UML Modeling

03.29.2004

冰云
icecloud@sina.com

Agenda

- 介绍Modeling：为什么要建模？
- UML的发展：
- UML Diagrams：12种UML图
- 面向对象的分析与设计（OOAD）
- Together建模实例
- 敏捷方法 ： Agile Alliance
- 设计模式 ： Design Patterns，内功
- UML2.0与MDA

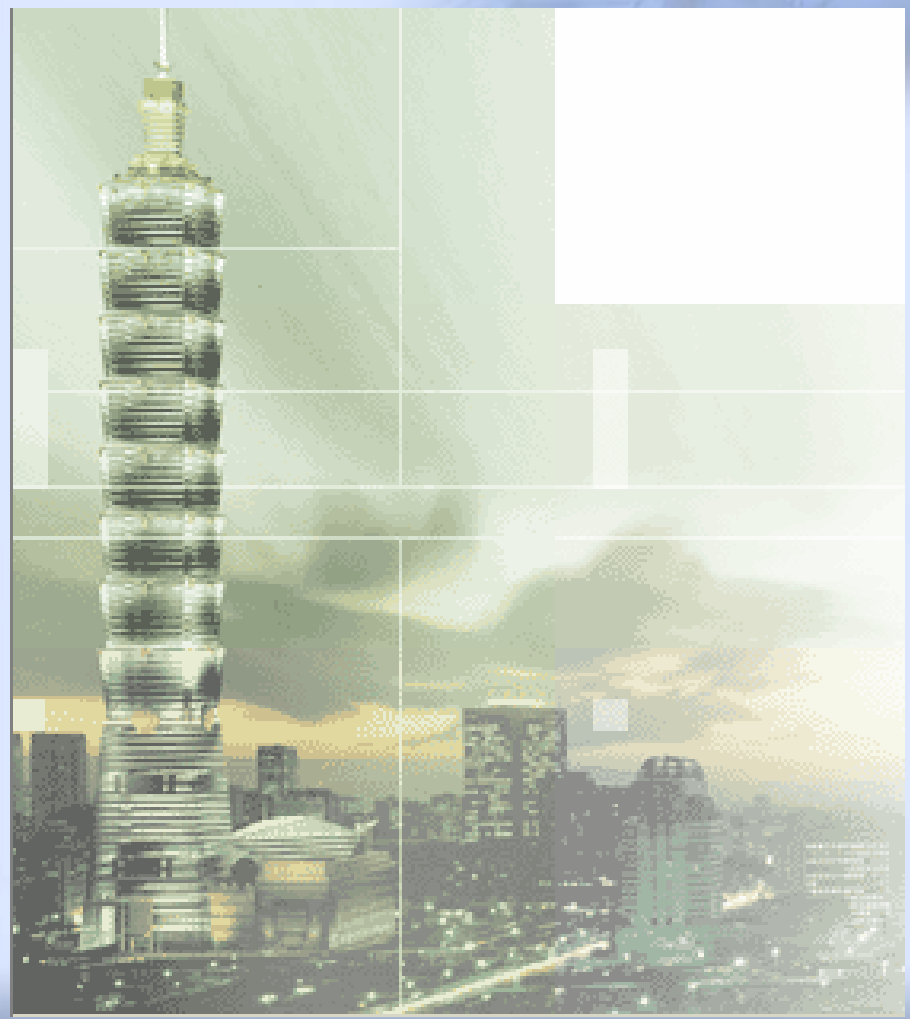
What's Modeling?

- 抽象
  - 问题的多个方面或可能的解决问题方案
- 设计蓝图：
  - Blueprints
- 交流工具
  - 与客户交流的手段
- 文档
  - 软件开发的必要文档

Why Modeling?

- 软件开发过程
  - 简单 vs 复杂
  - 个人 vs 团队
  - 非正式 vs 正式

< === >



Why Modeling?

- 与客户沟通业务需求
  - 需求建模：Requirement Model
- 理解一个复杂问题
  - Business Model
- 交流团队正在做的或已经做完的事情
  - Analysis Model – Platform Independent Model
  - Design Model – Blueprint for implementation

Features of Good Model

- 好的模型的特征
  - 满足创建者目的：交流还是理解
  - 易于理解：读者？客户还是程序员
  - 足够精确：过期的地图
  - 足够一致性：和制品一致
  - 足够详细：地图上的房子？
  - 积极的价值：不能过度建模
  - 简单：简单而有效

