本章主要内容

- 软件和软件工程的基本概念；
- 给出软件和软件工程的发展历程；
- 介绍不同类型的开发过程和生命周期模型。

- 软件的概念、特点及分类
- 软件危机
- 软件工程
- 软件生命期模型
- 软件工程的目标与原则
- 软件工程的教学意义

回顾“程序”

程序 = 数据结构 + 算法

handler_ticket.js

```
var template = require('./reply_template');
var model = require('../models/models');
var lock = require('../models/lock');
var urls = require("../address_configure");
var checker = require("./checkRequest");
var basicInfo = require("../weixin_basic/settings.js");

//Attentez: keep the activity::key unique globally.
var TICKET_DB = model.tickets;
var USER_DB = model.students;
var ACTIVITY_DB = model.activities;
var SEAT_DB = model.seats;
var db = model.db;

var alphabet = "qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM0123456789";

var act_cache={};
var rem_cache={};
var usr_lock={};

exports.clearCache=function()
{
    act_cache={};
}

function verifyStudent(openID,ifFail,ifSucc)
{
    db[USER_DB].find({weixin_id:openID,status:1},function(err,docs)
    {
        if (err || docs.length==0)
        {
            ifFail();
            return;
        }
        ifSucc(docs[0].stu_id);
    });
}
```

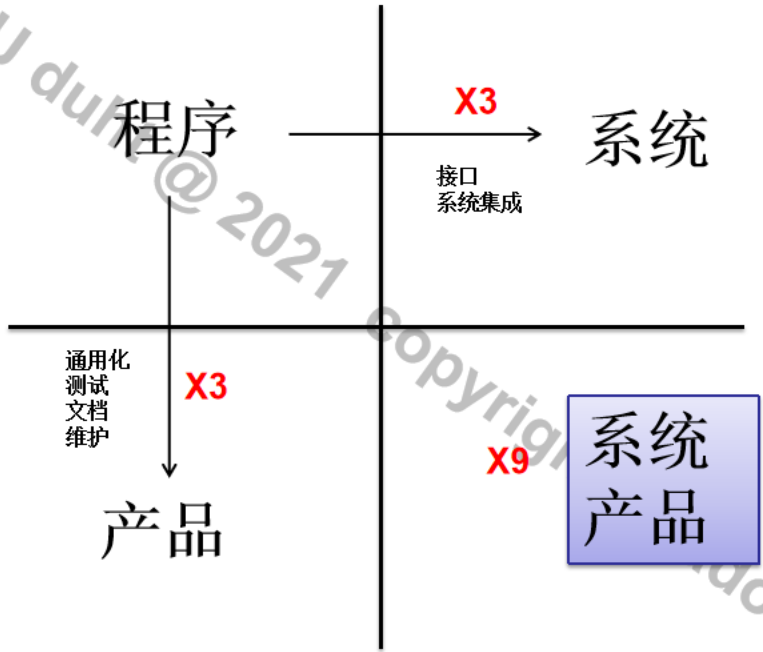
软件的概念、特点及分类

软件 是计算机系统中与硬件相互依存的另一部分；
它是包括程序、数据及其相关文档的完整集合。

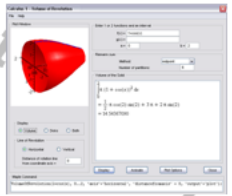
- **程序** 是按事先设计的功能和性能要求编写的指令序列
- **数据** 是使程序能正常操纵信息的数据结构
- **文档** 是与程序开发、维护和使用有关的图文材料

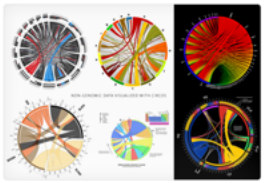
软件 =  **+**  **+** 

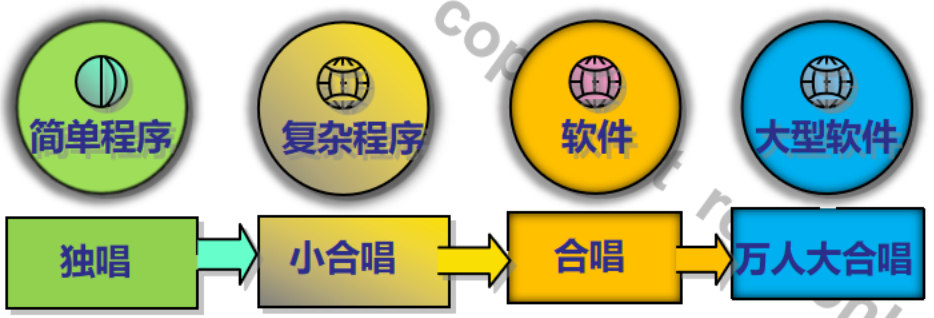
程序并不是软件，**程序**只是软件的组成部分。



软件的概念、特点及分类

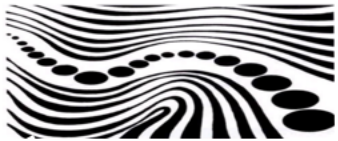






软件的概念、特点及分类

- 无形性(逻辑实体)、抽象性

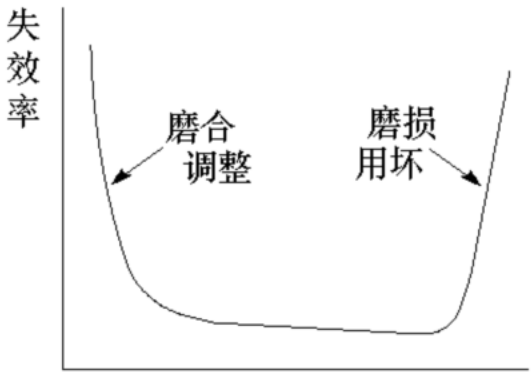


- 软件生产无明显制造过程

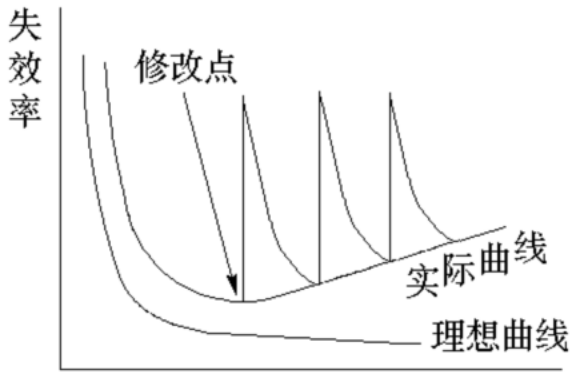
- 无磨损性

- 对计算机硬件依赖性
- 软件本身复杂性
- 成本高昂
- 软件的手工开发方式

- 涉及到社会因素

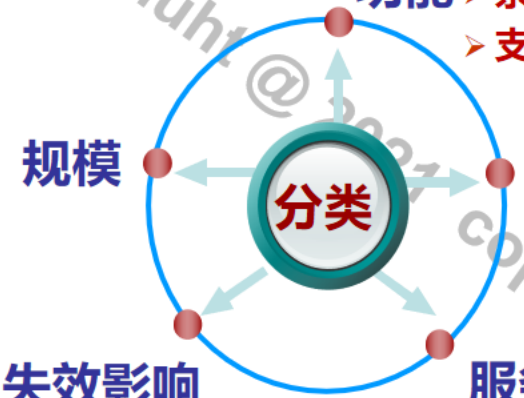


（a）硬件失效率曲线



（b）软件失效率曲线

软件的概念、特点及分类



功能
- 应用软件
- 系统软件
- 支撑软件

工作方式
- 分时软件
- 实时软件
- 交互式软件
- 批处理式软件

服务对象
- 项目软件
- 产品软件

失效影响
- 高可靠性软件
- 一般可靠性软件

规模

每一类软件在管理方面要求程度不同。

软件的概念、特点及分类

按规模分类

类别	参加人员数	研制期限	源程序行数	说明
微型	1	1～4周	0.5k	数值计算或数据处理，通常没有与其它程序的接口。需要按一定的标准化技术、正规的资料书写以及定期的系统审查。不需要特别严格。
小型	1-2	1～6月	1k～2k	
中型	2～5	1～2年	5k～50k	软件人员之间、与用户之间的联系、协调的配合关系。因而计划、资料书写以及技术审查需要比较严格地进行。应用程序和系统程序。系统的软件工程方法是完全必要的。
大型	5～20	2～3年	50k～100k	编译程序、小型分时系统、实时控制系统等。二级管理，若干小组，每组5人以下。人员调整往往不可避免。采用统一的标准，实行严格的审查是绝对必要的。
甚大型	100～1000	4～5年	1M(=1000k)	若干个子项目，每一个子项目都是一个大型软件。子项目之间具有复杂的接口。如：远程通信系统、多任务系统、大型操作系统、大型数据库管理系统、军事指挥系统通常现有这样的规模。很显然，这类问题没有软件工程方法的支持，它的开发工作是不可想象的。
极大型	2000～5000	5～10年	1M～10M	军事指挥、弹道导弹防御系统。

