UML (Unified Modeling Language)

1 Course Outline

Software Institute of
Nanjing University

Instructor

- 刘嘉 （LiuJia）
  - Email：liujia@software.nju.edu.cn
  - ext：509
  - Office：705

Objectives

- 了解并掌握UML(2.0)语言关键语法及概念
  - Class diagram
  - Sequence diagram
  - Use case diagram
- 具有利用UML进行软件系统的面向对象分析设计(OOA,OOD)的初步经验基础

Grading

- Attendance 5%
- Two marked assignments 40%
- Final Exam 55%

Reference book

- [1]《 The Unified Modeling Language User Guide》
  - by Grady Booch, Jim Rumbaugh, Ivar Jacobson
  - published by Addison-Wesley

- [2] 《The Unified Modeling Language Reference Manual》
  - by Grady Booch, Jim Rumbaugh, Ivar Jacobson
  - published by Addison-Wesley

Other Reference

- <UML Distilled > Martin Fowler
- <The Rational Unified Process : An Introduction > Philippe Kruchten
- <UML2 Toolkit> John.Wiley

- www.omg.org OMG UML Specification
- www.umlchina.com UML in China

软件开发建模

- 为什么要建模
  - 软件是产品而非“程序”
  - 对它的要求和所有其他工业产品是一样的
    - 使用者和制造者分离
    - 质量要求、文档、维护
  - 软件产品的生产和其他工业产品的生产也是一样
    - 生产：团队、工具的使用（Compiler,..）,技术复用
    - 如何满足？先设计，再生产！ =》 建模！

软件开发建模（cont’）

- 模型是什么
  - 简单的说：模型是对现实的简化
    - 模型提供了系统的蓝图。模型既可以包括详细的计划，也可以从很高的层次考虑系统的总体计划
- 建模的基本理由
  - 我们建模是为了能够更好地理解我们正在开发的系统

软件开发建模（cont’）

- 建模的基本原理
  - 选择创建什么模型对如何动手解决问题和如何形成解决方案有着意义深远的影响
  - 每一种模型可以在不同的精度级别上表示
  - 最好的模型是与现实相联系的
  - 单个模型是不充分的，对每个重要的系统最好用一组几乎独立的模型去处理
- 更多...
  - The Value of Modeling

什么是UML?

- UML: Unified Modeling Language
- UML is a family of graphical notations, backed by single meta-model, that help in describing and designing software systems, particularly software systems built using the object-oriented (OO) style.

- UML是由OMG管理控制的一种较为开放的标准

UML的使用方法

- Steve Mellor & Martin Fowler 提出关于UML使用方式的三种分类：
  - 用作草图绘制语言 UML as sketch
  - 用作蓝图绘制语言 UML as blueprint
  - 用作程序编制语言（程序设计语言）UML as programming language

UML的使用方法（cont’）

- 模型驱动体系结构
  - Model Driven Architecture (MDA)
  - 平台无关模型（Platform Independent Model，PIM）
  - 平台特定的模型（Platform Specific Model，PSM）

*如果由**PIM**到**PSM**再到最终代码的过程完全自动化，**UML**便是编程语言。如果其中任何一步是由人手工进行的，**UML**便是蓝图绘制语言。*

UML的使用方法（cont’）

- Steve Mellor：executable UML
  - 类似MDA的PIM等价的平台无关模型开始
  - 利用模型编译程序，把UML模型一步转化为可部署的系统
  - 模型的编译程序基于可复用的archetype

UML发展史

- 90年代，有一定影响的OOAD方法有50多种
- 1993年，由Booch开始工作并创建了Rational公司
- 1994年，OMT的重要元老Rumbaugh加入Rational公司
- 1995年10月，第一个版本，Unified Method 0.8
- 1995年Jacobson加入Rational公司
- 1996年6月，发布UML0.9
- 1997年，多家软件公司组成的UML联合组织成立,并把UML1.0提交到OMG
- 1997年11月4日，OMG发布UML1.1

