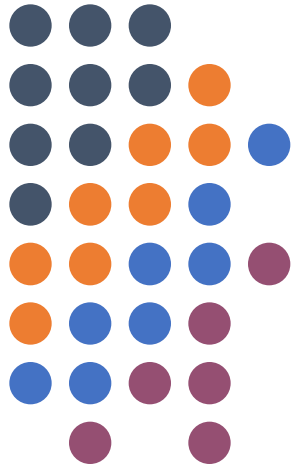
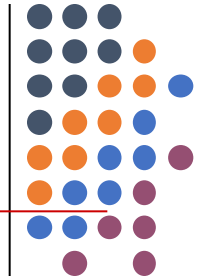
统一建模语言

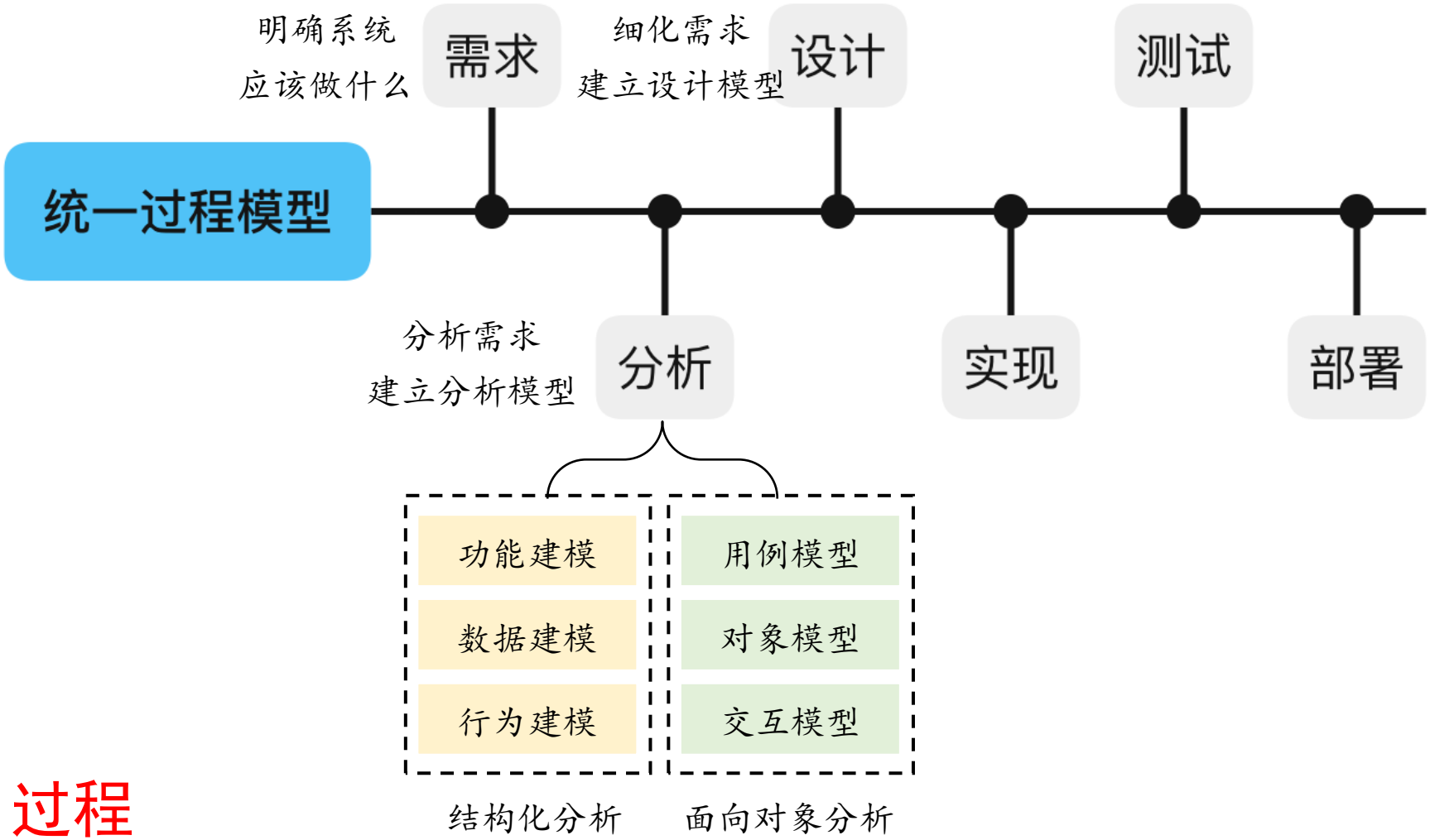


相关章节



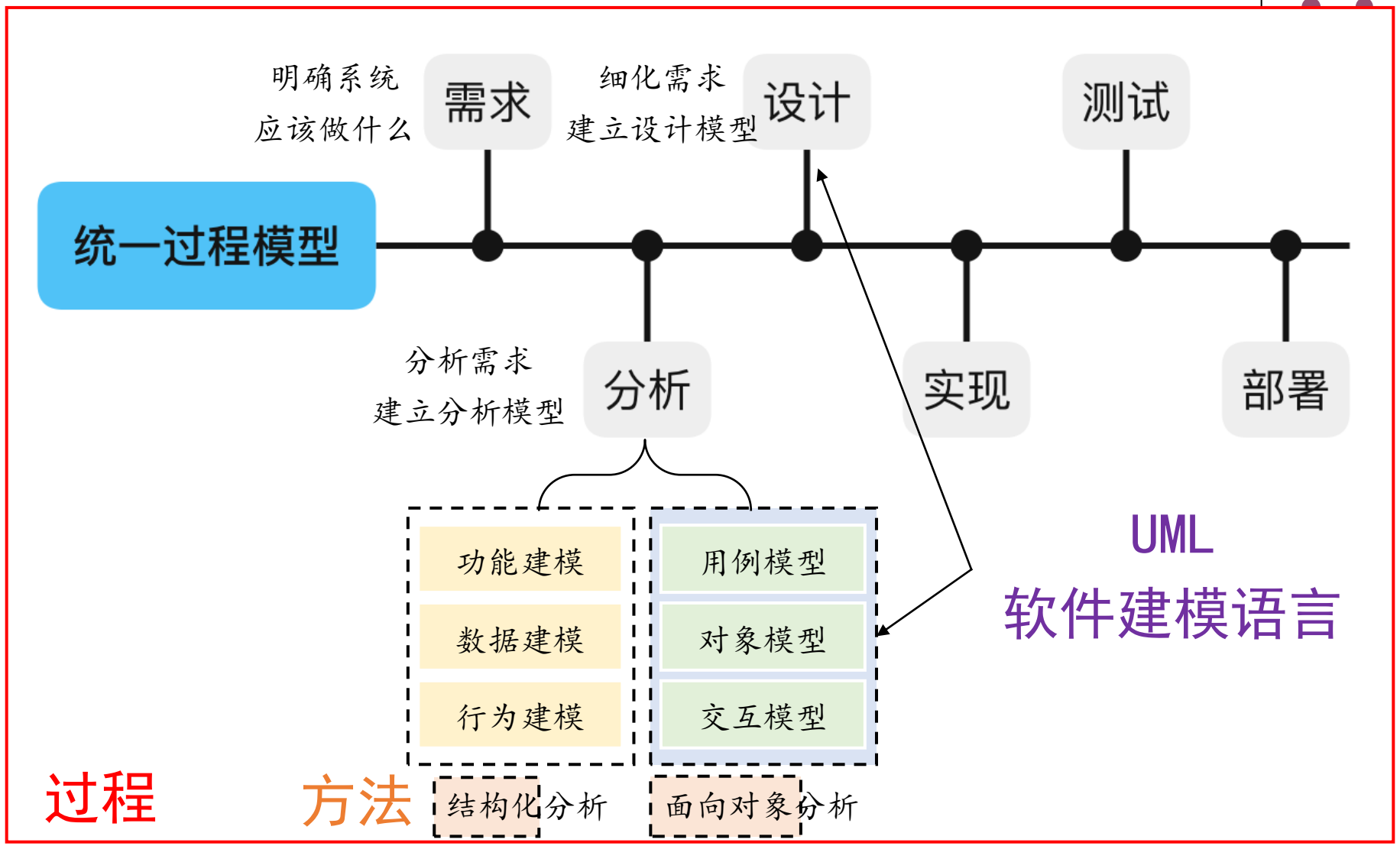
- 概论：1.1 ~ 1.4
- 软件生存期模型：2.1 ~ 2.6, 2.8
- 需求获取：3.1, 13.5.1, 13.5.2
- 结构化分析：3.2
- UML简介：5.2
- UML的事物：5.3

回顾RUP过程


明确系统
应该做什么
需求
细化需求
建立设计模型
设计
测试
统一过程模型
分析需求
建立分析模型
分析
实现
部署
功能建模
数据建模
行为建模
用例模型
对象模型
交互模型
结构化分析
面向对象分析