设置

软件产品与物质产品有很大的区别，软件产品是一种（）产品。

A 有形

B **消耗**

C **逻辑**

D **文档**

提交

设置

下列关于软件的说法正确的是（ ）

A 软件是通过定制进而生产制造出来的

B 软件没有磨损老化问题。

C 软件开发的成本很高

D 软件开发和运行必须依赖计算机环境

提交

设置

软件不会磨损但会逐渐退化，其原因在于（ ）

A 软件通常暴露在恶劣的环境下

B 软件错误在经常使用之后会逐渐增加

C 不断的变更使组件接口之间引起错误

D 软件备件很难订购

提交

设置

下列软件属于系统软件的是（）

A 操作系统

B 编译器

C 中间件

D 浏览器

提交

设置

下述软件属于支撑软件的是（）

A 财务管理软件

B 编译器

C **IDE(Integrated Development Environment)**

D **IBM的Rational**
https://www.ibm.com/developerworks/cn/rational/index.html

提交

设置

下述软件属于应用软件的是（）

A 财务管理软件

B 数据库

C 编译器

D 搜狗浏览器

提交

- 软件的概念、特点及分类
- **软件危机**
- 软件工程
- 软件生命期模型
- 软件工程的目标与原则
- 软件工程的教学意义

起源

20世纪60年代末70年代初，西方工业发达国家经历了一场“软件危机”。

这场软件危机表现在：

1) 软件十分复杂，价格昂贵，供需差日益增大；

2) 软件开发时又常常受挫，质量差，指定的进度表和完成日期很少能按时实现，研制过程很难管理，即软件的研制往往失去控制。我们称**软件开发**和**维护**过程中所中遇到的这**一系列严重问题为软件危机。**

软件工程概念的出现源自软件危机。

起源

美国IBM公司在1963年至1966年开发的IBM360机的操作系统。这一项目花了5000人一年的工作量，最多时有1000人投入开发工作，写出了近100万行源程序。……据统计，这个操作系统每次发行的新版本都是从前一版本中找出1000个程序错误而修正的结果。……

这个项目的负责人F. D. Brooks事后总结了他在组织开发过程中的沉痛教训时说：“……正像一只逃亡的野兽落到泥潭中做垂死的挣扎，越是挣扎，陷得越深，最后无法逃脱灭顶的灾难。……程序设计工作正像这样一个泥潭，……一批批程序员被迫在泥潭中拼命挣扎，……谁也没有料到问题竟会陷入这样的困境……”。IBM360操作系统的历史教训成为软件开发项目的典型事例为人们所记取。

表 现

软件需求的增长得不到满足
软件开发成本和进度无法控制
软件质量难以保证
软件不可维护或维护程度非常低
软件成本不断提高
软件开发生产效率的提高赶不上硬件的发展和应用需求的增长
软件通常缺少适当的文档资料

实际上几乎
所有软件
都在不同程度
上存在软件危机

案例1：军用软件

- 1963年美国飞往火星的火箭爆炸，造成1000万美元的损失。原因是FORTRAN程序：

  DO 5 I=1，3

  误写为：DO 5 I=1 . 3

- 1967年苏联“联盟一号”载人宇宙飞船在返航时，由于软件忽略一个小数点，在进入大气层时因打不开降落伞而烧毁。
- 1991年2月25日美军位于沙特阿拉伯宰赫兰的军营被一枚成功突防的“飞毛腿”击中，死伤28人
- 1996年6月4日，阿丽亚娜5型运载火箭终于进行了第一次发射，但不幸遭遇失败，这在世界上产生了重大影响。从1987年确定计划直到1996年首次飞行，阿丽亚娜5型运载火箭计划历时8年多时间，研制费用高达80亿美元以上。

案例2 - 12306系统

- 12306网络购票系统历时两年研发成功，耗资3亿元人民币，于2011年6月12日投入运行。
- 2012年1月8日春运启动，9日网站点击量超过14亿次，出现网站崩溃、登录缓慢、无法支付、扣钱不出票等严重问题。
- 2012年9月20日，由于正处中秋和“十一”黄金周，网站日点击量达到14.9亿次，发售客票超过当年春运最高值，出现网络拥堵、重复排队等现象。

案例3 - 证券系统

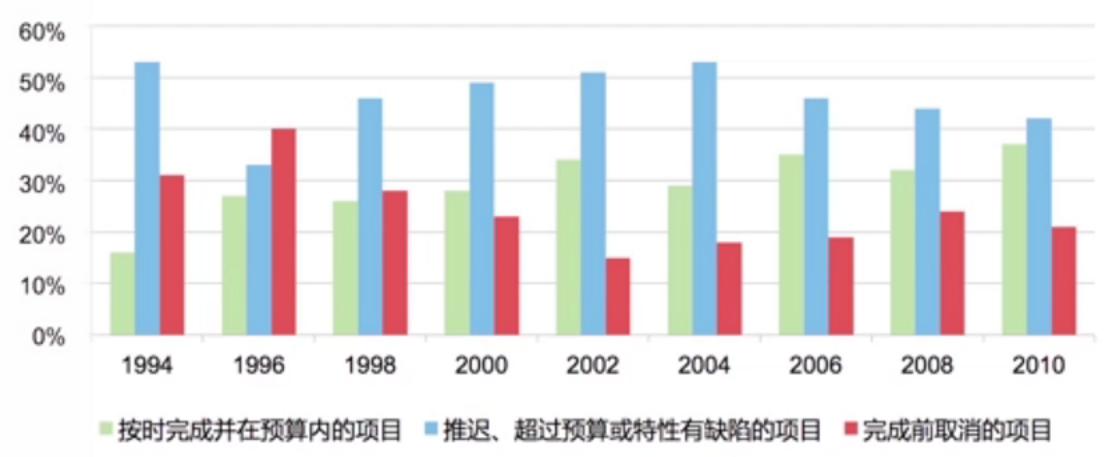
一个 **bug** 引发证券业 **400** 亿天价损失。

日本瑞穗证券“乌龙指”操盘手敲乱日本股市

程序员有风险、工程需严谨！

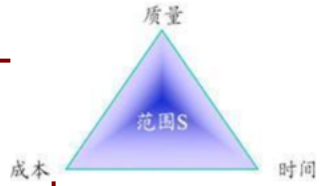
软件项目完成度调查

- 美国standish集团专门从事跟踪IT项目成功或失败的权威机构，在它每年的CHAOS Report报告中给出了IT项目相关调查数据结果。
- 统计结果显示，按时完成并在预算内的项目，只有不到三分之一。



特 征

软件开发周期大大超过规定日期;
软件开发成本严重超标;
软件质量难于保证;
软件难于维护。



可将软件危机归结为质量(Q)、成本(C)和时间(T)等问题

原 因

软件本身的复杂性
软件产品的特殊性
人们认识的局限性

原因

软件产品特殊性

一致性

软件必须遵从人为的惯例并适应已有的技术和系统。

1. 软件必须遵循各种接口、协议和标准
2. 有些情况下，兼容性是软件开发的目标

软件需要随接口的不同而改变，随时间的推移而变化。

可变性

软件产品扎根于文化的母体中，如各种应用、用户、自然及社会规律、计算机硬件等，后者持续不断地变化着，这些变化无情地强迫着软件随之变化。

人们总是认为软件是容易修改的，但忽视了修改所带来的副作用，不断的修改最终导致软件的退化。

不可见性

软件是不可见的和无法可视化的。

定义“需要做什么”成为软件开发的根本问题。人们一直试图使用不同的技术进行软件可视化：控制流程、数据流、依赖关系、UML、……

软件的复杂性

例如：Windows 95有1000万行代码，

Windows 2000有5000万行代码

Exchange2000和 Windows2000开发人员结构

	Exchange 2000	Windows 2000
项目经理	25人	约250人
开发人员	140人	约1700人
测试人员	350人	约3200人

原因

认识的局限性

- 忽视需求分析
- 软件开发 = 程序编写
- 软件产品的质量缺少度量的标准
- 轻视软件维护

认识的局限性

流传的“真正”的程序员据说是这样的

- 真正的程序员没有进度表，只有讨好领导的马屁精才有进度表，真正的程序员会让领导提心吊胆。
- 真正的程序员不写使用说明书，用户应当自己去猜想程序的功能。
- 真正的程序员几乎不写代码的注释，如果注释很难写，它理所当然也很难读。
- 真正的程序员不画流程图，原始人和文盲才会干这事。
- 真正的程序员不看参考手册，新手和胆小鬼才会看。
- 真正的程序员不写文档也不需要文档，只有看不懂程序的笨蛋才用文档。
- 真正的程序员认为自己比用户更明白用户需要什么。
- 真正的程序员不接受团队开发的理念，除非他自己是头头
- 真正的程序员的程序不会在第一次就正确运行，
  但是他们愿意守着机器进行若干个30小时的调试改错。
- 真正的程序员不会在上午9:00到下午5:00之间工作，
  如果你看到他在上午9:00工作，这表明他从昨晚一直干到现在。
- ……

