软件工程导论 -
Software Engineering Introduction

第1章 软件工程概述

2024.4月更新

软件工程导论 -
Software Engineering Introduction
第1章 软件工程概述

计算机学院

刘文洁

liuwenjie@nwpu.edu.cn

第一章 软件危机与软件工程 目录

- 1.1软件危机 ★
  - 1.1.1 软件危机的介绍
  - 1.1.2 产生软件危机的原因
  - 1.1.3 消除软件危机的途径
- 1.2软件工程
  - 1.2.1 软件工程的介绍
  - 1.2.2 软件工程的基本原理 ★
  - 1.2.3 软件工程方法学
- 1.3软件生命周期
- 1.4软件过程
  - 1.4.1 瀑布模型 ★
  - 1.4.2 快速原型模型 ★
  - 1.4.3 增量模型 ★
  - 1.4.4 螺旋模型
  - 1.4.5 敏捷过程与极限编程 ★

第一章 软件危机与软件工程

1.1 软件危机 -

1.1.1 软件危机的介绍

软件工程的发展

早期	第二阶段	第三阶段	第四阶段
●面向批处理	●多用户	●强大的桌面系统	●专家系统
●有限的分布	●实时	●面向对象	●人工神经网络技术
	●数据库	●低成本硬件	●并行计算
	●软件产品		●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
- 20世纪60年代中期的软件危机

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

软件工程的发展

- 控制机器（1956－1967）
  - 软件工程的史前时代
  - 批处理系统
  - 汇编语言，FORTRAN语言，COBOL语言
  - 已经认识到软件开发不仅是编码
- 控制过程（1968－1982）
  - 软件工程成为一个研究领域
  - 出现软件产品的定价和独立的软件产业
  - 结构化开发技术
  - Pascal语言，C语言，面向对象编程语言
  - 形成软件生命周期的概念，以瀑布模型为典型

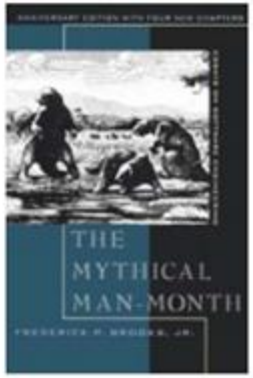
软件工程的发展

北约

- 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



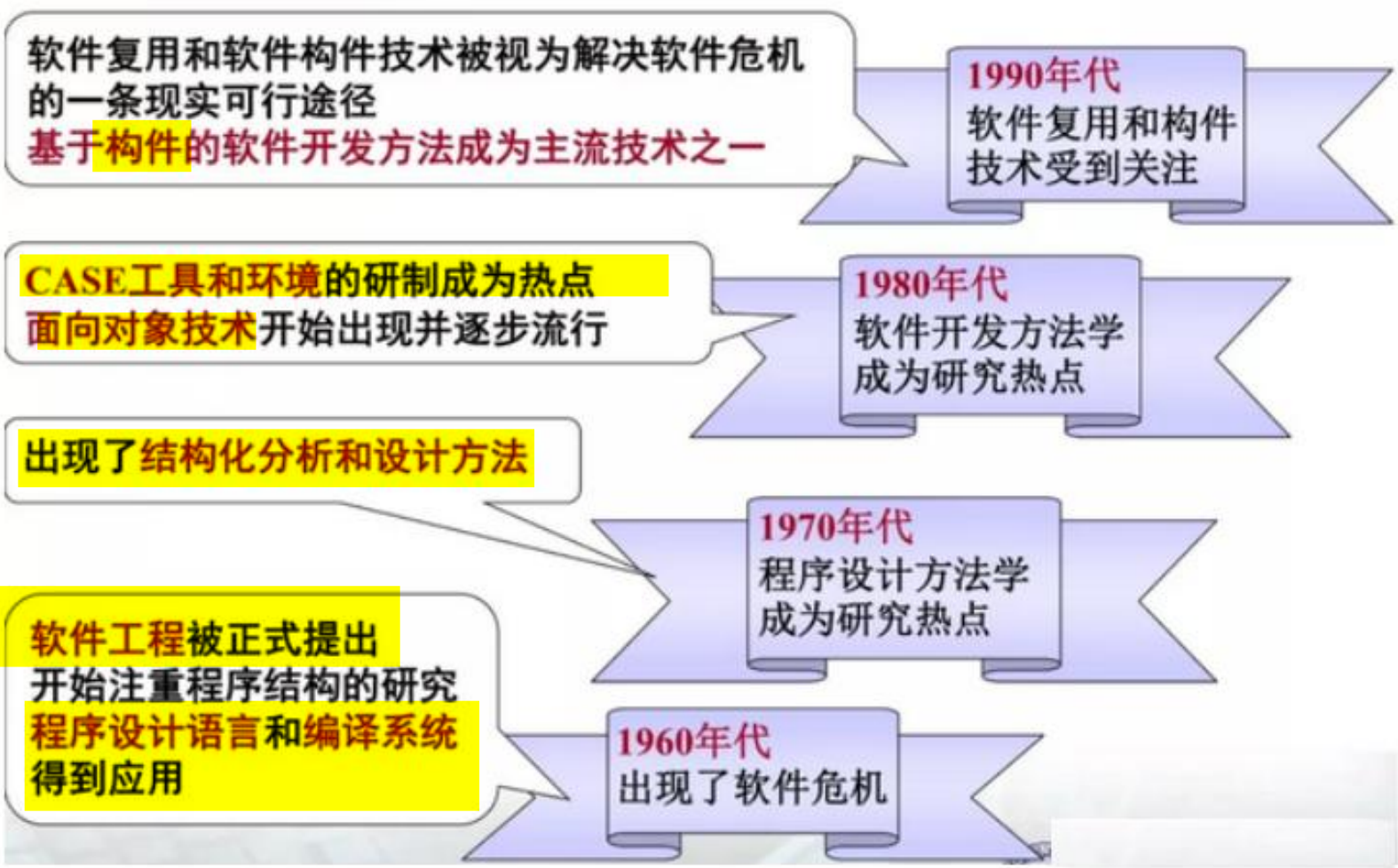
软件工程的发展

- 控制复杂性（1983－1995）
  - PC 时代的到来
  - CASE 工具的开发
  - 原型化开发技术，CMM，软件过程改进活动
  - 面向对象开发技术
  - Objective C，C++，JAVA 语言
- 1996－现在
  - 分布式计算
  - 应用领域的扩展
  - 网络环境的软件工程
  - 面向对象技术的进一步发展（设计模式、组件、中间件）

Fortran	COBOL	Basic	Pascal	C	C++	VB	delphi	java	C#	PHP	Python
1954	1960	1964	1968	1970	1983	1991	1993	1995	2000	1995	1991