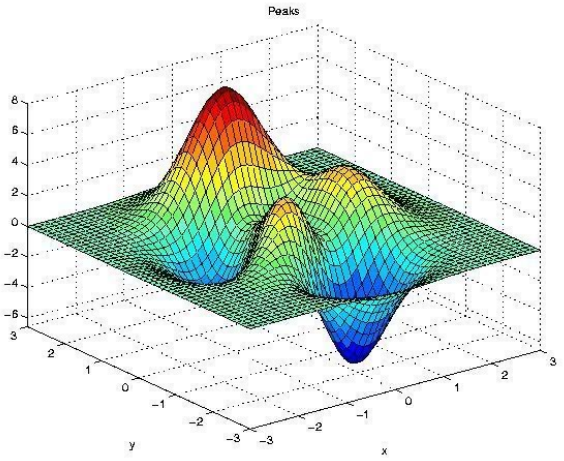
过程

本节所讲内容所处的位置



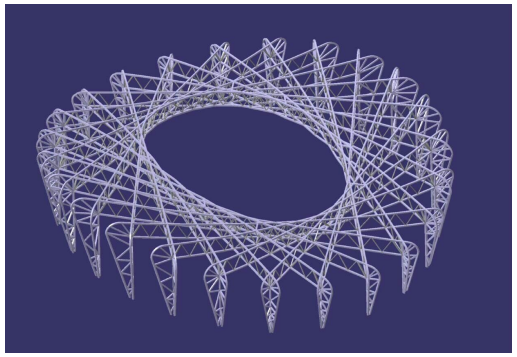
软件建模

模型是对某个客观事物、规律进行抽象后的一种形式化表达。



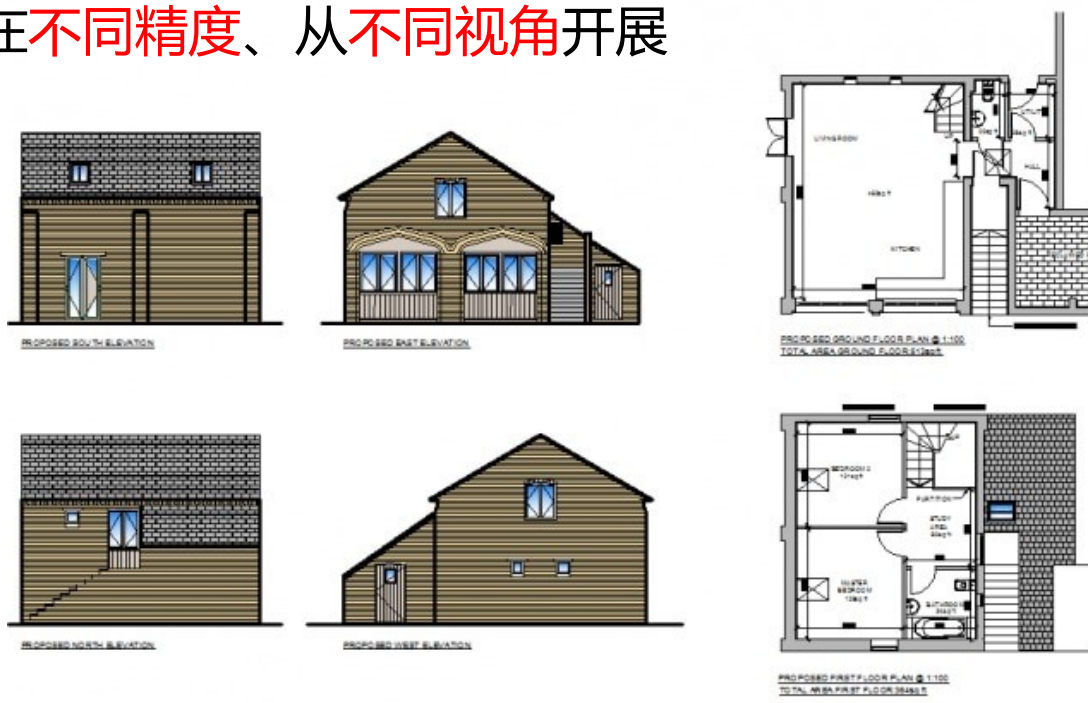
软件建模

- **建模的重要性**
  - 建模产生的结果就是模型，模型是对现实的简化
  - 模型有助于使人更好地了解事物本质
    - 在模型中，总是剔除那些与问题无关的非本质内容
  - 模型可以帮助人们对系统进行可视化



软件建模

- **建模的特点**
  - 每一个系统可以从不同的方面使用不同的模型进行描述
  - 每个模型都是对系统从语义上近似的抽象
    - 单个模型是不充分的
  - 建模可以在不同精度、从不同视角开展



