# 软件工程的相关概念

# 第一部分 软件开发方法

## 一、结构化的软件开发方法

软件工程的方法是基于软件危机的问题提出来的，最广泛使用的模型是结构化生命周期模型。它将软件开发分为几个不同的阶段：需求分析、系统设计、系统实现、系统测试、试运行及维护等。它的目的是模仿工程中的一些方法，将软件开发规范化，每阶段形成一定格式的文档，下一阶段以上一阶段的文档为原料。

结构化分析与设计方法是以数据流图为工具，实现对问题空间即需求的描述。它主要以数据流、数据变换为考虑对象，从这个角度来描述整个系统的状况。通过数据流图为蓝本对系统的功能加以**分解**，一直到最小的功能元素单元，然后开发人员据此进行程序设计。

|  |  |
| --- | --- |
| **优势** | **弊端** |
| 1. 可以控制软件开发的进度 2. 合理分配人力和资金 3. 软件的可靠性和维护性较高 | （1）结构化方法设计出的系统结构常常不稳定。  （2难把系统扩展到新的边界，系统难修改和扩充。  （3）几乎每开发一个新的软件系统都要针对具体系统做大量重复和复杂的工作，代码重用性差。 |

## 二、面向对象的软件开发方法

面向对象开发方法是基于所研究的问题，对问题空间进行自然分割，识别其中的对象及其相关关系，以对象为基础对软件进行处理的开发方法。它按照人类自己认识客观世界的一般方法和一般思维去分析和解决问题，是人类认识过程的计算机模拟。

**（一）、基本概念**

|  |  |
| --- | --- |
| **名称** | **概念** |
| 对象 | 描述客观事物的一个**实体**，是构成系统的基本单元，它由一组属性和一组服务（操作）组成。 |
| 类 | 是一组具有**相同**属性和相同操作的对象的集合。 |
| 实例 | 一个**具体的对象**就是类的一个实例。 |
| 消息 | 在交互的过程中，对象之间传送的**通信信息**。由三部分组成：接收消息的对象、消息名及实际变元。 |
| 封装 | 是把对象的属性和操作结合在一起，构成一个独立的对象，它的内部信息对外界是隐藏的，不允许外界直接存取对象的属性，而只能通过有限的接口与对象发生联系。对于对象的外界而言，只需要知道对象所表现的外部行为，不必了解对象行为的内部实现细节。即**统一外部接口**和**不公开内部实现**。 |
| 继承 | 子类（派生类）可以自动拥有父类（超类）的全部属性与服务。目的是为了**提高程序的可重用行**。 |
| 多态 | 在积累定义的属性和服务被其子类继承后，可以具有不同的数据类型或表现出**不同**的行为。 |
| 抽象 | 从众多的事物中**抽取共同的、本质性的特征**，而舍弃其非本质的特征。 |

**（二）、特点**

1、**封装性**：面向对象方法中，程序和数据是封装在一起的，**对象**作为一个实体，其操作**隐藏**在方法中，其状态由对象的“属性”来描述，并且只能通过对象中的“方法”来改变，从外界无从得知。面向对象方法的创始人Codd和YouMn认为，面向对象就是“**对象+属性+方法**”

2、**抽象性**：面向对象方法中，把从具有共同特性的实体中抽象出事物本质的特征和概念，成为“类（Class）”，**对象是类的一个实例**。类中封装了对象共有的属性和方法，通过实例化一个类创建的对象，自动具有类中规定的属性和方法。

3、**继承性**：是类具有的特性，类可以派生出子类，子类自动继承父类的属性和方法。这样在定义子类时，只需说明它不同于父类的特性，从而可以大大提高软件的可重用性。