编程语言的进化史其实就是软件工程思想的演化史

总结：软件工程的发展历程



软件工程面临的挑战

- 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

软件开发的发展过程总结

软件数量多
规模大

软件成本高
质量低

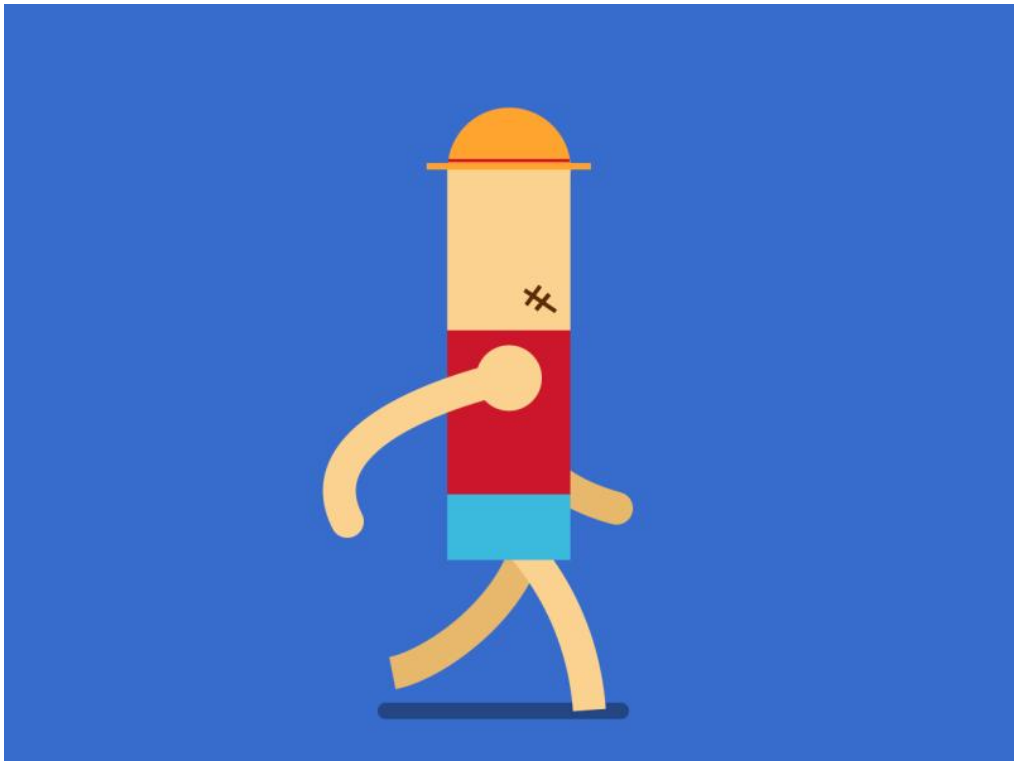
个体化软件开发方法 ⟶ 软件维护困难

⇒ 软件危机 ⇒ 软件工程

总结视频：软件工程的发展

- 总体分为4个发展阶段

做个小练习



填空题 3分

设置

按照程序设计语言的发展过程，程序设计语言分为三类，分别是：

A 机器语言；

B 汇编语言；

C 高级语言；

提交

软件的定义

- **程序、软件与软件产品**

独唱-->小合唱-->合唱-->万人大合唱

简单程序	较复杂程序	软件

★ - **软件包括**

程序：按事先设计的功能和性能需求执行的指令序列

数据：是程序能正常操纵信息的数据结构

文档：与程序开发、维护和使用有关的图文材料

回忆“程序”的定义

数据结构 + 算法 = 程序

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);
    });
}
```

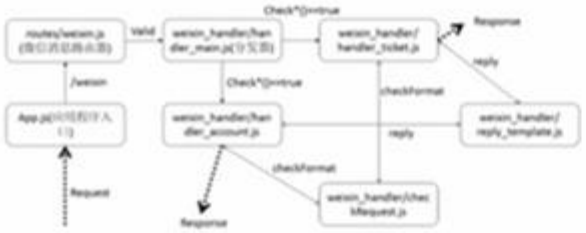
数据是软件的核心

字段名	类型	说明
_id	ObjectId	每个数据库表项的唯一序列号，自动生成
name	String	每个活动的活动名称
key	String	每个活动的活动代称，文字抢票的活动关键字，有唯一性
place	String	活动地点
description	String	活动描述
pic_url	String	活动
start_time	Int	活动 月1日经过的毫秒数
end_time	Int	活动 月1日经过的毫秒数
book_start	Int	抢票 月1日经过的毫秒数
book_end	Int	抢票 月1日经过的毫秒数
need_seat	Int	活动分配座位的方式，0表示此活动无需选座或选区（大礼堂），1表示此活动需要选区来非配座位（综体），2表示此活动需要选具体座位（新清华学堂）
remain_tickets	Int	活动抢票剩余的总票数

配套的文档是软件的一部分

	按顺序判断并执行第一个满足检验器条件的执行器	
weixin_handler/checkRequest.js	微信消息检查，对指定类型的消息（如菜单触发、订阅）时做出特殊判定	检测两种事件的函数
weixin_handler/reply_template.js	微信返回消息模板，根据微信官方API 制定，可以得到返回文字消息和图文消息的格式	返回文字消息以及返回图文消息的函数（图文消息超过 10 条会截剪）
weixin_handler/handler_ticket.js	微信票务相关事件处理内核，用于处理抢票、退票以及查票事件	对应三种事件的检验器和执行器，从而暴露给 handler_main 调用
weixin_handler/handler_account.js	微信账户相关事件处理程序，用于处理绑定、解绑、帮助、以及抢啥。（为调试使用增加了可删除的“获取实验账号”功能）	对应事件的检验器以及执行器，从而暴露给 handler_main 调用

模块间依赖图如下：



4.2.2 微信前端网页渲染后台

	的 post 请求
Route/ ticketsinfo.js	票务详情页[illegible]后台[illegible]参[illegible]数中的票号信息读出并传递给前端