UML发展史（cont’）

- 2003年6月2.0版本通过审议
- 2004年10月2.0版本稳定

图示法与元模型

- UML定义了一种图示法和一种元模型。
  - 图示法（notation）是你在模型中看到的图示材料；它是建模语言的图示语法。
    - 例如，类图图示法定义了如何表示类，关联以及重数等项目和概念。
  - UML的定义者定义元模型（meta model）是一种定义语言概念的图。
    - 元模型对于UML的用法以及编程语言都很重要。

UML图

- The superstructure defines the six structure diagrams, three behavior diagrams, four interaction diagrams,
- Structure Diagrams：
  - 类图(Class) *
  - 构件图(Component)
  - 复合结构图(Composite Structure) UML2中的新图
  - 部署图(Deployment)
  - 对象图(Object)
  - 包图(Package) UML1中非正式图

UML图（cont’）

- Behavior Diagrams：
  - 用例图(Use Case) *
  - 活动图(Activity) *
  - 状态机图(State machine) *
- Interaction Diagrams：
  - 顺序图(Sequence) *
  - 通讯图(Communication) UML1中的协作图
  - 交互概观图(Interactive overview) UML2中的新图
  - 定时图(timing) UML2中的新图

UML图例—用例图

- 用例图
  - 描述Actor与用例，以及用例与用例之间的关系
- 用例
  - 描述外部实体可见的动作，实现一个具体、完整的用户目标

USE-CASE DIAGRAM

Shows the system's use cases and which actors interact with them

Actor, use case, and association

UML图例（cont’）—类图

- 类图是UML中OO模型的基础，它描述系统的静态结构
  - 对象的类型、属性与操作
  - 类之间的静态关系：关联、子类型、
  - 聚集和构成、依赖等
  - 作用于上述关系之上的约束

UML图例（cont'）—类图

CLASS DIAGRAM Shows the existence of classes and their relationships in the logical view of a system

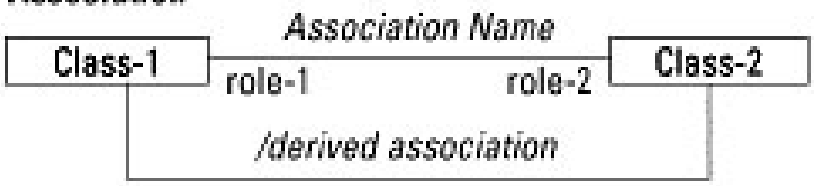
Class

Class Name

Class Name
attribute attribute : data_type attribute : data_type = init_value ...
operation operation (arg_list) : result_type ...