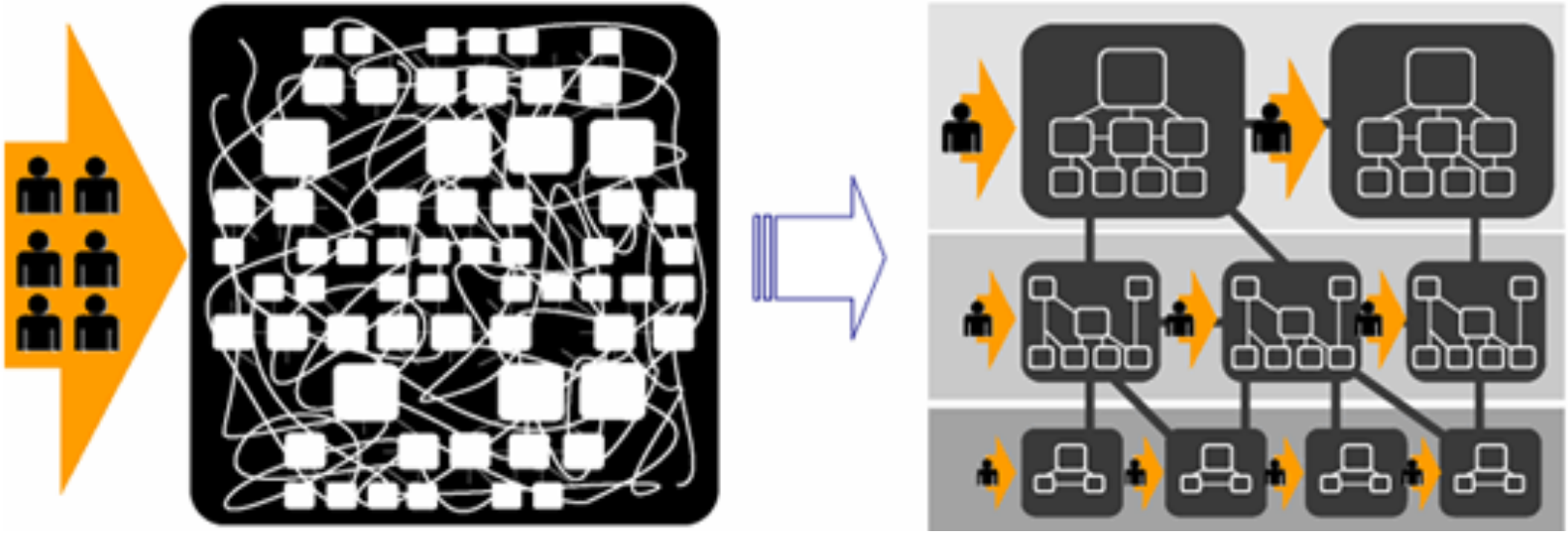
软件建模

软件分析建模从不同侧面描述软件系统的数据信息、处理功能以及运行的外部行为，有效简化和处理复杂性，将需求映射到软件结构中。

- 辅助沟通与组织
- 更早地发现问题或遗漏，为代码生成提供依据
- 一图胜过千言万语！

软件建模

软件分析建模从不同侧面描述软件系统的数据信息、处理功能以及运行的外部行为，有效简化和处理复杂性，将需求映射到软件结构中。

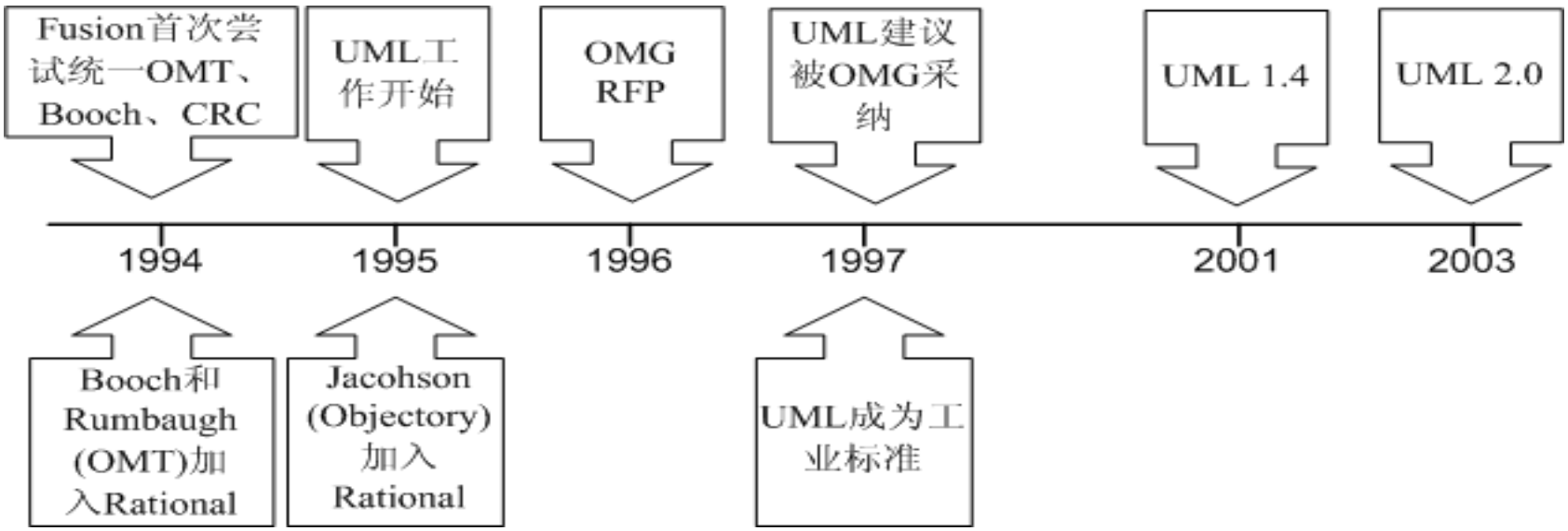


统一建模语言 UML

- 面向对象的建模语言很多，目前使用最广泛的是统一建模语言(UML，Unified Modeling Language)
- 主要贡献者
  - Grady Booch
  - Ivar Jacobsom
  - Jim Rumbaugh
  - Rational Software公司开发

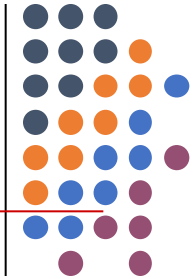
统一建模语言 UML

- UML 发展历程



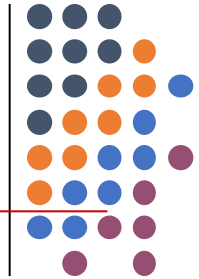
- 1997年11月被OMG采纳作为基于面向对象技术的标准建模语言
- 1998、2000、2001、2003、2005年分别发布了UML1.2、UML1.3、UML1.4、 UML1.5、 UML2.0
- 2011年发布了UML2.4，UML2.4.1
- **2013年发布了UML2.5**