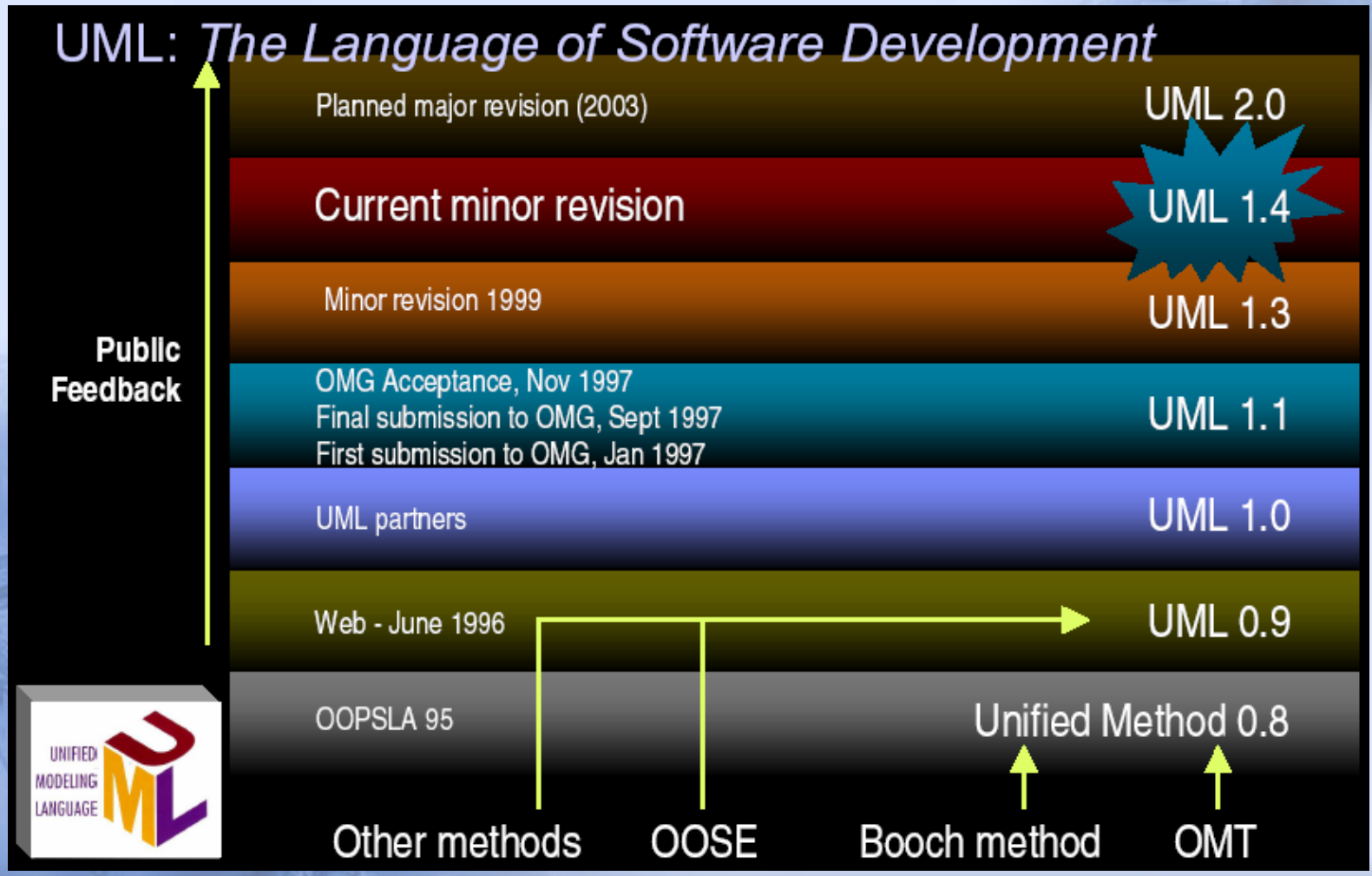
UML：统一建模语言

- UML: The Unified Modeling Language
  - The UML is the standard language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system
  - 可视化，说明，建造，文档
- 从实践发展而来
  - 理论与实践
  - OOAD：面向对象的软件工程

UML Methods?

- UML Three Amigos
  - Grady Booch：Booch Method，类似类图
  - Ivar Jacobson：OOSE等，用例
  - James Rumbaugh：OMT等
- UML没有提供方法论
  - 模型语义、图形表示法、使用规则
- Rational (IBM Rational)
  - Rational Unified Process

UML 发展过程


UML: The Language of Software Development
Planned major revision (2003)
UML 2.0
Current minor revision
UML 1.4
Minor revision 1999
UML 1.3
Public Feedback
OMG Acceptance, Nov 1997
Final submission to OMG, Sept 1997
First submission to OMG, Jan 1997
UML 1.1
UML partners
UML 1.0
Web - June 1996
UML 0.9
OOPSLA 95
Unified Method 0.8
UNIFIED MODELING LANGUAGE
Other methods
OOSE
Booch method
OMT

The Value of the UML

- 标准：UML is a standard
- 全部开发周期：Supports the entire software development lifecycle
- 不同领域应用：Supports diverse applications areas
- 基于经验和需要：Is based on experience and needs of the user community
- CASE工具多：Supported by many tools
  - CASE: Computer Aided Software Engineering

