Chapter 1

软件工程的要素：工具、方法、过程

三要素具体含义：工具：为软件工程方法提供自动化或半自动化的软件支撑环境和工具。

方法：为软件开发提供“如何做”的技术。

过程：软件过程是方法和工具综合起来以达到合理、高效地进行软件开发

的目的。

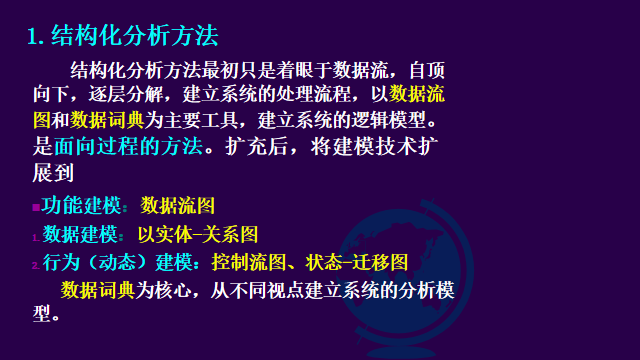
常用的方法有：

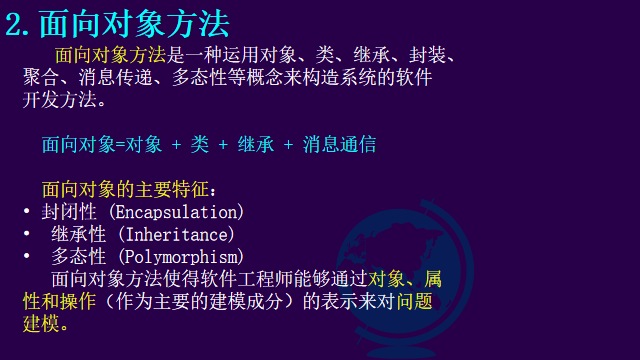
1.生命周期法（传统方法、结构化方法）

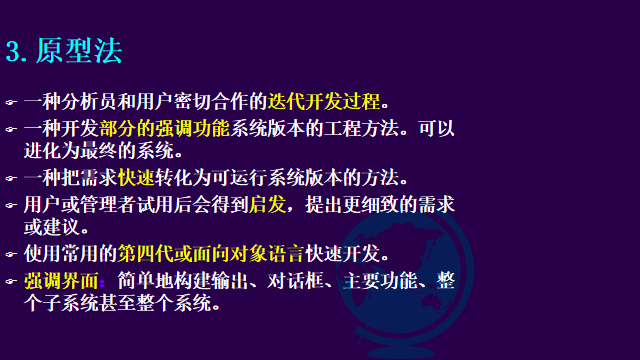
2.面向对象方法

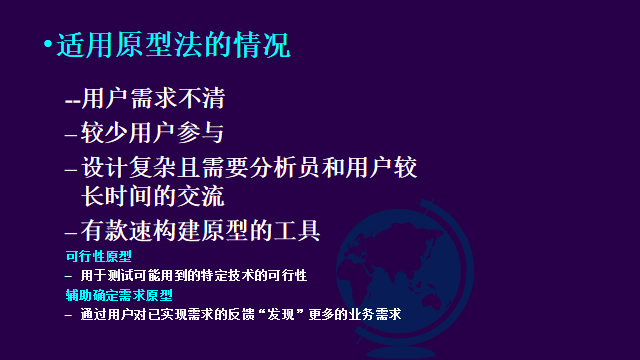
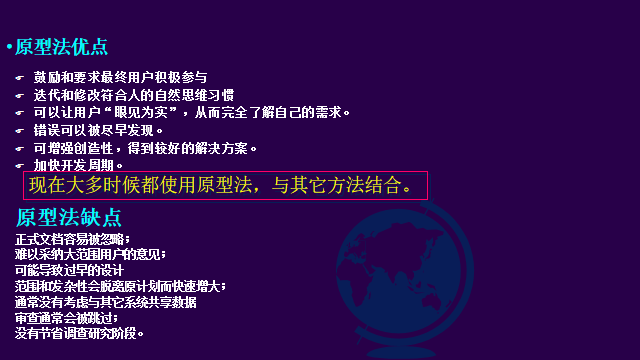
3.原型法

