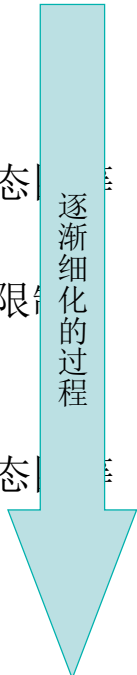
- ◆ 这一步骤主要是按照里程碑中确定的目标对需求规格说明进行评审。这一阶段里程碑的内容如下。
- ◆ 里程碑
 - 提交《软件需求规格说明》文档；
 - 各方对系统的范围和功能达成一致；
 - 用例描述中的活动步骤顺序正确、完整、可以理解；
 - 文档的完整性：文档是否包含了所有必要的内容；
 - 文档的一致性：文档描述中不存在前后不一致的信息，具体指
 - 在文档中没有相互矛盾的两条语句；
 - 在文档中一个给定术语、简称或缩略语始终指相同的事情；
 - 文档中对一个给定项目或概念始终采用相同的名字或叙述

2、系统分析和体系结构设计阶段

- ◆ 目标：对组成系统的各个软件模块及其接口关系做出明确的定义，作为任务划分的依据。
- ◆ 渐进式的迭代过程
- ◆ 4个视图
 - 逻辑视图
 - 并发进程
 - 开发视图
 - 物理视图（构件视图、部署视图）

渐进式的迭代过程

- ◆ 4个视图
 - 逻辑视图
 - 逻辑结构：包图、类图
 - 行为描述：交互图（顺序图和协作图）、状态图
 - 并发视图：
 - 活动图、交互图、状态图（、时序图—时间限制）
 - 开发视图
 - 开发结构：包图、类图
 - 行为描述：交互图（顺序图和协作图）、状态图
 - 内部接口关系
 - 构件视图：构件图
 - 部署视图：部署图



逐渐细化的过程

逻辑视图

- ◆ 主要是从满足功能需求的角度出发对系统进行分解，目的是向用户表明系统如何在功能上满足其最终的需求。
- ◆ 逻辑视图
 - 逻辑结构：包图和类图
 - 行为描述：交互图（顺序图和协作图）、状态图等

逻辑结构

- ◆ 从系统用户角度看到的系统结构。
- ◆ 逻辑结构的创建很大程度上依赖于设计师对领域知识的熟知程度，应建立在深入的领域分析基础之上。
- ◆ 逻辑结构可通过UML中包图和类图来描述。包表示可被进一步分解的元素，其内部可由其它包或者类组成，类表示在体系结构设计阶段不需要再对其进行分解的元素。



行为描述

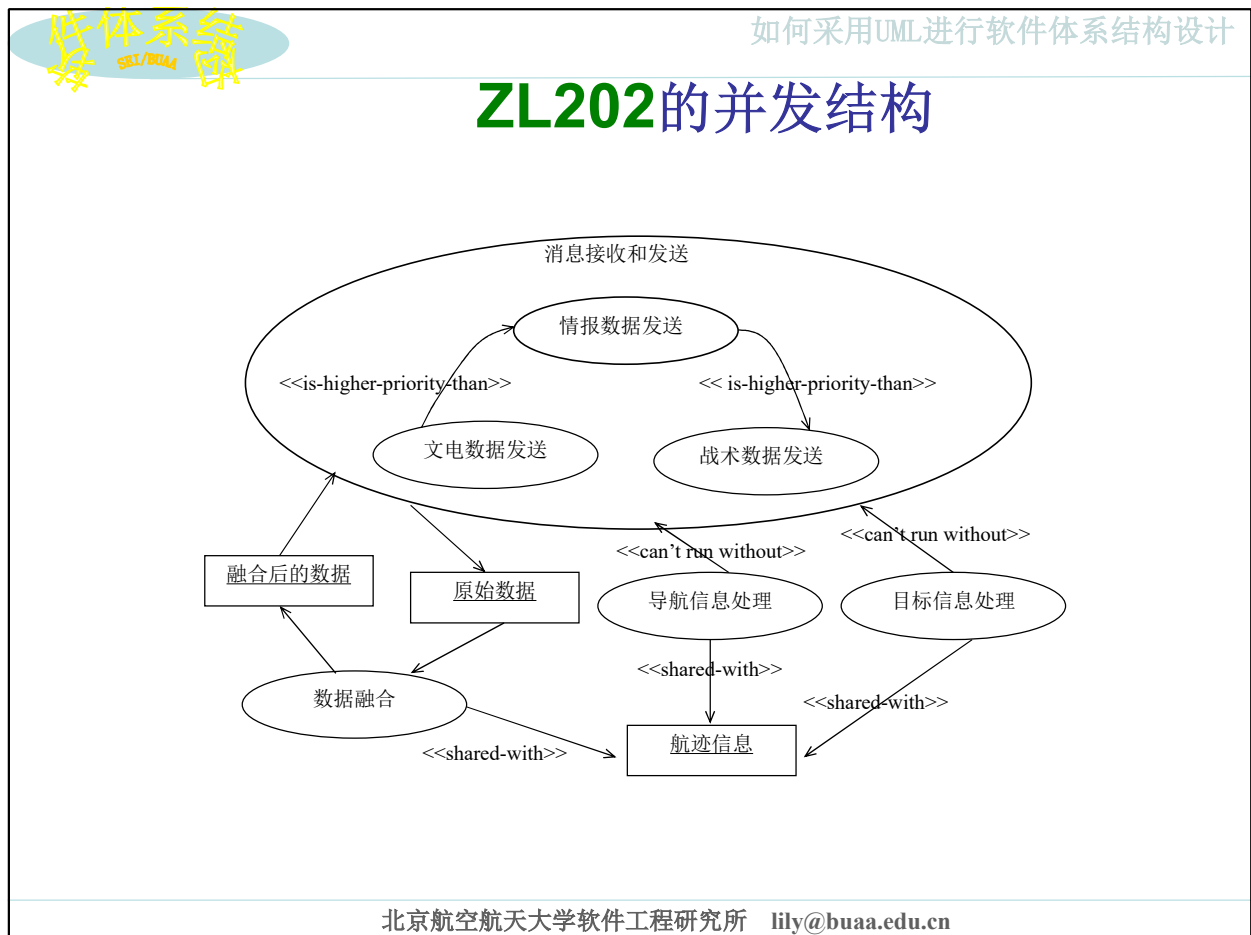
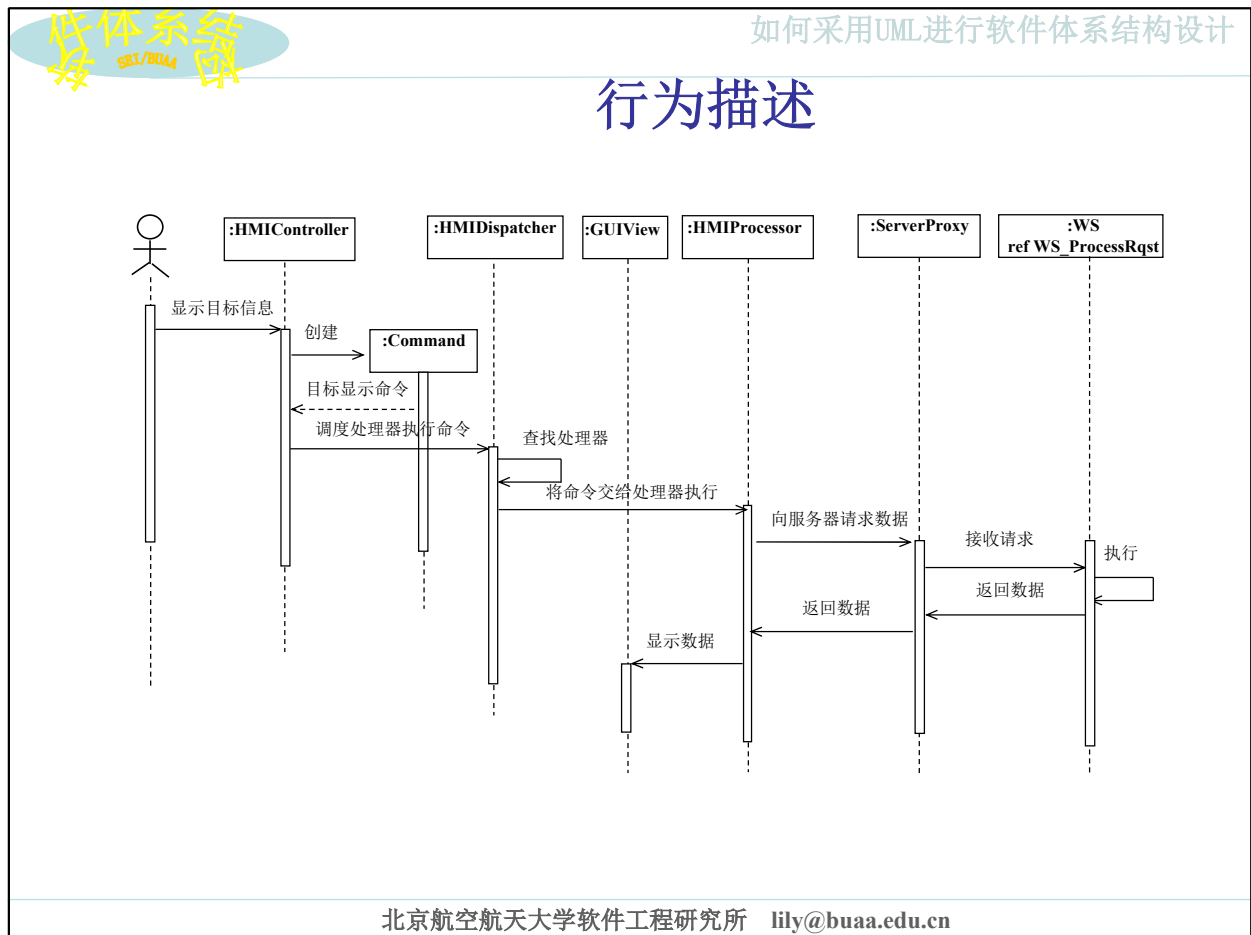
- ◆ 对系统逻辑行为进行描述的目的，一方面是为了验证这种结构设计是否能够真正满足系统的功能需求，另一方面是为了将用例所代表的功能分解成粒度更小、更易于理解的形式，并分别分配到各个逻辑组成元素上。
- ◆ 逻辑行为可以采用交互图（顺序图和协作图）、状态图等形式来描述。

北京航空航天大学软件工程研究所 lily@buaa.edu.cn



- ◆ 具体做法是：
- ◆ 1) 首先，“XX子系统逻辑视图”包中创建一子包，命名为“用例实现”；
- ◆ 2) 然后，在这个包中，创建一个用例图，针对每一个用例创建一个“用例实现（**Usecase Realization**）”，并建立该用例实现到其对应用例的追踪关系。
- ◆ 3) 以逻辑视图中划分的逻辑模块做为参与对象，通过交互图来描述每个用例的实现。这样就可以将用例的功能分解为这些逻辑模块的一个个职责。
- ◆ 4) 如果某个元素的行为比较复杂，在整个生命周期中会存在多种不同的状态，也可以用状态图对其行为进行细化。
- ◆ 通过创建逻辑行为模型，可以更深入的理解和交流需求，并且也为进一步的设计提供了基础。
- ◆ 逻辑视图仍然是在概念层次对系统进行的分析，不必过多的考虑系统实现方面的问题。
- ◆ 建议用协作图，因为。这时候不关心严格的时序关系，而更关心对象之间彼此的关系。

北京航空航天大学软件工程研究所 lily@buaa.edu.cn





渐进式的迭代过程

◆ 4个视图

➤ 逻辑视图

- 逻辑结构：包图、类图
- 行为描述：交互图（顺序图和协作图）、状态图

➤ 并发视图：

- 活动图、交互图、状态图（、时序图—时间限制

➤ 开发视图

- 开发结构：包图、类图
- 行为描述：交互图（顺序图和协作图）、状态图
- 内部接口关系

➤ 构件视图：构件图

➤ 部署视图：部署图

逐渐细化的过程

北京航空航天大学软件工程研究所 lily@buaa.edu.cn



开发视图

- ◆ 逻辑视图仍然是面向系统用户的，可以认为是系统设计人员在向用户解释系统是如何满足其功能需求的，所采用的表达方式仍然是用户易于理解的方式。
- ◆ 开发视图是从程序开发的角度描述系统中到底应该开发哪些模块，并对每个模块的规格说明做出详细的描述。
- ◆ 逻辑视图和开发视图的区别在于，逻辑视图是在用户可见的层次完成了对系统的横向分解，而开发视图则从信息隐藏、重用、易开发、易管理等开发方面的内部需求考虑了对系统的纵向分解。开发视图中的每个模块可以独立地被分配给一个或几个开发人员去实现，开发视图与开发组织的任务分解结构常常是相吻合的。
- ◆ 开发视图可以看作逻辑视图的物理实现。在开发视图中应给出对系统详细设计和实现形成约束的所有信息。