CASE Tools

- Borland Together 2002年10月
  - Together for .NET 1.0
  - Together for Eclipse 6.2
- IBM Rational Rose2002年12月
  - Rational Suite 2003
  - Rational XDE for .Net/Eclipse
- Microsoft Visio
  - Visio 2003
  - Whitehorse ?
- 其他中小公司

UML 12 Diagrams

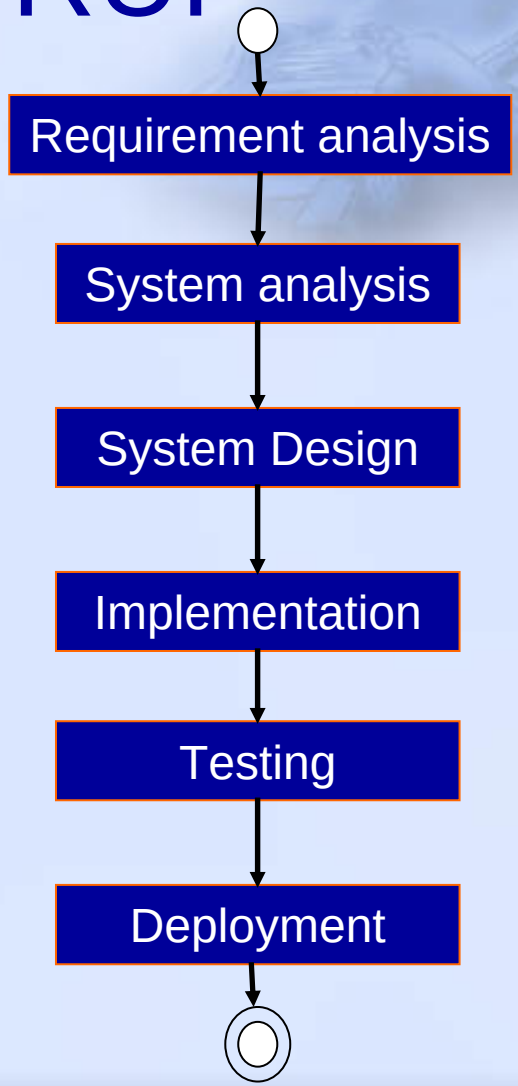
- Behavior :
  - Use Case ***
  - Activity **
  - Sequence ***
  - Collaboration *
  - State Chart **
- Structural:
  - Class ***
  - Component *
  - Deployment *
  - Object
- Model Management:
  - Packages (class diagram contains packages)
  - Subsystems (class diagram contains subsystems)
  - Models (class diagram contains models)

Diagrams

- 类图：
  - 描述一些类、包静态结构以及静态关系
- 组件图：
  - 描述可以部署的构件之间的关系
- 部署图
  - 描述系统的拓扑结构
- 用例图
  - 描述角色和用例及其关系，系统基本行为建模
- 活动图
  - 不同过程之间的动态接触。用例的具体化
- 状态图
  - 对象内部状态的变化和转移
- 顺序图
  - 交互图，描述不同对象之间消息传递
- 协作图
  - 交互图，描述收发信息的对象的组织结构

The Process - RUP

- Business Model
- Requirement Model
- User Experience Model
  - look-Feel & Interaction
- Analysis Model
- Design Model
  - Architecture design
  - Data Model
- Implementation Model
- Test Model
- Deployment Model



UML在软件开发周期

- Use Case
  - 需求获取
  - 定义需求模型
- Activity diagram
  - 需求分析
  - 业务过程分析
- Class diagram
  - 分析模型
- Subsystem，Package diagram，Class Diagram
  - 宏观设计，架构设计
- Sequence，Collaboration Diagrams，Class Diagram
  - 行为设计，设计模型
- State Chart
  - 对象设计
- Component Diagram
  - 实现模型
- Deployment Diagram
  - 部署

Use Case

- Use case 描述系统功能，是与客户沟通的工具
  - Use Cases: 系统行为或功能
  - Actors: 系统的各种使用者
- 组成部分
  - Use Cases Diagrams
  - Use-Case Reports or Use-Case Specifications

ATM系统

- 假设我们为客户制作一个ATM系统
- 客户这样说明他的系统：
  - 取款功能：用户用银行卡可以在提款机上取款。首先输入8位密码，如果三次输入错误，就退卡。如果正确则进入操作界面。可以取50元的整数倍的钱。用户取款后，询问是否要打印，如果不打印就退出卡。并提示操作结束。
- 最有价值的路径，涉众利益