面向对象建模语言



- 基于**面向对象的概念和抽象**，提供图形化的**图符**，用来**表示**软件系统的一种语言
- 目的
  - **用于建模**：绘制和描述软件系统模型(分析模型和设计模型)
  - **支持交流**：便于开发人员之间的交流、沟通和讨论
- 组成
  - **语法**：图形化的符号表示
  - **语义**：形式或半形式的语义
  - **语用**：如何使用语言来建立模型、提供策略和原则

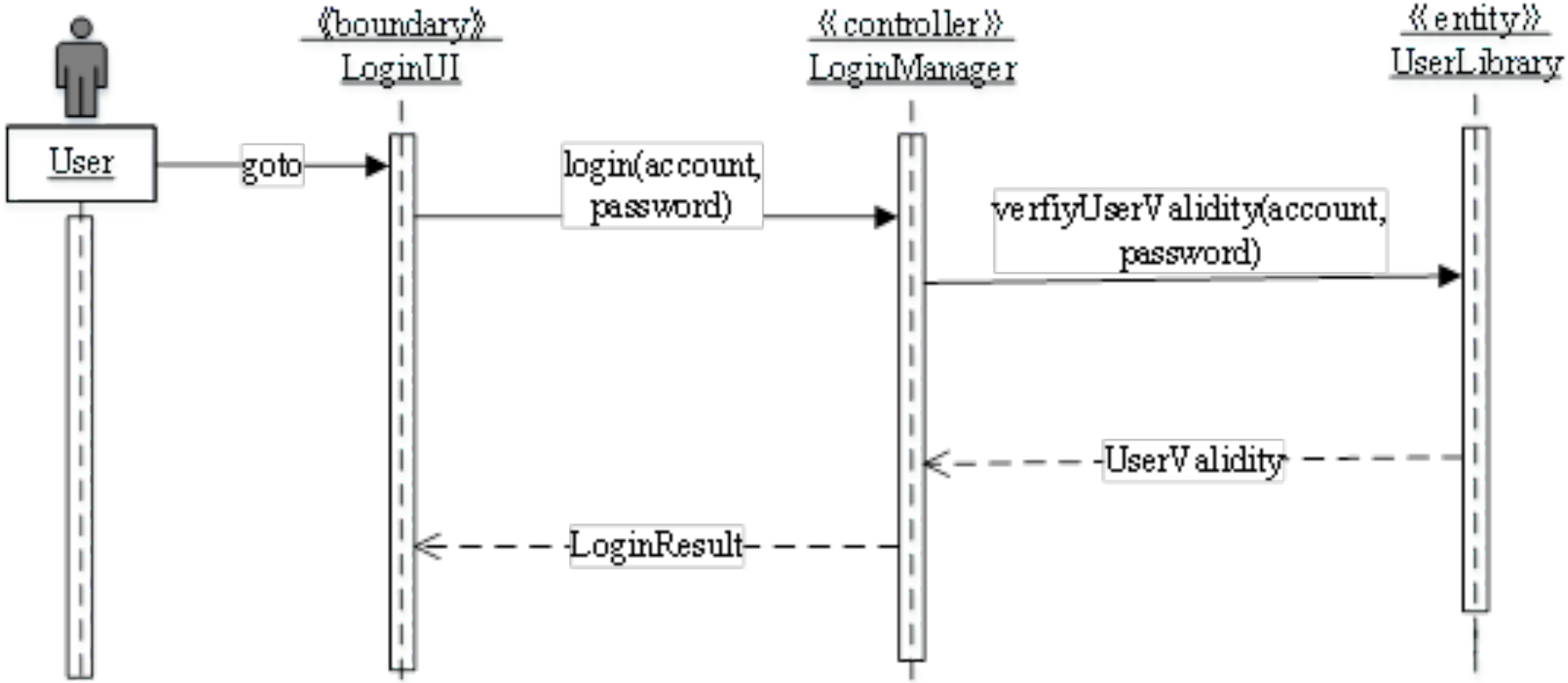
统一建模语言 UML



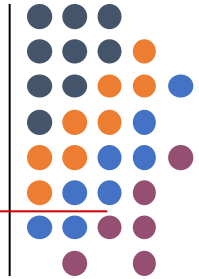
- Unified(统一)
  - 提取不同方法中最好建模技术，如OMT(James Rumbaugh)，Booch method(Grady Booch)和OOSE(Ivar Jacobson)
  - 采用**统一、标准化**的表示方式
- Modeling(建模)
  - 对现实系统和软件系统进行**可视化建模**
  - 建立系统模型
- Language(语言)
  - **图形化语言**：语法、语义和语用
  - 包括规则，约束 扩展机制

面向对象建模语言

- 例

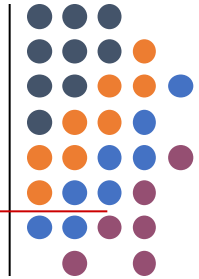

《boundary》
LoginUI
《controller》
LoginMnager
《entity》
UserLibrary
User
goto
login(account,
password)
verfiyUserValidity(account,
password)
UserValidity
LoginResult