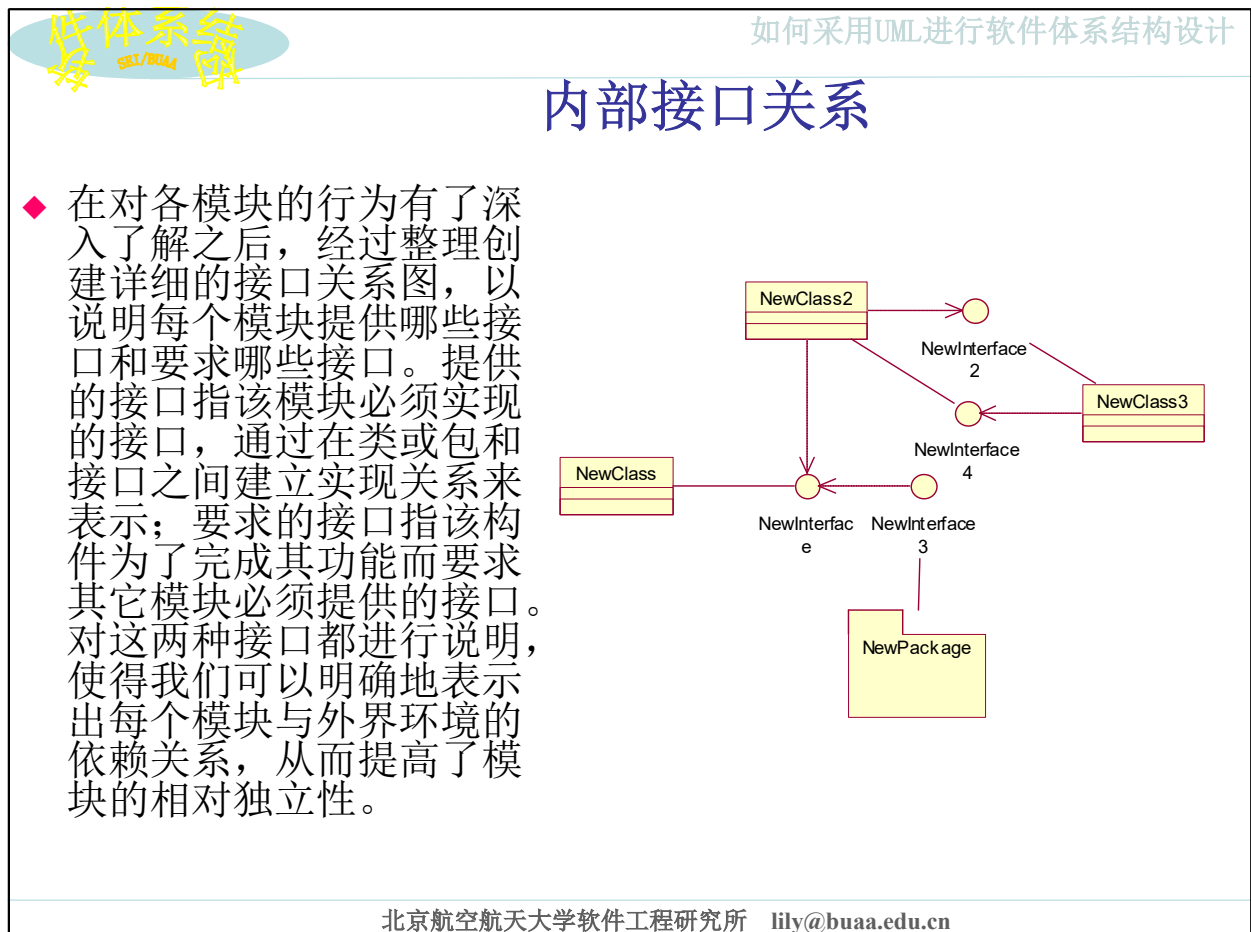
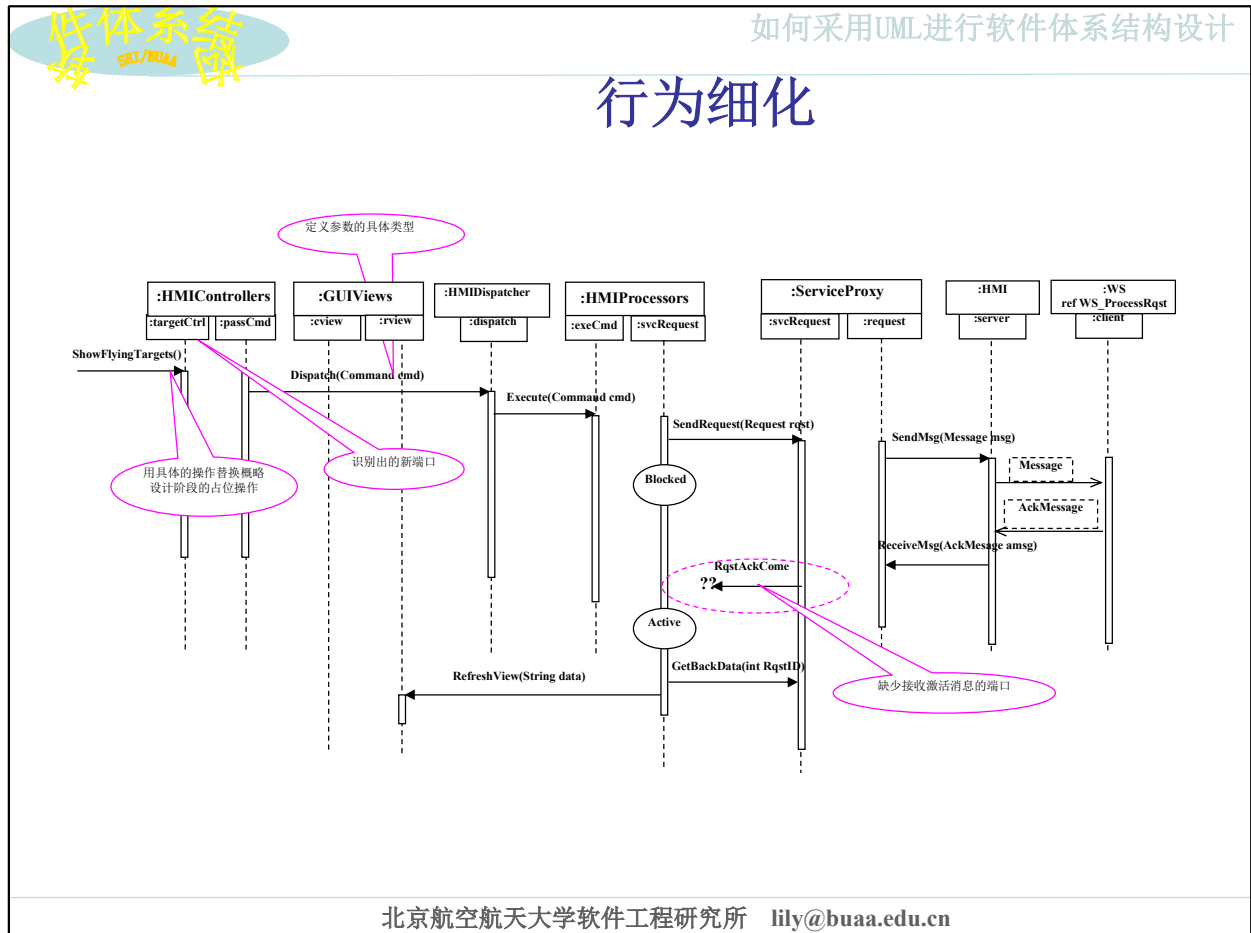
北京航空航天大学软件工程研究所 lily@buaa.edu.cn

开发结构

- ◆ 开发结构是建立在经验、对相关应用领域及软硬件技术的深入了解基础之上的。在创建开发结构的过程中，各子系统的设计人员应充分交流，以便彼此互相借鉴。
- ◆ 当不需要考虑重用、信息隐藏、开发性等内部需求时，开发结构与逻辑结构可以存在一一映射的关系。尽管如此，还是建议单独建立一个开发视图，以便于保留逻辑层次的设计结果。

行为描述

- ◆ 采用与逻辑行为相似的方法，以开发中定义的各个元素为参与对象，描述它们如何通过相互协作，完成用例中的功能。在开发视图中描述用例的实现时，建议采用顺序图，因为这时我们更关心消息的先后顺序。
- ◆ 但是，开发视图中的行为描述可以不仅仅局限于用例实现，当某个模块的行为比较复杂时，也可以用状态图来详细对其行为进行刻画。



渐进式的迭代过程

◆ 4个视图

➤ 逻辑视图

- 逻辑结构：包图、类图
- 行为描述：交互图（顺序图和协作图）、状态图

➤ 并发视图：

- 活动图、交互图、状态图（、时序图—时间限制

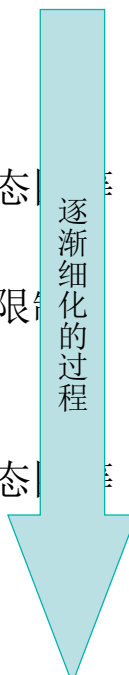
➤ 开发视图

- 开发结构：包图、类图
- 行为描述：交互图（顺序图和协作图）、状态图
- 内部接口关系

➤ 构件视图：构件图

➤ 部署视图：部署图

逐渐细化的过程

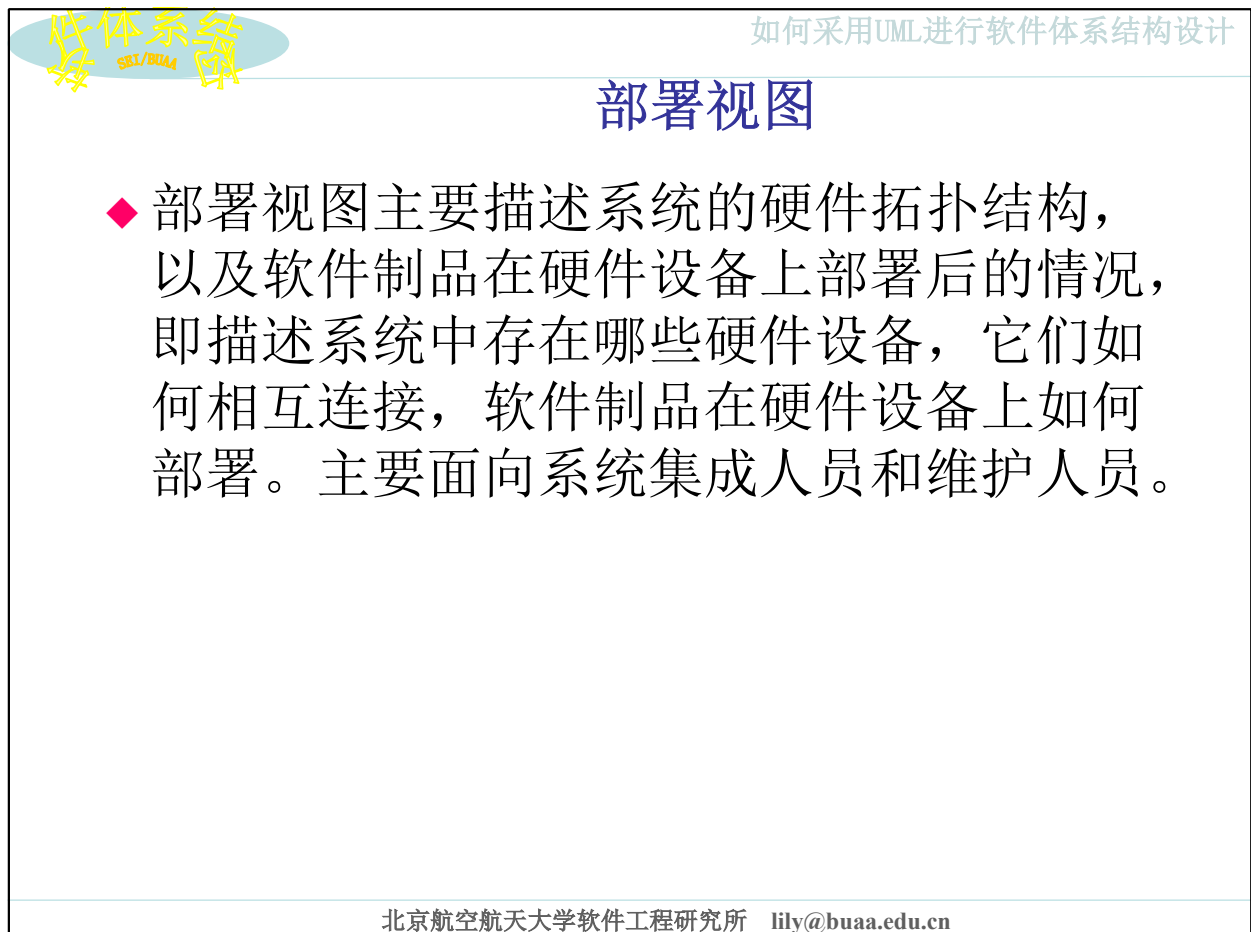
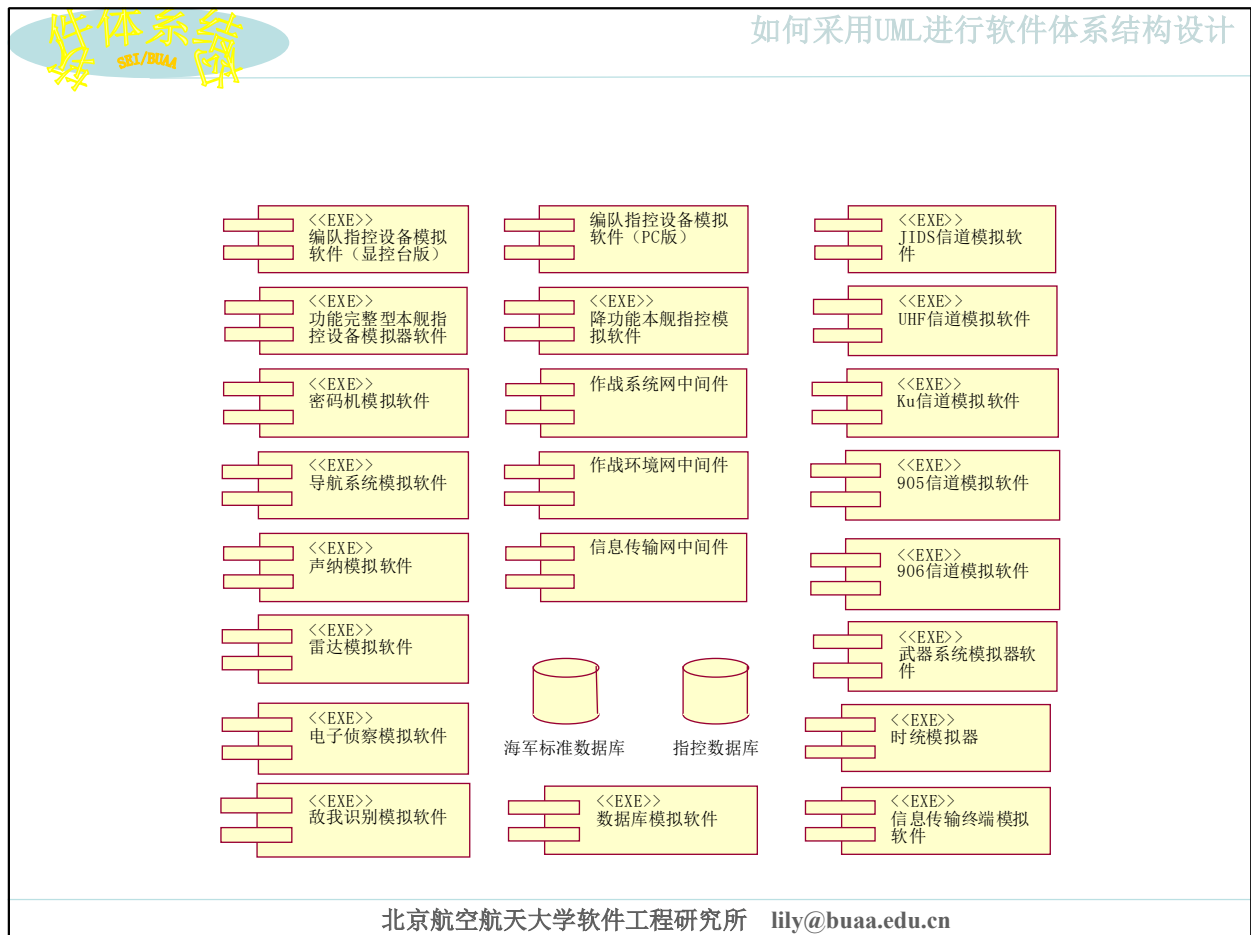