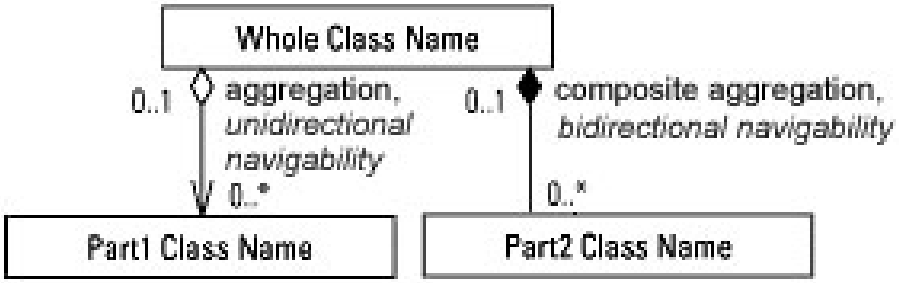
Role names and derived associations

Association

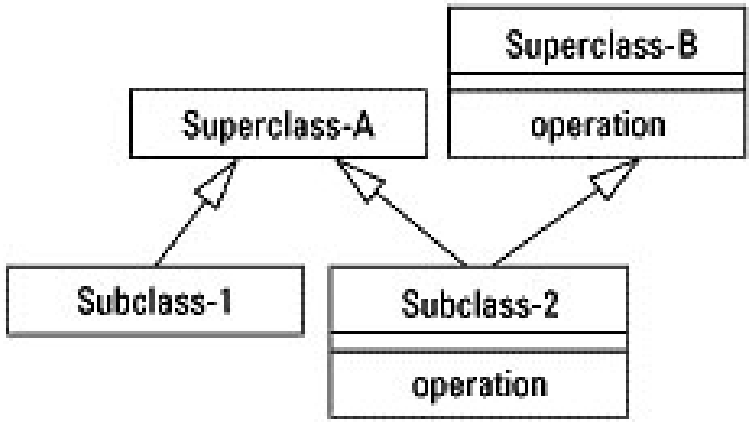


UML图例（cont'）－类图

Aggregation, navigability, and multiplicity

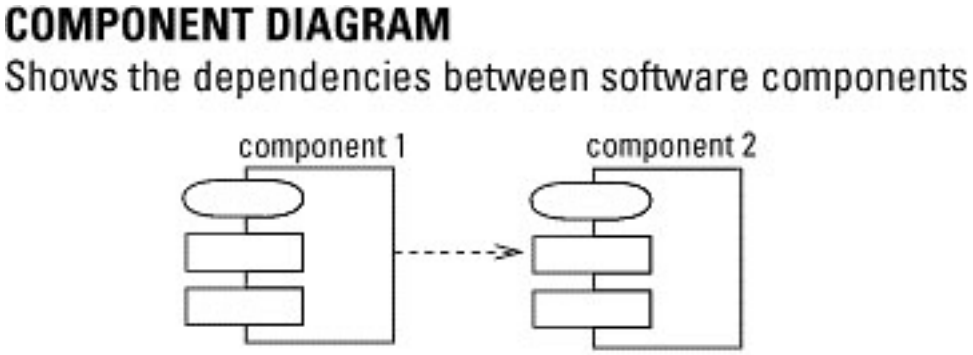


Generalization/specialization



UML图例（cont’）—构件图

- 构件图是实现层的软件结构
  - 构件：软件项目中的具有相对独立意义的单元（如源文件，二进制代码文件, DLL等）
  - 构件图用来显示编译、链接或执行时构件之间的依赖关系



UML图例（cont’）—部署图

- 配置图描述系统硬件的物理拓扑结构和在此结构上执行的构件

DEPLOYMENT DIAGRAM

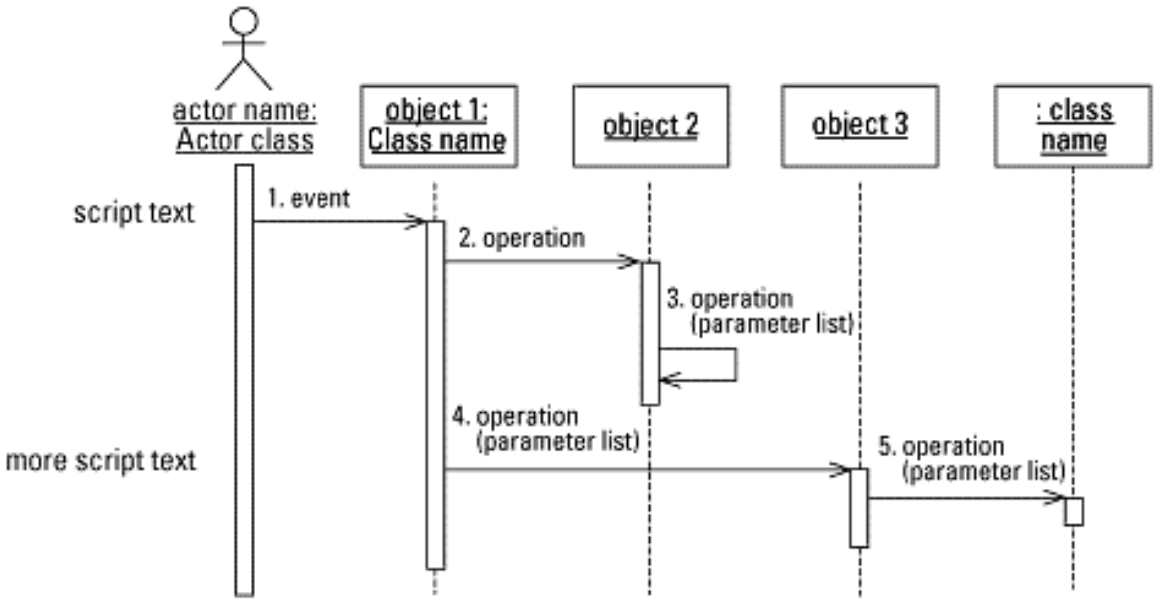
Shows the configuration of runtime processing elements