统一建模语言 UML 建模视角



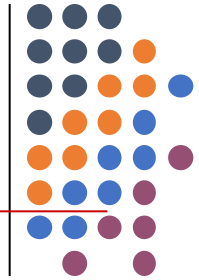
- 结构视角（Structural View）
  - **用于描述系统的构成**
  - UML提供了包图（Package Diagram）、类图（Class Diagram）、对象图（Object Diagram）和构件图（Component Diagram），从不同的抽象层次来表示系统的静态组织及结构
- 行为视角（Behavioral View）
  - **刻画系统的行为**
  - UML提供了交互图（Interaction Diagram）、状态图（Statechart Diagram）与活动图（Activity Diagram），以从不同侧面刻画系统的动态行为。

统一建模语言 UML 建模视角



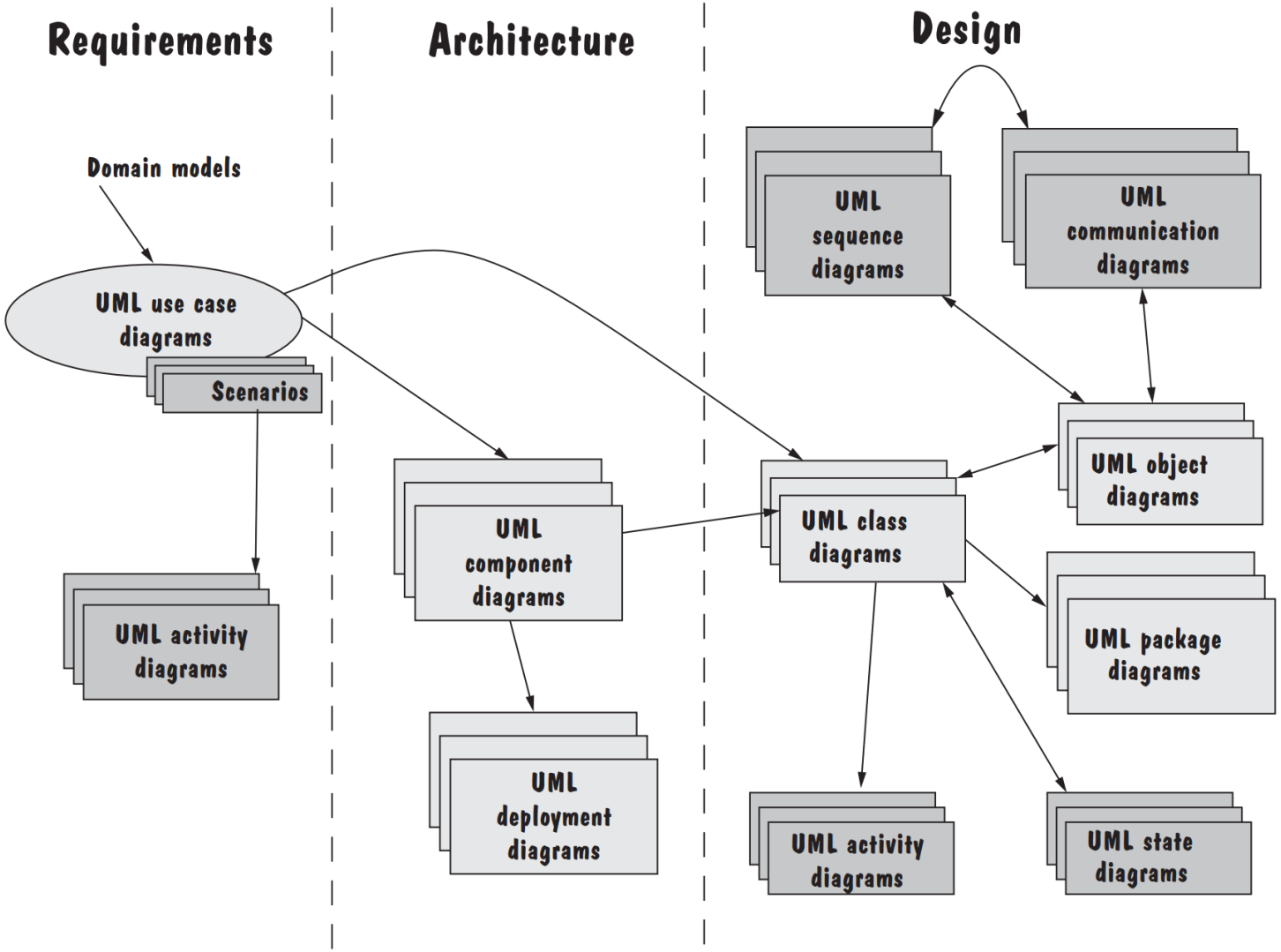
- 部署视角（Deployment View）
  - **刻画目标软件系统的软件制品及其运行环境**
  - UML提供了部署图（Deployment Diagram）来描述软件系统的部署模型
- 用例视角（Use Case View）
  - **刻画系统的功能**
  - UML提供了用例图（Use Case Diagram）以描述系统的用例及其与外部执行者之间的关系。