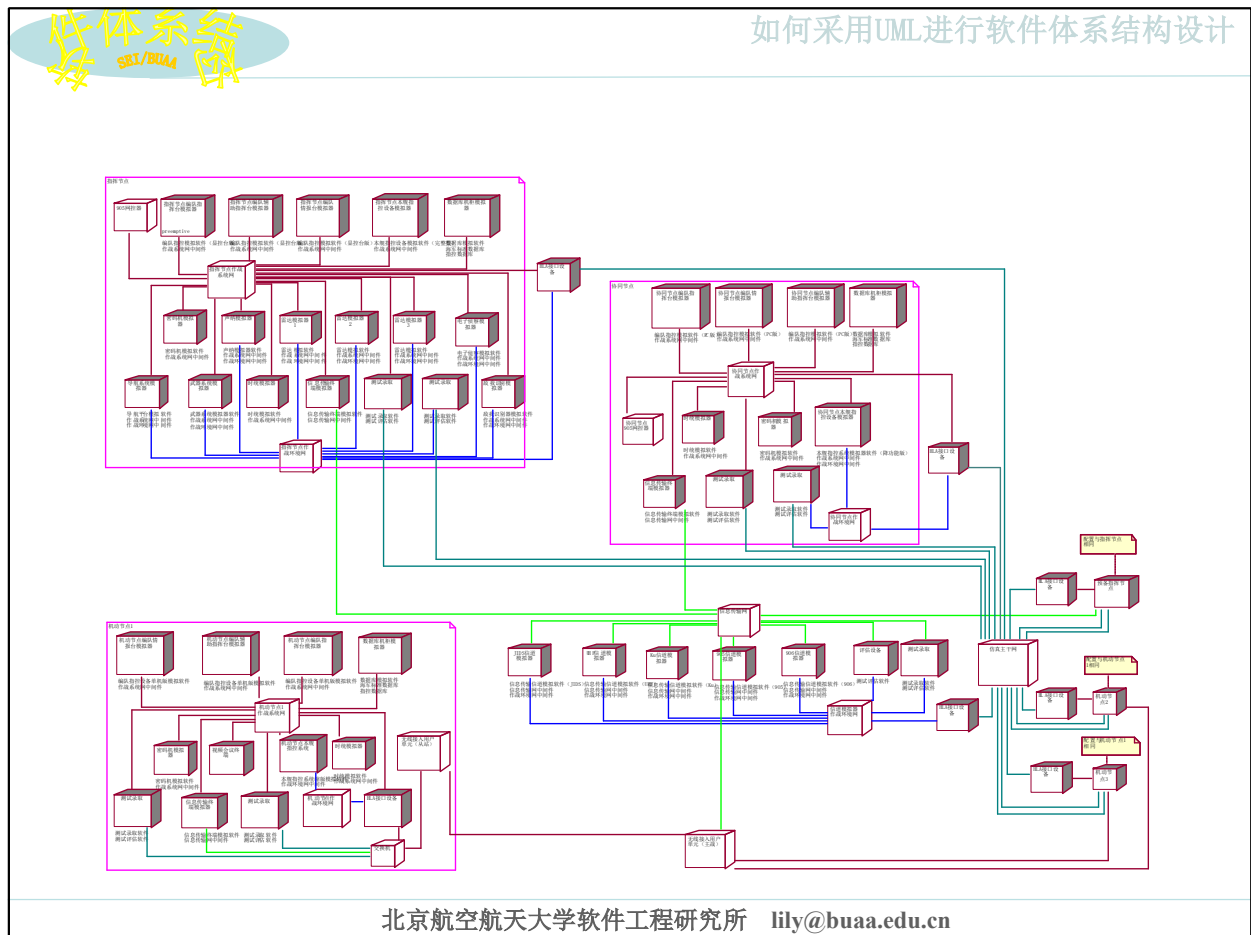
北京航空航天大学软件工程研究所 lily@buaa.edu.cn

构件视图

- ◆ 构件视图主要说明系统开发完成后将得到哪些软件制品，即最终系统将生成哪些 **.Exe**，**.dll** 或其它，并在必要时表明其间的依赖关系。在系统开发之前构件视图的主要作用是对系统最终将开发的软件制品进行的规划，作为软件打包的指导策略。在系统开发完成后，应根据实际情况进行修改，以反映实际的打包结果，为系统产品的配置管理提供依据。构件视图通过 **UML** 的构件图来描述，面向系统配置管理人员

北京航空航天大学软件工程研究所 lily@buaa.edu.cn





软件体系结构 SRI/BUAA

如何采用UML进行软件体系结构设计

系统硬件及软件配置列表

系统名称	系统标识	硬件名称	硬件标识	类型	数量	连接至	软件配置项	软件配置项标识	功能	备注
虚拟指挥模拟系统	ZL3	指挥台模拟器	H2. 06. 1	多功能显控台	6台	作战系统网	指控设备显控台版模拟软件	S2. 06	模拟指控设备的功能，实现信息共享和其它作战指挥辅助手段。	配置固定型节点2~4、机动型节点1~3
			H2. 18. 1	PC	6台	作战系统网	指控设备单机版模拟软件	S2. 18	模拟指控设备的功能。	
		情报台模拟器	H2. 06. 3	多功能显控台	5台	作战系统网	指控设备显控台版模拟软件	S2. 06	模拟情报台的功能。	
			H2. 18. 3	PC	5台	作战系统网	指控设备单机版模拟软件	S2. 18	模拟情报台的功能。	
		数据库机柜模拟器	H2. 12	PC	4台	作战系统网	数据库模拟软件	S2. 12	存储指控系统内部各类管理信息及相关标准数据库的模拟内容。	
		时统设备模拟器	H2. 21	PC	4台	作战系统网作战环境网	时统模拟软件	S2. 21	模拟时统信息发送。	固定节点2~4
		信传终端模拟器	H2. 20	PC	6台	信息传输网作战系统网	信传终端软件	S2. 20	实现编队固定节点向信息传输网的信息转发。	机动型节点1~3

北京航空航天大学软件工程研究所 lily@buaa.edu.cn



软件体系结构渐进设计过程

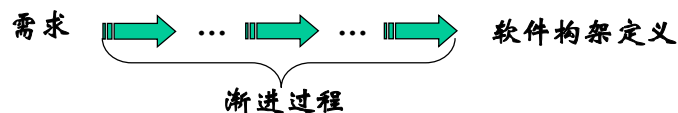
我们的一项研究
(2005年的一篇博士学位论文)



如何采用UML进行软件体系结构设计

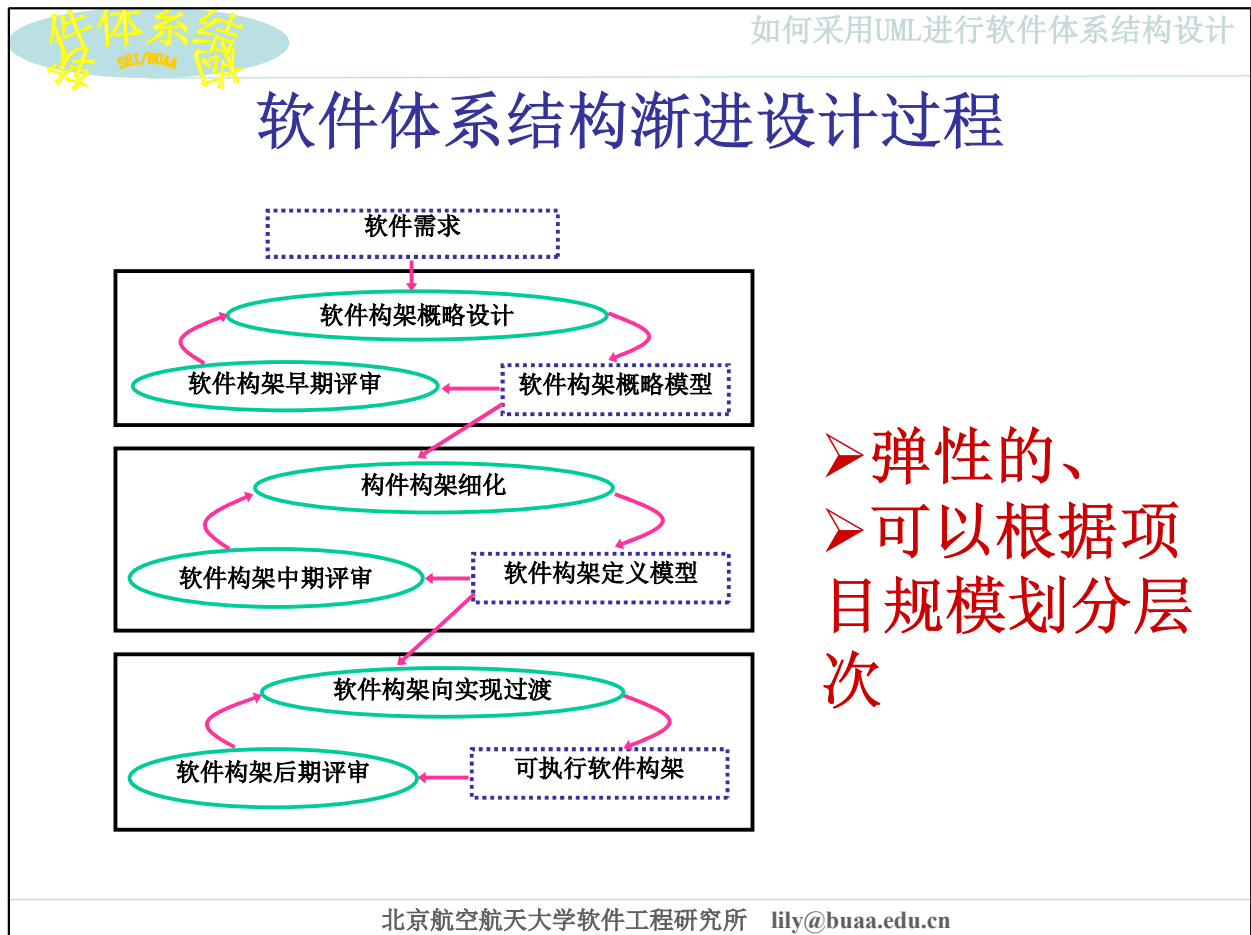
问题的提出——软件构架设计过程

- ◆ 软件构架设计是复杂的，不能一蹴而就地完成

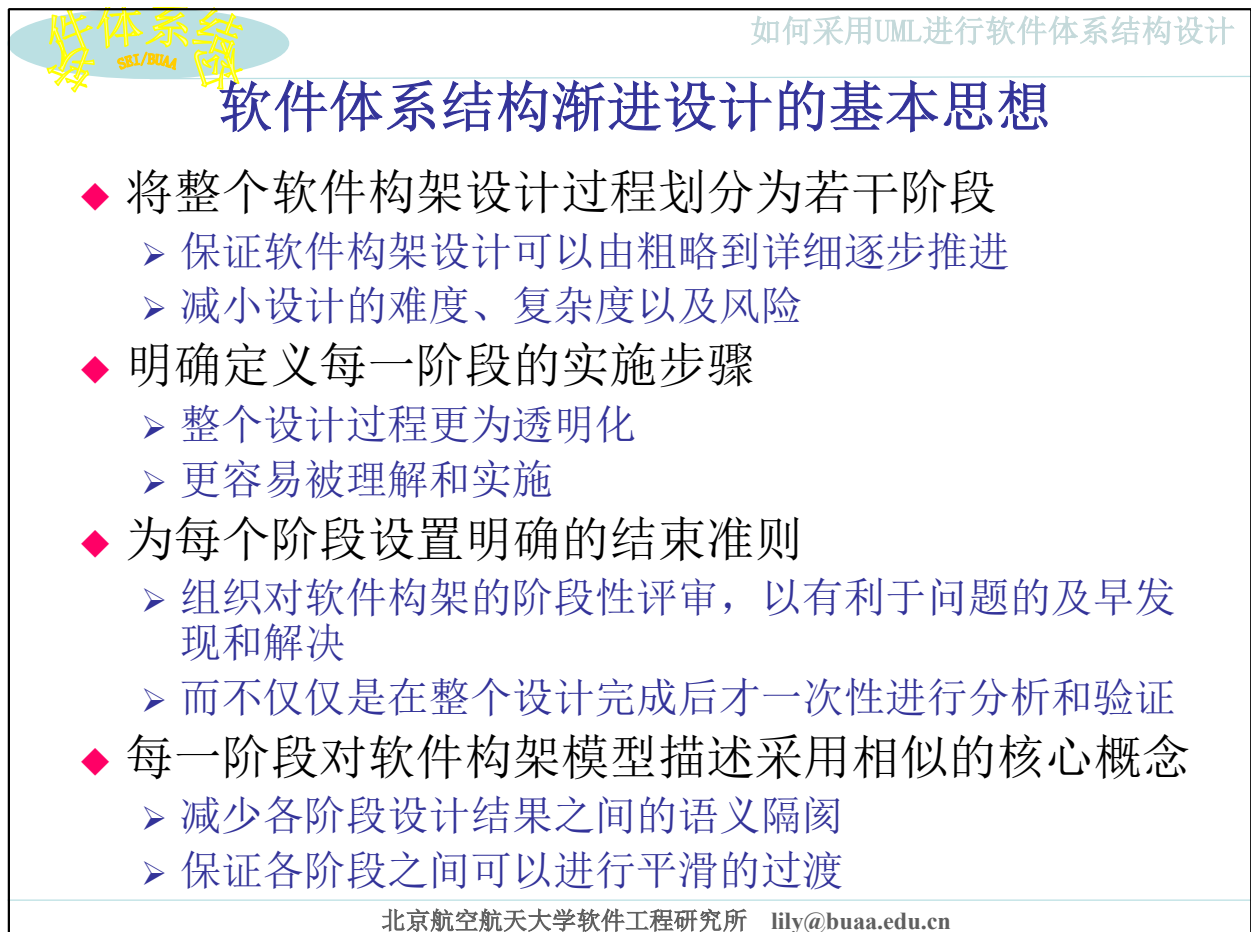


◆ 问题1

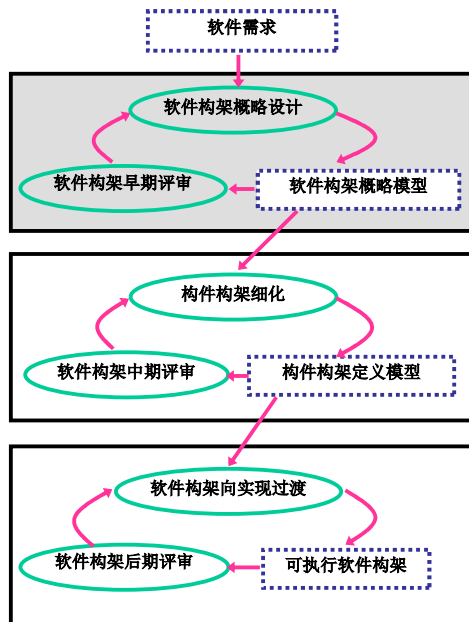
- 至今关于软件构架渐进设计过程具体实施步骤的定义仍然比较模糊
- 软件构架设计主要由设计师依靠自身的经验和直觉进行
- 软件构架设计带有很大的随机性
- 软件构架设计过程难以被重用被获得持续不断的改进



➤弹性的、
➤可以根据项目规模划分层次



软件构架概略设计



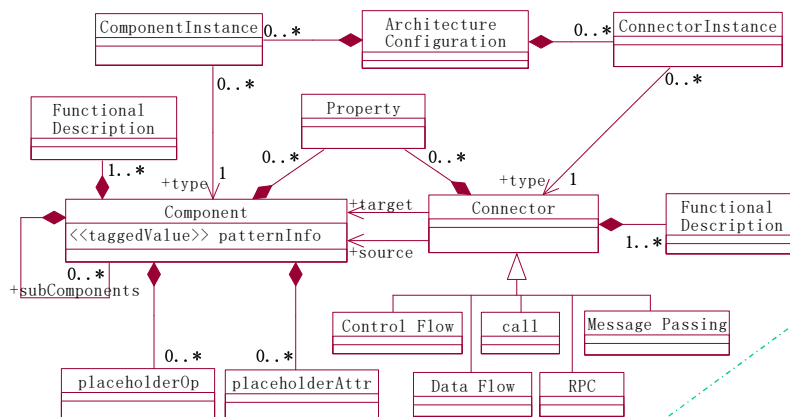
◆ 主要目的

- 构建一个包含较少细节的初始软件构架模型
- 在深入软件构架设计工作之前，使各相关人员可对系统的高层结构产生一个大致地了解
- 以便对系统能够满足各种软件构架需求的潜在可能性做出初步评价