4.其它新方法或综合方法











**软件生命**  周期包含了软件从概念形成到最终退役的所有活动，而对于一个具体的软件项目，开发人员更加关注的是开发过程中包含的活动以及其具体安排。

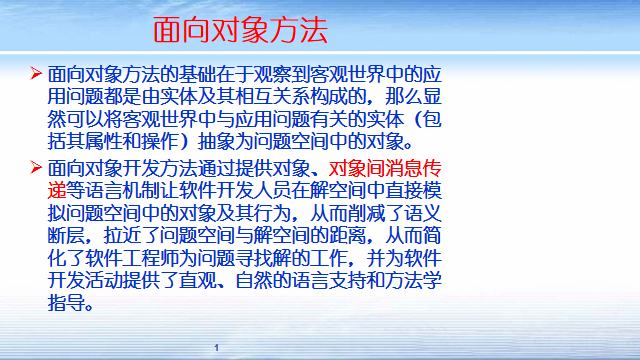
**软件体系结构**是软件系统的结构，包含软件元素、软件元素外部可见的属性以及这些软件元素之间的关系。

**Chapter 2**

UML（统一建模语言）是软件建模的表示法标准。

UML 2.0另一个显著特征就是加强了对模型驱动体系（MDA）的支持。

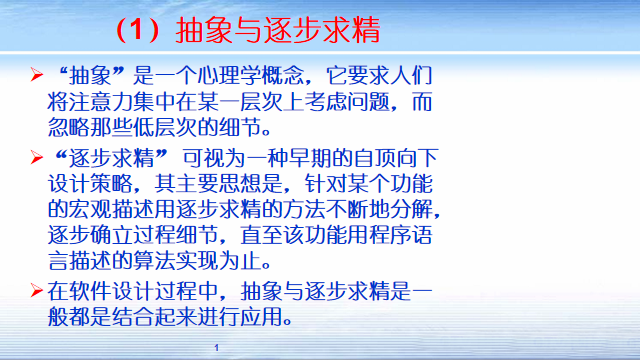
MDA 的目标是要实现从UML模型到最终代码的自动化生成，它将系统功能规范与该功能在某个特定平台上的实现规范分开，由同一个基础模型可以为不同的中间件平台产生应用程序。