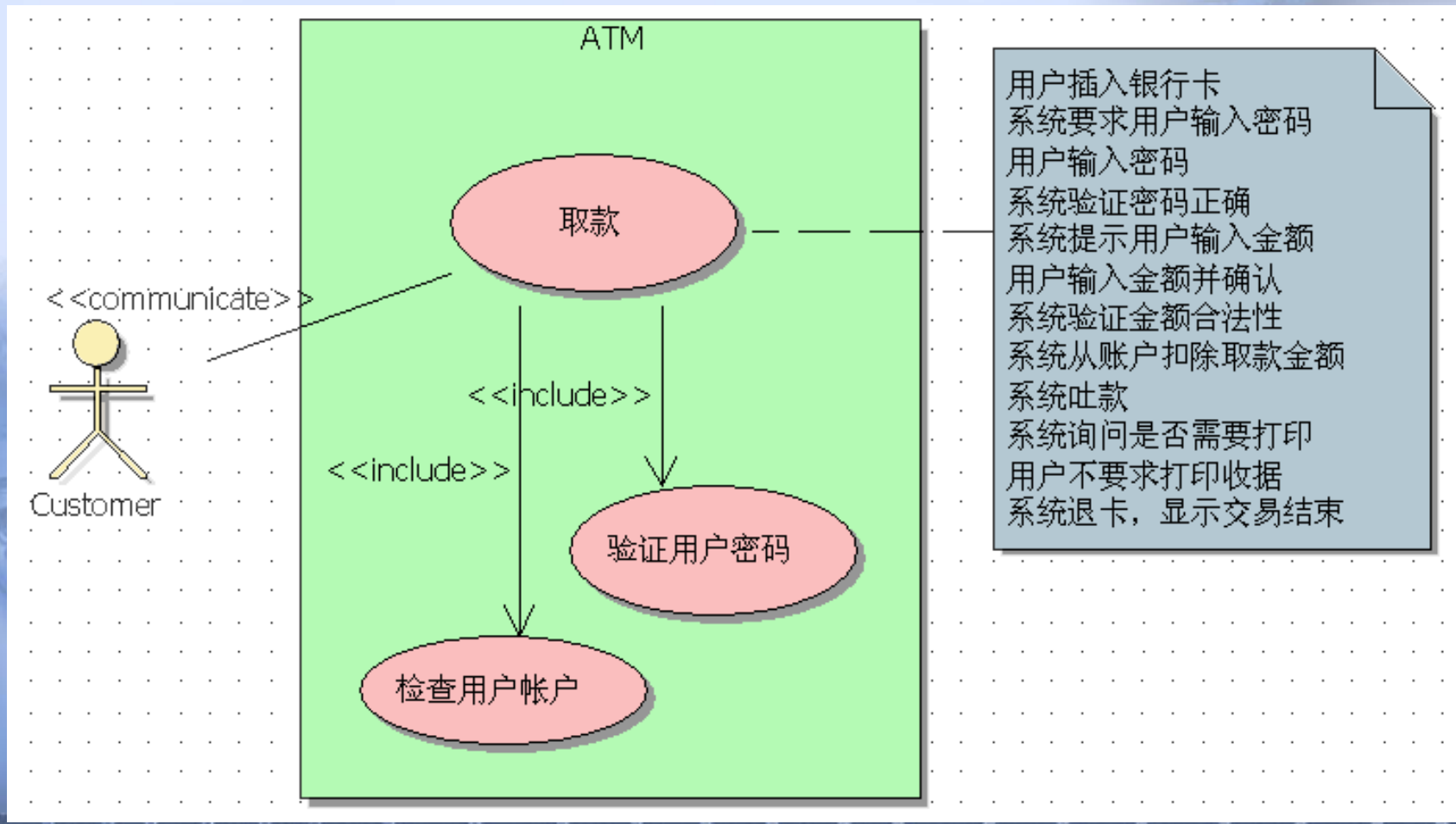
取款用例

- 立足用户视角建立用例
- 1. 场景:取款
- 2. 事件流：

  2.1 基本流：成功取款
  - 用户插入银行卡
  - 系统要求用户输入合法密码
  - 用户输入密码
  - 系统验证密码正确
  - 系统提示用户输入取款金额
  - 用户输入金额并确认
  - 系统验证金额合法性
  - 系统从账户扣除取款金额
  - 系统吐款
  - 系统询问是否需要打印
  - 用户不要求打印收据
  - 系统退卡，显示交易结束
- 3. 附加规约
  - 密码8位以上
  - 只能取50元整数倍