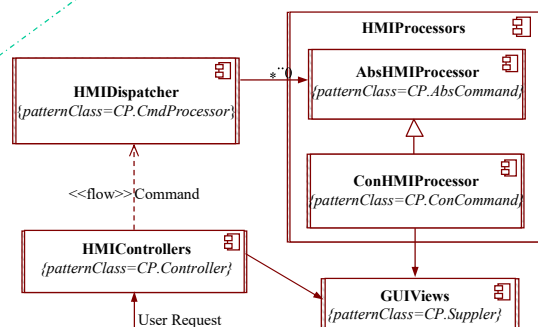
◆ 设计重点

- 识别软件构架设计策略
- 对系统的结构进行分解，功能进行分配
- 说明控制和数据在构件间传递

北京航空航天大学软件工程研究所 lily@buaa.edu.cn



SAS建模语言元模型示例



SAS模型示例

北京航空航天大学软件工程研究所 lily@buaa.edu.cn

SAS设计的具体实施步骤

步骤1: 识别软件构架需求

构架级重要用例
构架级重要质量属性



表 3.1 软件构架需求与候选策略对照表

步骤2: 识别候选软件构架设计策略

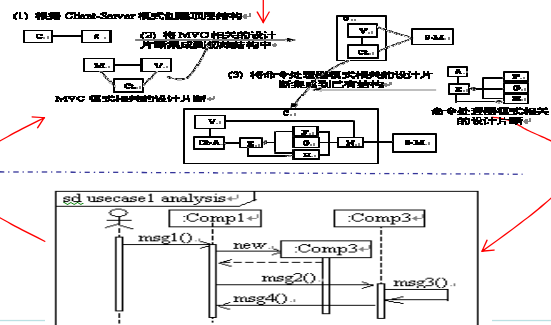
需求	需求标识号	策略标识号	策略名称	策略描述	策略状态	加强属性
性能	R_001	R_001_D1	PUSA.MVC+ PCSA.命令处理	能使用户界面和具体 的业务处理逻辑	建议	P_002 (0.8)

步骤3: 软件构架设计策略的权衡和取舍

步骤4: 创建软件构架概略模型

步骤4.1: 结构设计

步骤4.2: 行为描述



北京航空航天大学软件工程研究所 lily@buaa.edu.cn

软件构架早期评审

◆ 评审时机

- 获得软件构架概略模型之后

◆ 评审目的

- 对系统满足功能和质量需求的潜在可能性做出评价
- 不是精确预测系统的行为和质量指标
- 识别软件构架的设计风险
 - ☞ 对某个重要问题还没有做出决策
 - ☞ 虽然对问题做出了决策但对决策的后果还不清晰

◆ 评审手段

- 基于提问的方法（Checklist、场景）
- 功能需求的满足可通过用例分析来实现
- 质量需求的满足可采用SAAM、ATAM等方法通过质量场景来验证

北京航空航天大学软件工程研究所 lily@buaa.edu.cn

软件构架早期评审Checklist

- ◆ 所有软件构架需求都得到识别；
- ◆ 不存在代价太大而难于实现的需求；
- ◆ 软件构架覆盖了所有软件构架需求，即设计中不存在对重要设计问题的遗漏；
- ◆ 系统中所有关键构件及其间主要的连接关系都已被识别出来；
- ◆ 模型中每个构件都有一个名称和一个反映其在系统中作用的功能描述；
- ◆ 每个构架级重要的用例至少有一个交互图来反映该用例代表的功能如何得到满足；
- ◆ 针对每一个构架级重要的质量属性，能说明构架设计中为满足这一质量属性所采取的机制，包括相关的构件、连接件或约束；
- ◆ 用于实现改构架的所有技术都是可行的，即该构架在实现上步存在无法克服的技术风险。

北京航空航天大学软件工程研究所 lily@buaa.edu.cn

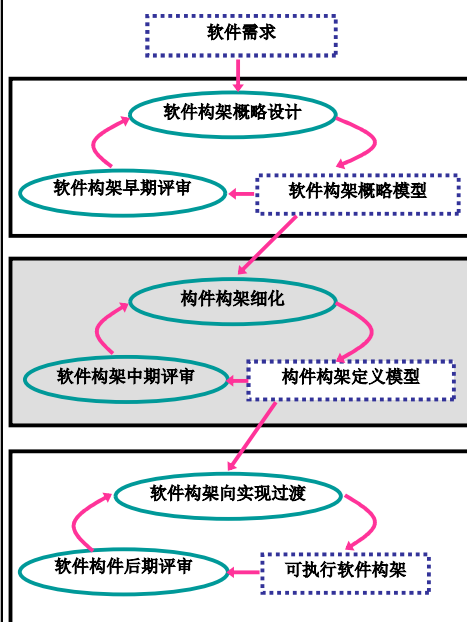
软件构架细化

◆ 主要目的

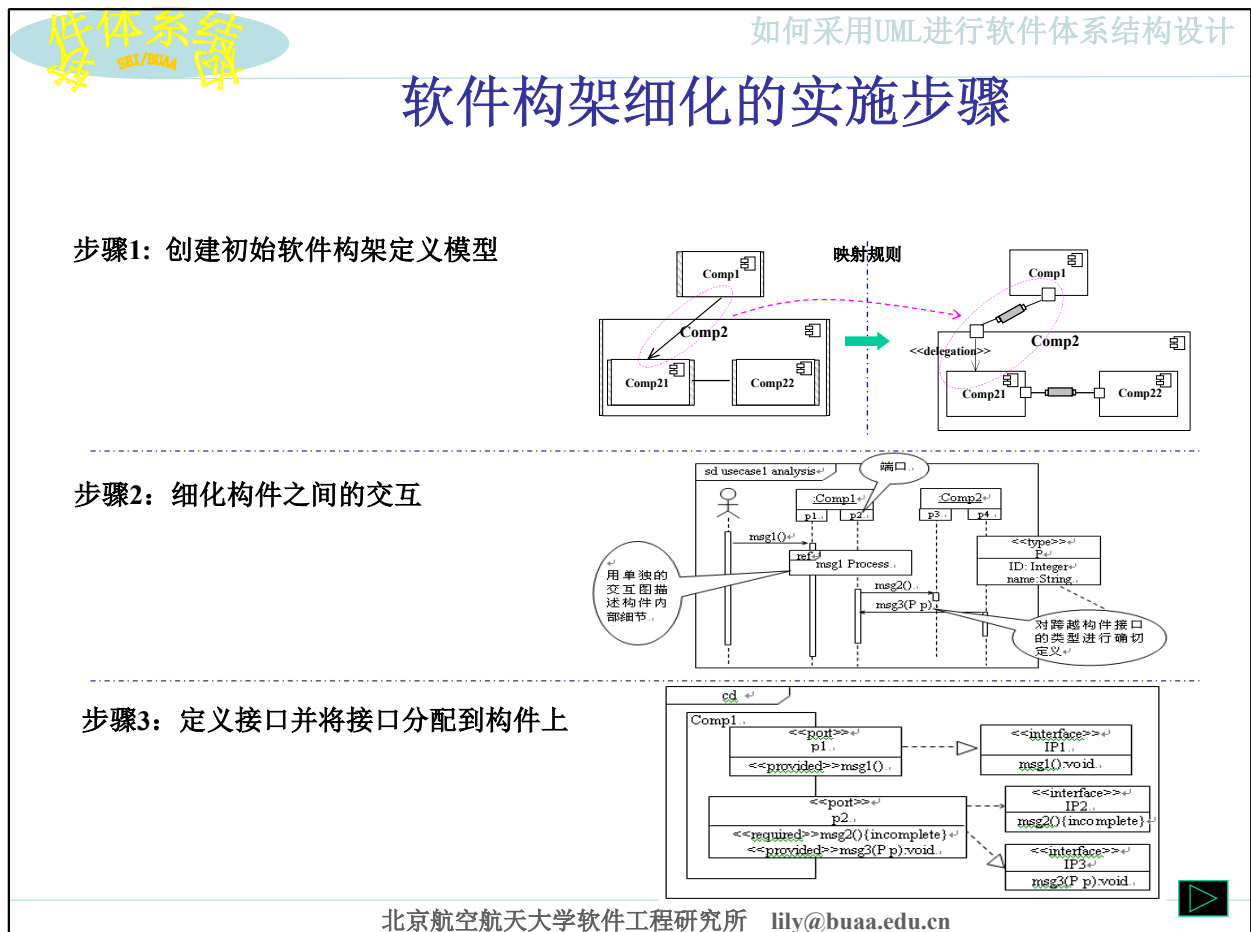
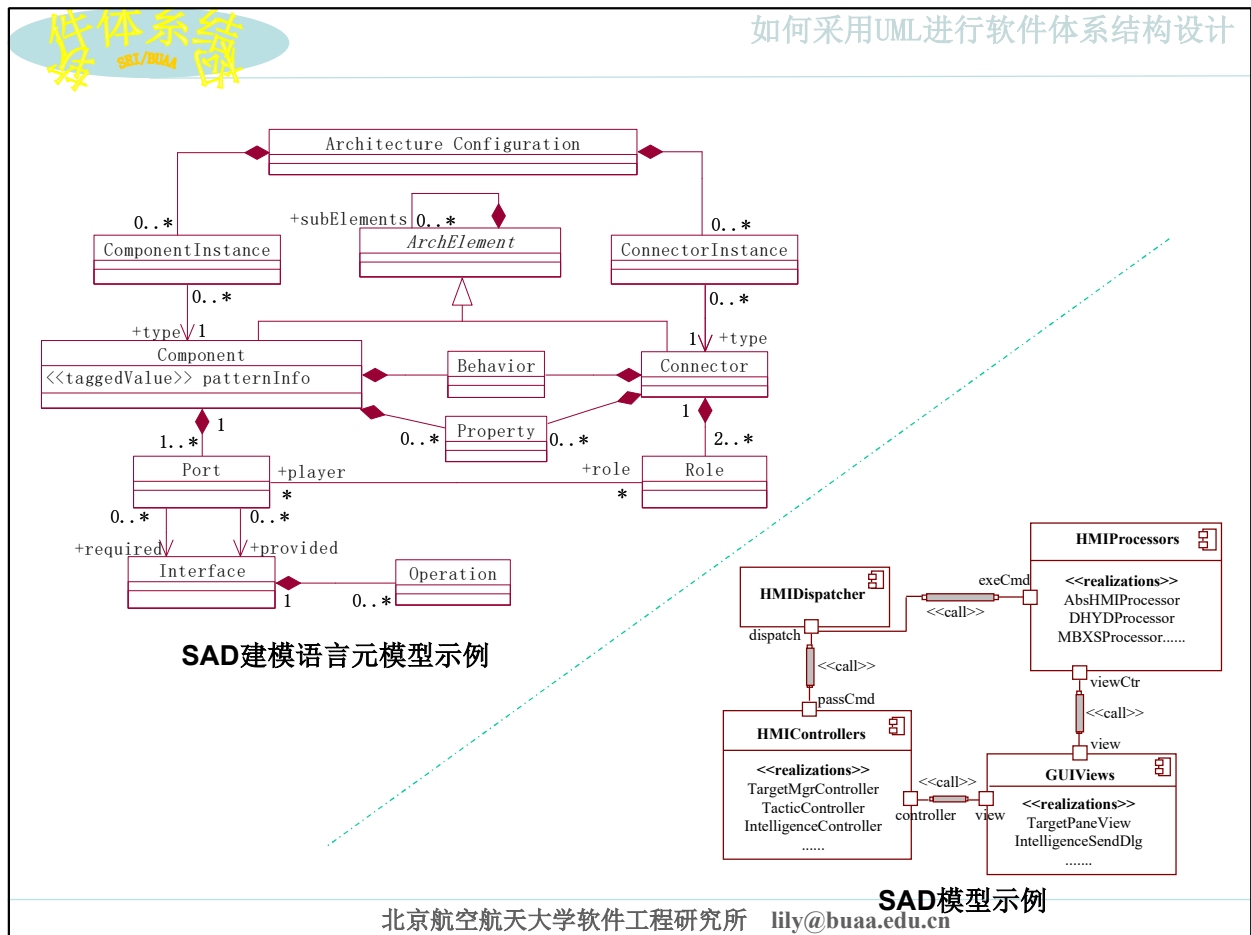
- 对SAS模型进行细化（SAD模型）
- 以更严格的形式对软件构架作出定义
- 为系统的开发和组装提供所需的约束信息

◆ 设计重点

- 做出构件获取方式的决策，即确定系统中哪些构件需要购买/复用/开发
- 对大粒度构件进一步进行分解
- 将构件之间抽象的连接关系具体化
- 为系统中的并发行为设计实现机制
- 识别和完善构件接口，采用严格的规格说明对接口操作进行定义



北京航空航天大学软件工程研究所 lily@buaa.edu.cn





SAS模型到SAD模型的映射规则

- ◆ SAS中的一个构件直接映射为SAD中的一个构件。
- ◆ SAS中的一个抽象构件在SAD中被多个具体的构件代替。
- ◆ SAS中一个构件被分解为SAD中多个粒度更小的构件。
- ◆ 如果SAS中两个构件之间存在连接关系，则为相连的构件分别创建一个端口，将构件之间的连接改为端口之间的连接，并确定具体的连接件类型。
- ◆ 如果一个构件（或其子构件）与另一个构件内部某个子构件存在连接关系，则在两个外部构件之间创建相应的端口和连接件。
- ◆ 将SAS中构件的占位操作<<placeholderOp>>分配到构件的端口上，成为进出端口的服务。
- ◆ SAS中如果一个构件包含属性信息<<placeholderAtt>>，则为对应的SAD构件创建相应的子构件，或映射为定义模型中相应构件的属性。

北京航空航天大学软件工程研究所 lily@buaa.edu.cn



软件构架中期评审

- ◆ 评审时机
 - 得到软件构架定义模型之后
- ◆ 评审目的
 - 对SAD与SAS之间的一致性进行检查
 - 对SAD本身的正确性、一致性和完备性进行检查
- ◆ 评审手段
 - 可以在工具的支持下自动完成

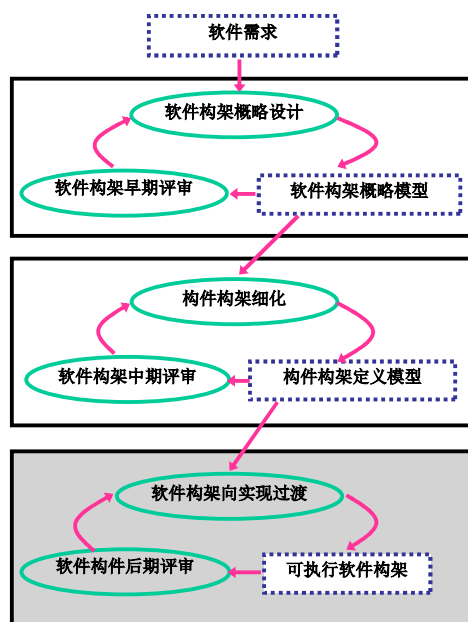
北京航空航天大学软件工程研究所 lily@buaa.edu.cn

软件构架中期评审Checklist

- ◆ 规则C1: 任何构件至少有一个端口。
- ◆ 规则C2: 任何构件只能通过端口与外界进行连接。
- ◆ 规则C3: 一个构件至少与一个其它构件存在连接。
- ◆ 规则C4: 同一构件提供的接口和要求的接口之间不能用连接件进行连接。
- ◆ 规则C5: 一个操作不能同时出现在同一个构件提供的接口和要求的接口当中。
- ◆ 规则C6: 跨越两个构件接口的数据类型必须得到明确的定义。
- ◆ 规则C7: 每个连接件至少与两个构件相连。
- ◆ 规则C8: 任何构件至少参与一个用例交互，否则该构件的功能是多余的。
- ◆ 规则C9: 用例交互中每个参与对象必须属于某种类型。
- ◆ 规则C10: 用例交互中，每个构件接收到的消息（通过其端口）必须是相应端口的一个提供的服务（<<providedOp>>或<<rcvdSignal>>）。