4.2.3 后台管理模块

后台管理后端(routes/user_manage.js)：

查找数据库中所有暂存、已发布的活动并将它们的信息返回给活动列表页进行渲染；导出 xlsx，根据前端发送的活动 id 查找数据库中对应活动的所有有效票，并将每张票的持有者学号、支付状态、入场状态、座位号等信息汇总生成 xlsx 表格返回前端；删除活动，根据活动 id 将某个活动删除（status 改为-1）；根据活动 id 查找 activity、seat 数据库，并将具体信息返回前端给活动详情页面进行渲染；活动详情写入数据库，根据前端 post 的内容，检查活动内容是否完整，格式是否正确，内容是否符合时间段的特点（某些信息在某些活动阶段里不允许更改），然后选择是否录入数据库，并将成功或失败信息返回前端。

后端对要录入数据库的内容进行了严格的审查，相当于前端检查过一次的内容，后端也进行一次审核（双审核），这样费周章的进行两次检验有必要吗？其实还是很有必要的，因为前端是很不靠谱的，用户可以改，甚至可以直接给后端 post 一些杂乱的内容，如果后端不做进一步的检查那么数据库很可能就会发生错误；即使用户并没有想恶意攻击，但也会有一些意外情况，比如用户的系统时间不正确，而前端是通过用户设备的时间来判断一个活动处于哪个阶段，进而判断哪些字段可以或不允许被更改，一旦用户时间和服务器时间不匹配，后端又不做检查，就可能发生一些严重错误——明明抢票开始了，总票数不允许更改，但用户系统时间是抢票前，导致该活动总票数在抢票中被改大或改小。因此后端的这个部分写了很长来进行严密的内容审查。

软件的分类1 ★

1、按软件的功能进行划分

- 系统软件
- 应用软件
- 支撑软件

能与计算机硬件紧密配合在一起，使计算机系统各个部件、相关的软件和数据协调、高效率地工作的软件。如操作系统、数据库管理系统、设备驱动程序以及通信处理程序等。系统软件是计算机系统中必不可少的一个组成部分。

是在特定的领域内开发，为特定目的服务的一类软件。如，CAD计算机辅助制造、计算机辅助教学系统、停车场管理系统，OA系统、ERP系统（财务、进销存、生产、人力资源）。

支撑各种软件的开发与维护的软件,又称为软件开发环境。它主要包括环境数据库、各种接口软件和工具组。

软件的分类2 ★

2、按软件的规模进行划分

按开发软件所需的人力、时间以及完成的源代码行数。

类别	参加人数	研制期限	产品规模(源代码行数)
微型	1	1-4周	约500行
小型	1	1-6周	约2000行
中型	2-5	1-2年	5000-50000行
大型	5-20	2-3年	5万-10万行
甚大型	100-1000	4-5年	100万行
极大型	2000-5000	5-10年	1000万行

软件的分类3 ★

3、按软件开发划分

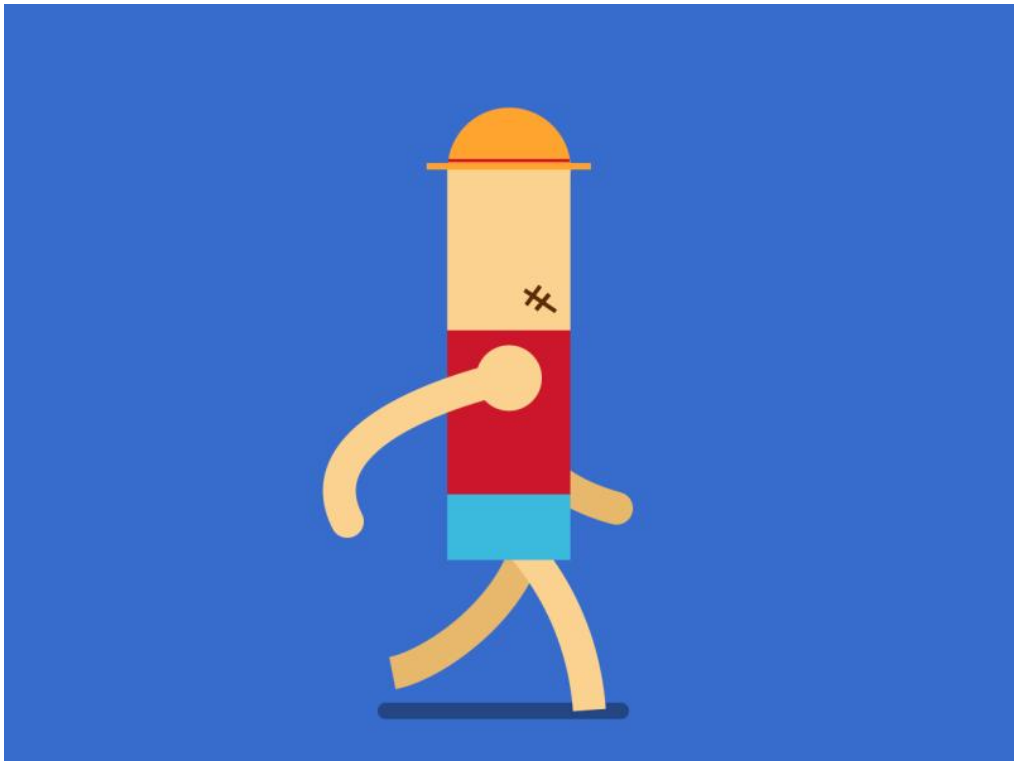
软件项目开发

也称定制软件，是受某个特定客户（或少数客户）的委托，由一个或多个软件开发机构根据合同的约定进行开发。

软件产品开发

可以被广大用户直接使用的软件系统。例如用友/金蝶的财务软件、QQ、微信等。

做个小练习



多选题 1分

设置

下列软件属于系统软件的是（）

好像是选A,B

A 操作系统

B 编译器

C 中间件

D 浏览器

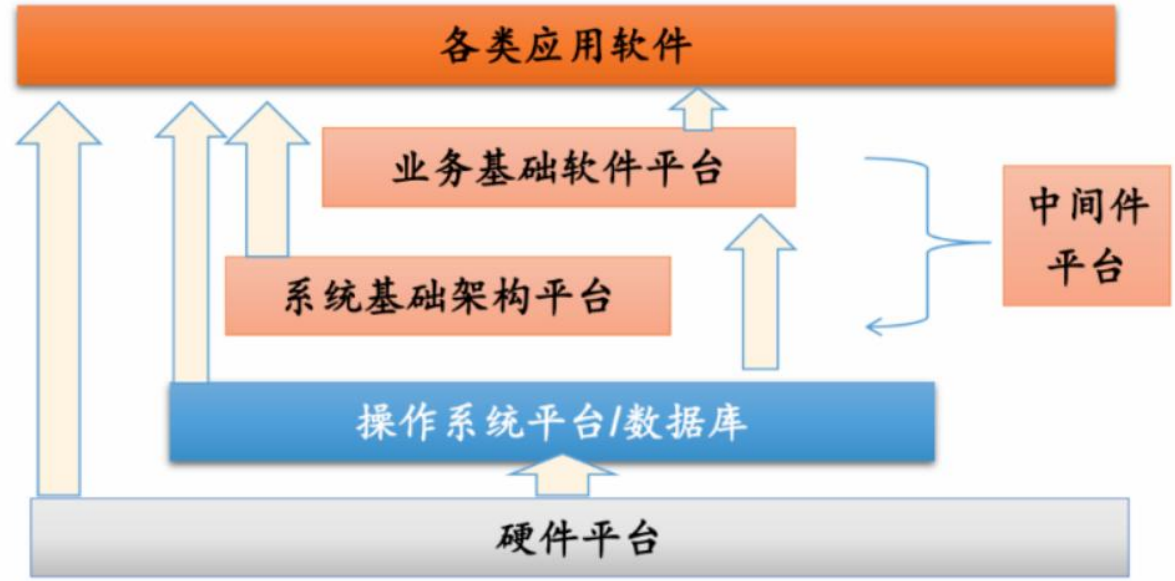
提交

中间件：

- **定义：**独立的系统级软件，连接操作系统层和应用程序层，将不同操作系统提供应用的接口标准化，协议统一化，屏蔽具体操作的细节。

