几种软件开发方法对比

# 综述

软件开发方法是一种使用早已定义好的技术集及符号表示习惯来组织软件生产的过程。

本文对净室方法、结构化方法、面向对象方法、原型法、逆向工程等方法进行梳理，并对各种开发方法特点、优点进行对比。

# 净室方法

## 特点

净室软件工程（Cleanroom Software Engineering, CSE)是软件开发的一种形式化方法，可以开发较高质量的软件。它使用盒结构规约进行分析和建模，并且将正确性验证作为发现和排除错误的主要机制，使用统计测试来获取认证软件可靠性所需要的信息。CSE强调在规约和设计上的严格性，还强调统计质量控制技术，包括基于客户对软件的预期使用测试。

## 缺点

CSE太理论化，需要更多的数学知识。其正确性验证的步骤比较困难且比较耗时。CSE要求采用[增量式开发](https://baike.baidu.com/item/%E5%A2%9E%E9%87%8F%E5%BC%8F%E5%BC%80%E5%8F%91/5133612)、采用盒子结构、采用统计测试方法，普通工程师必须经过加强训练才能掌握。

CSE开发小组不进行传统的[模块测试](https://baike.baidu.com/item/%E6%A8%A1%E5%9D%97%E6%B5%8B%E8%AF%95)，这是不现实的。工程师可能对编程语言和开发环境还不熟悉，而且编译器或操作系统的bug也可能导致未预期的错误。

CSE毕竟脱胎于传统软件工程，不可避免地带有传统软件工程的一些弊端。

# 结构化方法

## 特点

结构化方法的主要特点是：开发目标清晰化、开发工作阶段化、开发文档规范化和设计方法结构化。结构化方法特别适合于数据处理领域的问题，但是不适应于规模较大、比较复杂的系统开发。

由结构化分析、结构化设计、结构化程序设计组成，是一种面向数据流的开发方法。

结构化分析是根据分解与抽象的原则，按照系统中数据处理的流程，用数据流图来建立系统的功能类型，从而完成需求分析工作。结构化方法分析模型的核心是数据字典，围绕这个核心，有三个层次的模型，分别是数据模型、功能模型和行为模型（也称为状态模型）。在实际工作中，一般使用E-R图表示数据模型，用DFD表示功能模型，用状态转换图表示行为模型。这三个模型有着密切的关系，它们的建立不具有严格的时序性，而是一个迭代的过程。

结构化设计是根据模块独立性准则、软件结构优化准则将数据流图转换为软件的结构体系，用软件结构图来建立系统的物理模型，实现系统的概要设计。

结构化程序设计使用3种基本控制结构构造程序，任何程序都可以由顺序、选择和重复3种基本控制结构构造。

自顶向下方法是一种决策策略。软件开发涉及作什么决策、如何决策和决策顺序等决策问题。

自顶向下方法在任何时刻所作的决定都是当时对整个设计影响最大的那些决定。如果把所有决定分组或者分级，那么决策顺序是首先作最高级的决定，然后依次地作较低级的决定。同级的决定则按照随机的顺序或者按别的方法。一个决策的级别是看它距离要达到的最终目的（因此是软件的实际实现）的远近程度。从问题本身来看，或是由外(用户所见的）向内（系统的实现）看，以距离实现近的决定为低级决定，远的为高级决定。

## 缺点

对系统分析和设计人员的要求较高。

开发周期长，系统复杂，一般属于一种高成本、大投资的工程。

对于大系统而言，自上而下的规划对于下层系统的实施往往缺乏约束力，

从经济角度来看，很难说自顶向下的做法在经济上市合算的。

# 面向对象方法

## 特点

面向对象方法是系统的描述及信息模型的表示与客观实体相对应，符合人们的思维习惯，有利于系统开发过程中用户与开发人员的交流和沟通，缩短开发周期，提高系统开发的正确性和效率。

面向对象方法包含面向对象分析OOA、面向对象设计OOD、面向对象程序设计OOP。

面向对象的分析模型主要由顶层架构图、用例与用例图、领域概念模型构成。用例建模用来描述待开发系统的功能需求，主要元素是用例和参与者。参与者是指系统以外的，需要使用系统或与系统交互的事物，包括：人或组织、设备、外部系统等，比较隐晦的参与者包括：时间、温度；用例是对系统行为的动态描述，用例获取是需求分析阶段的主要任务之一，用例之间的关系包括：包含、扩展、泛化。

面向对象的设计模型则包含以包图表示的软件体系结构图、以交互图表示的用例实现图、完整精确的类图、针对复杂对象的状态图和用以描述流程化处理过程的活动图等。在面向对象设计中，类可以分为三种类型：实体类、边界类和控制类。类之间的关系包括6种：关联、聚合、组合、依赖、泛化、实现。