北京航空航天大学软件工程研究所 lily@buaa.edu.cn

软件构架向实现过渡



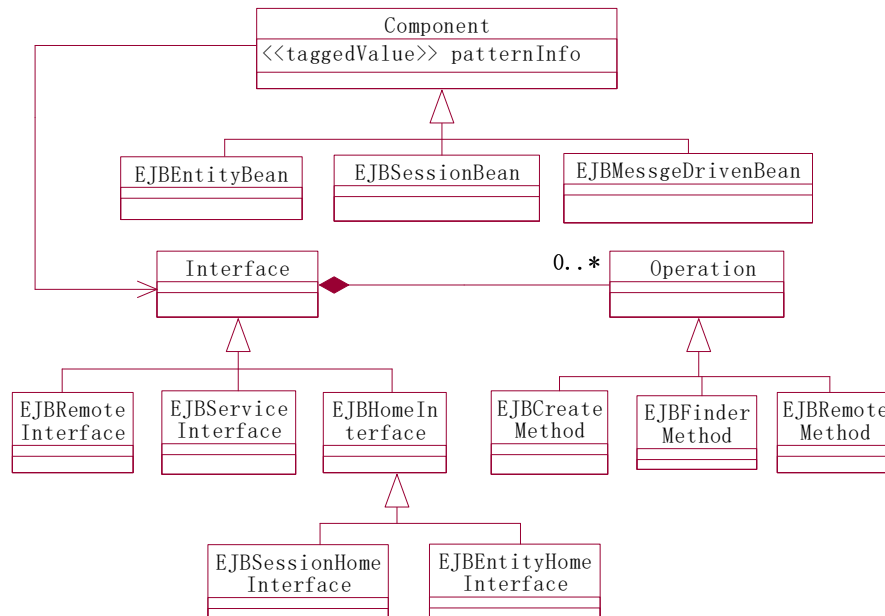
◆ 主要目的

- 完成软件构架定义向特定构件实现技术和运行支撑平台的映射
- 缩短软件构架设计与实现之间的距离

◆ 主要任务

- 以特定构件实现技术相关的概念对软件构架进行描述

北京航空航天大学软件工程研究所 lily@buaa.edu.cn



EJB 构件模型建模语言元模型

北京航空航天大学软件工程研究所 lily@buaa.edu.cn

软件构架后期评审

◆ 评审时机

- 得到软件构架代码框架并被实现和部署之后进行

◆ 评审目的

- 通过对软件构架的模拟运行来对其运行时特性（如性能、并发性、调度、循环、死锁等）进行考察和分析

北京航空航天大学软件工程研究所 lily@buaa.edu.cn



4 软件构架建模语言VAML

- ◆ VAML的设计思想
 - 设计目标
 - 核心概念
 - 实现手段
- ◆ UML2.0软件构架建模能力分析
 - 构件、连接件、接口/端口、行为、构架配置、模式
- ◆ VAML的定义
 - SAS Profile
 - SAD Profile
 - HPSM Profile
 - PDL Profile

北京航空航天大学软件工程研究所 lily@buaa.edu.cn



VAML的设计目标

- ◆ 为设计人员提供简单、易于使用的建模符号
 - 可视化的
 - 与设计人员在实际开发中使用的术语保持一致
 - 具备对语言进行扩展的能力
- ◆ 支持设计人员对软件构架由模糊到清晰、由粗略到详细的认知过程
 - 提供在不同层次以不同细节程度对软件构架进行描述的能力
- ◆ 具备对软件构架模式进行定义和表达的能力
 - 模式是软件构架建模过程中一个非常重要的概

北京航空航天大学软件工程研究所 lily@buaa.edu.cn



VAML中的核心概念

- ◆ 构件
 - 构件指系统中负责进行计算或数据存储的单元
- ◆ 连接件
 - 连接件是对构件之间通信、协调和合作机制的一种抽象
- ◆ 接口
 - 接口是对一组公共特征和职责的声明
- ◆ 构架配置
 - 构架配置用于定义构件和连接件实例互相连接构成的系统的拓扑结构
- ◆ 协作
 - 协作是对构件之间交互场景的描述
- ◆ 细化映射

北京航空航天大学软件工程研究所 jily@buaa.edu.cn



实现手段

- ◆ VAML的定义通过对UML2.0进行扩展来实现
- ◆ UML2.0具有很多可作为软件构架建模语言的潜在优势
 - 作为一种标准建模语言，已为广大开发人员所熟知
 - 本身已具备了软件构架建模的部分支持能力
 - 设计结果与开发过程的其它阶段更容易衔接

北京航空航天大学软件工程研究所 jily@buaa.edu.cn



UML2.0软件构架建模能力分析

◆ 构件建模能力

- 已具有丰富的语义来表示软件构架设计中构件的概念
- UML2.0中Component是一个“大而全”的通用构件，包含了各个阶段建模所需的信息，没有针对不同设计阶段的需要给出关于构件语义的不同约束
- 容易造成不同阶段的设计信息相互混杂，或在设计早期就陷入对细节信息的关注

◆ 连接件建模能力

- UML2.0新增了Connector元类来建模连接件，区分为委托连接件和装配连接件
- UML2.0中Connector只是一种特征(Feature)，而不是一个独立的实体

北京航空航天大学软件工程研究所 lily@buaa.edu.cn



UML2.0软件构架建模能力分析（续）

◆ UML2.0行为建模能力

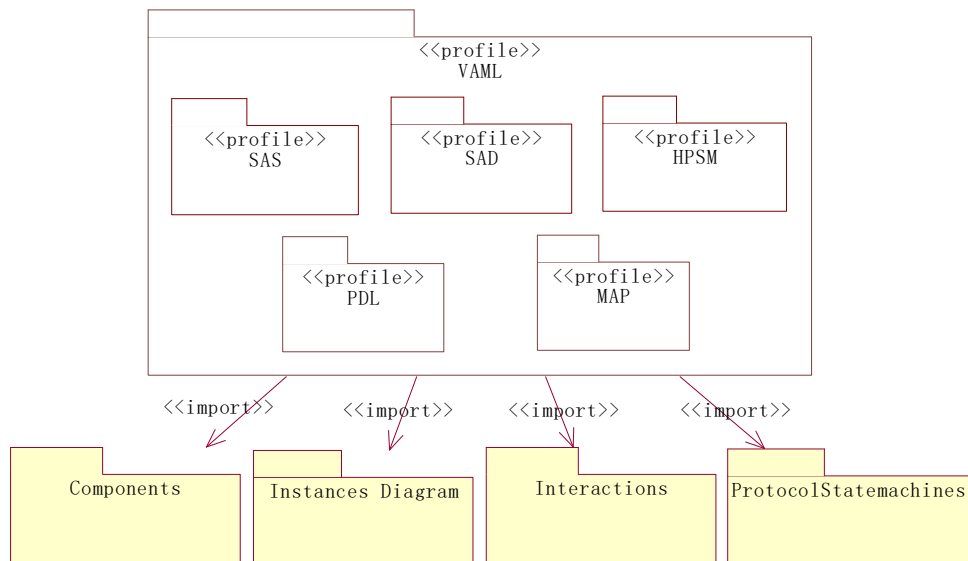
- 顺序图、通信图、交互概览图、时序图、状态机和活动图
- 顺序图用于描述多个构件为完成某一功能而进行的相互协作
 - 用以检验构件之间的控制流程是否合理
 - 用来辅助设计人员识别和捕获构件的潜在职责或接口需求
- 状态图主要用于在不限任何特定行为实现技术的条件下，强化构件、连接件、端口、接口等的合法使用场景

◆ 构架配置建模

- 用组装连接件来表示构件之间的装配关系

北京航空航天大学软件工程研究所 lily@buaa.edu.cn

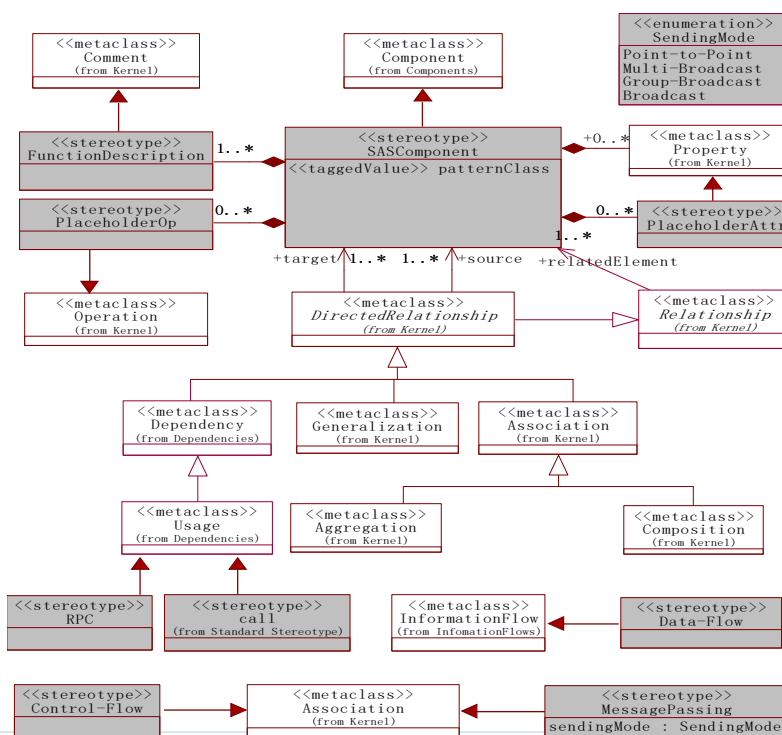
VAML的定义



VAML Profile顶层组织结构

北京航空航天大学软件工程研究所 lily@buaa.edu.cn

SAS Profile中主要建模概念



• <<SASComponent>>

主要侧重对构件功能性的描述，使设计人员可以在不必对构件接口和行为细节进行深入考察的情况下了解构件在系统中的主要作用和职责

• 关系

侧重于抽象表达构件之间在控制和数据上的相互联系，而不揭示构件之间交互机制的具体细节

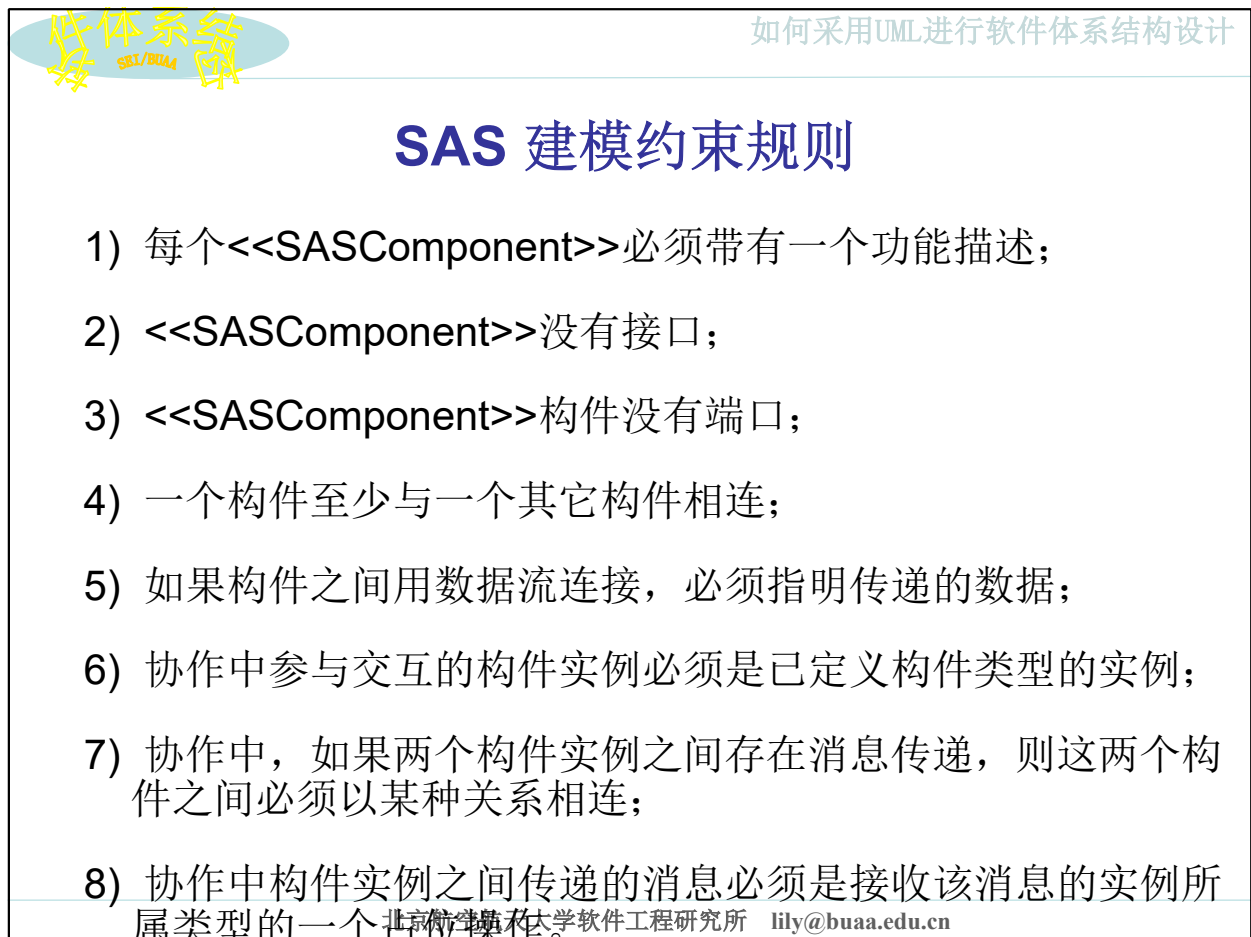
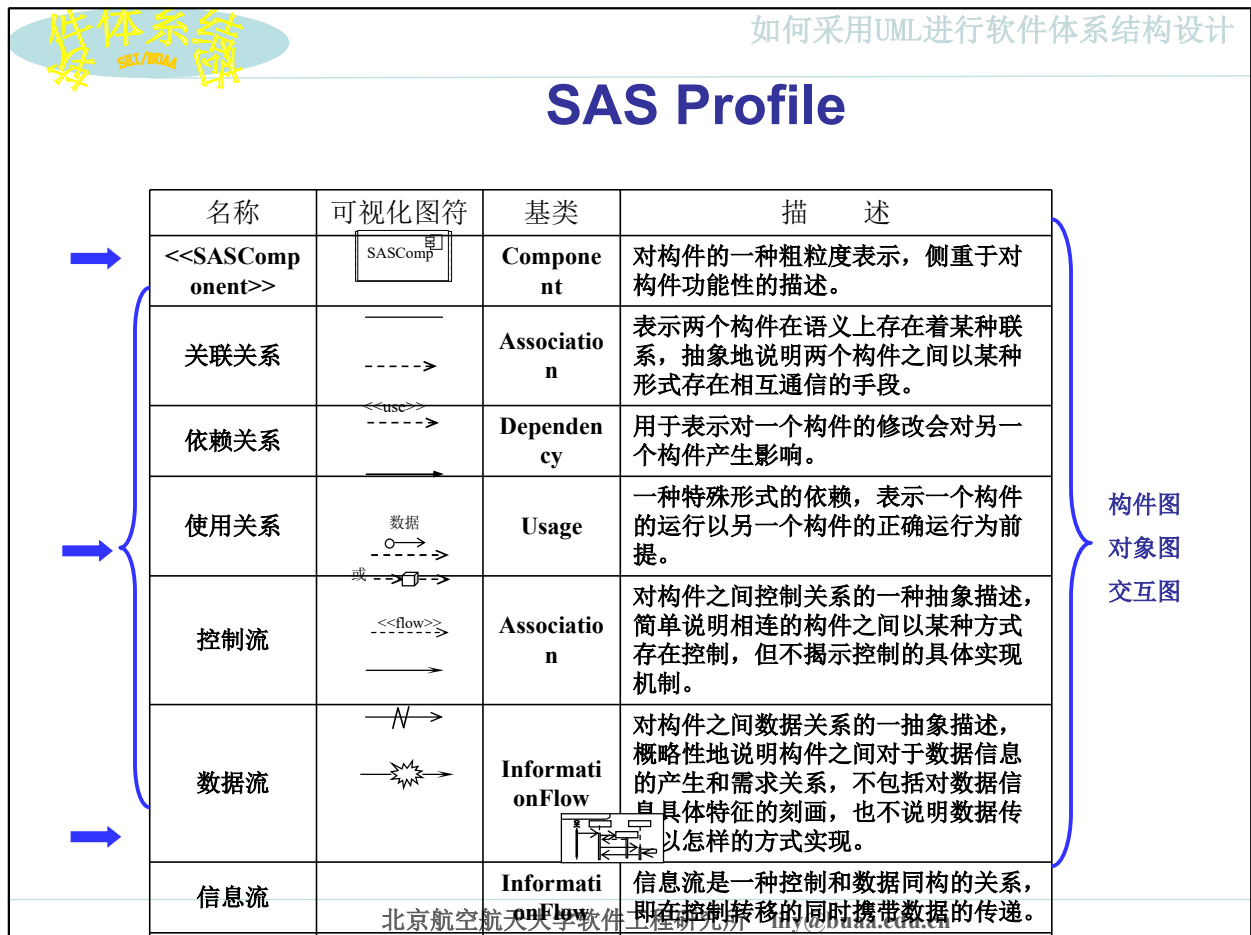
• 协作