UML图例（cont’）—顺序图

- 顺序图用来描述对象之间动态的交互关系，主要强调对象间消息传递的时间序及并发

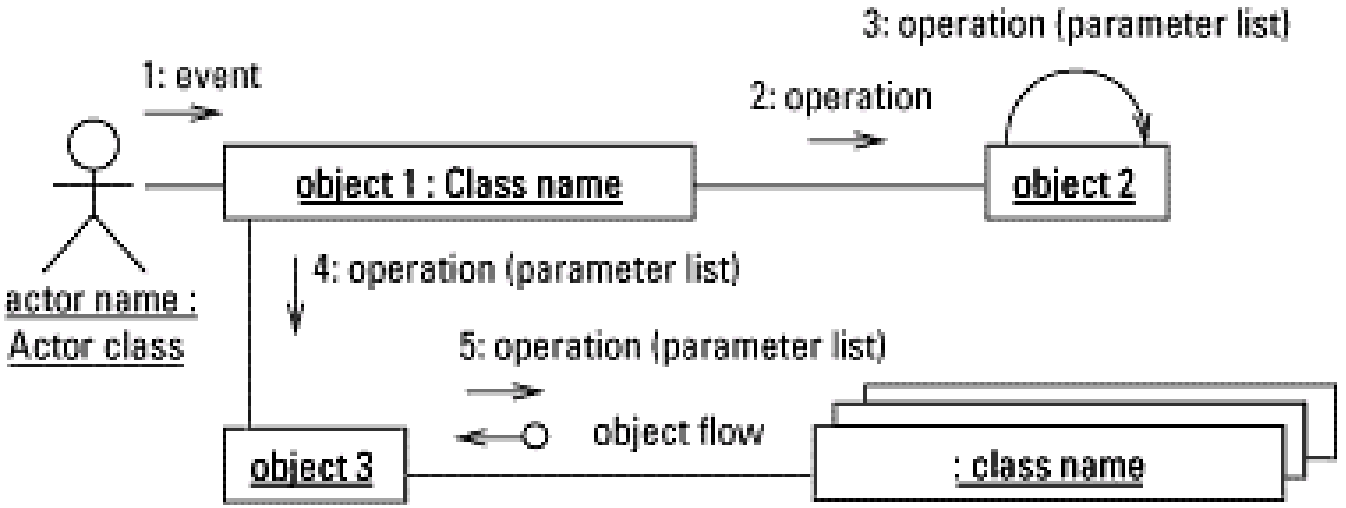
INTERACTION DIAGRAMS Show objects in the system and how they interact