认识的局限性


《神秘的程序员们》-32
临时方案
http://c.xiqiao.info
0
这部分很重要，要好好写。
来不及了，这次出的是demo，用不了多久这部分就会重写的。
1
这命名不太规范啊
没时间了，差不多就行了，肯定要重写的
2
反正要重写，也不用注释得太仔细，deadline 要紧。
3
见鬼，这段真是写得太烂了，一定要重构。
4
老板，这部分我们需要重构一下。
5
这么关键的时候，哪有功夫弄这个，现在不是好好地跑着呢嘛。这个版本更新完了再说。
6
这是核心代码，又一直在运行中，注释和文档都不清楚，谁敢重写啊……
这部分不是早就应该重构了吗
两年后...
mm_segment_t old_fs = get_fs();
ssize_t result, copied = 0;
if (oldval && oldlen) {
loff_t pos = 0;
char buf[15], *nodep;
unsigned long area, node;
__le16 dnaddr;
//这部分必须在2010年5月前重构
set_fs(KERNEL_DS);
result = vfs_read(file, buf, sizeof(buf) - 1, &pos);
set_fs(old_fs);

原因

认识的局限性


《神秘的程序员们》
高效程序
http://c.xiqiao.info
C++
Asp
Python
Ruby
Perl
JAVA
C#
Php
0
实习生
能帮我把这份数据和标准样本做个对比，找出来异常部分吗？我急要，最好能今天下班之前让它运行起来。
1
三个小时后……
你怎么还没搞定？
标准样本库太大了，我想需要一个缓存以便让它加载快一些。
2
载入标准样本需要多久？
15分钟，所以我觉得需要缓存。
3
这个程序每周只需运行一次，且标准样本不会增加，15分钟不算什么。
但我想让程序高效一点。
好吧。。。。
4
这小子悲剧了，我先弄个自己用着。
搞定了，可以下班了。
1个小时后……
5
第二天
我的cache好像有bug弄错了数据。
6
第三天
我的cache不是线程安全的，所以程序死锁了。
7
第四天
我的cache好像比没有还慢，一定是什么地方错了
8
第五天
9
你还在写那个缓存吗？
是啊。。。我只是想让它更高效一点。
根据用途，每种程序都有适当的复杂级别，如果你无法判断，过度努力，只会比低于这种级别更不专业。

伴随着软件危机的存在引入软件工程

- 软件的概念、特点及分类
- 软件危机
- **软件工程**
- 软件生命期模型
- 软件工程的目标与原则
- 软件工程的教学意义

软件工程的历史

早期

- 面向批处理
- 有限的分布

第二阶段

- 多用户
- 实时
- 数据库
- 软件产品

第三阶段

- 强大的桌面系统
- 面向对象
- 低成本硬件

第四阶段

- 专家系统
- 人工神经网络技术
- 并行计算
- 网络计算机

- 软件开发没有方法可循
- 软件设计是在开发人员头脑中完成的隐藏过程
- 60世纪中期的软件危机

史前时代

1956 - 1967

- 1968年提出“软件工程”
- 结构化开发方法
- 瀑布式软件生命周期模型成为典型

“软件工程”学科诞生

瀑布过程模型

1968 - 1982

- 面向对象开发方法
- 软件过程改进运动
- CMM/ISO9000/SPICE等质量标准体系

质量标准体系

1983 - 1995

- 敏捷开发方法流行
- 更紧密的团队协作
- 有效应对需求变化
- 快速交付高质量软件
- 迭代和增量开发过程

20世纪90年代至今

软件工程的历史

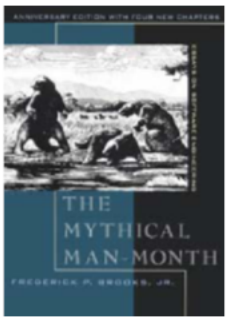
软件工程的诞生

1968 年10 月，NATO 科学委员会在德国的加尔密斯举行会议讨论大型软件项目的若干问题。提出了“软件工程”和“软件危机”的术语。

一本经典的著作

《人月神话(Mythical Man Month)》

- Brooks 在1975 年出版
- 描写了大型软件开发中的许多关键问题
- Brooks 法则：
  向进度落后的项目中增加人手，
  只会使进度更加落后。people ≠time



定义

1968年

Fritz Bauer的定义：“建立并使用完善的工程化原则，以较经济的手段获得能在实际机器上有效运行的可靠软件的一系列方法。”

1983年

IEEE的软件工程定义：“软件工程是开发、运行、维护和修复软件的系统方法。”

1993年

IEEE的一个更加综合的定义：“将系统化的、规范的、可度量的方法应用于软件的开发、运行和维护的过程，即将工程化应用于软件中。”

定义

软件工程是应用计算机科学、数学及管理科学等原理开发软件的工程。它借鉴传统工程的原则、方法，以提高质量，降低成本为目的。

软件工程所包含的内容**不是一成不变**的，随着人们对软件系统的研制开发和生产的理解。应用**发展的眼光**看待它。

工程的含义

是将理论和所学的知识应用于实践的科学，以便经济有效地解决实际问题。

花园小道

树上小屋

加法程序

VS.

规模上的差异

高速公路

摩天大楼

医疗档案系统

手工：小规模的设计与建造

- 简单问题与单一目标
- 个人控制与个人技能

工程：大规模的设计与建造

- 复杂问题与目标分解
- 多人参与，需要考虑运营、管理、成本、质量控制、安全等

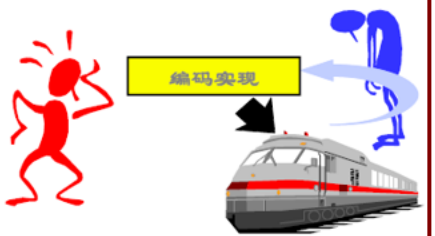
工程的特征

- 团队协同工作
  - 注重训练有素，并以团队的形式进行有效的工作。
- 角色分工
  - 多重角色：研究、开发、设计、生产、测试、构造、实施、管理以及其他诸如销售、咨询和教学等。
- 最佳实践
  - 通过专业团体不断地开发和确认工程原则、标准和实践。
- 强调重用
  - 工程师应该重用设计和设计制品。

定义

工程化的软件开发

只有编码的开发过程



- 明确的工作步骤
- 详细具体的规范化文档
- 明确的质量评价标准

本质特征

- 软件工程关注于**大型程序**的构造。
- 软件工程的中心课题是**控制复杂性**。
- 软件经常变化。
- 开发软件的效率非常重要。
- **和谐地合作**是开发软件的关键。
- 软件必须有效地支持它的**用户**。
- 在软件工程领域中是由具有一种文化背景的人替具有另一种文化背景的人完成一些工作。