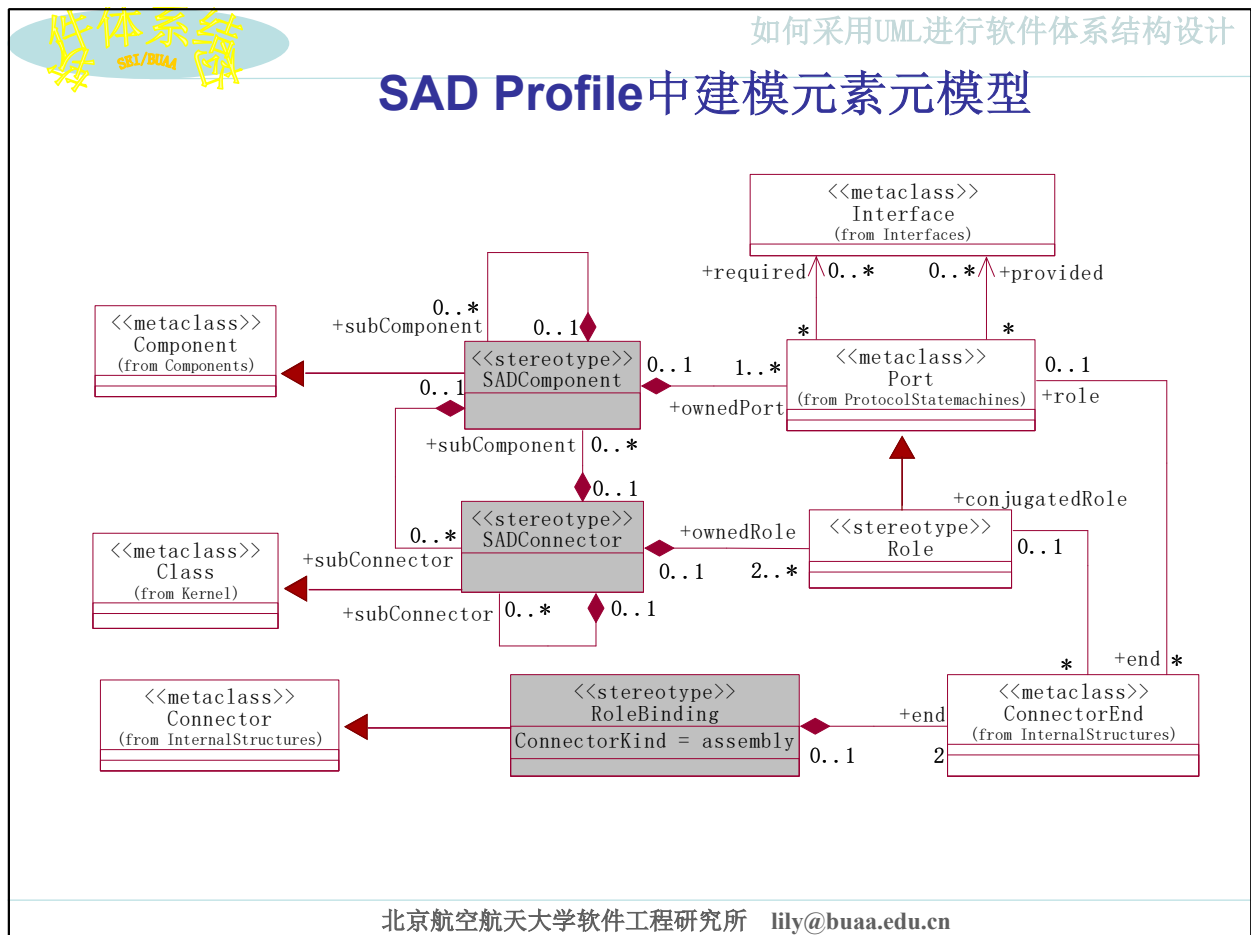
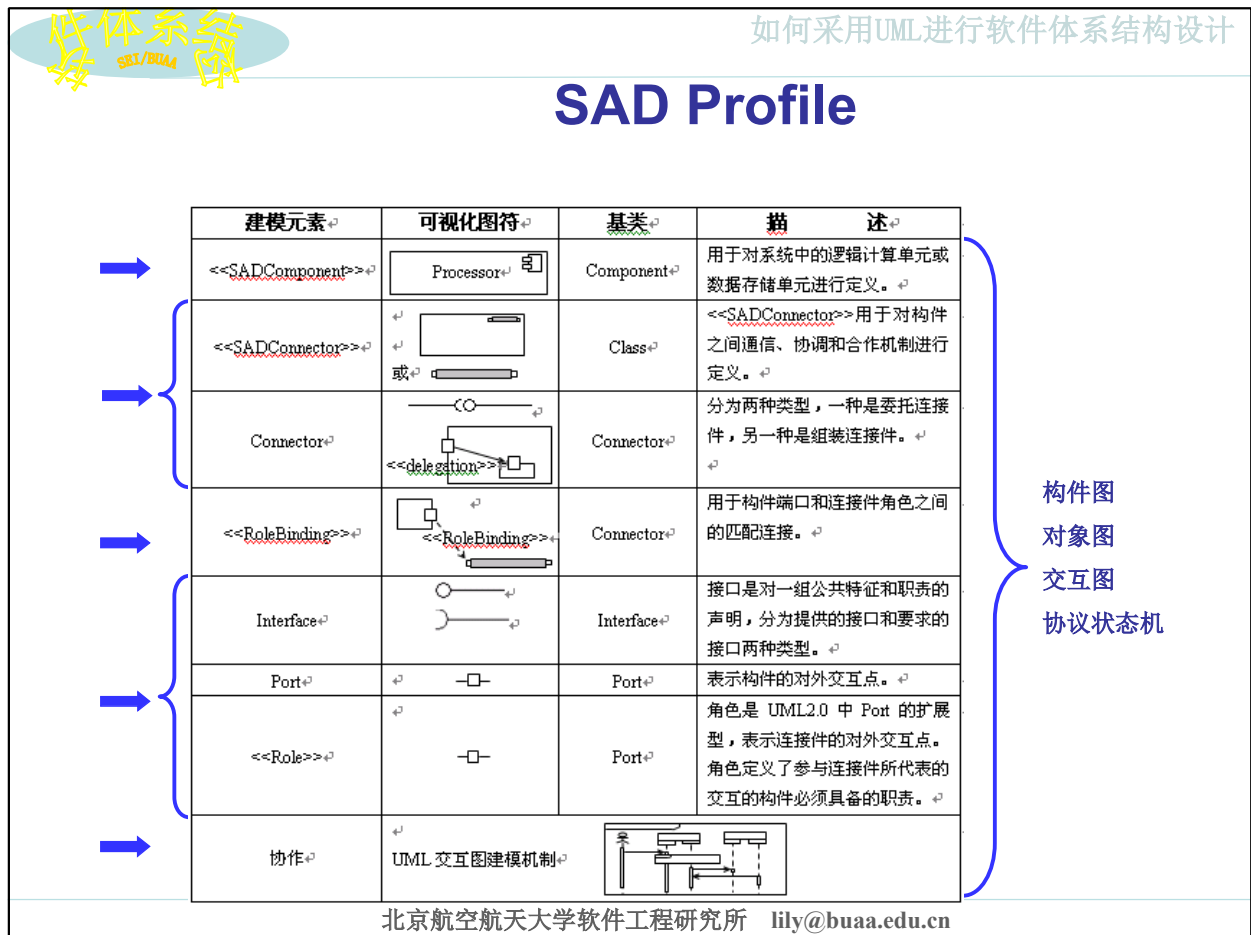
引入UML交互图建模机制

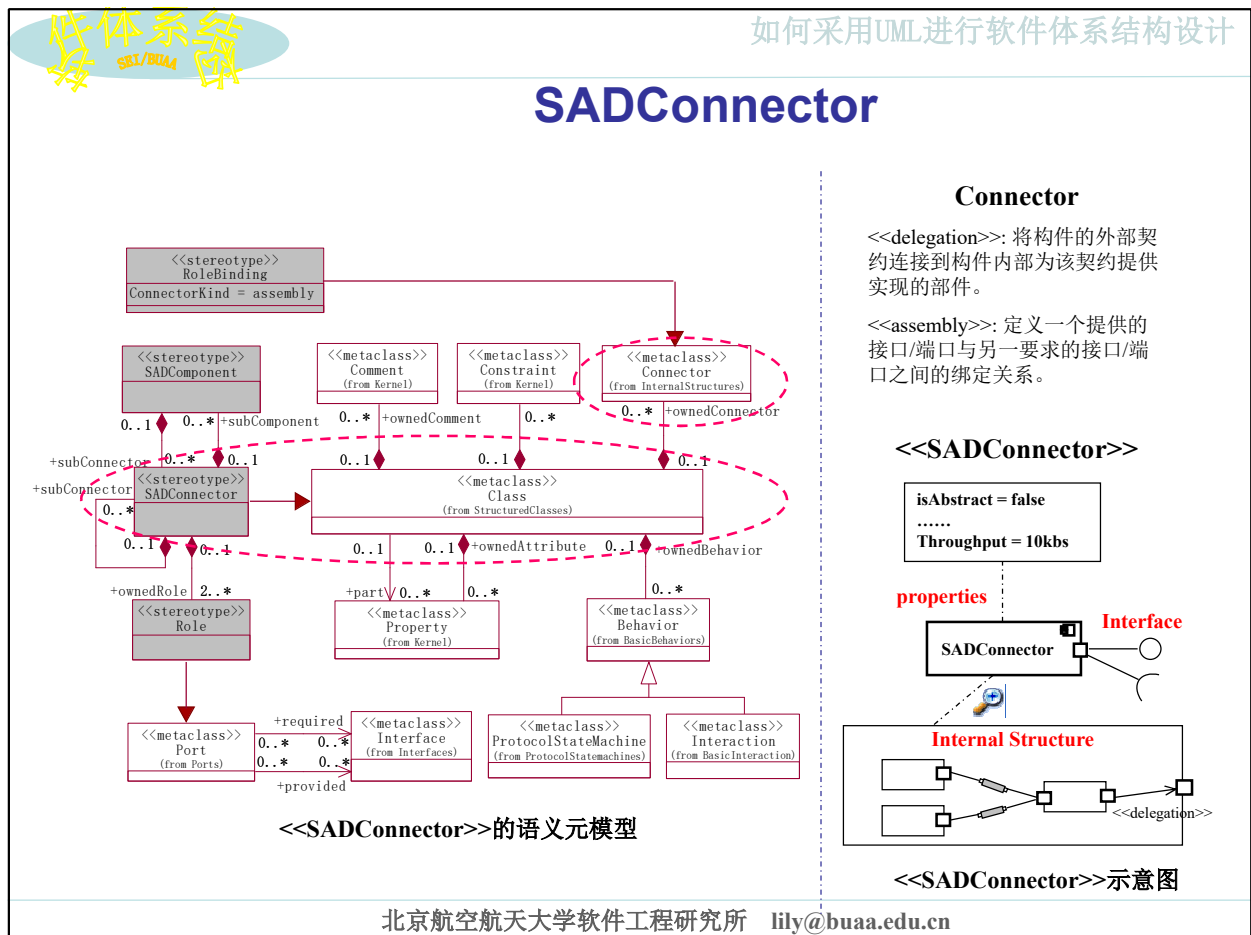
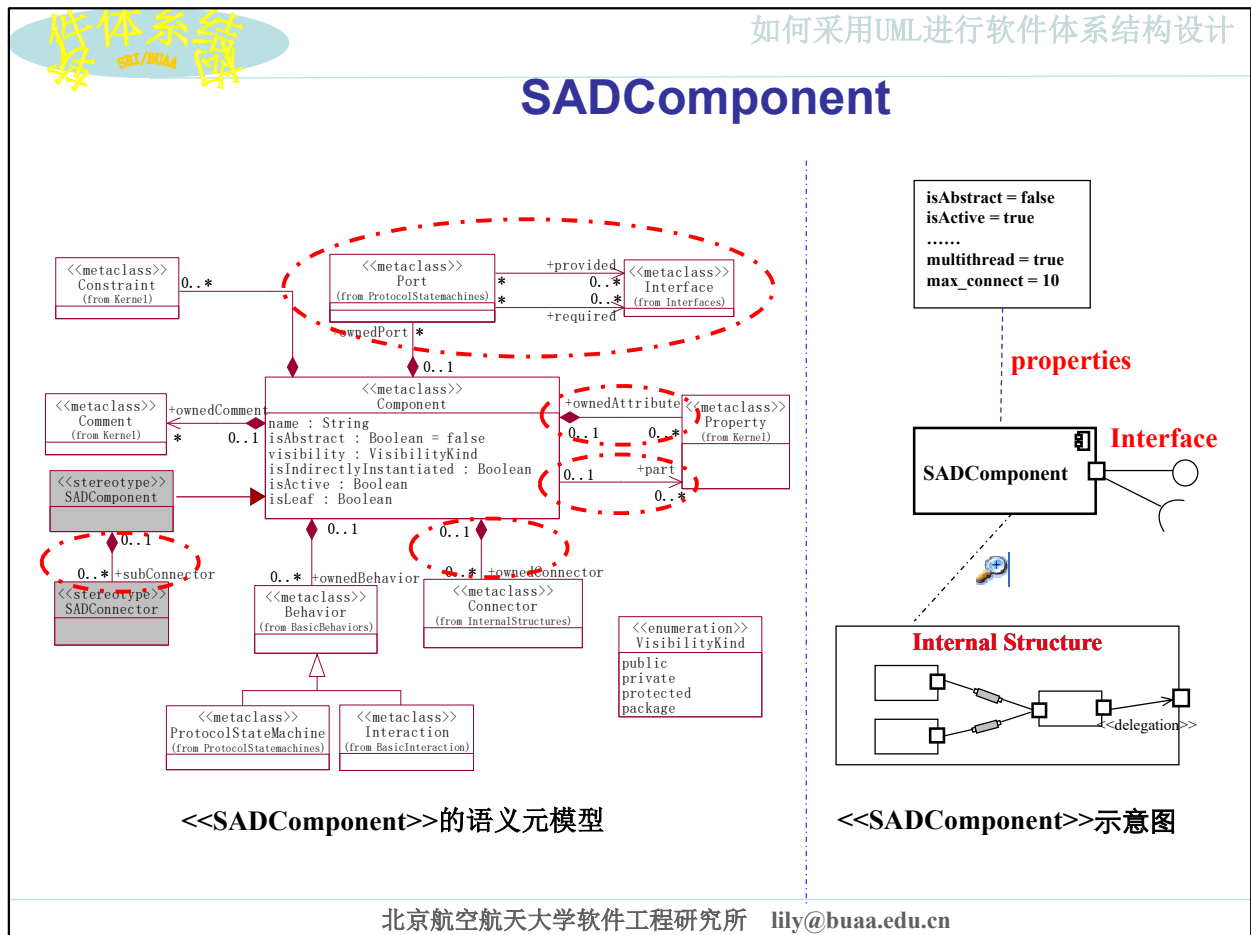
分析和验证控制和数据在构件之间的传递流程是否合理