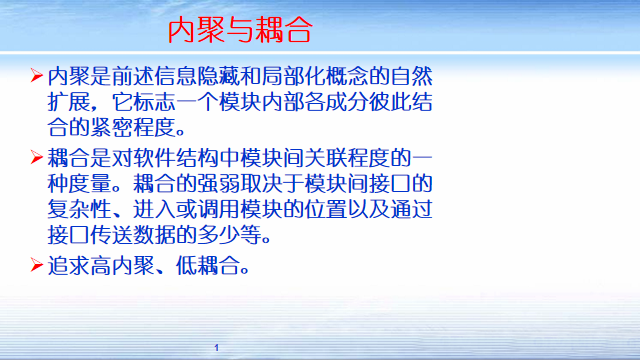
面向对象 = 对象 + 类 + 继承 + 聚集 + 多态 + 消息

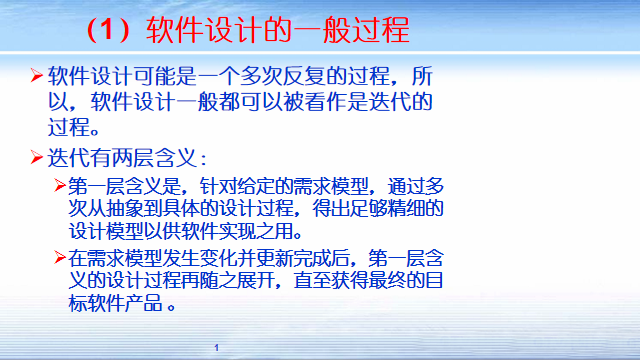
**Chapter 3**

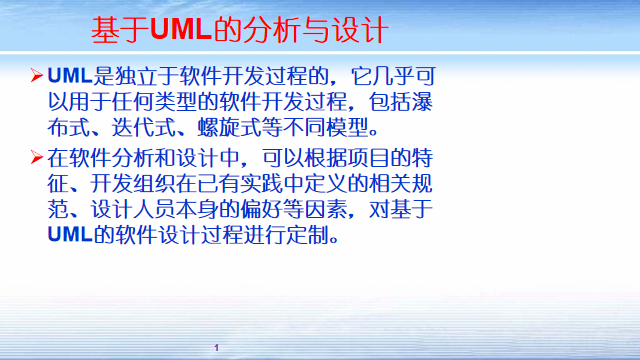
**软件设计 主要针对需求分析过程得到的软件需求规格说明**

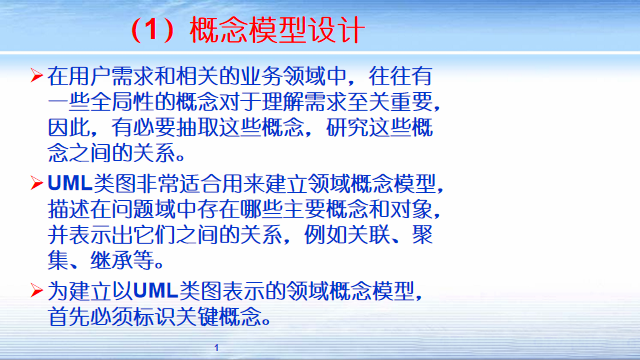


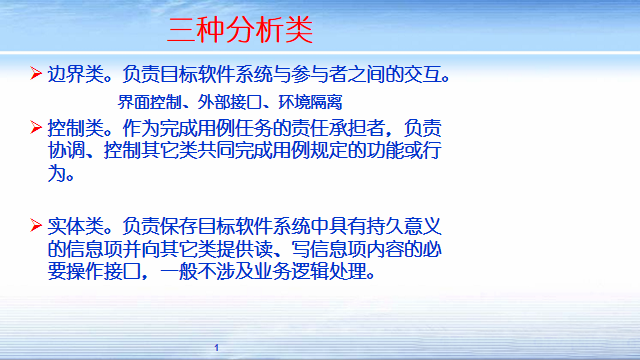
逐步求精 自底向上方法是先编写基础程序语段，然后再扩大、补充、升级。

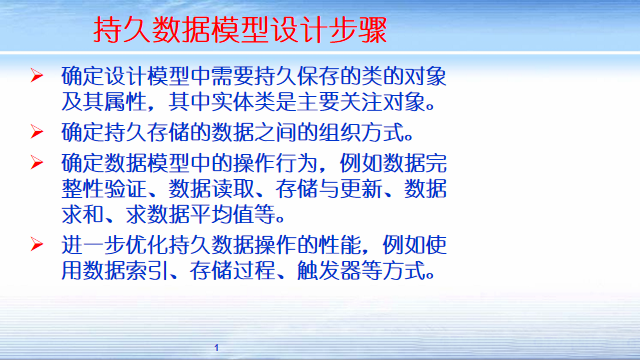












**面向对象设计原则有什么意义？**

是指导面向对象设计的基本指导思想

是评价面向对象设计的价值观体系

是设计模式的出发点和归宿

**面向对象的基本设计原则**

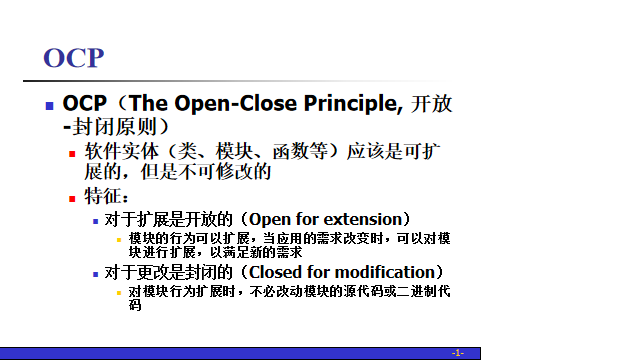
SRP：单一职责原则