4、**动态连接性**：对象间的联系是通过对象间的**消息传递**动态建立的。

**（三）、面向对象方法的开发过程**

**1、面向对象的分析（OOA）**

（1）问题领域分析（标识对象）

（2）发现和定义对象、类（标识结构，确定内部特征）

（3）识别对象的外部联系（标识主题，识别外部联系）

（4）建立系统的静态结构模型（定义属性，建立系统的静态结构模型）

（5）建立系统的动态行为模型（定义方法，建立系统的动态行为特征）

**2、面向对象的设计（OOD）**

（1）设计对象与类（具体设计对象与类的属性、服务操作）

（2）设计系统结构（设计组件与子系统及其动静态关系）

（3）设计问题域子系统（确定系统的主要业务）

（4）设计人机交互子系统（设计用户界面及详细交互过程）

（5）设计数据管理子系统（数据的管理）

（6）设计任务管理子系统（负责协调和管理进程）

（7）设计优化，提高系统的性能

**3、面向对象的编程（OOP）**

**4、面向对象的测试（OOT）**

**5、面向对象的维护（OOM）**

**（四）、总结**

（1）客观世界是由对象组成的

（2）对象由属性和操作组成

（3）对象可按其属性进行分类

（4）对象之间的联系通过传递消息来实现

（5）对象具有封装性、继承性和多态性。

## 三、结构化方法和面向对象方法的比较

|  |  |  |
| --- | --- | --- |
| **项目** | **面向对象** | **结构化设计** |
| 基本思想 | 自下而上设计库类 | 自上而下设计过程库，逐步求精 |
| 概念或者术语名词 | 对象、类、消息、继承等 | 过程、函数、数据等 |
| 逻辑工具 | 对象模型图、数据字典动态模型图、功能模型图 | 数据流图、系统结构图、数据字典状态转移图、实体关系图 |
| 处理问题出发点 | 面向问题 | 面向过程 |
| 控制程序方式 | 通过“事件驱动”来激活和运行程序 | 通过设计调用或返回程序 |
| 可扩展性 | 只需修改或增加操作，而基本对象结构不变，扩展性好 | 功能的变化会危及整个系统，扩展性差 |
| 重用性 | 好 | 不好 |
| 层次结构的逻辑关系 | 用类的层次结构来体现类之间的继承和发展 | 用模块的层次结构概括模块和模块之间的关系和功能 |
| 分析、设计、编码转换方式 | 平滑过程，无缝连接 | 按规则转换，有缝连接 |
| 运行效率 | 相对低 | 相对高 |

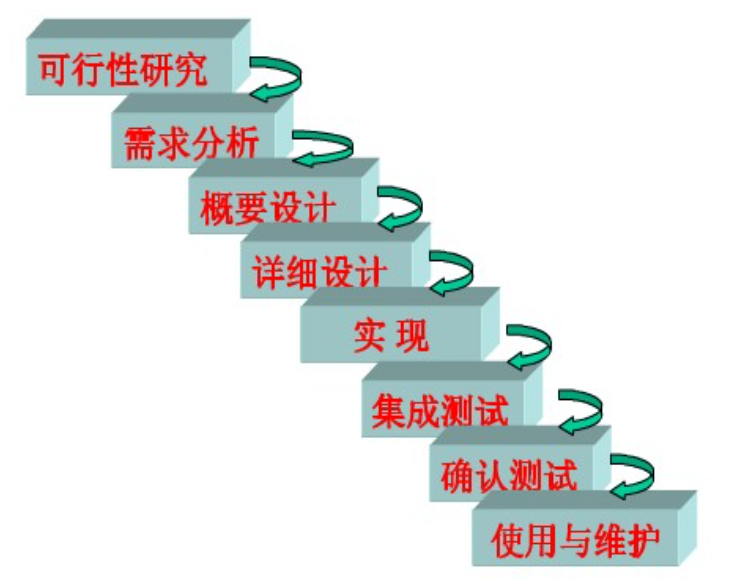
# 第二部分 软件开发模型

## 基本概念

软件开发模型是软件开发全过程、软件开发活动以及他们之间关系的结构框架。它是一种策略，这种策略是由软件工程师在具体的实践工程活动当中设计并提炼出来，能够覆盖软件生命周期的基本阶段，确定设计的方法、过程及工具。