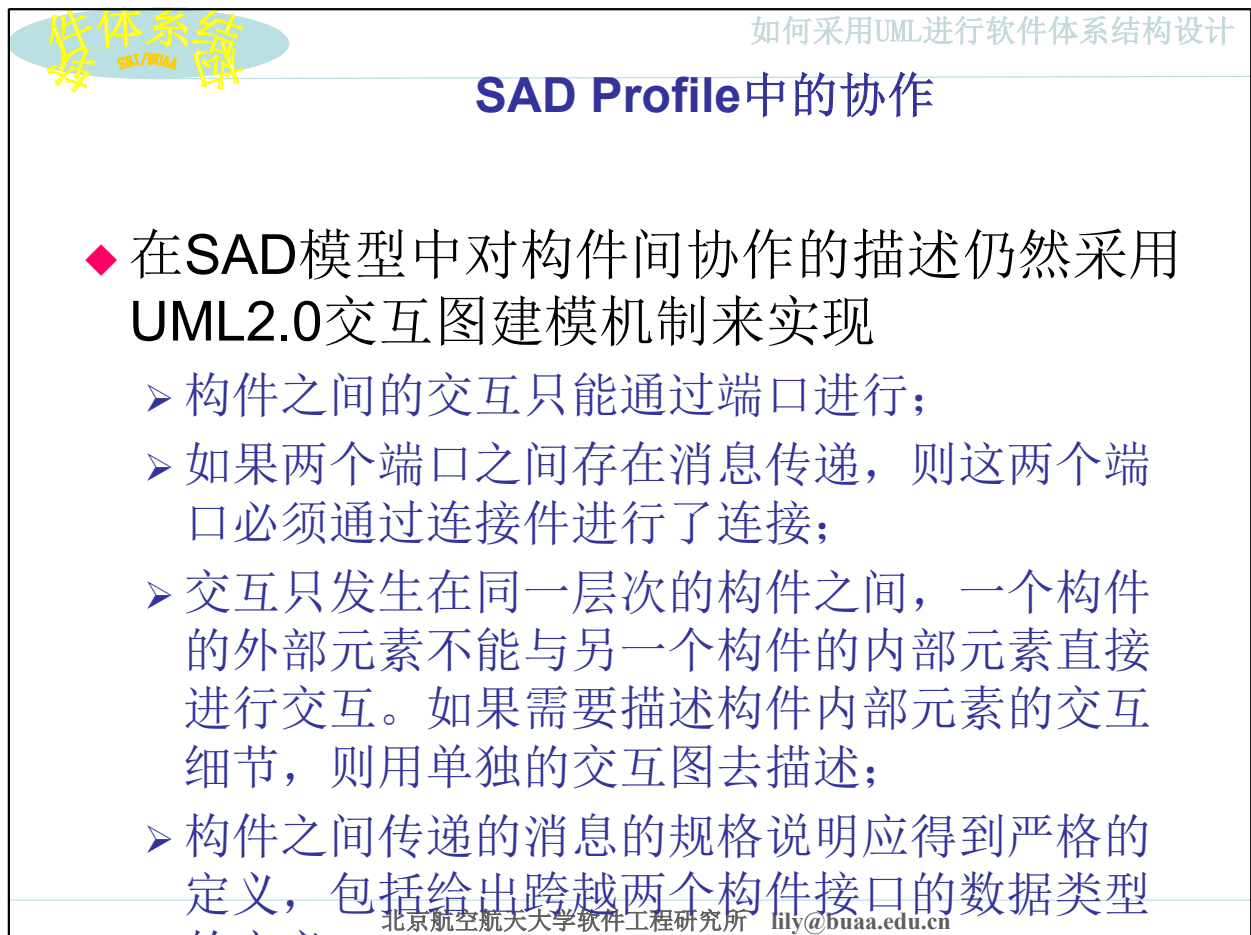
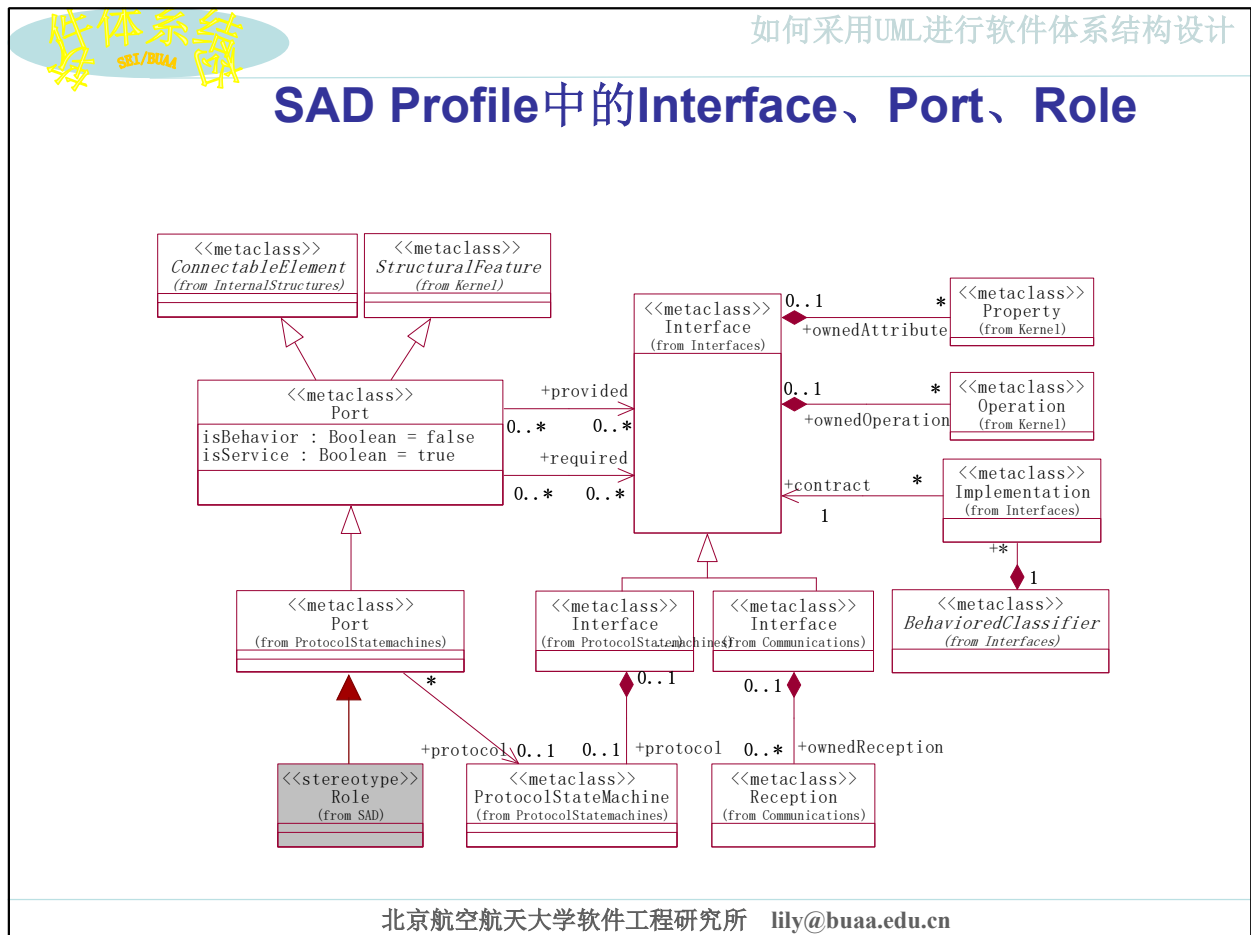
不刻画控制和数据传递的具体细节

北京航空航天大学软件工程研究所 lily@buaa.edu.cn

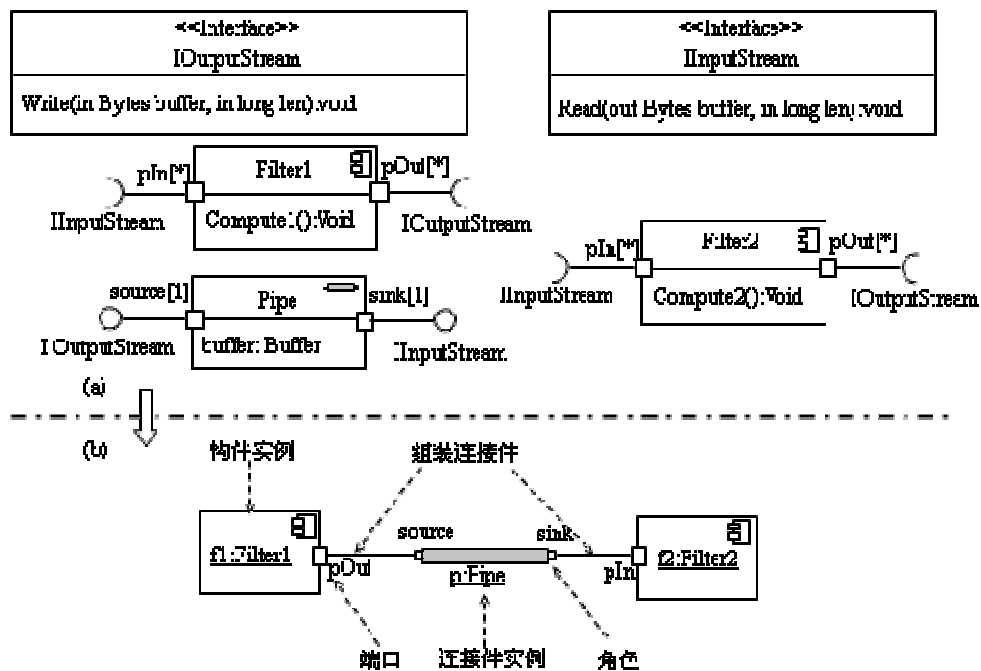








SAD Profile中构架配置的描述

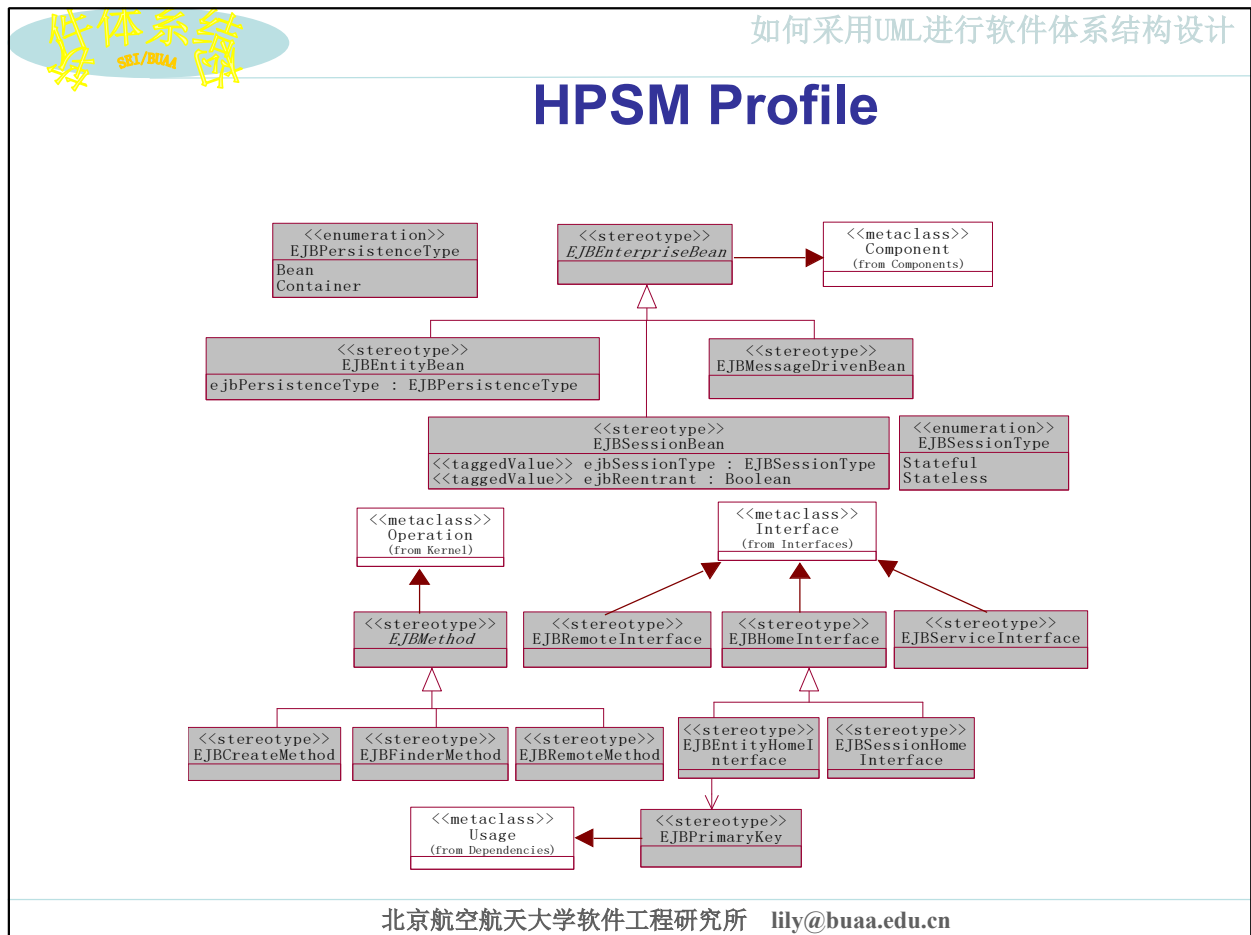


北京航空航天大学软件工程研究所 lily@buaa.edu.cn

SAD 约束规则

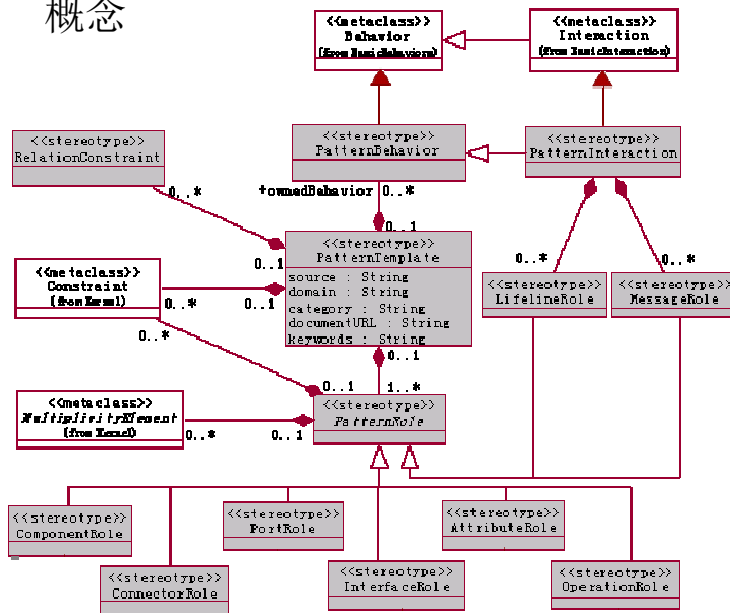
- 1) 每个<<SADComponent>>至少有一个端口，每个端口至少与一个接口相关联；
- 2) 端口不能独立被创建，只能在拥有该端口的构件被创建之后才能创建，并随构件的删除而被删除；
- 3) 角色不能独立被创建，只能在拥有该角色的连接件被创建之后才能创建，并随连接件的删除而被删除；
-
- 9) 角色绑定只能用于连接构件的端口和连接件的角色；
- 10) 如果构件的端口与连接件的角色通过角色绑定来连接，则构件的端口必须提供或要求角色所要求或提供的接口；
- 11) 接口定义了一组声明，因此不能被实例化；