软件工程要素

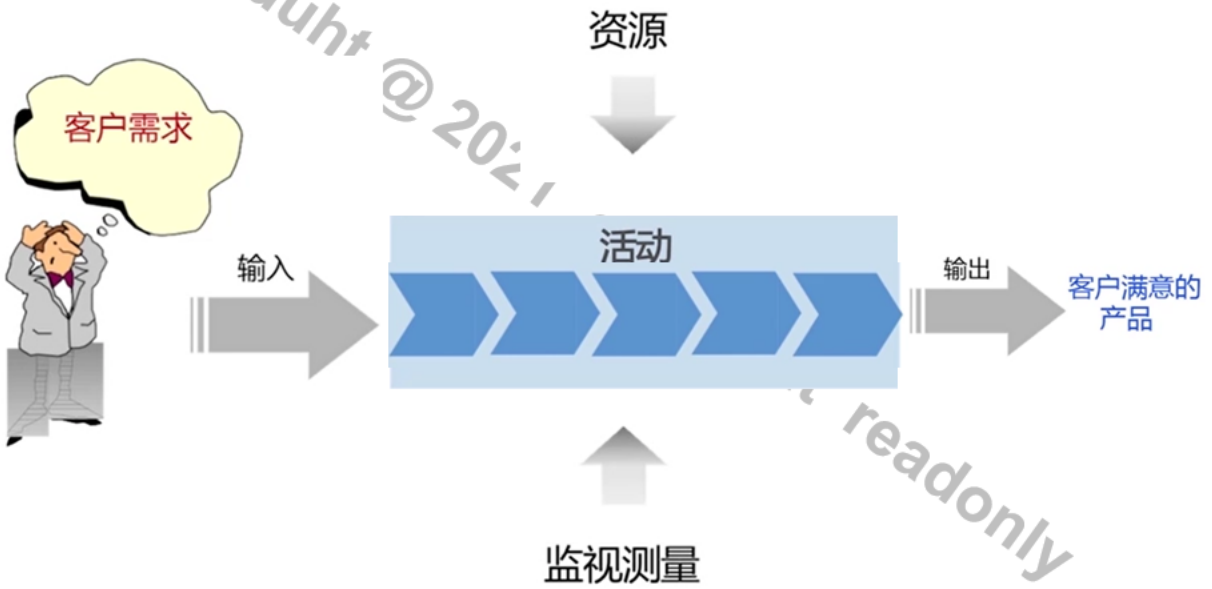
软件工程以关注软件质量为目标，包括过程、方法和工具三个要素。

过程 支持软件生命周期的所有活动。

方法 为软件开发过程提供“如何做”的技术。

工具 为软件开发方法提供自动的或半自动的软件支撑环境。

理解过程



过程与方法

软件开发管理

（软件项目管理计划、软件配置管理计划、软件质量保证计划、评审记录……）

工具

需求开发	软件设计	软件构造	软件测试	软件维护	开发管理
• 软件建模工具 • 数据库设计工具		• 程序编辑器 • 程序编译器 • 程序解释器 • 程序调试器 • 集成开发环境	• 单元测试工具 • 静态分析工具 • 自动化测试工具 • 性能测试工具 • 缺陷跟踪工具	• 代码重构工具 • 逆向工程工具	• 需求管理工具 • 项目管理工具 • 配置管理工具 • 测试管理工具
ENTERPRISE ARCHITECT Microsoft Visio MySQL Workbench	Notepad++ Visual Studio eclipse	Selenium xUnit BugFree Loadrunner JMeter	Visual Studio eclipse ALTOVA umodel	Trello tower REDMINE git GitHub	

质量

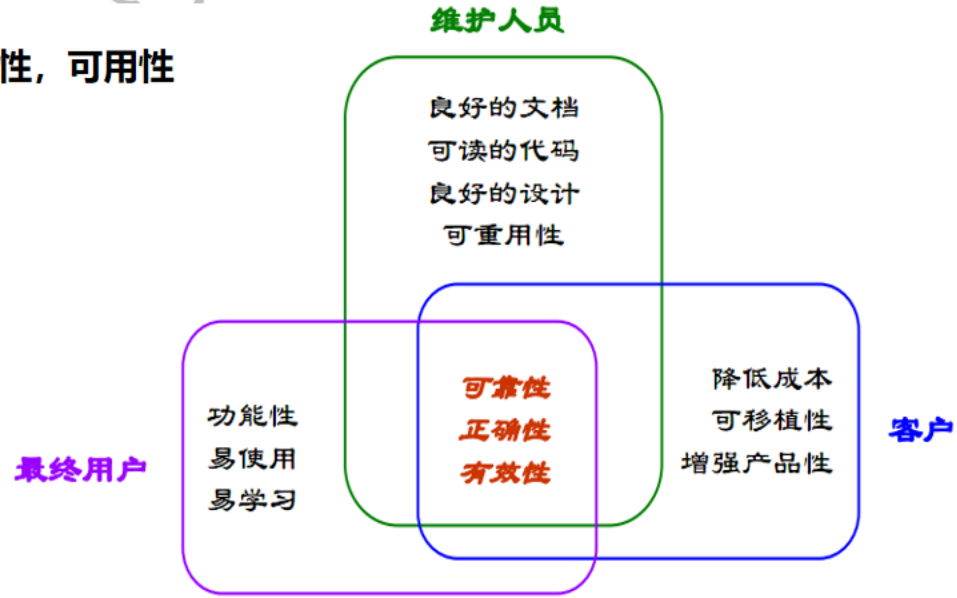
- 你同意以下说法吗？为什么？

  “运行正确的软件就是高质量的软件。”

- 软件除了提供用户所需的功能以外，还应该具有一系列反映质量的属性，包括可维护性、可依赖性、有效性和可用性等。
  - 可维护性：软件必须能够不断进化以满足客户的需求变化
  - 可依赖性：软件必须是可靠的、保密的、安全的
  - 有效性：软件不应该浪费内存和处理器等系统资源
  - 可用性：软件必须是可用的，用户可以很方便地使用

理解质量

- **可靠性**：正确性和健壮
  正确性和对异常值边界值的处理能力
- **可维护性**
  可读性，可修改性，可测试性，完整性
- **可理解性**
  简单性，清晰性，可用性
- **效率**



ISO9126 质量模型

知识结构

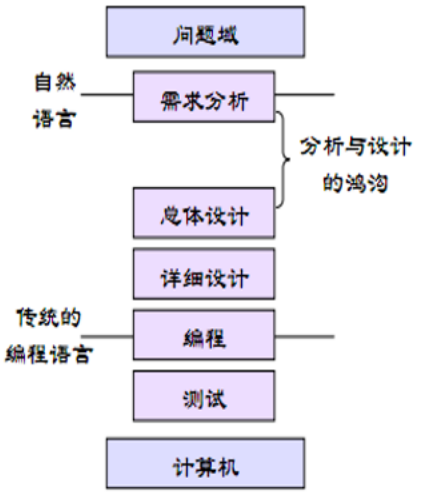
- 2001年5月ISO/IEC JTC1发布了《SWEBOK指南V0.95（试用版）》，即Guide to the Software Engineering Body of Knowledge。
- SWEBOK把软件工程学科的主体知识分为10个知识领域。这10个领域包括：

① 软件需求　② 软件设计
③ 软件构造　④ 软件测试
⑤ 软件维护　⑥ 软件配置管理
⑦ 软件工程管理　⑧ 软件工程过程
⑨ 软件工程工具和方法　⑩ 软件质量

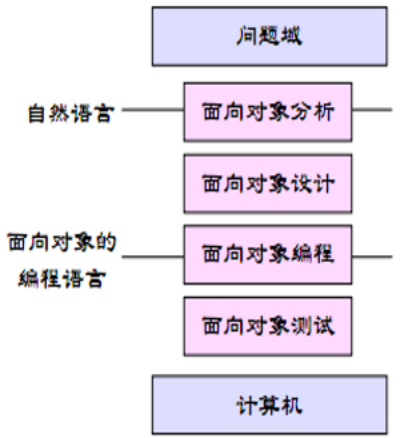
知识结构


软件工程知识体系（SWEBOK）
软件需求
需求工程过程
需求获取
需求分析
需求规格说明
需求验证
需求管理
软件设计
基本概念
关键问题
结构与体系结构
质量分析与评价
设计符号
策略与方法
软件构造
降低复杂性
预知变化性
结构化验证
使用外部标准
软件测试
基本概念和定义
测试级别
测试技术
测试相关度量
测试过程管理
软件维护
基本概念
维护过程
关键问题
维护技术
软件配置管理
配置过程管理
配置标识
配置控制
配置状态报告
配置审计
软件发布管理与交付
软件工程管理
组织管理
过程／项目管理
软件工程度量
软件工程过程
软件过程概念
过程基础设施
过程度量
过程定义
定性过程分析
过程实施与变更
软件工程工具与方法
软件工具
软件方法
软件质量
软件质量概念
SQA和V&V目的与计划
SQA和V&V活动与技术
适用于SQA和V&V的度量