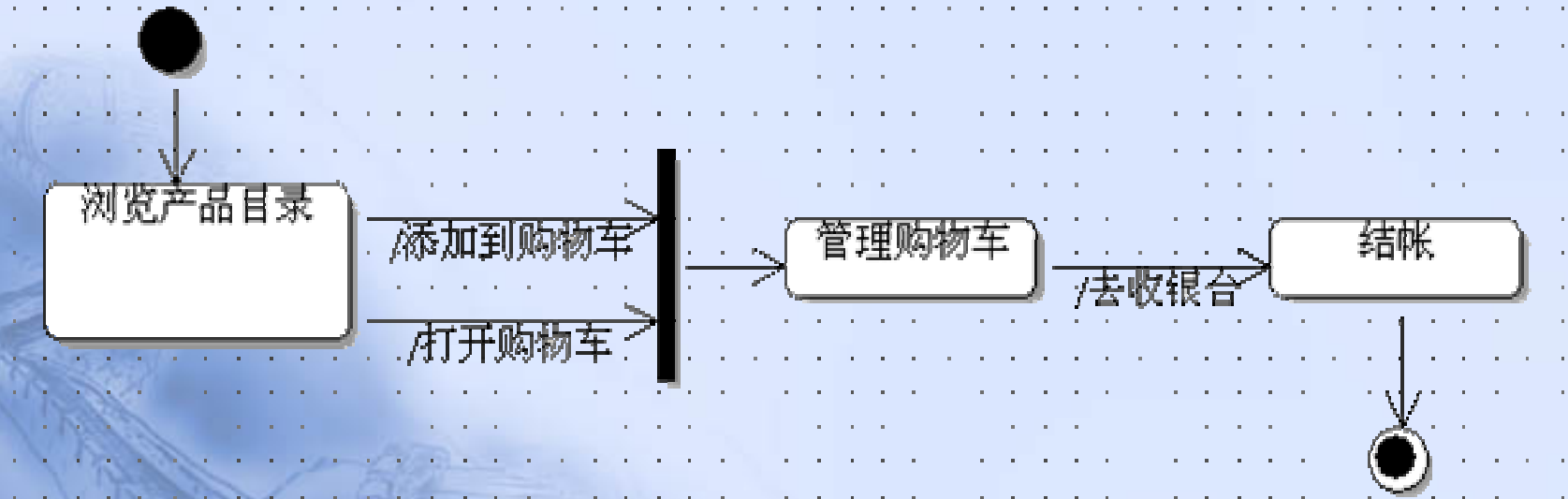
Use Case Diagram

- 用例说明是重点，用例图仅作为直观表述



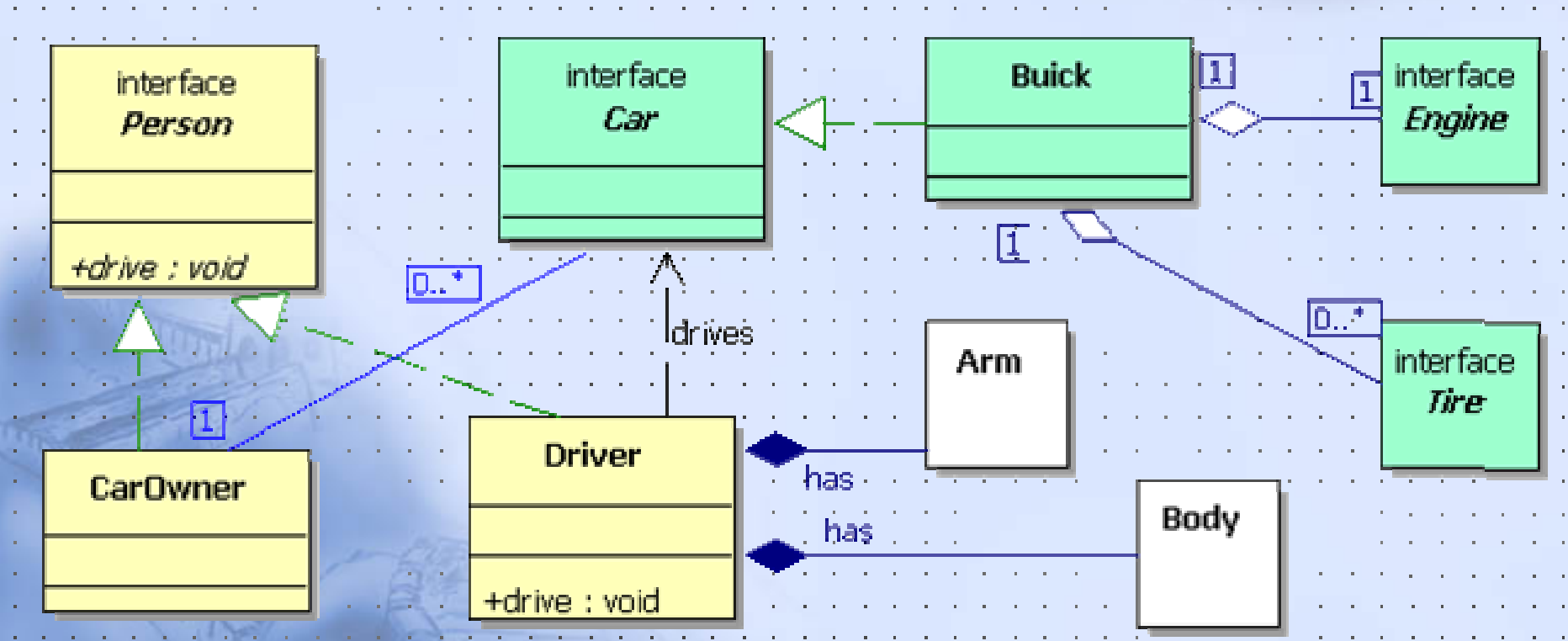
Activity Diagram

- 描述业务过程或Use Case的操作流程
  - 业务过程：多个用例之间的交互



Class Diagram

- 类图是最重要的UML图



类之间的关系

- 泛化（Generalization），继承关系
- 关联关系（Association）
  - CarOwner拥有Car，person.getCar(); 实例变量
- 聚合关系（Aggregation）
  - 强关联：Car由Engine和Tire组成
- 合成关系（Composition）
  - 强关联，负责生命周期。Person由Arm和Body
- 依赖关系（Dependency）
  - Person驾驶Car，局部参量

