

第二节 项目管理与需求分析

詹文翰

zhanwenhan@163.com

课程安排

1

项目概述与技术基础

2

项目管理与需求分析

3

系统设计

4

软件配置与项目实施

5

需求变更与敏捷开发

6

软件测试

7

知识拓展与项目完善

8

项目答辩及演示

软件项目管理

2

项目管理与需求分析

软件项目管理的特殊性：

- 1、软件是纯知识产品，其开发进度和质量很难估计和度量，生产效率也难以预测和保证。
- 2、软件系统的复杂性也导致了开发过程中各种风险的难以预见和控制。

例：Windows这样的操作系统有1500万行以上的代码，同时有数千个程序员在进行开发，项目经理都有上百个。这样庞大的系统如果没有很好的管理，其软件质量是难以想象的。

软件项目管理的内容主要包括：人员的组织与管理，软件度量，软件项目计划，软件任务跟踪，软件配置管理等。

强大的工具往往可以帮助我们事半功倍。

常用项目管理软件：Project、Teambition、PingCode

项目管理与需求分析

任务：是指软件开发中的一个具体工作。

工期：是完成某项任务所需活动工作时间的总长度，通常是从任务开始日期到完成日期的工作时间量。

里程碑：是一个工期为零，用于标识关键项目节点的重要日程事项。

[illegible]

制定和追踪项目计划

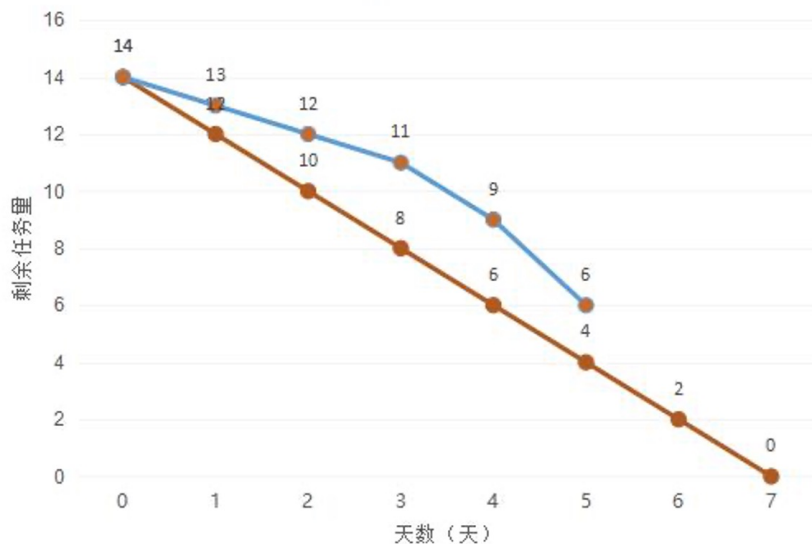
2

项目管理与需求分析

燃尽图 (burn down chart)：是用于表示剩余工作量的工作图表，横轴表示时间，纵轴表示剩余工作量，可以直观的预测工作的完成进度，常用于敏捷开发当中。

若实际工作线高于理想工作线时，则意味着剩余的工作量比原先预测的多，并且项目落后于计划。

若实际工作线低于理想工作线，则意味着剩余的工作量比原先预测的少，并且项目提前完成。



Teambition演示

软件工程方法学

软件危机

2

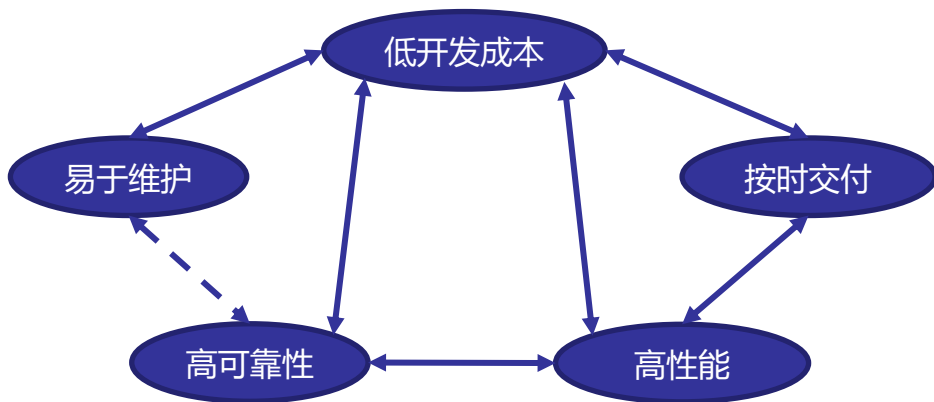
从20世纪60年代末至70年代初，“软件危机”一词在计算机行业广泛流传，指代在软件开发过程和维护中出现的大量问题。这些问题的产生都是由于落后的软件生产方式无法满足迅速增长的计算机软件应用需求所导致。