Sequence diagram

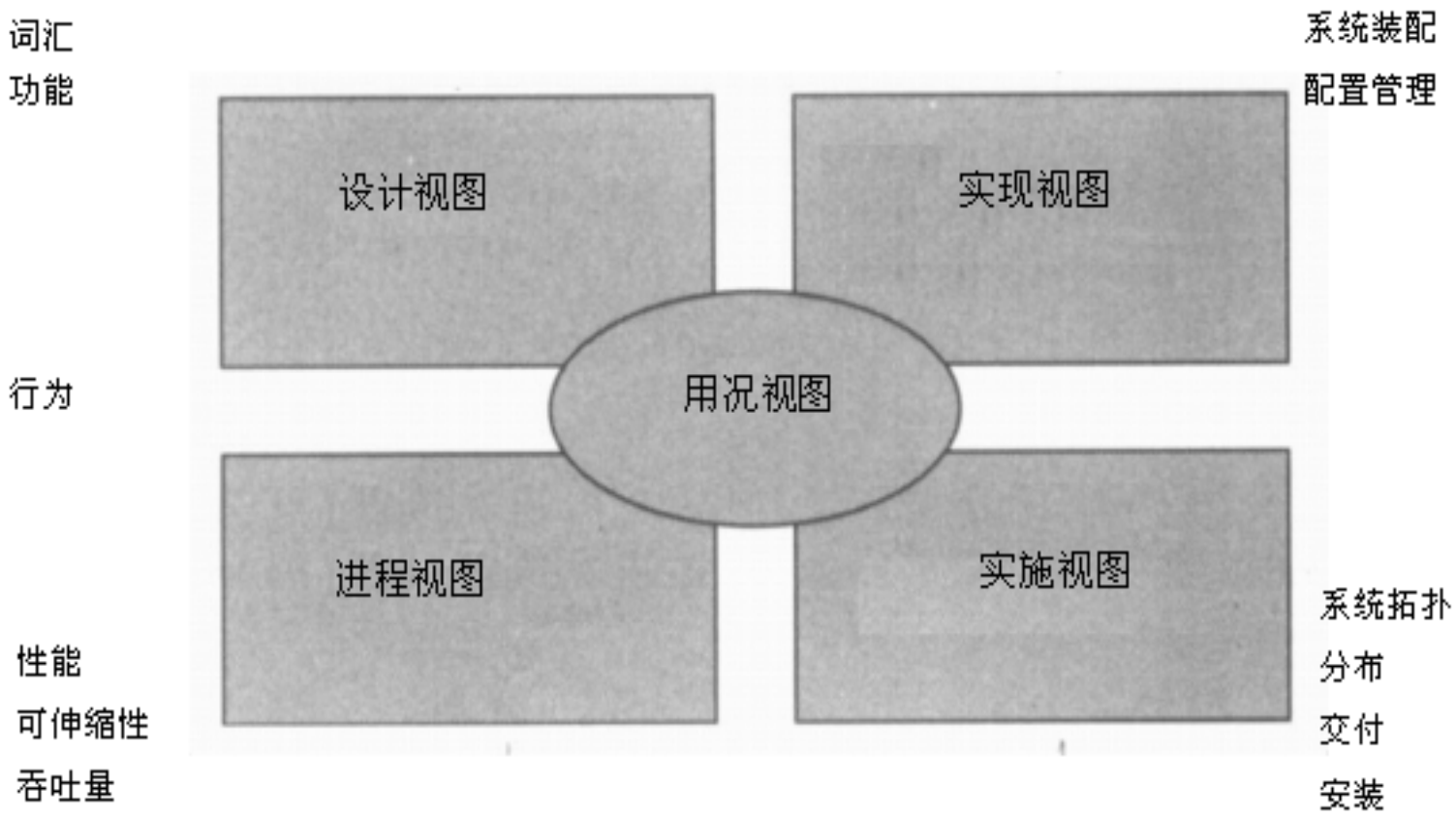


UML图例—通讯图

- 通讯图描述相互合作的对象间的交互和链接关系



UML体系结构视图


词汇
功能
行为
性能
可伸缩性
吞吐量
设计视图
实现视图
用况视图
进程视图
实施视图
系统装配
配置管理
系统拓扑
分布
交付
安装

UML视图

- Use Case View
  - 描述系统行为，用户和系统的交互
  - 面向最终用户，分析员和测试人员
  - 静态：Use Case图
  - 动态：交互图，状态图和活动图
- Logic View
  - 系统的面向对象模型
  - 面向最终用户，分析设计人员
  - 静态：类图，对象图
  - 动态：交互图，状态图，活动图