- **所处位置：**中间件在操作系统、网络和数据库之上，应用软件的下层。

- **中间件功能：**
  1）通信支持
  2）应用支持 API、库和服务
  3）公共服务



安全管理、事务处理、负载均衡、日志管理等

中间件：分类

- 中间件可以分为基础中间件、集成中间件和行业领域应用平台。
- 三大基础中间件：交易中间件、消息中间件和应用服务器中间件。

多选题 1分

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

多选题 1分

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

软件危机的定义 ★

- **定义**　　考试的时候写一句话！一定要背！

  计算机软件的开发和维护过程所遇到的一系列严重问题。

- **表现**
  - （1）对软件开发成本和进度的估算很不准确
  - （2）用户很不满意
  - （3）质量很不可靠
  - （4）没有适当的文档难于维护
  - （5）软件成本比重上升
  - （6）供不应求：软件开发生产率跟不上计算机应用迅速深入的趋势

软件危机的一句话概括

在计算机软件的开发和维护过程中所遇到的一系列严重问题。

软件问题的实例1

- 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。

软件问题的实例2 - Windows Vista

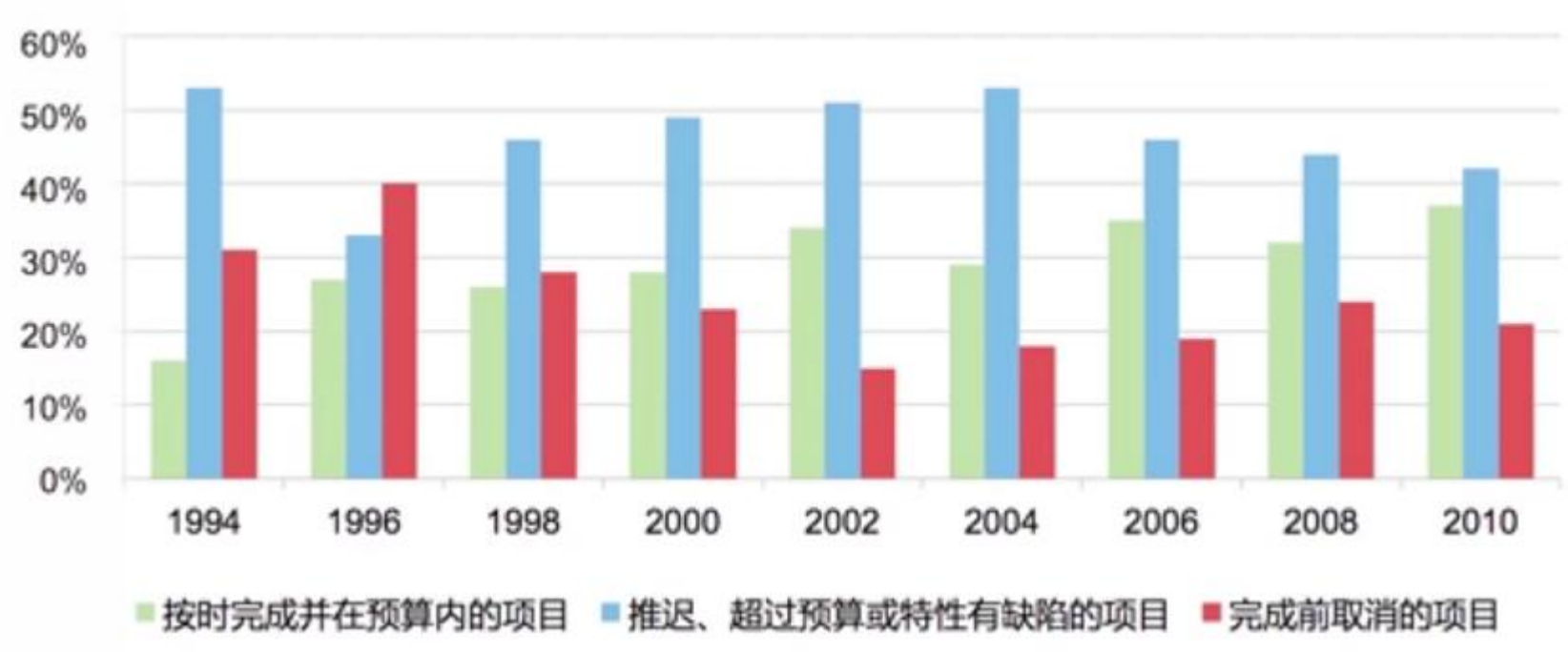
- 该系统从2001年开始研发，整个过程**历时5年半**，先后有**9000位**开发人员投入其中，**耗资60亿美元**，代码规模超过**5000万行**。
- 按照微软公司最初的计划，该系统面世时间应该在2003年，之后推迟到2004年下半年再到2005年初，最终在取消一些高级功能后于2006年11月正式发布。
- 由于整个系统过于庞杂，其开发管理相当混乱，以致于很多时间用在互相沟通和重新决定上。
- 从Vista Beta1进入公开测试以来，程序错误总数已经超过2万个，这其中还不包括微软内部未公开的一些错误。

软件问题的实例3 - 12306网络购票系统

- 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。

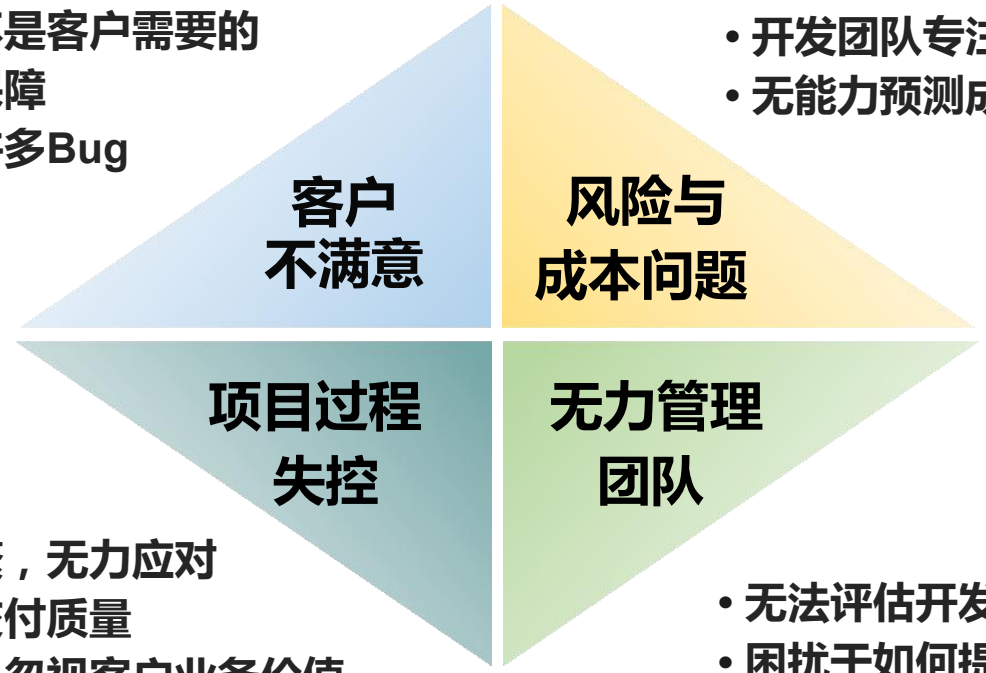
美国standish集团的调查报告

- 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。



总结：软件开发面临的挑战

- 交付的许多功能不是客户需要的
- 交付的日期没有保障
- 客户使用时发现许多Bug



- 开发团队专注技术，忽视风险
- 无能力预测成本，导致预算超支

- 客户需求变化频繁，无力应对
- 无法预见软件的交付质量
- 对流程盲目遵从，忽视客户业务价值

- 无法评估开发人员能力及工作进度
- 困扰于如何提升团队的能力与效率

第一章 软件危机与软件工程

1.1 软件危机 -

1.1.2 产生软件危机的原因

软件 VS 硬件的生产

一架客机由数百万个单独的部件组成，需要上千人组装，但通常都能够按时按预算交付使用。

微软于1989年11月发布的Word最初版本，花费了55人年，大约有249，000行源代码，却晚了4年交付使用。

软件的特点 ★

（1）软件是一种逻辑实体。

（2）软件的开发，是人的智力的高度发挥，而不是传统意义上的硬件制造。