软件工程

“软件工程”的概念是1968年在德国召开的NATO（北大西洋公约组织）会议上首次提出的，强调应用工程化的原则和方法进行软件的开发和管理。

软件工程方法学

软件工程的目标



软件工程的三要素

软件开发过程：研究软件开发的生命周期，已提出了多种软件开发过程模型，如瀑布模型、原型模型、增量模型等。

软件开发方法：软件开发在分析、设计和编程的过程中所基于的方法学，主流的方法学包括：结构化方法和面向对象方法。

软件开发工具：为支持不同的软件开发过程和方法学而研制的计算机软件系统和开发环境。

软件过程模型

2

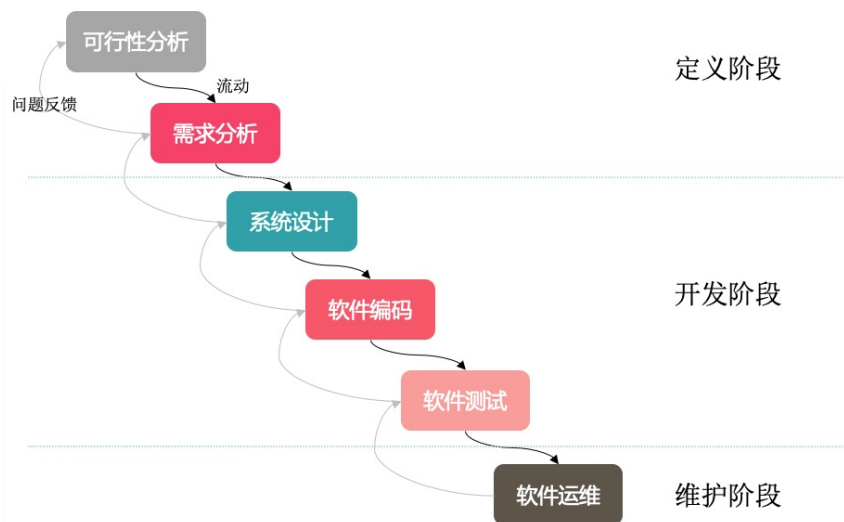
项目管理与需求分析

瀑布模型

瀑布模型（Waterfall Model）是所有软件开发过程模型的基础。其是一个软件项目的理想开发架构，开发过程通过一系列的阶段顺序展开，从系统需求分析开始直到产品发布和维护。

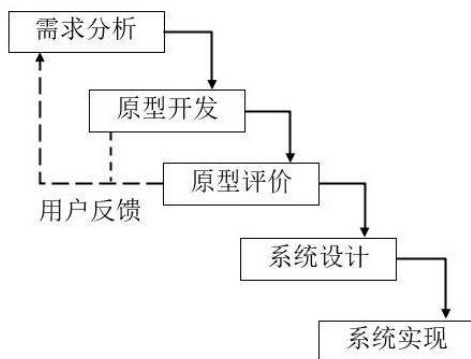
特点：

- 以里程碑的方式严格定义各开发阶段的输入和输出。如果上一阶段的输出达不到要求，则不开展下一阶段的工作。
- 各阶段定义清晰，强调文档，便于阶段评审、审计、管理和跟踪。
- 软件生命周期前期造成的Bug的影响比后期的大的多。
- 无法解决软件需求不明确、不准确和完善性问题。