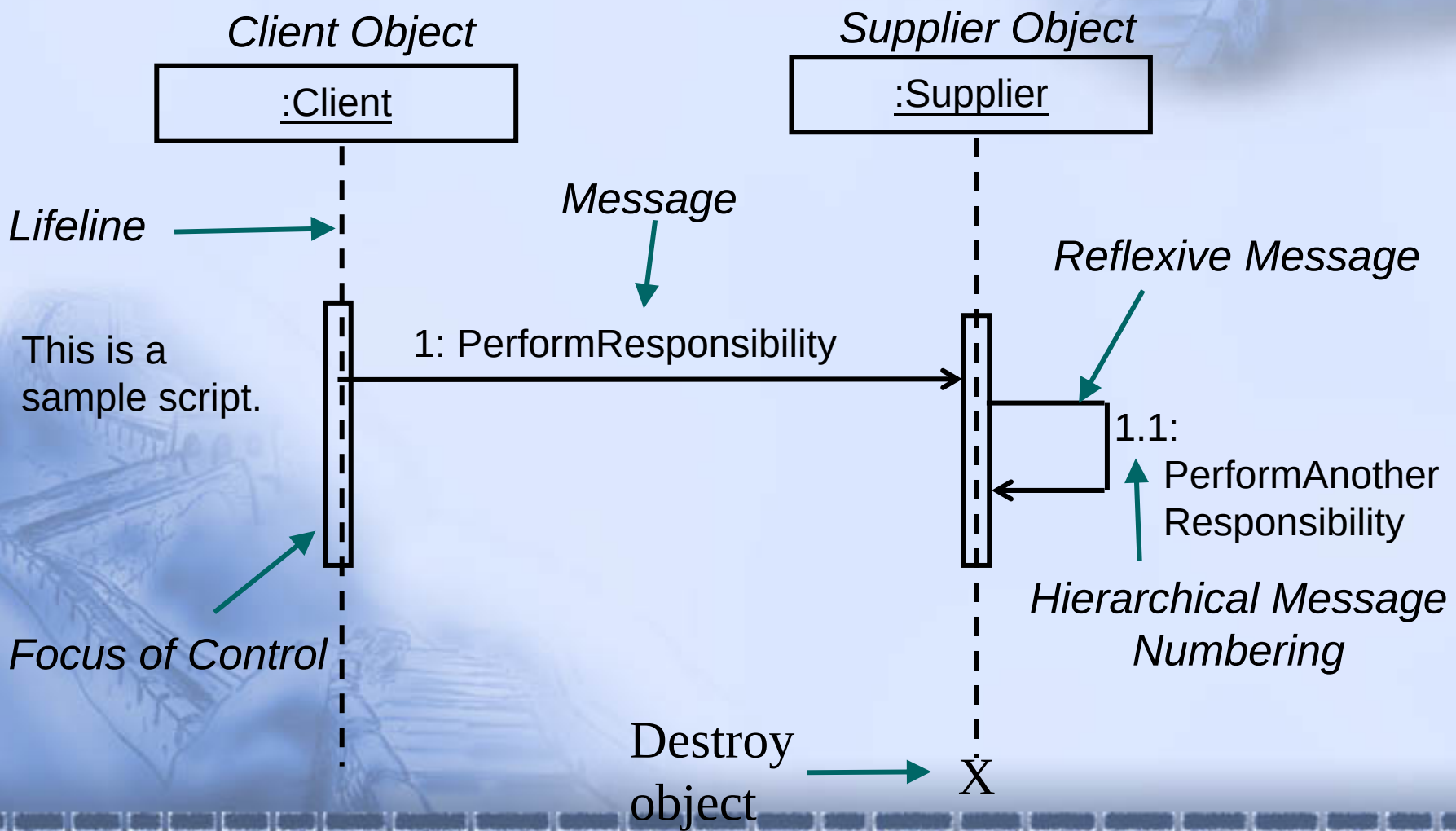
Modeling in Color with UML

- Peter Coad：
  - *Java Modeling in Color with UML*
    by Coad, Lefebvre, and De Luca
- How to Build Better Models and Have More Fun Doing So
- A **pink** moment-interval：变化的量，价格
- A **yellow** role：角色，人，参与者
- A **green** party-place-thing：事物，具体物品
- A **blue** description：描述，附加信息