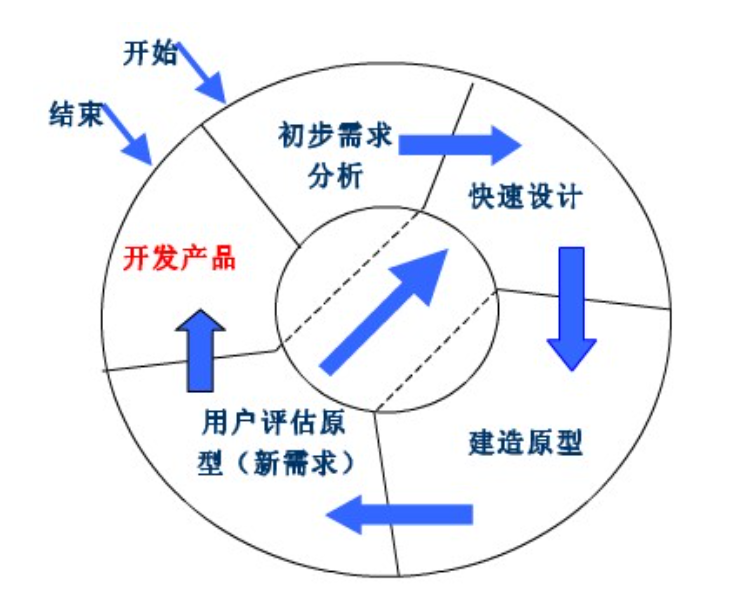
**（一）、瀑布模型**

瀑布模型的软件开发过程与软件生命周期是一致的并且它是由文档驱动，两相邻阶段之间存在因果关系，需要对阶段性的产品进行审批。瀑布模型**假定用户的需求是不变的**，因此缺乏灵活性。



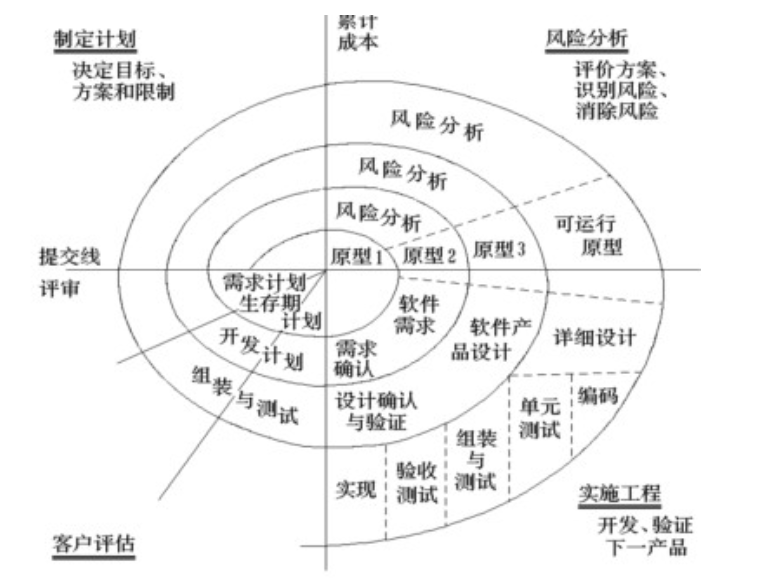
**（二）、快速原型模型**

快速原型模型在功能上等价于产品的一个子集。根据客户的需要在很短的时间内解决用户最迫切的需要，此时只是部分功能的实现，快速原型模型最重要的目的是**确定用户真正的需求并支持需求的动态变化**。一般不会单独使用，而和瀑布模型或螺旋模型一起使用。