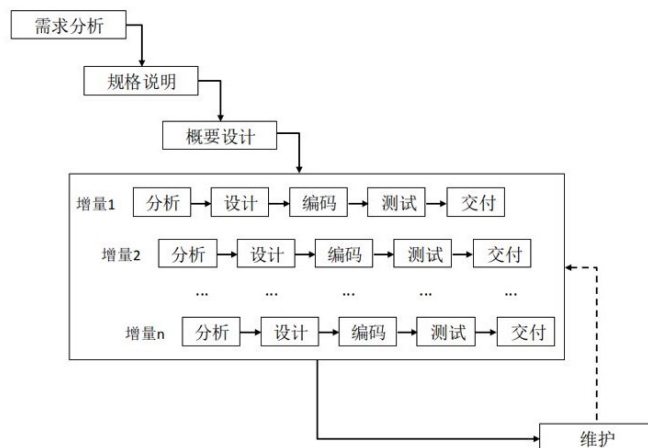


软件过程模型

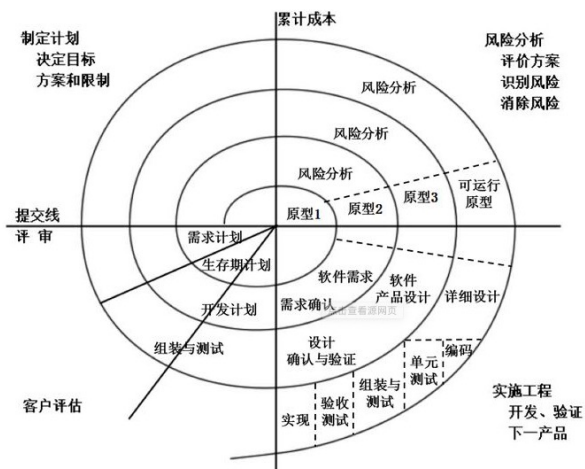
原型模型



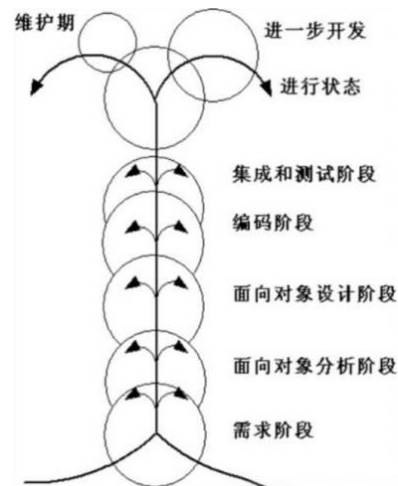
增量模型



螺旋模型



喷泉模型



2

项目管理与需求分析

软件开发方法

结构化方法

最受广大软件工程师所熟知的软件开发方法。该方法是20世纪60年代末在结构化编程的基础上发展而来的，由**结构化分析**、**结构化设计**和**结构化编程**三部分有机组成。

基本思想：自顶向下，逐步求精。通过把一个复杂问题（系统）进行抽象，将大问题（系统）分解为小问题（子系统），再将小问题（子系统）分解为更小的、相对独立的问题（任务）进行实现。

缺点：围绕“**过程**”来构造系统，而非“**功能**”本身；而用户的需求变化主要是针对功能的。因此，需求变化所导致的软件实现代价过大。

面向对象方法

建立在“**对象**”概念基础之上，将面向对象的思想应用于软件开发过程中。该方法起源于面向对象的程序设计模式，包括**面向对象分析**、**面向对象设计**和**面向对象编程**三部分。

基本思想：封装、继承和抽象。客观世界由对象组成。将系统中的任何事物都抽象为对象，使开发软件的过程尽可能接近人类的自然思维方式。

由于客观世界的对象之间的关系相对比较**稳定**，因此，面向对象分析和设计的结果也比较稳定，从而在一定程度上提高了软件的**生产率**、**可靠性**、**易重用性**和**易维护性**。面向对象方法正在逐步成为计算机领域软件开发的主流方法。