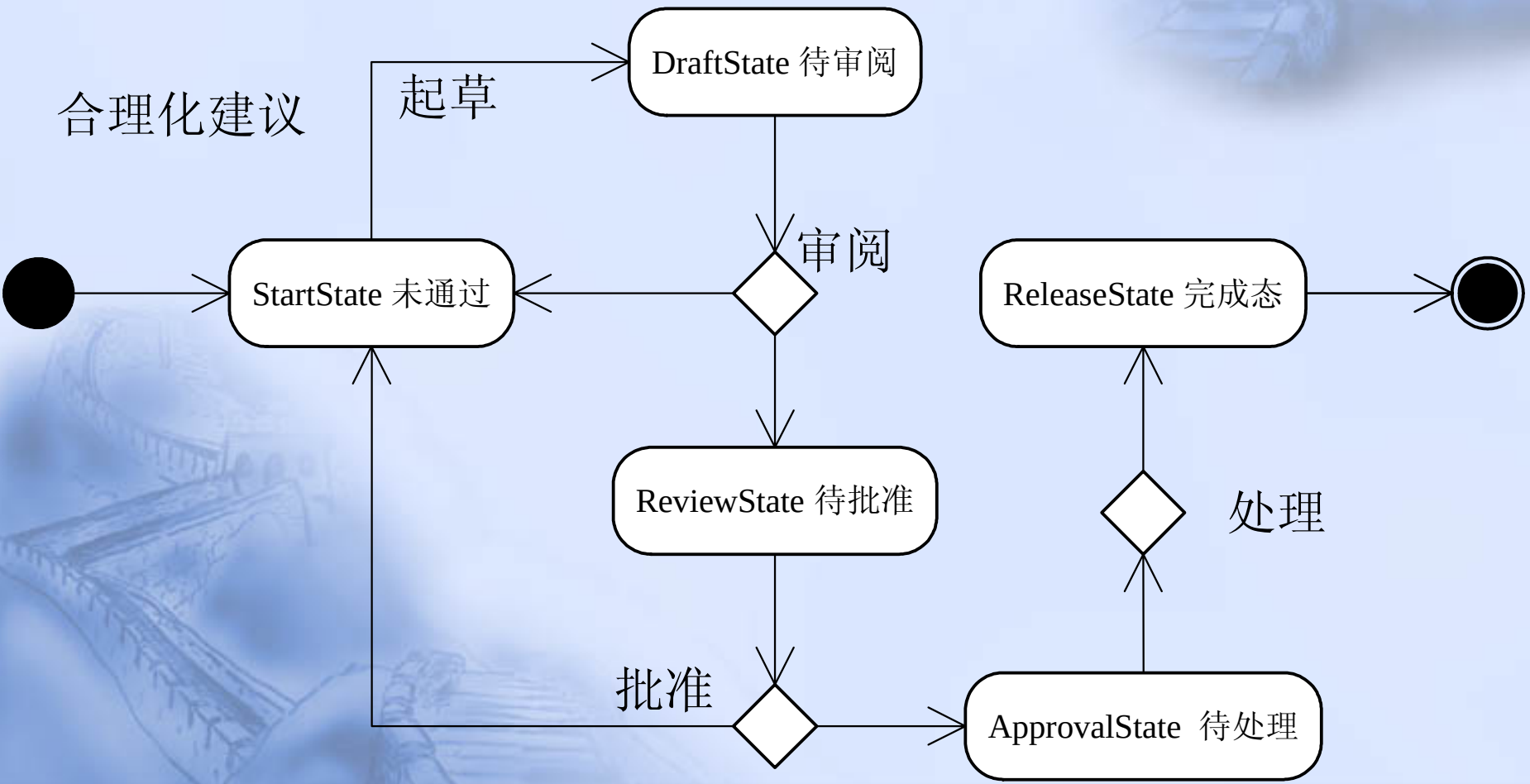
分析模型：实体类

- Platform Independent Model
- 找出完成use cases 所需要的类
  - Analysis classes
- 规划Classes的relationship
- 指派use-case behaviors给适当的类别
  - 找出Operations
- 找出每个class的属性

顺序图


Client Object
:Client
Supplier Object
:Supplier
Lifeline
Message
Reflexive Message
This is a sample script.
1: PerformResponsibility
1.1: PerformAnother Responsibility
Hierarchical Message Numbering
Focus of Control
Destroy object
X

状态图


DraftState 待审阅
起草
合理化建议
审阅
StartState 未通过
ReleaseState 完成态
ReviewState 待批准
处理
批准
ApprovalState 待处理

Agile Alliance

- 敏捷宣言，2001.2
  - 个体和交互　胜过　过程和工具
  - 可以工作的软件　胜过　面面俱到的文档
  - 客户合作　胜过　合同谈判
  - 响应变化　胜过　遵循计划
- 核心：
  - 强调人与人之间的合作因素
  - 以敏捷性应对变化