UML视图

- Process View
  - 描述系统的并发和同步机制，包括进程、线程的组织
  - 面向集成人员
  - 静态和动态图同逻辑视图，但是关注的是表示进程和线程的主动类
- Implementation View
  - 描述用来发布实际系统的文件和软件部件，关注配置管理和系统组装
  - 面向编程人员
  - 静态：构件图　　动态：交互图，状态图，活动图

UML视图

- Deployment View
  - 描述硬件拓扑结构和分布。
  - 面向系统工程师
  - 静态：部署图
  - 动态部分同Process View

什么是合法UML？（Legal UML）

- 合法UML就是在规范中定义为组成良好的UML
- 任何一种语言的约束分为两种：
  - 指定性规则(prescriptive rule)
  - 描述性规则(descriptive rule)
- UML是一种精确的语言，而且相当复杂。本课程中，会提到习惯用法以及标准和规范之间的一些差异

宁愿有一个用不合法的UML书写的好的设计，而不愿意有一个用合法的UML书写的蹩脚的设计

UML是富有表现力的语言

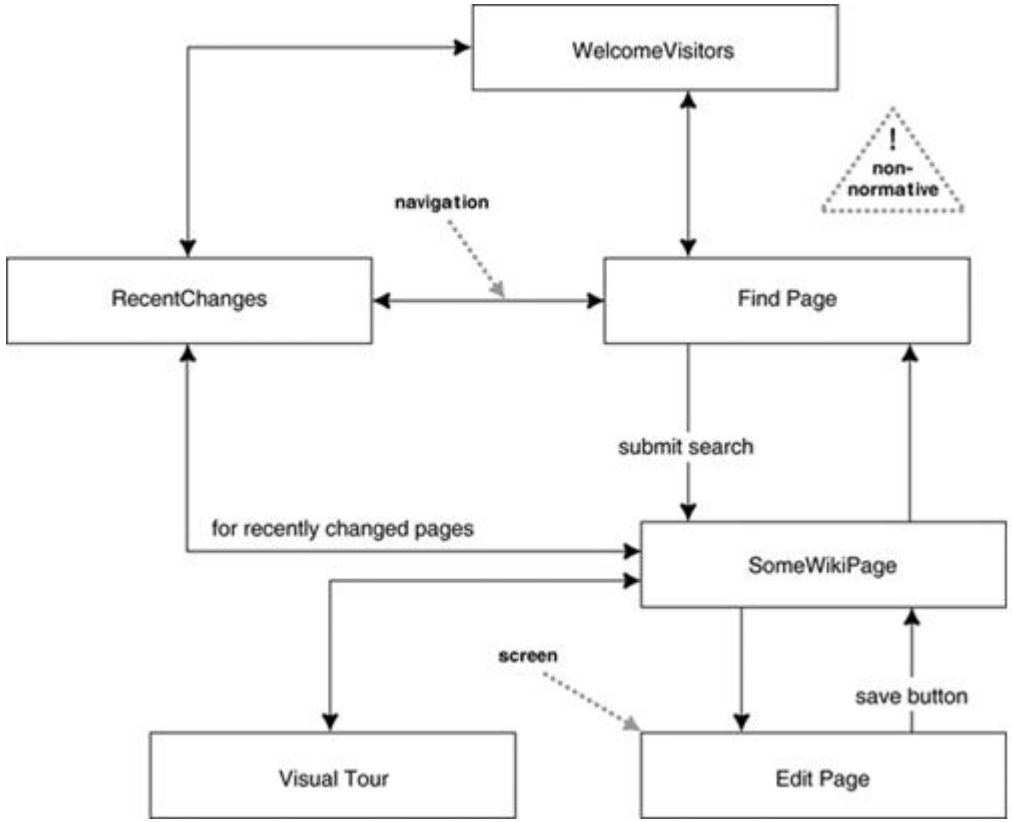
- 从并发系统到基于Web的分布式应用，从企业信息系统甚至到严格的实时嵌入式系统都适合用UML来建模
- UML是十分庞大的语言，提供了多种模型和足够的扩展性以对各种复杂系统建模