OCP：开放-封闭原则，关键在于抽象

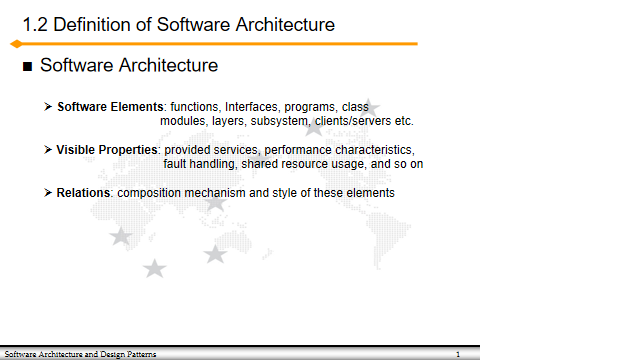
LSP：Liskov替换原则

ISP：接口隔离原则

DIP：依赖倒置原则







**4+1视图模型:逻辑视图、开发视图、过程视图、物理视图、场景视图**

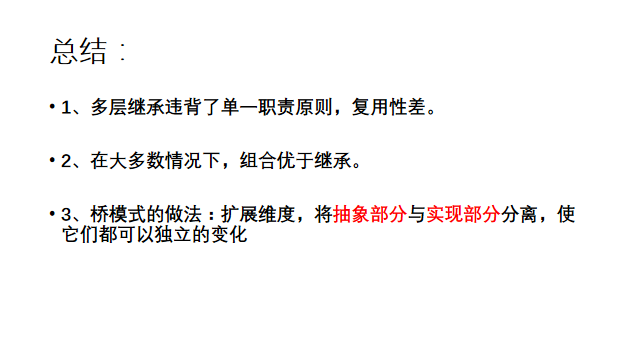
**五种模型： 结构、框架、动态、过程、功能**

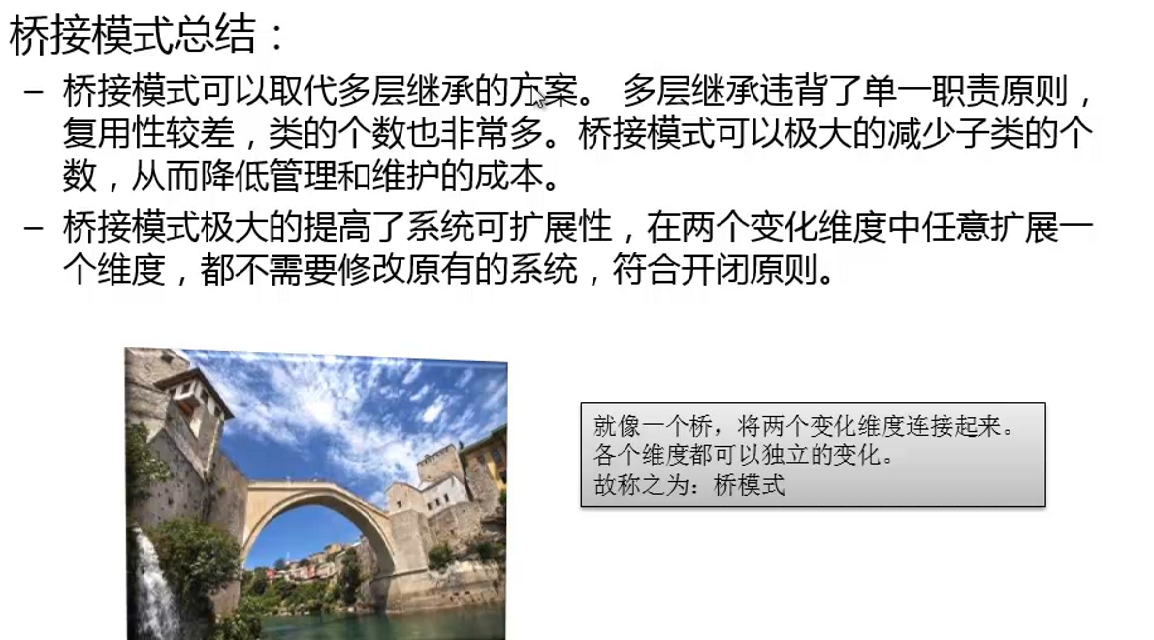
**设计模式**

**桥模式：**

**1.1 桥接模式核心要点：**

**处理多层继承结构，处理多维度变化的场景，将各个维度设计成独立的继承结构，使各个维度可以独立的扩展在抽象层建立关联。**





例子：

购买电子产品，可抽象为品牌、颜色、产品三种维度

**观察者模式：**

定义

观察者模式（又被称为发布-订阅（Publish/Subscribe）模式，属于行为型模式的一种，它定义了一种一对多的依赖关系，让多个观察者对象同时监听某一个主题对象。这个主题对象在状态变化时，会通知所有的观察者对象，使他们能够自动更新自己。