传统的方法学



面向对象方法学



面临的挑战

遗留系统的问题

遗留系统是指那些过时或存在问题的计算机系统，通常是许多年以前开发的。

挑战：既要以合理的成本维护和更新系统，又要能够继承系统中重要的商业信息和服务。

异构系统的问题

网络环境下包含不同的硬件平台和软件系统。

挑战：需要提出新的开发技术，能够使所开发的软件系统运行在不同的硬件平台和系统环境下。

面临的挑战

高可信软件开发的要求

软件的重要作用要求正确性、可靠性、安全性等可信性质。

挑战：如何在软件的开发和运行中保证其具有高可信的性质。

软件开发方式的变化

网络时代带来的冲击：开源软件开发技术；Web 工程。

挑战：研究分布式的软件体系结构和开发模式，探索与之相适应的软件工程策略。

Google搜索

社交网络分析系统

物联网数据处理系统

填空题 1分

设置

软件工程概念的提出是为了解决（ [填空1] ）

作答

正常使用填空题需3.0以上版本雨课堂

设置

下面的（ ）是正确的。

A 运行正确的软件就是高质量的软件。

B 软件质量是在开发过程中逐渐构建起来的。

C 软件产品质量越高越好，最理想的情况是达到“零缺陷”。

D 软件质量是由产品的功能、性能、易用性等外在特性决定的。

提交

设置

软件可靠性是指

A 软件产品提供了让用户产生惊喜的特性

B 软件实现了用户需要的功能和性能

C 软件在规定时间和条件下无故障持续运行

D 软件符合国家或行业的相关标准

提交

- 软件的概念、特点及分类
- 软件危机
- 软件工程
- **软件生命期模型**
- 软件工程的目标与原则
- 软件工程的教学意义

软件生存期

软件有一个孕育、诞生、成长、成熟、衰亡的生存过程。这个过程即为软件的生存周期。

软件生存周期

软件定义、软件开发及软件运行维护。

软件生存周期是软件工程思想的具体化，是跨越软件生存周期的系统开发、运行、维护所实施的全部活动和任务的过程框架。

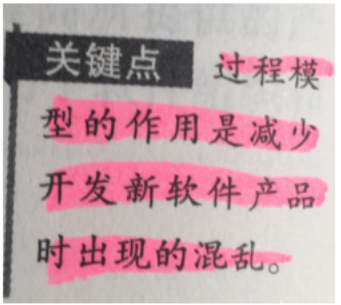
软件生存期

软件定义	软件开发	软件运行维护
问题定义：这是软件生存期的第一个阶段，主要任务是弄清用户要计算机解决的问题是什么。 **可行性研究**：任务是为提出的问题寻求一种至数种在技术上可行、且在经济上有较高效益的解决方案。	**需求分析**：弄清用户对软件系统的全部需求，主要是确定目标系统必须具备哪些功能。 **总体设计**：设计软件的结构，即确定程序由哪些模块组成以及模块间的关系。 **详细设计**：针对单个模块的设计。 **编码**：按照选定的语言，把模块的过程性描述翻译为源程序。 **测试**：通过各种类型的测试(及相应的调试)使软件达到预定的要求。	**软件安装运行** **维护**：软件使用后的不足点的修正、性能和其它属性的提高；另外为适应环境变化的修正。

结构分析设计过程总结

阶段	关键问题	结束标准
问题定义	问题是什么？	规模和目标的报告书
可行性研究	有可行解吗？	系统的高层逻辑模型：数据流图 成本/效益分析
需求分析	系统必须作什么？	系统的逻辑模型：数据流图，数据字典，算法描述。
总体设计	概括地说应该如何解决这些问题？	可能的解决方法：系统流程图 推荐的系统结构：层次图或结构图
详细设计	怎样具体地实现这个系统？	编码规格说明： HIPO图或PDL
编码和单元测试	正确的程序模块	源程序清单；单元测试方案和结果
综合测试	符合要求的软件	综合测试方案和结果：完整一致的软件配置
维护	持久地满足用户需要的软件	完整准确的维护记录