UML并非足够

- 虽然UML提供了有助于定义各种应用的各种图，但这决不完备。在很多场合，别的图也有可能有用。

*如果没有适合你的目标的**UML**图，你就应毫不犹豫地去使用那些非**UML**的图。*

UML并非足够（cont’）



开发过程

- UML源于一组面向对象分析和设计方法
- 如果不了解建模技术如何适应过程，建模技术也就不会有什么意义，使用UML的方式很大程度上取决于使用的过程的风格
- UML可以用于各种过程

迭代过程与瀑布过程

- 迭代过程与瀑布过程之间的本质区别是如何将一个项目分解成小块
  - 瀑布风格是基于活动来分解项目的
  - 迭代风格根据功能子集来分解项目
- 也有混合性的方法
  - McConnel描述了阶段提交（staged delivery）生命周期
- 使用UML草图绘制和蓝图绘制

迭代过程与瀑布过程（cont’）

- 迭代的常用技术
  - 时间框定法（time boxing）
  - 重做（rework）
    - 自动回归测试（automated regression test）

      http://junit.org
      http://www.xprogramming.com/software.htm
    - 结构改组（refactoring）

      http://www.refactoring.com
    - 连续集成（continuous integration）

预见性计划制定与适应性计划制定

- 人们软件开发中对预见性的渴望，制定预见性计划（predictive planning）
- 核心是需求分析，软件项目复杂性的根源就是了解软件系统需求的困难。大多数项目都会经历需求折腾（requirement churn）
  - 更多致力于需求过程本身，得到更为精确的需求，从而会使折腾减少
  - 需求折腾是不可避免的，主张适应性计划制定（adaptive planning）强调变动是经常的，控制变化，是项目能够提交最好的软件，虽然项目是可控制的，但是是不可预见的。

敏捷过程

- 覆盖了很多过程，这些过程共享由《敏捷软件开发宣言》（http://agile manifesto.org）所规定的一组价值和原则
  - Extreme Programming (XP), Scrum,
  - Feature Driven Development (FDD), Crystal,
  - DSDM (Dynamic Systems Development Method).
- 大多数将UML用于草图绘制

RUP Rational统一过程

- RUP是一个过程框架，需要选择一个开发案例（development case）
- RUP本质上是一个迭代过程
  - 初始（inception）
  - 细化（elaboration）
  - 构作（construction）
  - 转换（transition）

过程适配项目

- 没有适用于任何项目的万能过程
- 进行迭代回顾（iteration retrospective）
  - Keep
  - Problem
  - Try
- 正式项目回顾（project retrospective）
  - http://www.retrospectives.com

UML适配过程—需求分析

- 用例图
- 从概念视图绘制的类图
- 活动图
- 状态图

UML适配过程—设计

- 类图
- 常用用例的顺序图
- 包图
- 具有复杂生命周期的类的状态图
- 部署图

UML适配过程—文档

- UML图可以使得对系统有一个全局的了解
  - 草图突出系统最为重要的部分
  - 包图和部署图也是很好的全局图
- 详细文档应该根据代码生成

更多的资源...

- http://www.omg.org

  UML 2.0 Superstructure Specification
  OMGUML04-10-02.pdf