（3）软件维护与硬件的维修有着本质的差别。

（4）软件的开发和运行常常受到计算机系统的限制，对计算机系统有着不同程度的依赖性。

（5）软件的开发至今尚未完全摆脱手工艺的开发方式，使软件的开发效率受到很大限制。

（6）软件的开发是一个复杂的过程。

（7）软件的成本非常高昂。

（8）相当多的软件工作涉及到社会因素

软件的本质特性

软件具有的复杂性、一致性、可变性、不可见性等特征，使得软件开发过程变得难于控制，开发团队如同在焦油坑中挣扎的巨兽。

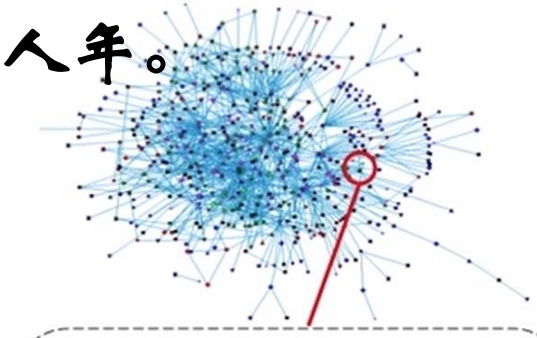
- 复杂性（*Complexity*）
- 一致性（*Conformity*）
- 可变性（*Changeability*）
- 不可见性（*Invisibility*）

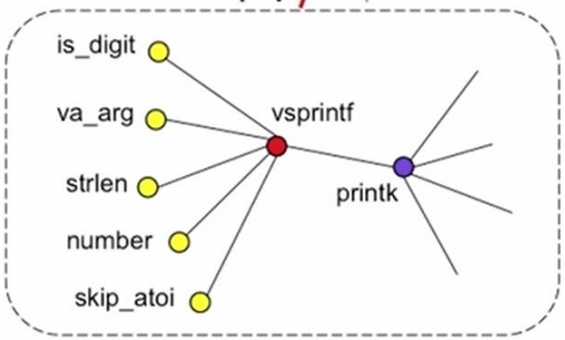
软件的本质特性（1）- 复杂性

- 复杂性
  - 软件在规模上可能比任何由人类创造的其他实体都要复杂，复杂性是软件的本质特性。
  - 软件的复杂性是必要属性
    - 大量的组合状态
    - 丰富的结构和相互依赖性
    - 良好的接口用以封装内部的复杂性
  - 开发问题也会增加复杂性
    - 高效率的代码通常是复杂的
    - 重用通用化的组件意味着复杂的状态连接
    - 复杂的代码难以维护，导致设计上的更复杂

软件复杂性的例子

- 1 Windows程序超过1000万行
- 2 WWMCCS（军事和控制）花费3500多人拖了几年，交付后发显出100个错误。最后失败。
- 3 城市银行出纳机程序7.8万行，150人年。

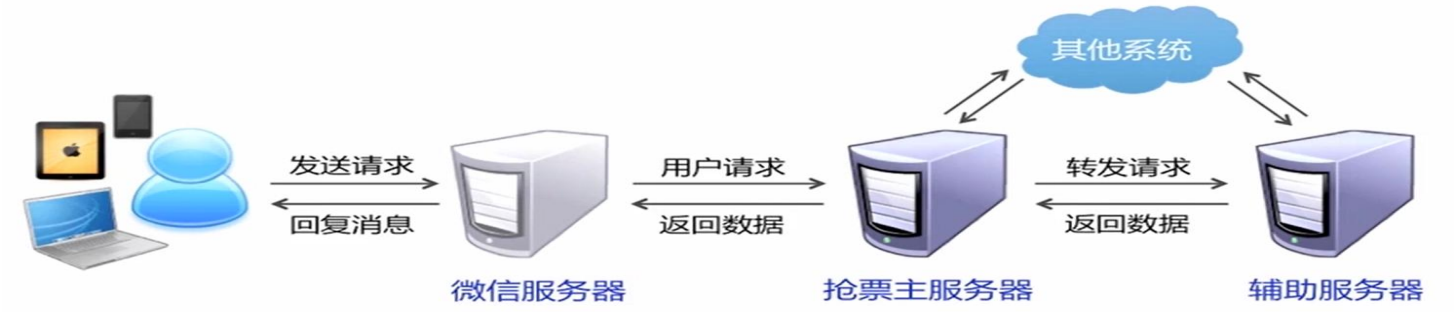




软件的本质特性（2） - 一致性

- 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
    - 软件必须遵循各种接口、协议和标准
    - 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，而这些变化是不同的人设计的结果。

  - 许多复杂性来自保持与其他接口的一致，对软件的任何再设计，都无法简化这些复杂特性。



软件的本质特性（3） - 可变性

- 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。
  - 所有成功的软件都会发生变更！
    - 当人们发现软件很有用时，会在原有应用范围的边界，或者在超越边界的情况下使用软件；
    - 功能扩展的压力主要来自那些喜欢基本功能，又对软件提出了很多新用法的用户们。

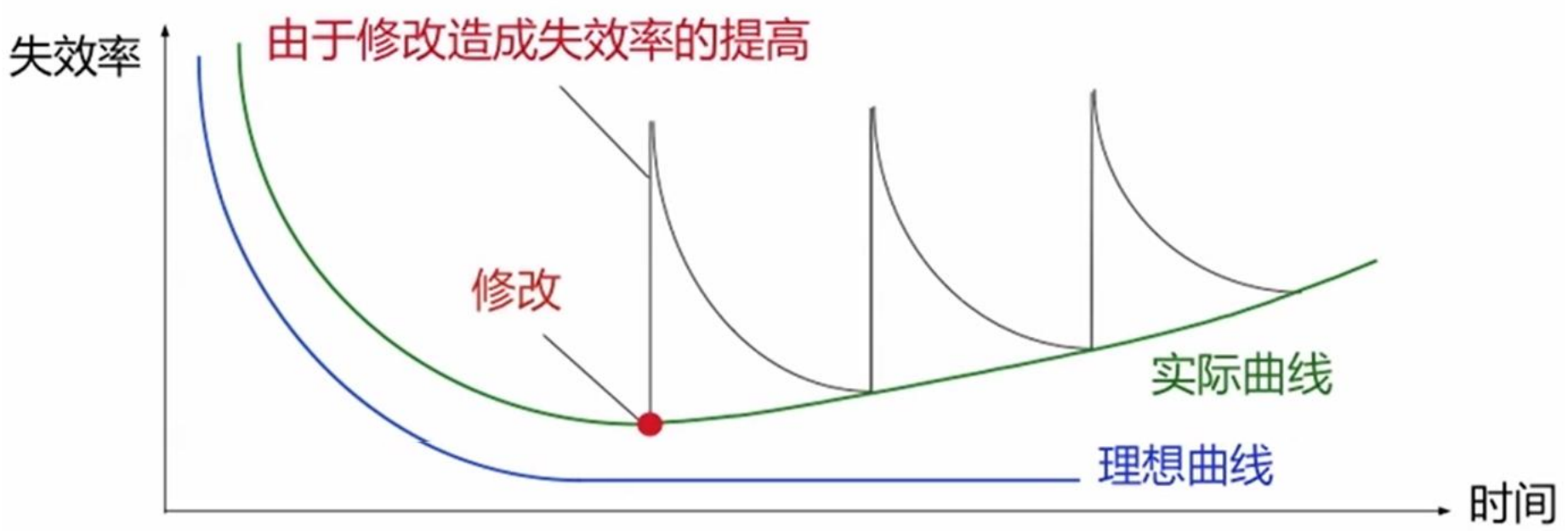
例：微信软件的演化


• 2011.1.21，微信1.0发布
（文字短信、QQ好友、头像）
• 2011.2，支持多人会话
• 2011.3，增加图片功能
• 2011.5.10，微信2.0发布
（语音对讲、QQ邮箱提醒）
• 2012.4.19，微信4.0发布
（朋友圈、开放API、地理位置）
• 2012.7.19，微信4.2发布
（视频、网页版、朋友圈回复）
• 2012.8.18，微信公众平台开放
• 2012.9.25，微信4.3发布
（摇一摇传图、解绑、扫一扫）
• 2014.1.26，微信5.2发布
（聊天记录搜索、银行卡新增服务、图片墙、@提醒）
• 2014.5.8，微信5.3发布
（面对面建群、收藏内容添加标签、外文翻译）