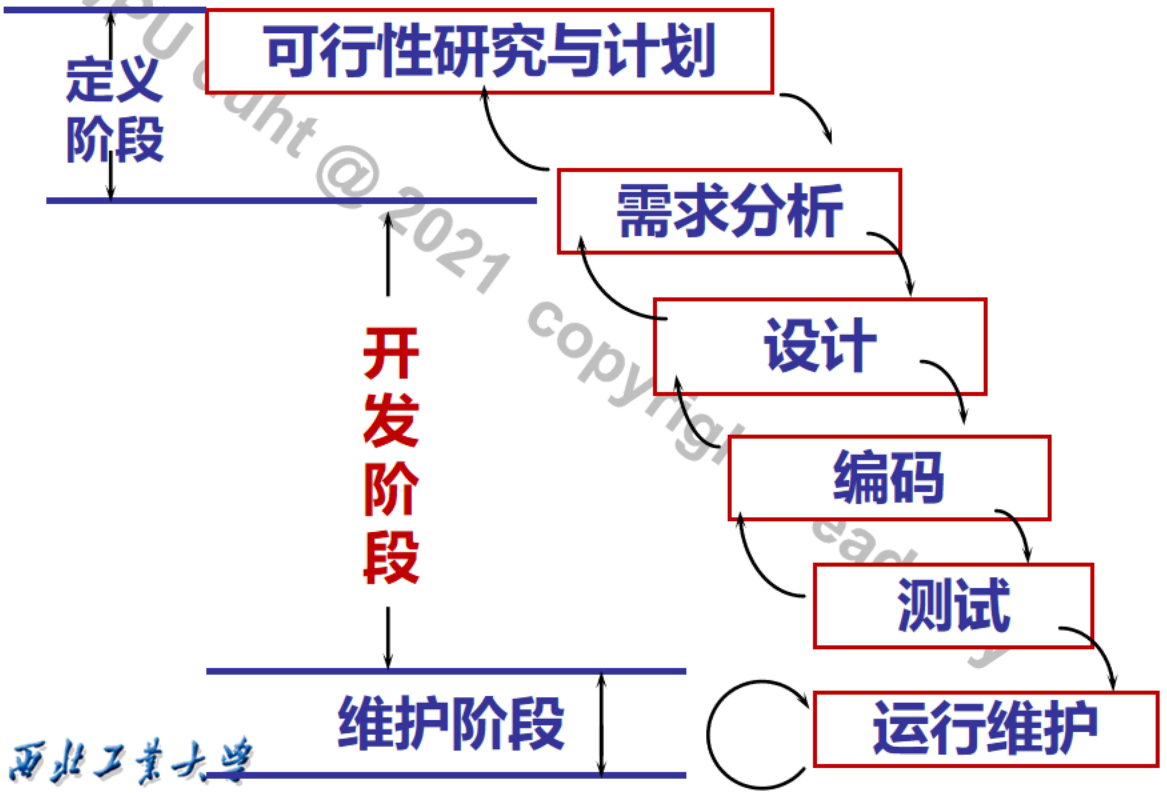
软件生命期模型



常用的软件生命期模型有：

- 瀑布模型
- 原型模型
- 螺旋模型
- 增量模型

瀑布模型



瀑布模型的特点

1) 各项活动按自上而下，相互衔接的固定次序，如同瀑布逐级下落。

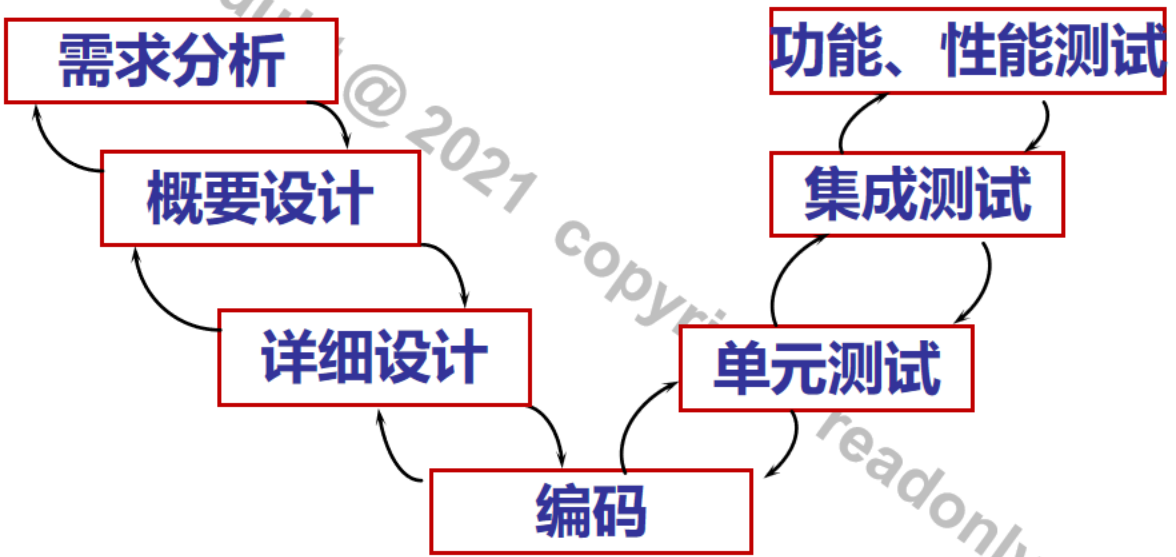
   每项活动均处于一个质量环（**输入-处理-输出-评审**）中。

2) 阶段间具有顺序性和依赖性。

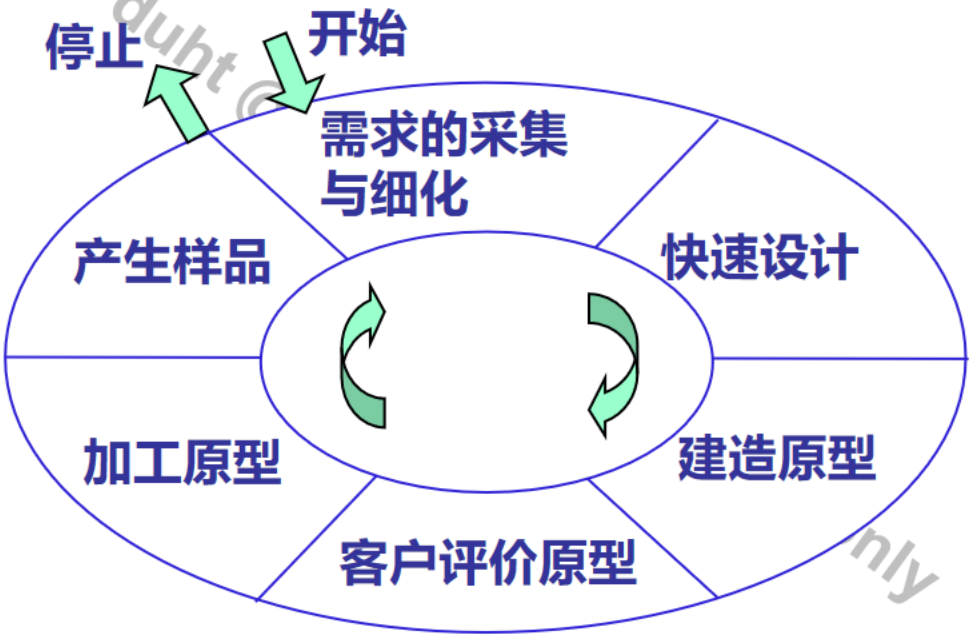
3) 推迟实现的观点。

4) 每个阶段必须完成规定的文档；每个阶段结束前完成文档审查。

瀑布模型-U型演化



原型模型



原型模型的特点

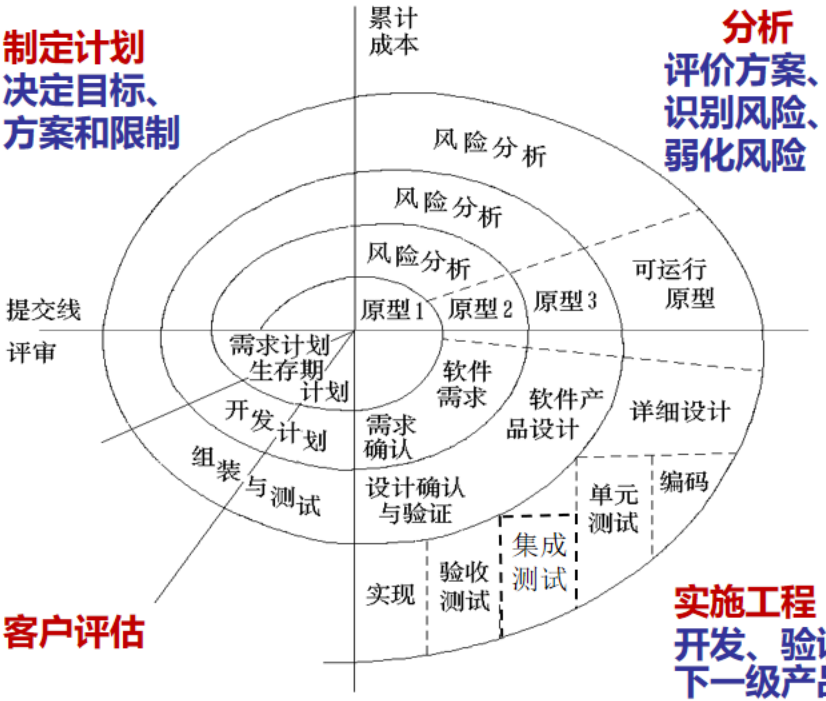
1)原型模型是迭代的。因为软件与所有的复杂系统一样，必须经过不断演化才能完善。

2)原型模型先开发一个“原型”软件，完成部分主要功能，展示给用户并征求意见，然后逐步完善，最终获得满意的软件产品。

3)业务和产品需求在变化中，采用线性开发方式是不实际的。

4)快速实现和提交一个有限的版本，可以应付市场竞争的压力。

螺旋模型


制定计划
决定目标、
方案和限制
累计
成本
分析
评价方案、
识别风险、
弱化风险
风险分析
风险分析
风险分析
可运行
原型
原型1
原型2
原型3
提交线
评审
需求计划
生存期
计划
开发计划
组装
与测试
需求
确认
软件
需求
软件产
品设计
详细设计
设计确认
与验证
单元
测试
编码
集成
测试
验收
测试
实现
客户评估
实施工程
开发、验证、
下一级产品

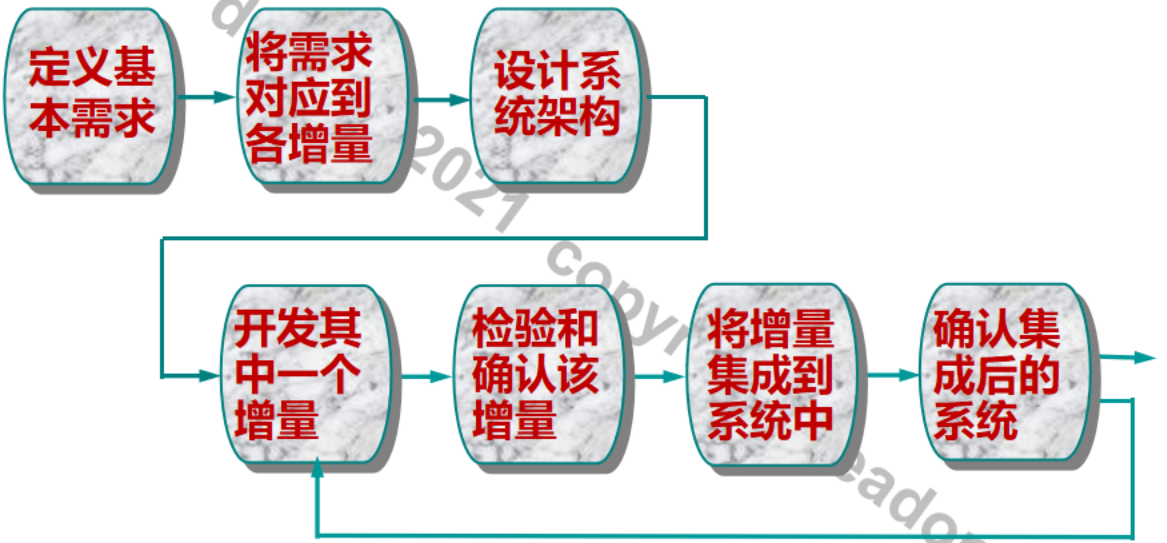
螺旋模型的特点

1)螺旋模型将瀑布模型与原型模型结合起来，并且加入两种模型均忽略了的风险分析。

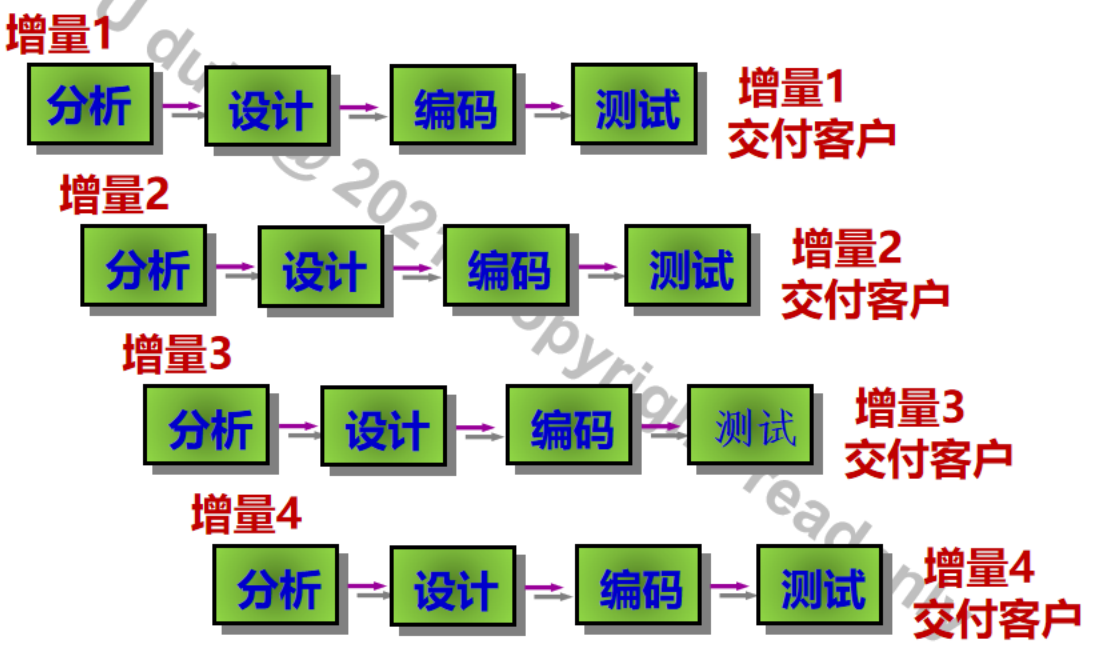
2)螺旋模型沿着螺线旋转，自内向外每旋转一圈便开发出更完善的一个新版本。

- 制定计划 确定软件目标，选定实施方案，弄清项目开发的限制条件；
- 风险分析 分析所选方案，考虑如何识别和消除风险；
- 实施工程 实施软件开发
- 客户评估 评价开发，提出修正建议。