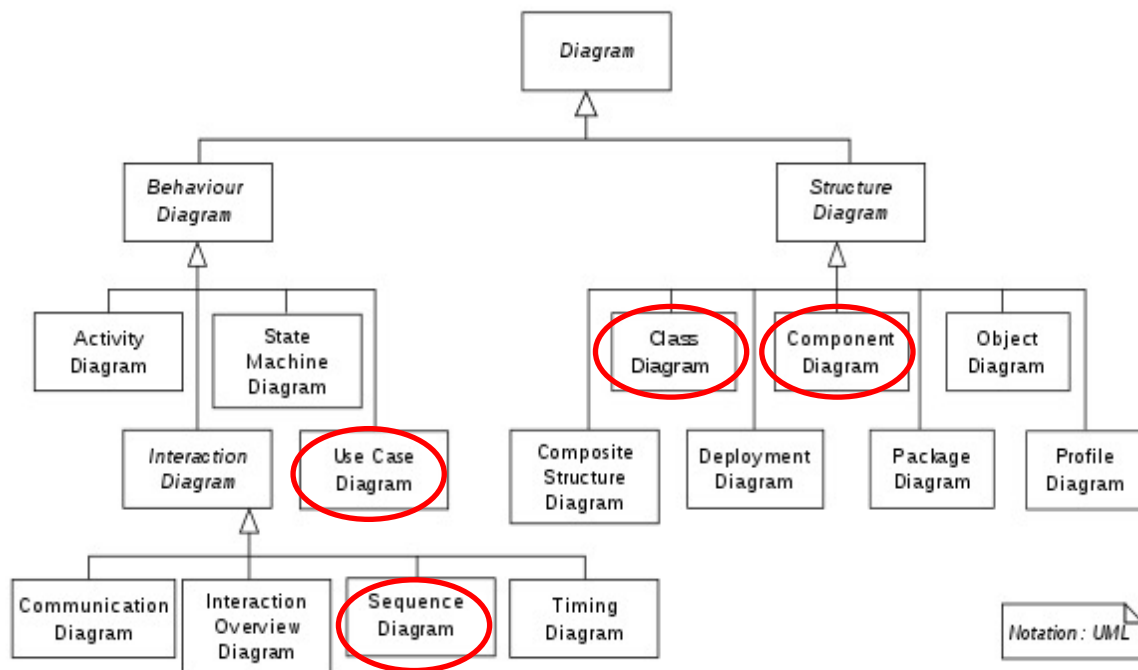
统一建模语言UML

面向对象分析与设计的有力工具

2

UML产生于20世纪90年代中期，是一个通用的、可视化的标准建模语言，用于对使用面向对象方法所开发的系统进行说明。

UML是一种形式化语言，定义了一系列的图形符号来描述软件系统。这些图形符号有严格的语义和清晰的语法，可以同时对该系统的静态结构和动态结构进行建模。



面向对象的需求分析

2

项目管理与需求分析

面向对象需求分析的核心是用例模型。用例模型是一个包含参与者、用例及它们之间关系的模型，描述了系统应为其用户做些什么以及在哪些约束限定之下的模型。

参与者：指系统以外的，使用系统或与系统产生交互的人或者实体。



用例：一个完整的、可以被参与者感受到的系统功能单元。



面向对象的需求分析

关系：关联、包含、泛化、扩展四大关系。

2

关联关系：参与者与用例之间的通信路径。

—————

包含关系：在基础用例之上插入附加行为；支持在用例模型中复用功能。

——<<include>>——>

泛化关系：指示子代将接收父代中定义的所有属性、操作和关系，并可以增加新的特性。

—————>

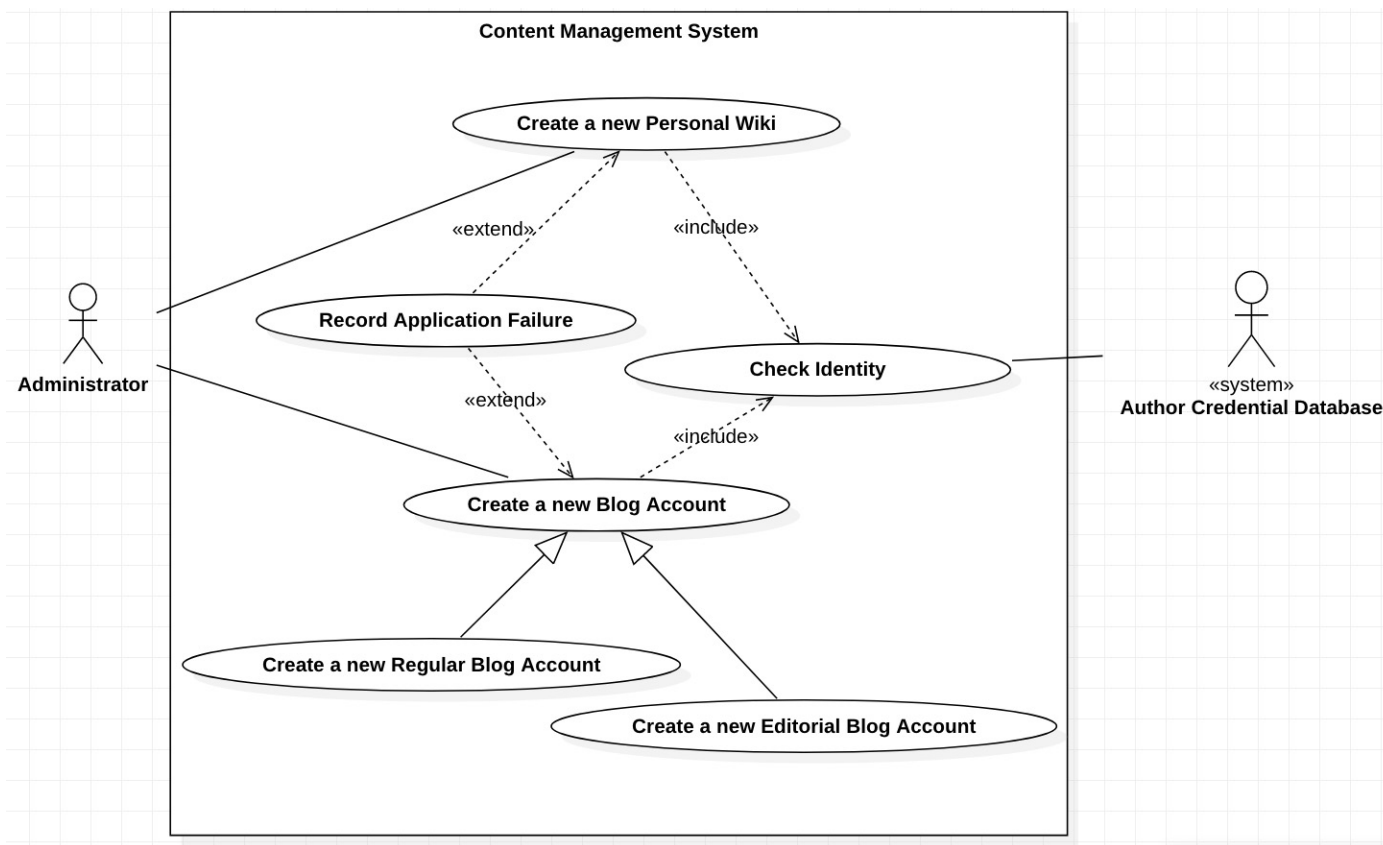
扩展关系：对基础用例的功能进行可选的扩充。

——<<extend>>——>

用例图举例

2

项目管理与需求分析




内容发布管理系统（出自《UML学习指南》）

对于关系的进一步说明

包含关系与扩展关系的区别：  

1、**include**通常发生在“多个用例中有可以提取出来的公共部分”，类似于提取公因式。包含方知道有被包含方（箭头指向方）的存在，但被包含方不知道包含方的存在。

2、**extend**表示扩展用例可以扩展被扩展用例（箭头指向方）在某方面的行为，被扩展用例需要留出扩展接口，交由扩展用例来实现。扩展方知道被扩展方的存在，但被扩展方不知道扩展方的存在。

依赖关系： 

正确的用例图是不存在依赖关系的。