• 2014.8.14，微信5.4发布
2011
2012
2013
2014
2010.11.20
微信正式立项
• 2011.8，微信2.5发布
（查看附近的人、语音记事本）
• 2011.10.1，微信3.0发布
（摇一摇、漂流瓶）
• 2011.12，微信3.5发布
（二维码扫描、自定义表情）
• 2013.1.21，微信4.5发布
（语音聊天室、摇一摇搜歌、语音提醒、位置导航）
• 2013.8.5，微信5.0发布
（折叠公众账号、游戏中心、新版扫一扫、微信支付）

软件的本质特性（3） - 可变性

- 可变性（续）

软件环境的变化以及**用户新的要求**，必须要多次修改（维护）软件，而每次修改**必不免地引入新的错误，就这样**一次次修改，从而导致软件的失效率升高，如图所示，以至造成软件退化。因此说，软件的维护要比硬件的维护要复杂的多，与硬件的维修有着本质上的差别。



在软件开发的不同阶段进行修改需要付出的代价是很不相同的。

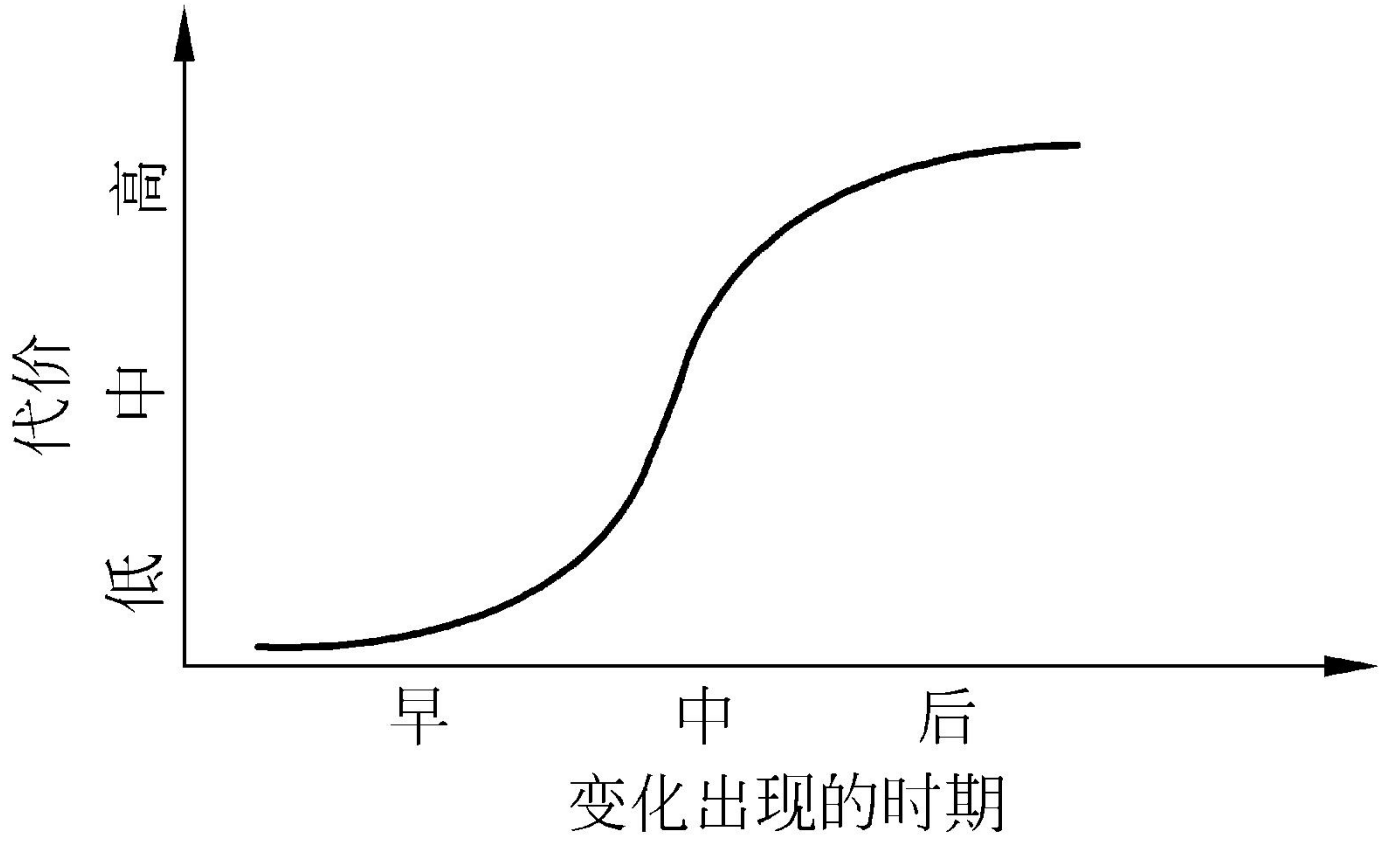
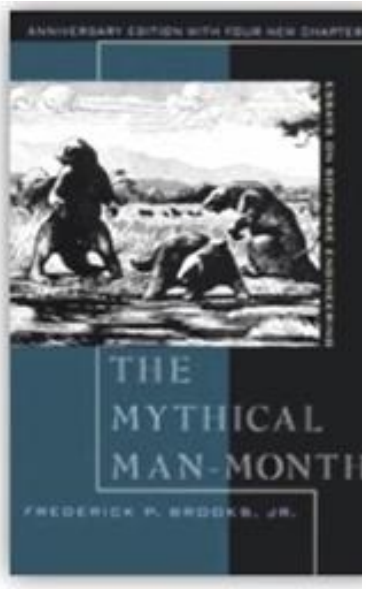


图**1.1** 引入同一变动付出的代价随时间变化的趋势

软件的本质特性（4） - 不可见性

- 不可见性
  - 软件是不可见的和无法可视化的
    - 软件的客观存在不具有空间的形体特征
    - 开发人员可以直接看到程序代码，但是源代码并不是软件本身
  - 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、……
    - 这些技术仍然无法给出准确的、完整的描述

软件的本质特性（4） - 不可见性



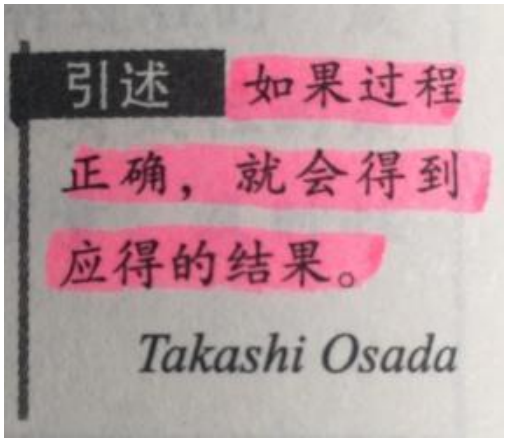
软件人员太像“皇帝的新衣”故事中的裁缝了。当我来检查软件开发工作时，所得到的回答好象对我说：我们正忙于编织这件带有魔法的织物。只要等一会儿，你就会看到这件织物是及其美丽的。但是我什么也看不到，什么也摸不到，也说不出任何一个有关的数字，没有任何办法得到一些信息说明事情确实进行得非常顺利，而且我已经知道许多人最终已经编织了一大堆昂贵的废物而离去，还有不少人最终什么也没有做出来。

软件危机的原因 *

- 客观：软件本身特点
  - 逻辑部件
  - 规模庞大
  - 维护数量不断膨胀（软件维护通常意味着改正或修改原来的设计）
- 主观：不正确的开发方法
  - 忽视需求分析
  - 软件开发=程序编写
  - 轻视软件维护

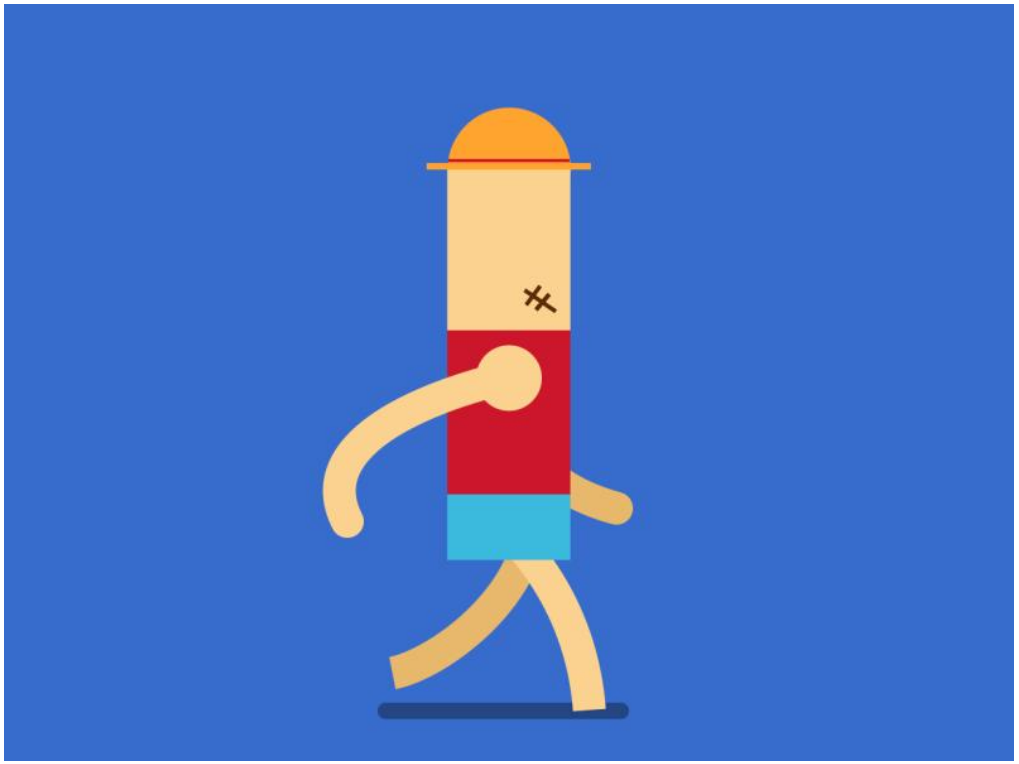
在中国接受了ISO的思想之后，过程和质量的控制能力在国内的制造业中逐渐体现出来。但是在软件行业中，拥有这种过程能力的软件组织仍然是少的可怜，大多数的软件组织奉行的还是一种在八九十年代的个人英雄主义，开发软件单靠个人的力量，能力强的程序员能够成功的完成软件，能力差的则失败。**大多数的软件组织中，少数人掌握着代码，他们就是一切**，如果他们因为私人原因离开所在的组织，手上的代码则是他们的资本，原有的组织将受到沉重的打击。

1.1.3 软件危机的解决途径★



- **组织管理**
  - 工程项目管理方法
  - 编写程序所需的工作量只占软件开发全部工作量的10%～20%
  - 做好软件定义时期的工作
- **技术措施**
  - 软件开发技术与方法（例如面向对象技术等）
  - 软件工具（例如配置管理工具、测试工具等）

做个小练习



单选题 1分