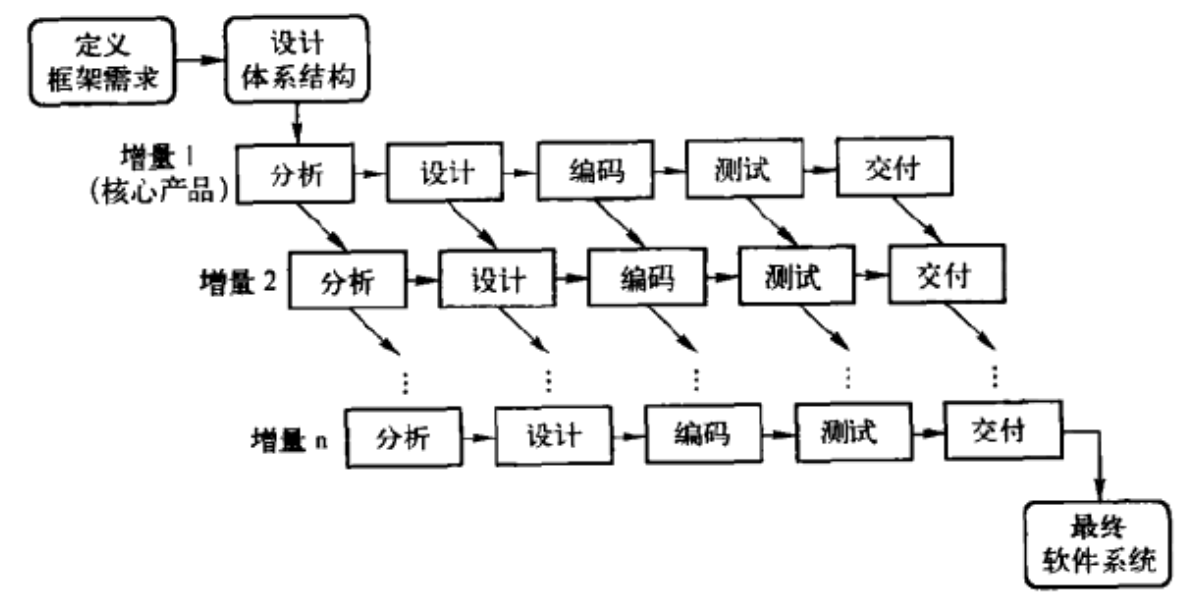
（**三）、螺旋模型**

螺旋模型是瀑布模型和快速原型模型的有机结合，它是**由风险驱动**的，它强调了其他模型所忽略的风险分析，适合较大的系统，但是缺乏对风险的评估。



**（四）、增量模型**

螺旋模型是一种采用周期性的方法来进行软件系统开发的模型，它结合了瀑布模型与原型模型的特点，强调了其他模型所忽视的**风险分析**，常用于大型复杂系统的开发。

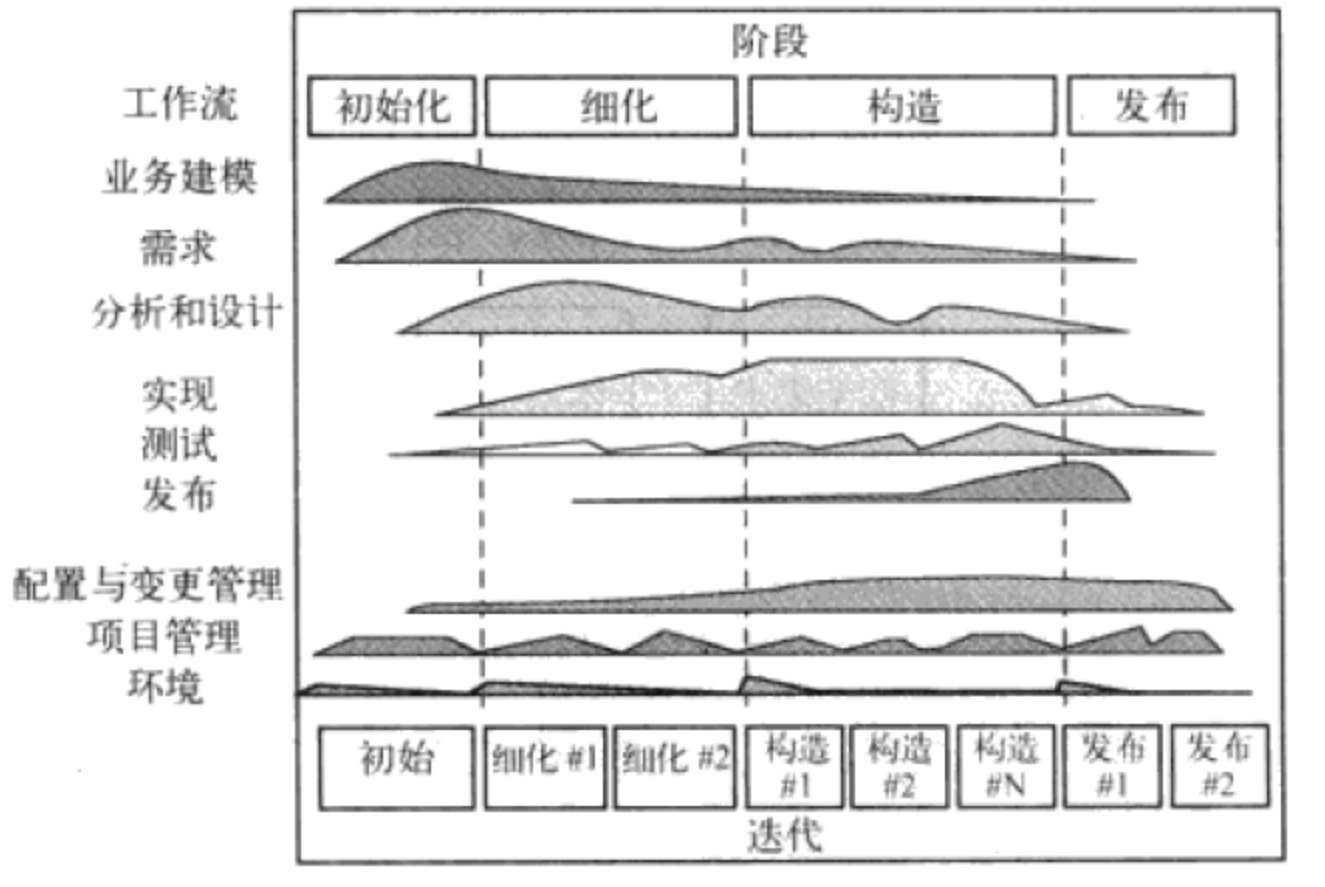


**（五）、统一软件过程RUP模型**

RUP（Rational Unified Process 统一过程）是Rational公司提出的一套软件开发过程模型。它将用户需求转化为软件系统所需活动的集合，描述了一系列相关的软件工程流程，他们具有相同的结构，即相同的流程框架。

RUP可以用二维坐标来描述。**横轴**通过**时间组织**，是过程展开的生命周期特征，体现开发过程的动态结构，用来描述它的术语主要包括周期、阶段、迭代和里程碑；**纵轴**以**内容**来组织为自然的逻辑活动，体现开发过程的静态结构，用来描述它的术语主要包括活动、产物、工作者和工作流。

RUP中的软件生命周期在时间上被分解为4个阶段，即初始阶段、细化阶段、构建阶段以及交付阶段。每个阶段结束于一个主要的里程碑，而每个阶段本质上是两个里程碑之间的时间跨度。



## 二、各模型的对比

|  |  |  |
| --- | --- | --- |
| **模型** | **特点** | **适用范围** |
| 瀑布模型 | 分阶段，各个阶段完成后都有评审，允许反馈，要求预先确定需求 | 需求定义完善，不易变更的系统 |
| 螺旋模型 | 具有瀑布模型、快速原型模型的特点，并引进了风险分析活动 | 需求难以获取和确定、开发风险较大的系统 |
| 原型模型 | 不要求需求预先完备定义，支持用户参与，能够适应用户需求的变化 | 需求复杂、动态变化、难以确定的系统 |
| 增量模型 | 增量式开发，允许开发活动并行 | 技术风险大、用户需求较为稳定的系统 |
| RUP模型 | 可改造、扩展和剪裁，可以对它进行设计、开发和维护 | 复杂、需求不确定的系统，软件开发项目组拥有丰富的软件开发和管理经验 |

**参考资料：**

<http://blog.csdn.net/shadowyelling/article/details/7406442>

<http://blog.csdn.net/lairyi/article/details/79325921?foxhandler=RssReadRenderProcessHandler>

<http://blog.csdn.net/omenglishuixiang1234/article/details/51728276>