Extreme Programming

交流·简单·反馈·勇气

- 现场客户 On-site customer
- 用户故事 User story
- 计划游戏 Planning game
- 单元测试 Unit test
- 测试优先设计 TDD
  Test-first design
- 重构 Refactoring
- 验收测试 Acceptance test
- 结对编程
  pair programming
- 简单设计 Simple design
- 探索方案 Spike solution
- 集体所有
  Collective ownership
- 编码规范
  Coding standard
- 持续集成
  Continuous integration
- 隐喻 Metaphor
- 每周40小时
  Forty-hour week

敏捷过程

- CMM 的弊病：强调过程与计划
- 精益生产-消除浪费：
  - 部分完成的工作 – 库存
  - 额外过程
  - 额外特性 – 生产过剩
  - 任务调换 – 运输
  - 等待
  - 移动
  - 缺陷
- 制造业：
  - 泰勒-生产装配线
  - 顾客、竞争和变化 **3C**
- 敏捷方法
  - XP，Scrum，ASD，Crystal，FDD，DSDM
- 敏捷建模
- KISS：
  - Keep It Simple Stupid

设计原则

- 开闭原则：OCP
  - 对扩展要开放，对修改要封闭
- 里氏代换原则：LSP
  - 基类出现的地方，子类一定可以出现
- 依赖倒转原则：DIP
  - 要依赖于抽象，不要依赖于实现
- 单一职责原则：SRP
  - 类或方法职责要尽量少
- 接口隔离原则：ISP
  - 为客户端提供尽可能小的单独接口，而不是大总接口
- 组合复用原则：CARP
  - 尽量使用合成或聚合，而不是继承关系
- 迪米特法则：LOD
  - 软件实体尽可能少的与其他实体发生相互作用

源代码就是设计

- 软件工程师 类似于？ 建筑工程师
- 工程过程的结果是文档，而不是实物
- 一切都是设计过程的一部分，调试，测试
- 原因：廉价构建，编译器
- UML图不是真正的设计
- 测试—〉编码—〉重构—〉模式
  - 设计模式为重构提供了目标
- Clean code that works

设计模式

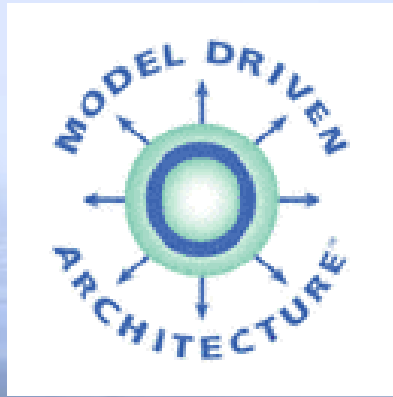
- 写程序的内功，程序员的交流语言
  - GoF 《设计模式》
  - Core J2EE Patterns - EBay
  - EJB Design Patterns
  - PEAA：Patterns of Enterprise Application Architecture – Martin Fowler
  - Business Modeling with UML
- 模式是描述特定场景（context）、特定约束（force）下描述以某种特定的方式解决特定问题（problem）之方案（solution）的专用语言。

实例：工厂方法

- 水果工厂　Fruit Factory
  - 生产Apple, Orange
  - 增加新产品？
- Simple Factory
  - 具体工厂—〉抽象产品
- Factory Method
  - 抽象工厂—〉抽象产品
- Abstract Factory
  - 抽象工厂 —〉抽象产品族

UML2.0与MDA

- MDA (Model Driven Architecture)
  - 史前：算盘
  - 石器时代：机器码
  - 铁器时代：源代码->机器码
  - 工业时代：模型->源代码
- 蓝图：
  模型->最终产品



- MDA基于UML2.0
  - Unified Modeling Language (UML);
  - Meta-Object Facility (MOF);
    - 杂志->文章->汉语->汉字
    - 人->染色体->基因->碱基
  - XML Meta-Data Interchange (XMI);
    - 用XML形式表现对象
  - Common Warehouse Meta-model (CWM).
    UML2.0

No Silver Bullet 没有银弹

- 狼人传说
  - 只有用银质子弹才能制服这些举止无常的怪兽。
  - “银弹“: 制服软件项目这头难缠的怪兽的法宝。
  - “没有一种单纯的技术或管理上的进步，能够独立地承诺在10年内大幅度地提高软件的生产率、可靠性和简洁性”
- 《人月神话 (The Mythical Man-Month)》
  - Frederick Brooks
- 没有银弹，只有实践

推荐书目

- Refactoring：Martin Fowler, Kent Beck
- Test Driven Development：Kent Beck
- Effective Java：
- Design Patterns：GoF，Eric Gamma
- UML Distiller 2nd： Martin Fowler
- Write Effective Use Cases：Alistair Cockburn
- Agile Software Development：Robert C Martin
- JAVA与模式：阎宏

The End

完

欲知后事如何，且听下回分解