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

多选题 1分

设置

下列关于软件的说法正确的是（ ）

软件项目才是这样，软件产品是面向所有人的，不是定制的

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

单选题 1分

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

第一章 软件危机与软件工程

1.2 软件工程-

1.2.1 软件工程的介绍

定义

- 工程方法+管理技术+技术方法

1983年美国《IEEE软件工程标准术语》对软件工程下的定义为：软件工程是开发、运行、维护和修复软件的系统方法，

1993年IEEE进一步给出了一个更全面更具体的定义：

“软件工程是：

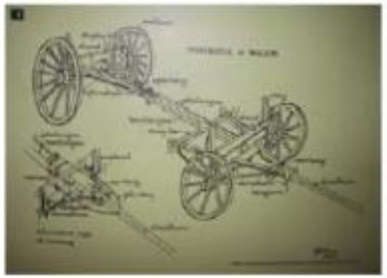
①把系统的、规范的、可度量的方法应用于软件开发、运行和维护，也就是工程化应用于软件；

②研究①中提到的途径、方法。”

（The Application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.）

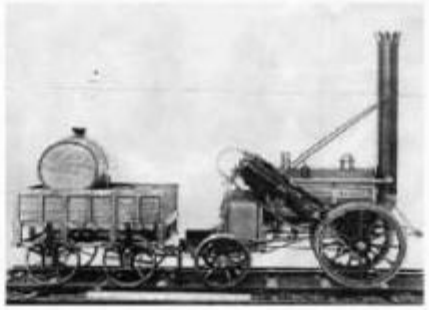
工程的含义

- 工程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。



- craft
  - personal skill
  - experience
  - flexible materials

- engineering
  - tables and tools
  - recorded knowledge
  - controlled materials



工程的含义

- 规模上的差异
  - 花园小道 vs. *汽车高速公路*
  - 树上小屋 vs. *摩天大楼*
  - 加法程序 vs. *医院档案系统*

- 手工（*Craft*）：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（*Engineering*）：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。

软件的工业化生产过程应具备的特点：

- 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

- 强调规范化
- 强调文档化

“一个好的工业，应有一套良好的标准来配套”

软件工程的本质特征

1. 软件工程关注于大型程序的构造。
2. 软件工程的中心课题是控制复杂性。
3. 软件经常变化。
4. 开发软件的效率非常重要。
5. 和谐地合作是开发软件的关键。
6. 软件必须有效地支持它的用户。
7. 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。

软件工程包含的内容 ★

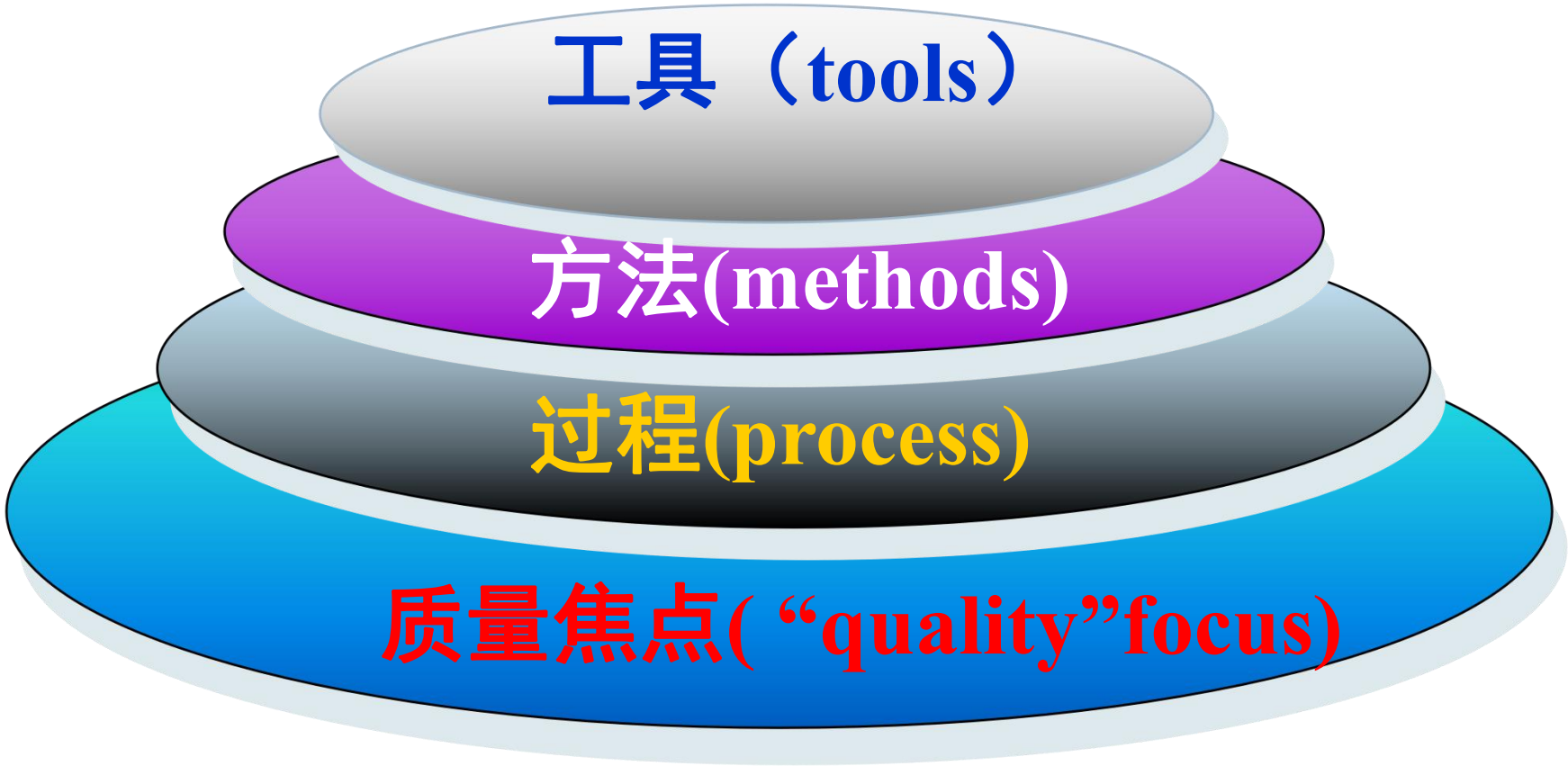
- **一个过程**
  - 方法使用的顺序
  - 要求交付的文档资料
  - 为保证质量和适应变化所需要的管理
  - 软件开发各个阶段完成的里程碑
- **一组方法（如何做）**
- **一系列工具**

  软件工具为软件工程方法提供了自动的或半自动的软件支撑环境

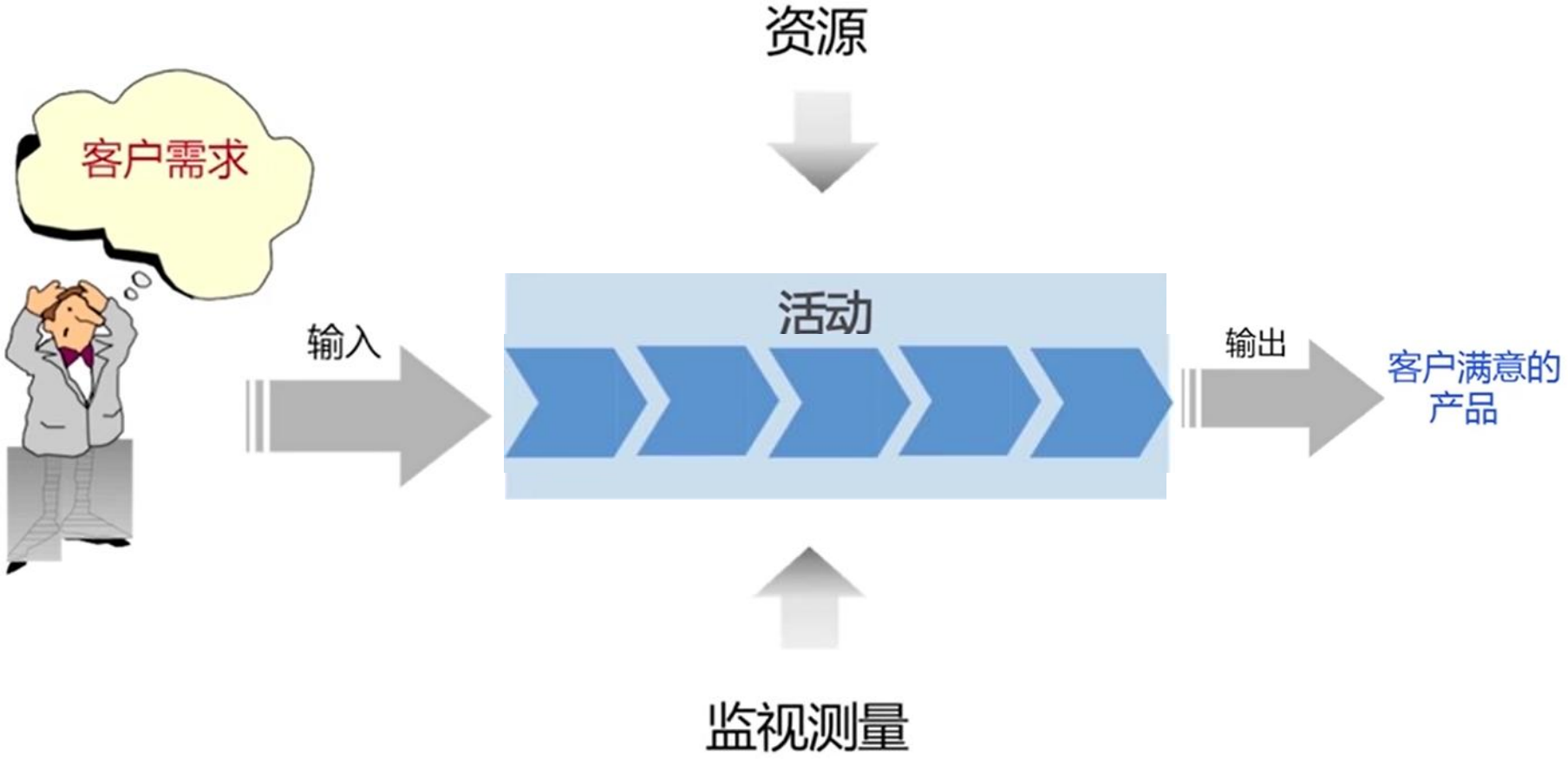