统一建模语言 UML 建模视角



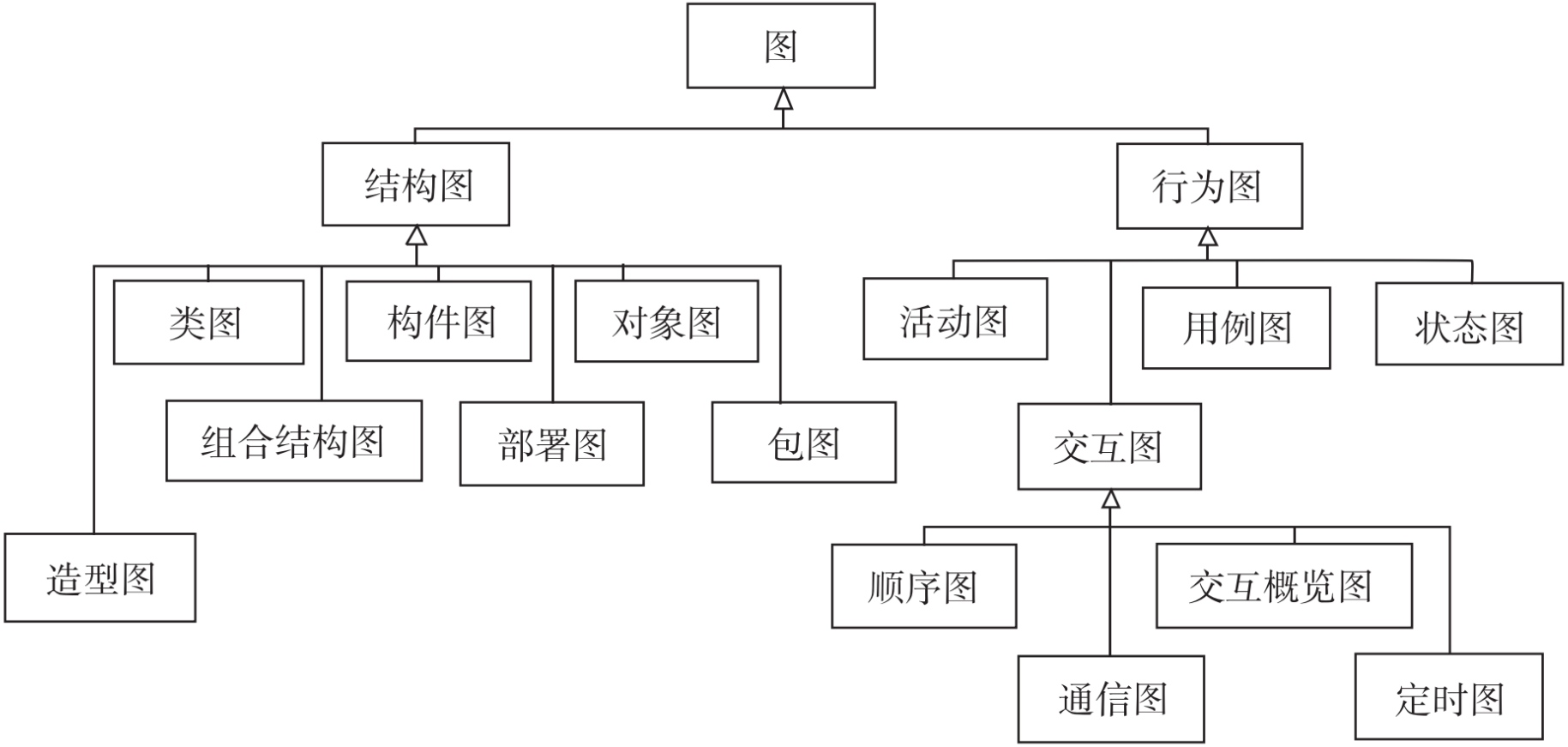
视点	图（diagram）	说明
结构	包图（package diagram）	从包层面描述系统的静态结构
	类图（class diagram）	从类层面描述系统的静态结构
	对象图（object diagram）	从对象层面描述系统的静态结构
	构件图(component diagram)	描述系统中构件及其依赖关系
行为	状态图(statechart diagram)	描述状态的变迁
	活动图(activity diagram)	描述系统活动的实施
	通信图(communication diagram)	描述对象间的消息传递与协作
	顺序图(sequence diagram)	描述对象间的消息传递与协作
部署	部署图（deployment diagram）	描述系统中工件在物理运行环境中的部署情况
用例	用例图（use case diagram）	从外部用户角度描述系统功能