增量模型



增量模型



增量模型

增量模型的特点

1)增量模型是迭代和演进的过程。

2)增量模型把软件产品分解成一系列的增量构件，在增量开发迭代中逐步加入。

3)每个构件由多个相互作用的模块构成，并且能够完成特定的功能。

4)早先完成的增量可以为后期的增量提供服务。

复合模型

迭代增量开发

增量开发

分几个阶段开发和集成系统，每一个阶段完成整个系统的可执行的一个子集。

迭代开发

重复的循环开发过程来提炼和详细实现整个系统。

迭代增量开发

结合迭代与增量，每一次都可以有整个方案的一部分可执行的程序，并不断的提炼和详细化。

敏捷方法

产生背景：

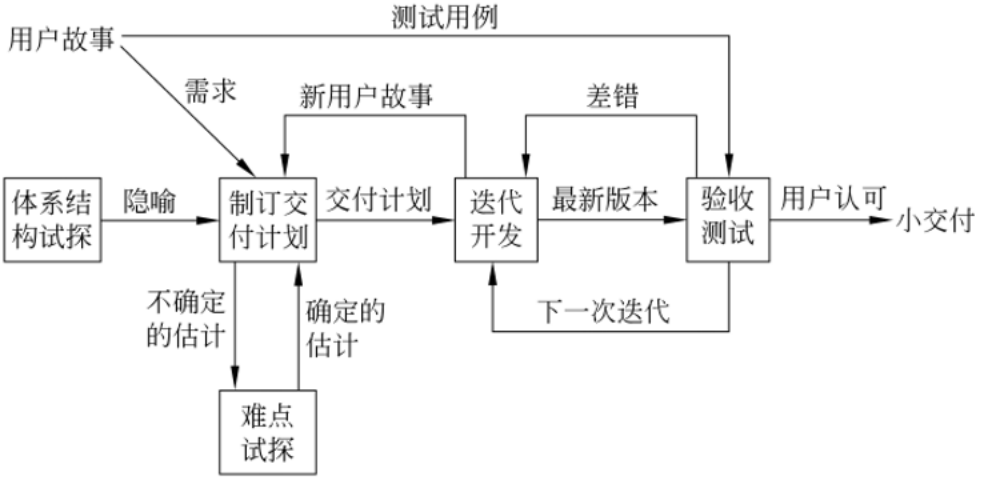
- 快速的市场进入时间，要求高生产率
- 快速变化的需求
- 快速发展的技术

价值观：

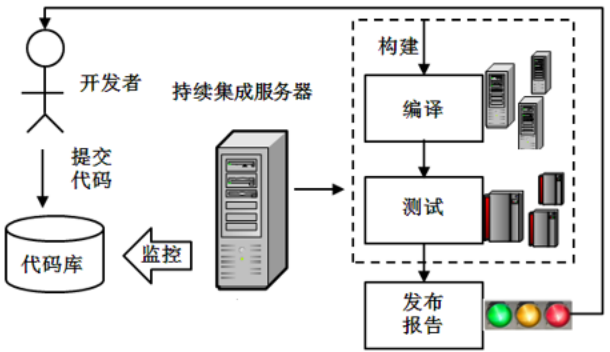
1. 个体和交互胜过过程和工具
2. 可以工作的软件胜过面面俱到的文档
3. 客户合作胜过合同谈判
4. 响应变化胜过遵循计划

极限编程（XP）

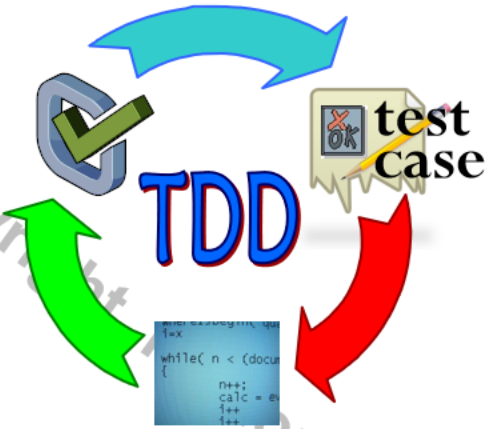
极限编程（eXtreme Programming,XP）是敏捷过程中最富盛名的一个，其名称中“极限”二字的含义是指把好的开发实践运用到极致。
目前，极限编程已经成为一种典型的开发方法，广泛应用于需求模糊且经常改变的场合。



极限编程（XP）- 最佳实践示例



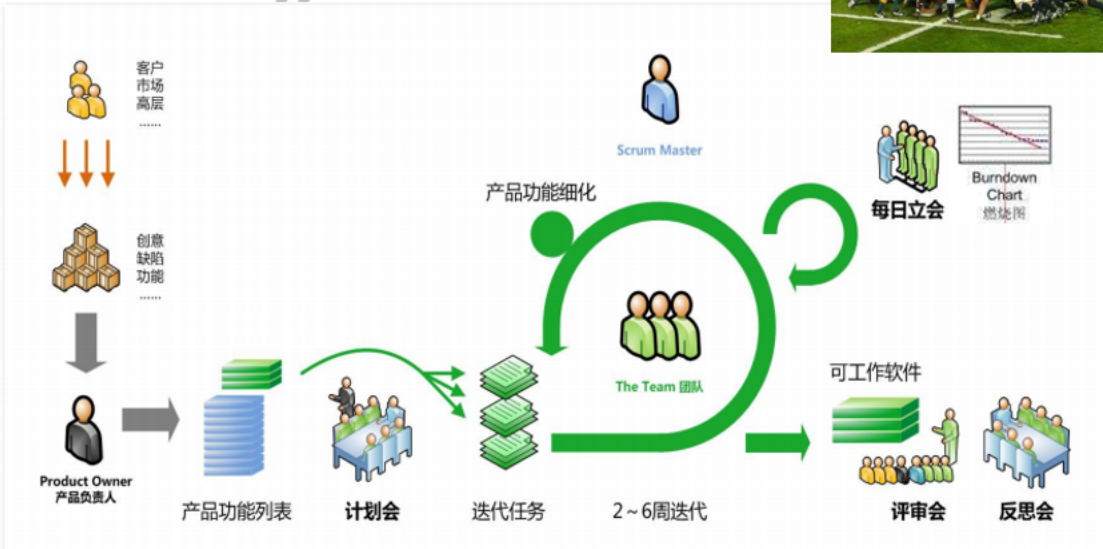
持续集成



测试驱动

敏捷方法

Scrum


客户
市场
高层
.....
创意
缺陷
功能
.....
Product Owner
产品负责人
产品功能列表
计划会
迭代任务
Scrum Master
产品功能细化
The Team 团队
2～6周迭代
每日立会
Burndown
Chart
燃烧图
可工作软件
评审会
反思会

- 软件的概念、特点及分类
- 软件危机
- 软件工程
- 软件生命期模型
- **软件工程的目标与原则**
- 软件工程的教学意义

软件工程的目标

- 软件工程需要解决的问题：软件成本、软件可靠性、软件维护、软件生产率和软件复用。
- 软件工程需要达到的基本目标：
  - 付出较低的开发成本
  - 达到要求的软件功能
  - 取得较好的软件性能
  - 开发的软件易于移植
  - 需要较低的维护费用
  - 能按时完成开发，及时交付使用