## 缺点

类和继承等特点使得程序会多很多指针操作来定位函数入口和自身要维护虚拟方法表等额外的工作，程序的处理效率相对要低（但程序开发效率高）。

# 原型法

## 特点

原型是软件系统的初始版本，用来演示概念并尝试设计选择，通常用来发现更多的问题和可能的解决方案。快速迭代式的原型开发能够有效控制成本，根据原型与最终产品之间的关系，原型开发分为三类：抛弃式原型开发利用原型验证和澄清系统的需求描述，重新构造系统：演化式原型开发逐步改进和细化原型，将原型进化直至产生出目标系统；增量式原型开发在建立软件总体设计的基础上，采用增量开发方法，使原型成为最终系统。

以原型开发思想为基础，提出了螺旋模型、敏捷方法等。

## 缺点

原型法不适用于开发大型的信息系统；系统难于维护；如果用户合作不好，盲目纠错，会拖延开发进程。

# 逆向工程

## 特点

逆向工程与重构工程是目前预防性维护采用的主要技术。所谓软件的逆向工程就是分析已有的程序，寻求比源代码更高级的抽象表现形式。一般认为，凡是在软件生命周期内将软件某种形式的描述转换成更为抽象形式的活动都可称为逆向工程。逆向工程导出的信息可以分为：实现级、结构级、功能级、领域级，信息的抽象级别越高，它与代码的距离就越远，通过逆向工程恢复的难度亦越大，而自动工具支持的可能性相对变小，要求人参与判断和推理的工作增多。

# 敏捷方法

Scrum

Scrum是橄榄球比赛中“争球”的意思，你可以脑补争球时的敏捷、激情和你争我夺。现在Scrum是广泛应用于IT界的一套项目管理工具

Scrum的核心，是把整个项目分成若干个冲刺，每次2-4周，冲完一次再来一次。

Scrum的本质，是把一次漫长的长跑，分割成一段段全力以赴的冲刺，通过一套标准流程方法，提高效率。

Scrum是由三个角色（产品负责人，Scrum Master，团队成员）、四个仪式（冲刺计划会，每日站会，冲刺评审会，冲刺回顾会）和三个物件（产品积压，冲刺积压，燃尽图）组成的一套项目管理方法。

产品负责人，召开冲刺计划会，定下三件事：冲刺目标。定下目标后，把它们从“产品积压”，移入“冲刺积压”。

冲刺方法。选定冲刺方法。

分配任务。团队成员若干人，在6个步骤中，各自主动领取任务。

燃尽，是烧完了的意思。随着时间推移，剩余工作量越来越少。把计划进度，画成一根从左上到右下的直线。然后，把实际进度用红线标在旁边，看着工作量像蜡烛一样，不断燃尽。

冲刺评审会，由主编负责主持，一起审阅最终交付的文章。冲刺回顾会，讨论开始做什么，停止做什么，继续做什么。

极限编程

XP (Extreme Programming，极限编程)在所有的敏捷型方法中，XP是最引人瞩目的。它源于Smalltalk圈子，特别是Kent Beck和Ward Cunningham在20世纪80年代末的密切合作。XP在一些对费用控制严格的公司中的使用，已经被证明是非常有效的。

水晶方法

Cockburn的水晶系列方法，水晶系列方法是由AlistairCockburn提出的。它与XP方法一样，都有以人为中心的理念，但在实践上有所不同。Alistair考虑到人们一般很难严格遵循一个纪律约束很强的过程，因此，与XP的高度纪律性不同，Alistair探索了用最少纪律约束而仍能成功的方法，从而在产出效率与易于运作上达到一种平衡。也就是说，虽然水晶系列不如XP那样的产出效率，但会有更多的人能够接受并遵循它。

开源界的开发方法

开放式源码，这里提到的开放式源码指的是开放源码界所用的一种运作方式。开放式源码项目有一个特别之处，就是程序开发人员在地域上分布很广，这使得它和其他敏捷方法不同，因为一般的敏捷方法都强调项目组成员在同一地点工作。开放源码的一个突出特点就是查错排障(debug)的高度并行性，任何人发现了错误都可将改正源码的“补丁”文件发给维护者。然后由维护者将这些“补丁”或是新增的代码并入源码库。

FDD

Coad的功用驱动开发方法(FDD-Feature Driven Development)，FDD是由Jeff DeLuca和大师Peter Coad提出来的。像其他方法一样，它致力于短时的迭代阶段和可见可用的功能。在FDD中，一个迭代周期一般是两周。在FDD中，编程开发人员分成两类：首席程序员和“类”程序员(classowner)。首席程序员是最富有经验的开发人员，他们是项目的协调者、设计者和指导者，而“类”程序员则主要做源码编写。