统一建模语言 UML 建模阶段

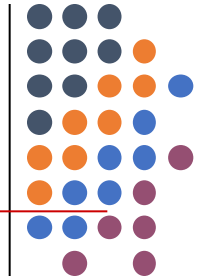


统一建模语言 UML

- UML图的划分

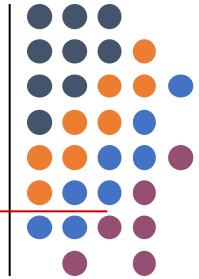


需求工程的CASE工具



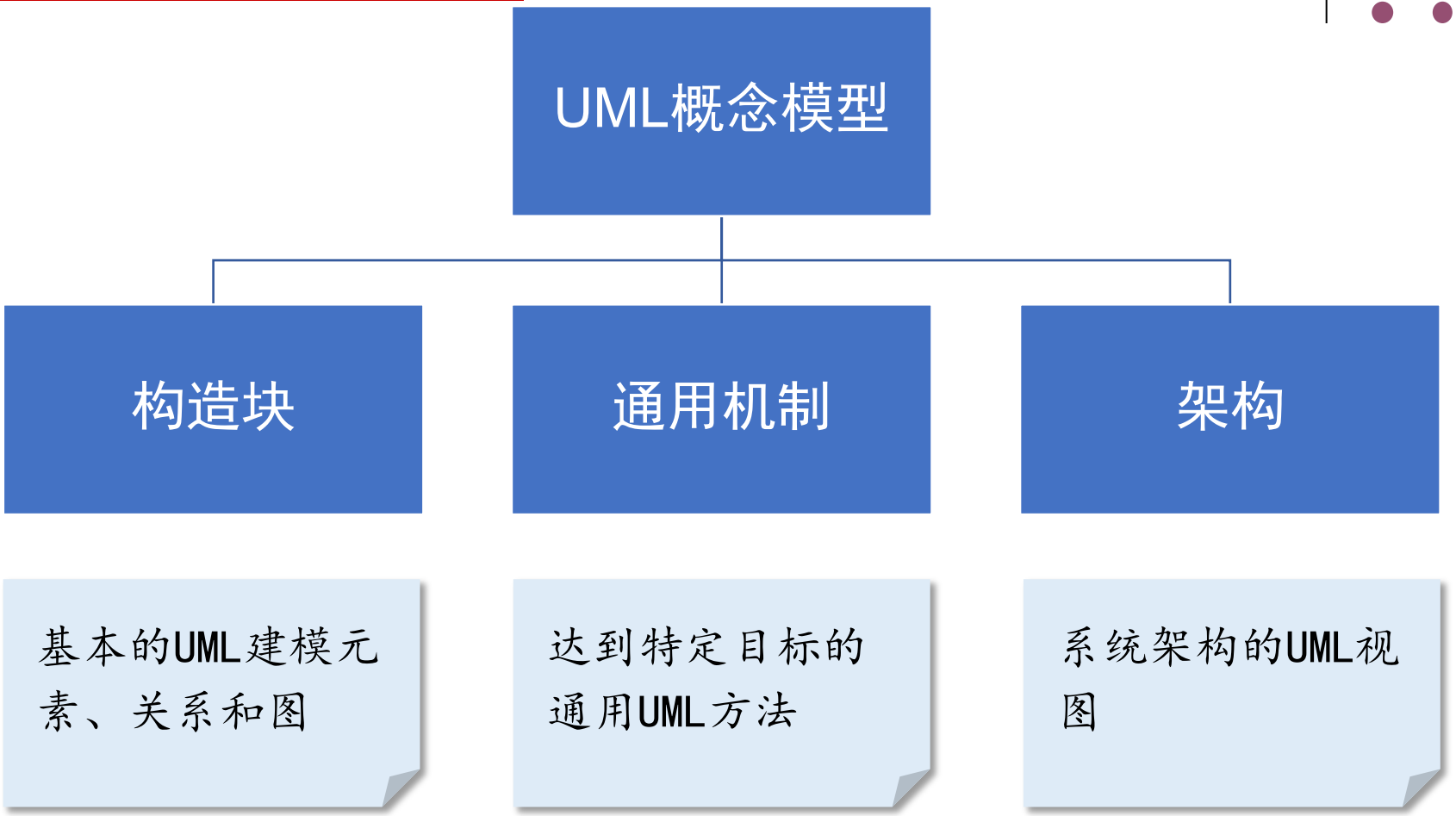
- 需求文档撰写工具，如借助于Microsoft Office、WPS
- 需求建模工具，如利用Microsoft Visio、Rational Rose、StarUML
- 软件原型开发工具，如Mockplus、Axure RP Pro、UIDesigner
- 需求分析和管理专用工具，如IBM Rational RequisitePro
- 配置管理工具和平台，如Git、Github、Gitlab、PVCS、Microsoft SourceSafe等

本学期课程设计采用的UML工具

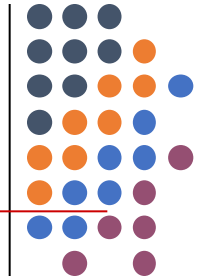


- 可根据个人偏好，自行选择可用的UML绘图工具
- 例
  - Enterprise Architect
  - StarUML
  - GitMind
  - ...

UML概念模型


UML概念模型
构造块
通用机制
架构
基本的UML建模元素、关系和图
达到特定目标的通用UML方法
系统架构的UML视图

UML的构造块



- 事物
  - 结构、分组、行为、注释
- 关系
  - 依赖、关联、泛化、实现
- 图
  - 静态
    - 类图、对象图、构件图、部署图、包图、组合结构图、外廓图
  - 动态
    - 顺序图、通信图、时间图、交互纵览图、活动图、状态机图、用例图