北京航空航天大学软件工程研究所 lily@buaa.edu.cn



PDL Profile

PDL Profile中定义了一组用于对模式进行定义和表示的建模概念



一个模式由一组角色及这些角色在结构和行为上的约束关系组成。

每个角色代表模式的一个重要特征，如拥有的构件、连接、接口、操作、属性等。

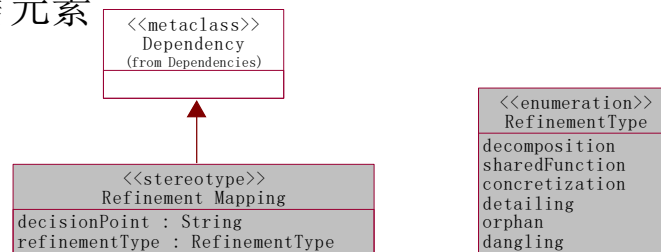
模式被实例化时，模式角色将被实际的构件、连接、接口、操作、属性等所代替。

<<RelationConstraint>>定义在模式实例化时扮演相应角色的实际模型元素之间必须具有的关系

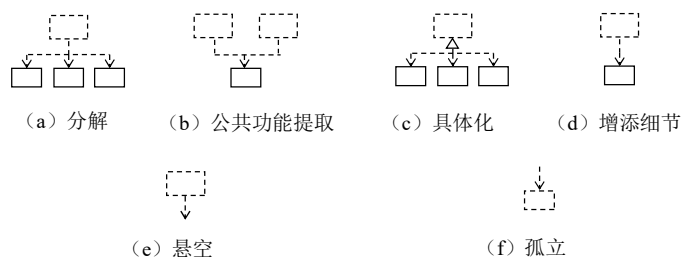
北京航空航天大学软件工程研究所 lily@buaa.edu.cn

MAP Profile

◆ MAP Profile中定义了对不同层次模型间细化关系进行描述的语言元素



MAP Profile元模型



北京航空航天大学软件工程研究所 lily@buaa.edu.cn



各Profile中的主要建模概念

profile 核心 概念	SAS	SAD	HPSM	PDL	MAP
构件	<<SASComponent>>	<<SADComponent>>	<<EJBEnterpriseBean>> <<EJBEntityBean>> <<EJBMessageDrivenBean>> <<EJBSessionBean>>	<<PatternTemplate>> <<PatternRole>> <<ComponentRole>> <<ConnectorRole>>	<<RefinementMapping>> 具体映射类型类型包括: {decomposition, sharedFunction, concretization, detailing, dangling, Orphan}
连接件	<<Control-Flow>> <<Data-flow>> InformationFlow Association Dependency Usage <<call>> <<RPC>> <<MessagePassing>>	<<SADConnector>> Connector {assembly, delegation} <<RoleBinding>>	<<RMI>> <<call>>	<<PortRole>> <<InterfaceRole>> <<AttributeRole>> <<OperationRole>> <<RelationConstraint>> <<PatternInteraction>> <<LifetimeRole>>	
		Interface Port	<<EJBEntityHomeInterface>>		

北京航空航天大学软件工程研究所 lily@buaa.edu.cn



5 软件构架建模支持工具ArchME

- ◆ 软件构架建模支持工具的设计需求
- ◆ 软件构架建模环境ArchME功能概述
- ◆ ArchME的设计和实现方案



软件构架建模支持工具的设计需求

◆ 软件构架模型的表达能力

- 能为软件构架模型的表达提供有效的建模符号
 - 应简单、易掌握、支持软件构架由抽象到具体、由粗略到详细的逐步演化
 - 最好是可视化的
- 建模语言的选择是建模工具设计中必须考虑的重要问题之一

◆ 不同层次设计结果的衔接

- 下层模型通常在上层模型的基础上得到
- 应提供由上层模型快速获得下层模型初始结构的机制

北京航空航天大学软件工程研究所 lily@buaa.edu.cn



软件构架建模支持工具的设计需求 (续)

◆ 为设计人员提供辅助决策的必要手段

- 帮助设计人员搜寻和产生明智的决策，增加设计决策的效率和质量

◆ 模型检查和分析能力

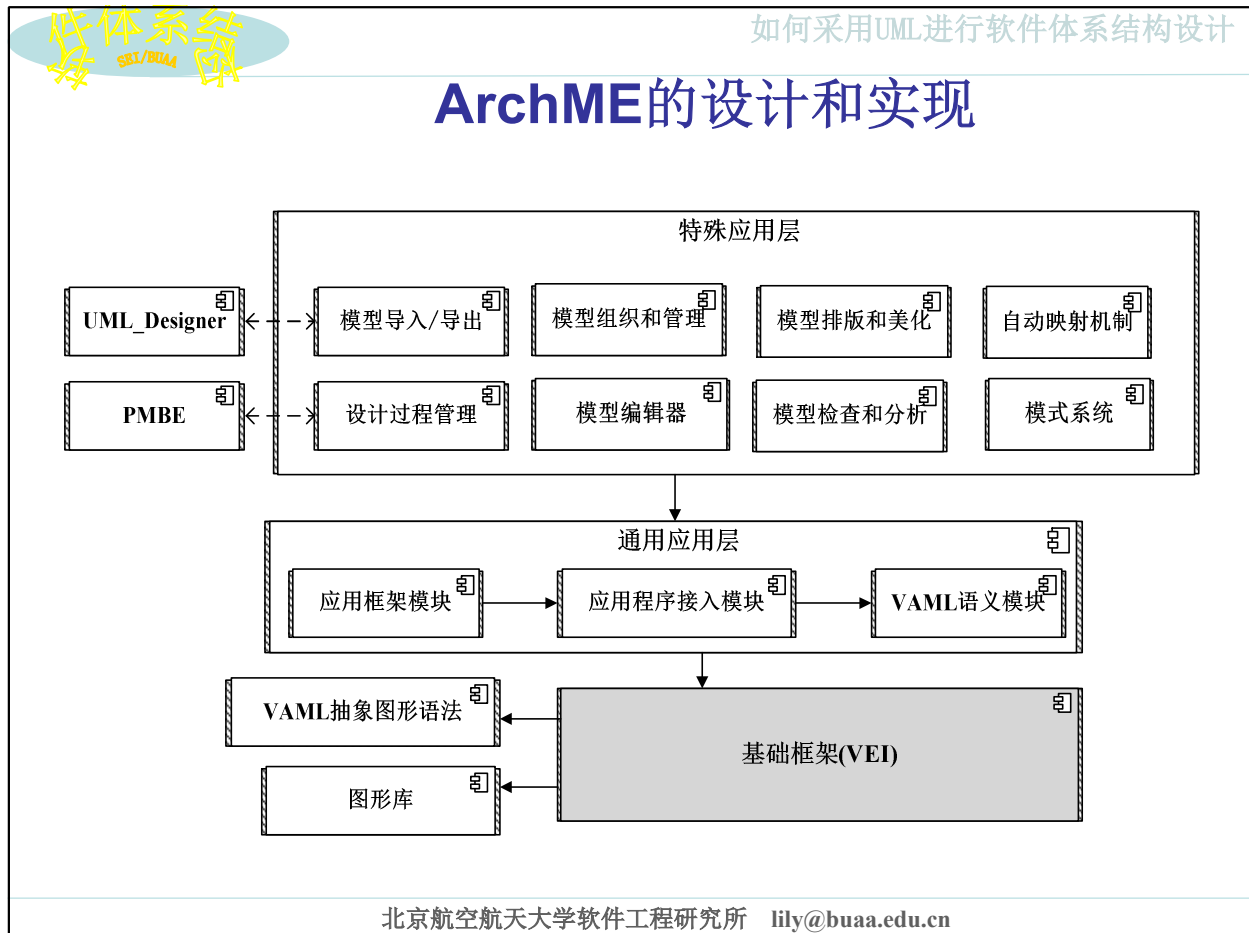
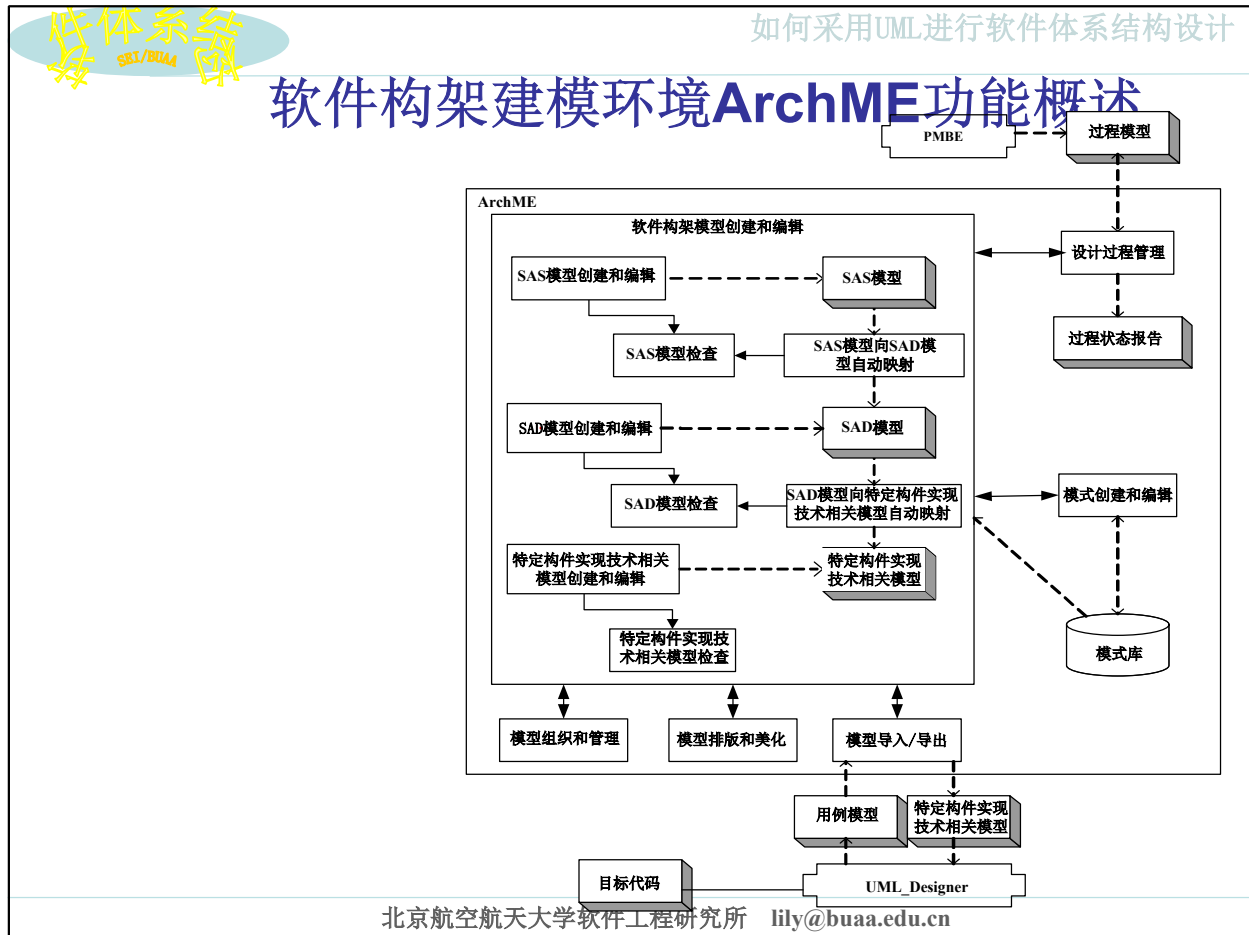
- 保证模型语法上的正确性、一致性和完备性

◆ 工具易用性方面的考虑

- 模型创建：减少用户因频繁地进行鼠标移动而分散设计精力的可能性
- 排版支持：允许用户以相对比较随意的方式创建模型元素
- 信息过滤：允许用户根据需要进行选择显示或隐藏

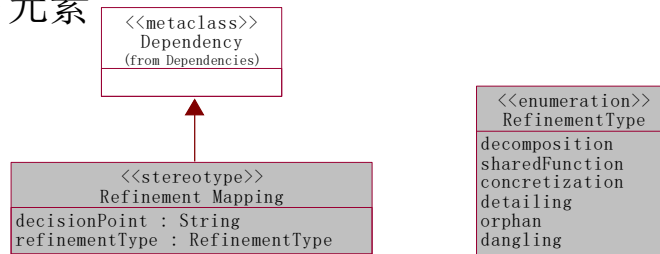
甘米信自

北京航空航天大学软件工程研究所 lily@buaa.edu.cn

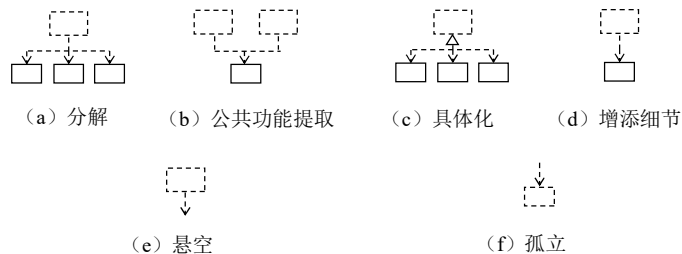


MAP Profile

- ◆ MAP Profile中定义了对不同层次模型间细化关系进行描述的语言元素



MAP Profile元模型



北京航空航天大学软件工程研究所 lily@buaa.edu.cn

各Profile中的主要建模概念

profile 核心 概念	SAS	SAD	HPSM	PDL	MAP
构件	<<SASComponent>>	<<SADComponent>>	<<EJBEnterpriseBean>> <<EJBEntityBean>> <<EJBMessageDrivenBean>> <<EJBSessionBean>>	<<PatternTemplate>> <<PatternRole>> <<ComponentRole>> <<ConnectorRole>> <<PortRole>> <<InterfaceRole>> <<AttributeRole>> <<OperationRole>> <<RelationConstraint>> <<PatternInteraction>> <<LifecycleRole>>	<<RefinementMapping>> 具体映射类型类型包括: {decomposition, sharedFunction, concretization, detailing, dangling, Orphan}
连接件	<<Control-Flow>> <<Data-flow>> InformationFlow Association Dependency Usage <<call>> <<RPC>> <<MessagePassing>>	<<SADConnector>> Connector {assembly, delegation} <<RoleBinding>>	<<RMI>> <<call>>		
		Interface Port	<<EJBEntityHomeInterface>>		

北京航空航天大学软件工程研究所 lily@buaa.edu.cn