2. 意义

此设计模式最重要的作用就是 解耦！将观察者与被观察者解耦，使得他们之间的依赖性更小。

优点

解除耦合，让耦合的双方都依赖于抽象，从而使得各自的变换都不会影响另一边的变换。

缺点

在应用观察者模式时需要考虑一下开发效率和运行效率的问题，程序中包括一个被观察者、多个观察者，开发、调试等内容会比较复杂，而且在Java中消息的通知一般是顺序执行，那么一个观察者卡顿，会影响整体的执行效率，在这种情况下，一般会采用异步实现。

例子：

观察者模式这种发布-订阅的形式我们可以拿微信公众号来举例，假设微信用户就是观察者，微信公众号是被观察者，有多个的微信用户关注了程序猿这个公众号，当这个公众号更新时就会通知这些订阅的微信用户

组合模式：

定义：

将对象组合成树形结构以表示“部分-整体”的层次结构。Composite模式使得用户对单个对象和组合对象的使用具有一致性。

解决问题：

　　组合模式解决这样的问题，当我们的要处理的对象可以生成一颗树形结构，而我们要对树上的节点和叶子进行操作时，它能够提供一致的方式，而不用考虑它是节点还是叶子。

1.优点

　　（1）、简化客户端操作。客户端只需要面对一致的对象而不用考虑整体部分或者节点叶子的问题。

　　（2）、具有较强的扩展性。当我们要更改组合对象时，我们只需要调整内部的层次关系，客户端不用做出任何改动。

　　（3）、方便创建出复杂的层次结构。客户端不用理会组合里面的组成细节，容易添加节点或者叶子从而创建出复杂的树形结构。

　　2.缺点

1. 、要求较高的抽象性，如果节点和叶子有很多差异性的话，比如很多方法和属性都不一样，难以实现组合模式。

例子：

把学校作为根节点，学院做普通节点，专业就是叶子。

首先，从上面的类图中知道，对于客户端来说，其需要面对的就是一个组件，而不用管是否节点还是叶子。

**代理模式：**

代理模式的定义：由于某些原因需要给某对象提供一个代理以控制对该对象的访问。这时，访问对象不适合或者不能直接引用目标对象，代理对象作为访问对象和目标对象之间的中介。

**代理模式的主要优点有**：

代理模式在客户端与目标对象之间起到一个中介作用和保护目标对象的作用；

代理对象可以扩展目标对象的功能；

代理模式能将客户端与目标对象分离，在一定程度上降低了系统的耦合度；

**其主要缺点是：**

在客户端和目标对象之间增加一个代理对象，会造成请求处理速度变慢；

增加了系统的复杂度；

**例子：**

举个例子来说明：假如说我现在想买一辆二手车，虽然我可以自己去找车源，做质量检测等一系列的车辆过户流程，但是这确实太浪费我得时间和精力了。我只是想买一辆车而已为什么我还要额外做这么多事呢？于是我就通过中介公司来买车，他们来给我找车源，帮我办理车辆过户流程，我只是负责选择自己喜欢的车，然后付钱就可以了

**适配器模式：**

适配器模式的定义：将某个类的接口转换为客户所需的类型。换句话说，适配器模式的作用是：将原本由于接口不兼容而不能一起工作、不能统一管理的那些类变为可以在一起工作、可以进行统一管理。

**适用场景：**

系统需要使用一些现有的类，而这些类的接口（如方法名）不符合系统的需要，甚至没有这些类的源代码。

想创建一个可以重复使用的类，用于与一些彼此之间没有太大关联的一些类，包括一些可能在将来引进的类一起工作。