花费，时间，品质平衡关系

软件工程的原则—复用

原则1—复用

软件开发过程中必须遵循

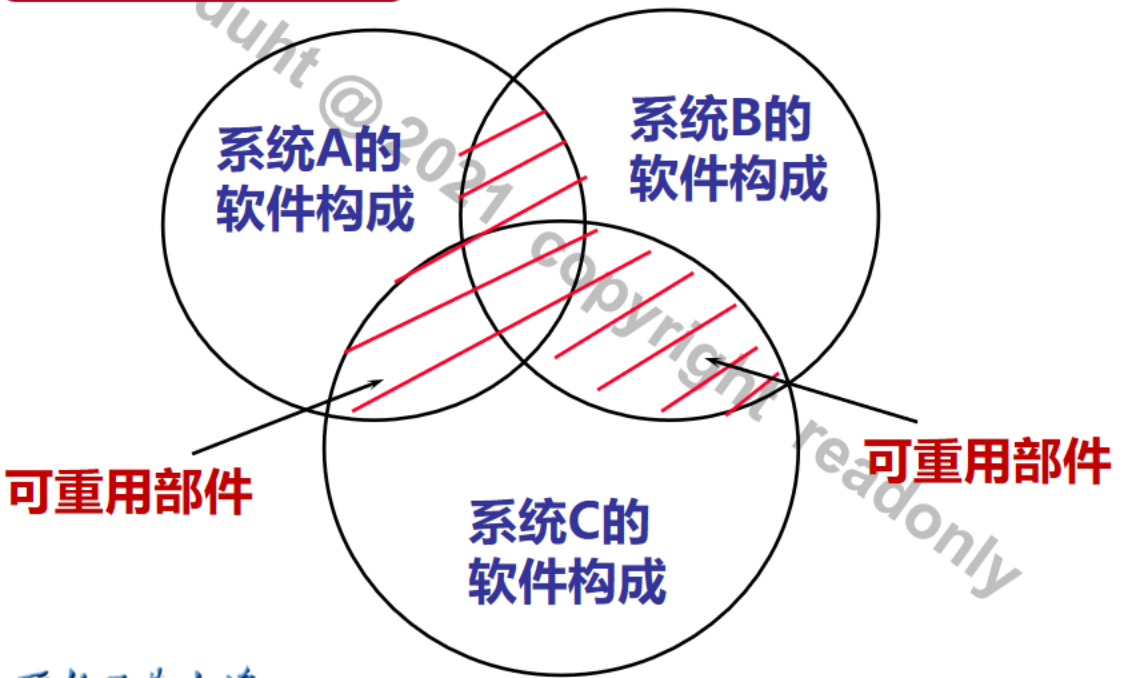
复用就是利用某些已开发的、对建立新系统有用的软件元素来生成新的软件系统。

软件复用就是直接使用已有的**软构**件(具有一定集成度并可以重复使用的软件组成单元)，即可组装(或加以合理修改)成新的系统，而可以不必每次从零做起。

复用就是指“**利用现成的东西**”，文人称之为“**拿来主义**”。复用不仅要使自己拿来方便，还要让别人拿去方便，是“**拿来拿去主义**”。

软件工程的原则—复用

可重用部件组装模型

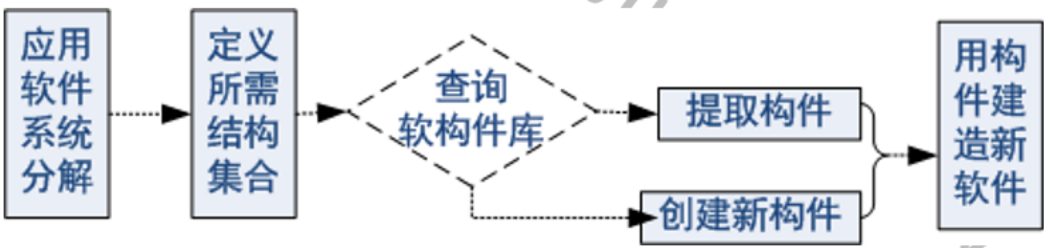


软件工程的原则—复用

原则1—复用

复用的优点：
降低了软件的成本、提高了生产率而且新系统也具有较高的质量。

使用构件开发软件的过程：



软件工程的原则—分而治之

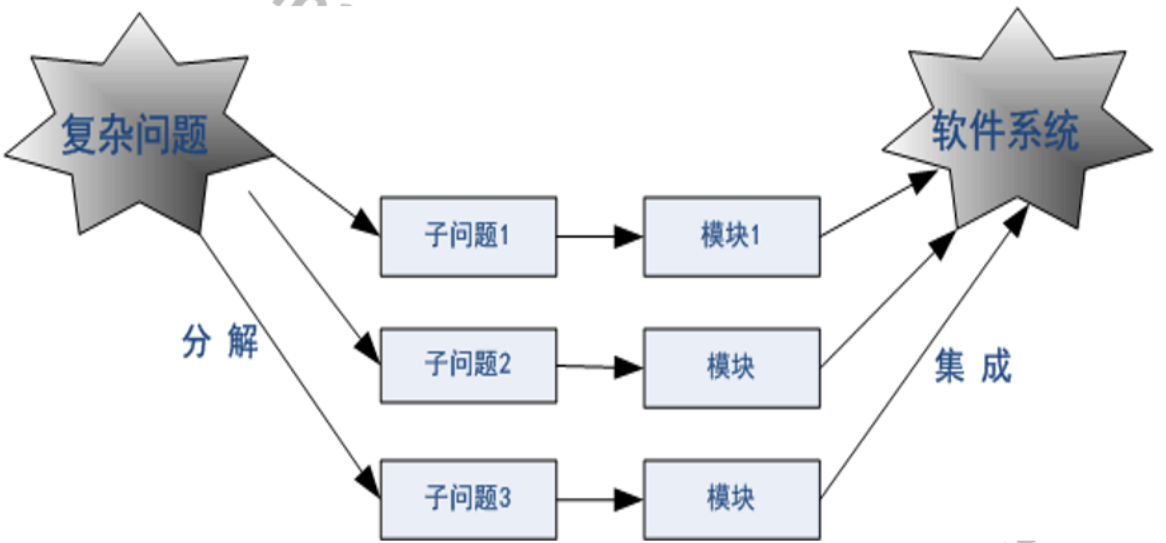
原则2—分而治之

分而治之是指把一个复杂的问题**分解**成若干个**简单**的问题，然后逐个解决。诸如软件的体系结构设计、模块化设计都是分而治之的具体表现。

1) **抽象与自顶向下、逐层细化**：采用分层抽象的方法，有效控制软件开发的复杂性。
2) **模块化**：把问题分解为若干较小的较易解决的模块，有助于信息隐蔽和抽象。
3) **信息隐蔽和数据封装**：将模块中的软件设计决策封装在模块内部，使得模块实现与使用分离，有助于控制修改局部化。

软件工程的原则—分而治之

原则2—分而治之



软件工程的原则—优化—折衷

原则3—优化—折衷

优化：为了提高软件质量，程序员会不断改进软件中的算法，数据结构和程序组织。

优化工作是十分复杂的，有时很难实现所有目标的优化，这时就需要“**折中**”策略。软件的折衷策略是指通过协调各个质量因素，实现整体质量的最优。

运行速度要高　内存占用要少

用户界面友好

三维效果逼真

软件工程过程管理原则

软件工程过程管理原则

1）按软件生存期分阶段**制定计划并认真实施**
把整个软件开发过程视为一项工程，把工程划分为若干阶段，分别制定每个阶段的计划，逐个实施。

2）**坚持进行阶段评审**　前一阶段的结果将成为下一阶段的依据。坚持阶段的评审才能保证错误不传播到下一阶段。

3）**使用现代程序设计技术**　先进的程序设计技术带来的是生产率和质量的提高。使用合适的开发模式和工具可以有效地建立功能强大的系统。

软件工程过程管理原则

软件工程过程管理原则

4）**明确责任**　使得工作结果能够得到清楚的审查。开发组织严格划分责任并制定产品的标准，使得每个成员的工作有据。

5）**用人少而精**　开发组织不在人多，在于每个人的技能适合要求。同时用人少而精，可减少沟通路径，提高生产率。

6）**不断改进开发过程**　在开发的过程中不断总结经验，改进开发的组织和过程，有效地通过过程质量的改进提高软件产品的质量。

- 软件的概念、特点及分类
- 软件危机
- 软件工程
- 软件生命期模型
- 软件工程的目标与原则
- **软件工程的教学意义**

软件工程的教学意义

软件在当今社会中发挥着重要的作用

- 社会经济的发展依赖于软件
- 更多的系统需要软件控制，软件质量和成本成为关键因素

软件工程是一个正在兴起的年轻学科

- 工业界形成了CMMI和ISO9000系列标准
- IEEE 提出了软件工程知识体系
- IEEE 提出了软件工程本科教程，成为独立学科

软件工程的教学意义

“软件工程”课程与其它软件专业课的区别

(1) 立足于系统的整体。

(2) 讲授系统分析、系统设计、测试及维护的理论和方法。

(3) 构筑一个软件系统，实践软件开发全过程。

软件工程的教学意义

软件工程

- 是一门学科，一种科学理论来指导软件系统开发，标准化，自动化的过程
- 考虑如何分解一个系统，以便各人分工开发；考虑如何说明每个部分的规格要求；怎样才能易于维护

编　程

- 单纯的代码编写
- 是软件工程发展的前身
- 是软件工程中占据很少时间和空间的一部分

什么是软件？软件的特点？
产生软件危机的原因？
什么是软件工程？
软件工程基本概念，关注焦点是什么？
什么是软件生命期？
软件生命期模型各自的特点？

课后思考题

1) 有人认为“软件工程过于耗费时间，并且妨碍开发人员的编程效率。”你是否认同这种观点？请阐述理由。

2）查阅资料，试论述敏捷开发产生的背景、特征和目标。比较常见敏捷开发模式（XP，Scrum等）的异同。

3）试举例说明一个你将开发的产品（要简述产品的功能和面向的客户等信息），你如何选择开发模型，如果选择敏捷模式，如何考虑迭代的次序。

谢 谢！