依赖关系说明用例之间存在流程上的相互覆盖。针对这种情况，应将用例进一步细化，提取出公共部分，然后使用包含关系（**include**）来表示用例之间的关系。

用例描述

2

项目管理与需求分析

在进行需求分析时，仅用例图并不能提供足够的需求细节，必须假以用例描述，才能让客户以及设计者充分理解真实的需求。

用例描述举例（接上例）：

用例标识		xxx	用例名称	Create a new Regular Blog Account
创建人		xxx	创建日期	2020年11月2日
语境目标		一位新的或现有作者通过线下向管理员请求一个新的普通博客账户		
前置条件		作者需有适当的身份证明		
成功的结束状态		成功创建该作者的新普通博客账户		
失败的结束状态		该作者的新普通博客账户申请被拒绝		
参与者		Administrator		
触发器		管理员要求CMS建立一个新的普通博客账户		
基础用例		Create a new Blog Account		
包含用例		Check Identity		
扩展用例		Record Application Failure		
事件流	基本流程	1、管理员要求系统建立一个新的博客账户		
		2、管理员选择普通博客账户类型		
		3、管理员输入作者的详细数据		
		4、检查该作者的详细数据		
		include::Check Identity		
	扩展流程	5、成功创建新的博客账户		
		4.1、不允许该作者建立新的博客		
		4.2、博客账户申请被拒绝		
4.3、此申请失败情况被记录在该作者的历史数据里				
非功能需求		安全性		
补充规格说明书		无		

用例图演示

实验环节

2

项目管理与需求分析

完成项目的需求分析（小组讨论）：

- 思考项目需要实现的功能
- 设计软件的交互使用方式
- 绘制项目用例图
- 撰写用例描述
- 推荐使用**Teambition**进行项目管理
 - 项目任务分配
 - 制定项目计划
 - 项目文档分享
 - 跟踪项目执行状态
- 完成项目需求分析（根据实验模板撰写）