虽然生产一件产品的相关人员有千千万万，但是生产出来的产品却只有一种。这种能力并不是从天上掉下来的，是通过制定极为详尽的生产过程规定得到的。

软件过程提供稳定性，控制和组织。期望像流水线一样生产软件。

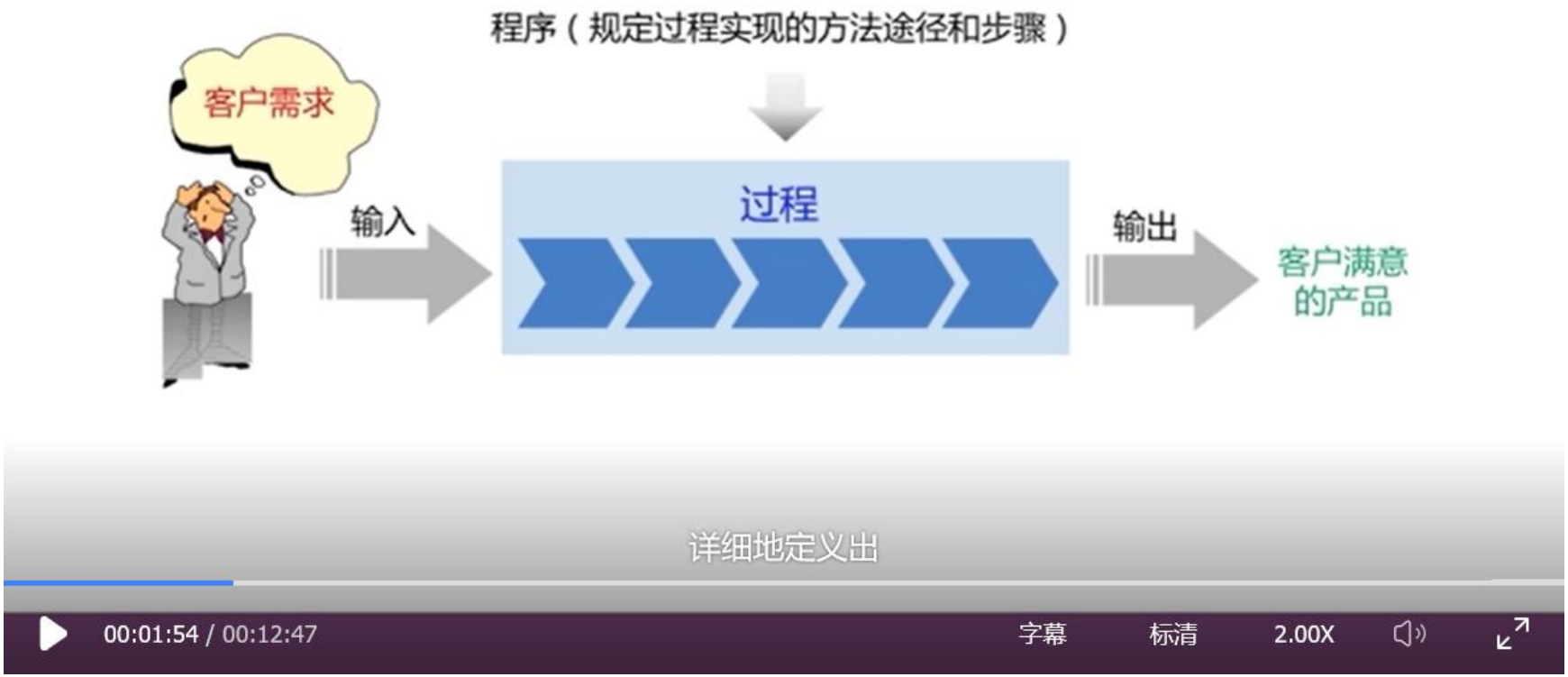
软件工程 — 一种层次化技术 ★



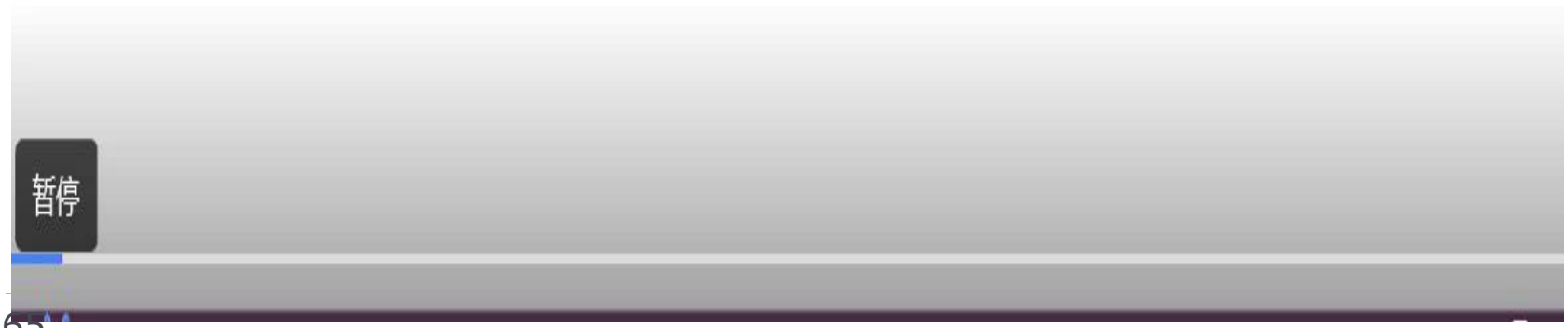
理解软件工程过程（process）



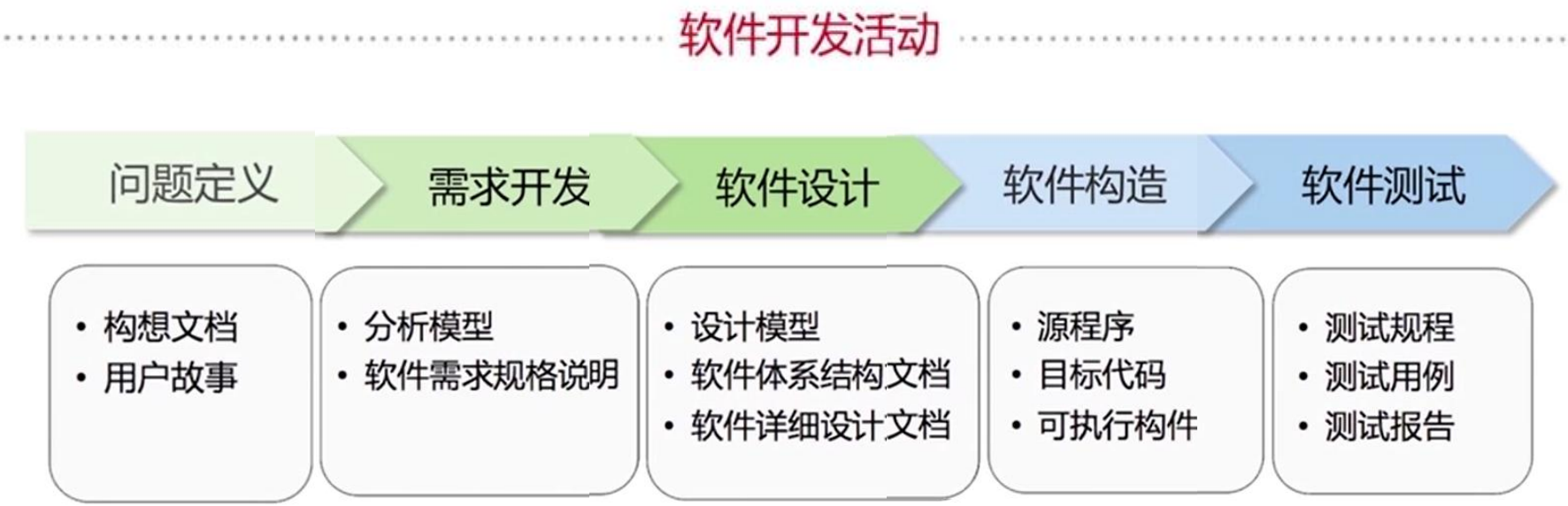
看视频，理解过程（Process）



过程是一组将输入转化为输出的相互关联或相互作用的活动。



软件工程过程与方法（Method）



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……）

理解软件工具（Tool）

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	Visual Studio eclipse	Selenium xUnit BugFree Loadrunner JMeter	Visual Studio eclipse ALTOVA umodel	Trello tower REDMINE git GitHub	

理解软件质量 ★

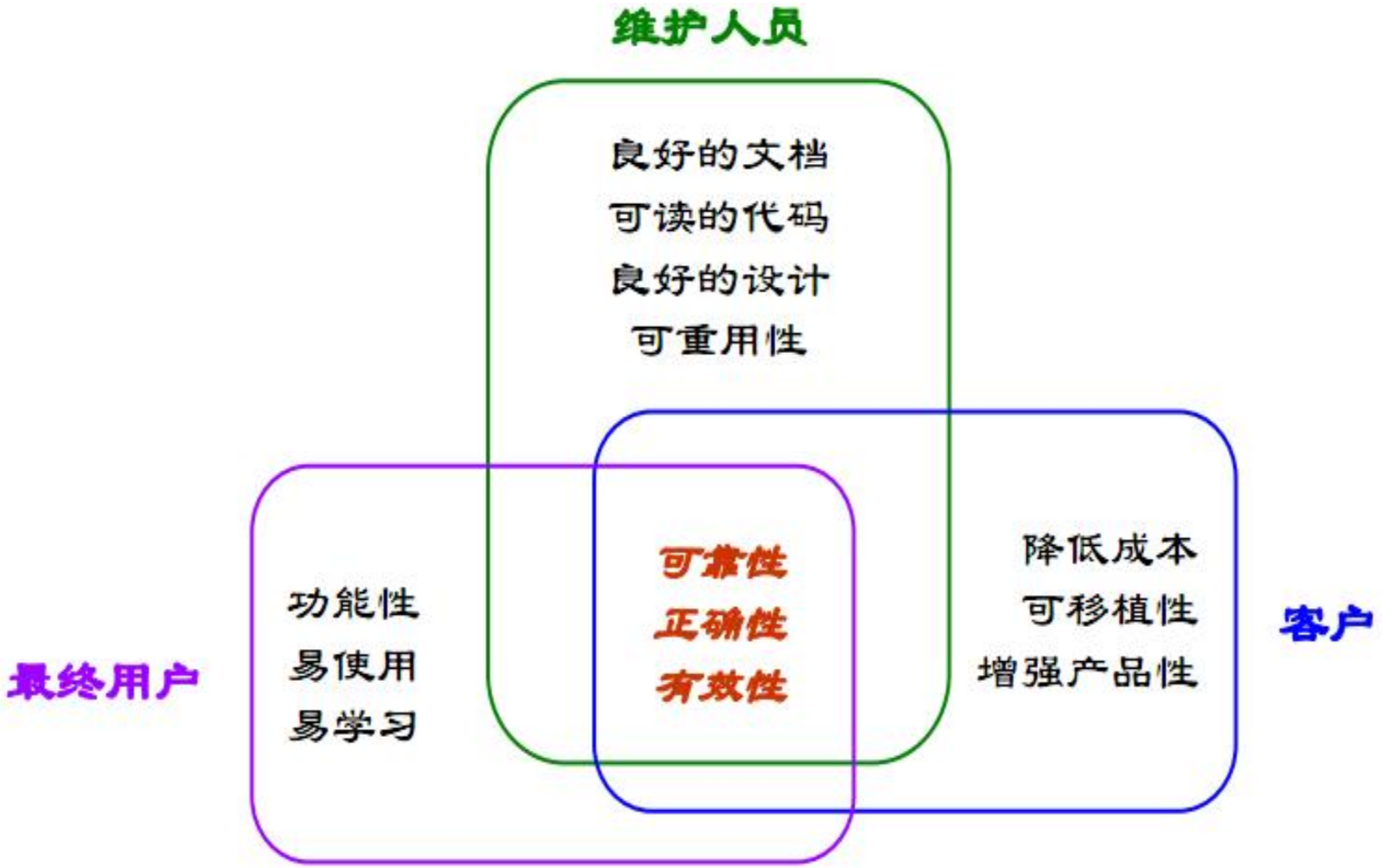
- 你同意以下说法吗？为什么？

  “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

理解软件质量


维护人员
良好的文档
可读的代码
良好的设计
可重用性
最终用户
功能性
易使用
易学习
可靠性
正确性
有效性
降低成本
可移植性
增强产品性
客户

软件质量的评价（以下内容需要掌握）★

- **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

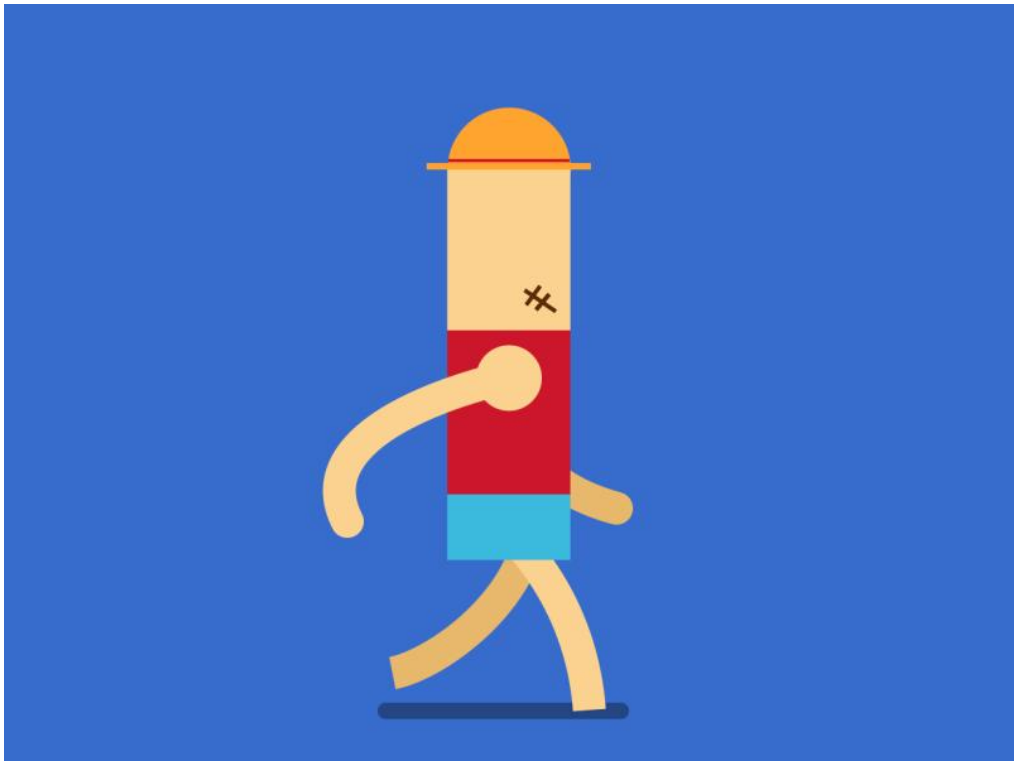
- ISO/IEC 9126 (1991) 软件质量模型是评价软件质量的国际标准，由6个特性和27个子特性组成。

业界人士谈软件工程的应用

- 软件不仅仅是做出来能运行，而是要能用好用。


北京清软海芯科技有限公司
创始人 施侃乐
我们是一家专门从事于
00:00:08 / 00:11:49
字幕

做个小练习



填空题 1分

设置

软件工程概念的提出是为了解决（ [填空1] ）

软件危机

作答

正常使用填空题需3.0以上版本雨课堂

单选题 1分

设置

下面的（B）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。 可理解性还有可维护性

提交

单选题 1分

设置

软件可靠性是指 C

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

第一章 软件危机与软件工程

1.2 软件工程-

1.2.2 软件工程的基本原理

软件工程的基本原理 ★

— 确保软件产品质量和开发效率的原理的最小集合。

美国著名软件工程专家B.W.Boehm于1983年提出了软件工程的以下7条基本原理：

1. 用分阶段的生命周期计划严格管理
2. 坚持进行阶段评审
3. 实行严格的产品控制
4. 采用现代程序设计技术
5. 结果应该可以清楚地审查
6. 开发小组的人员应该少而精(1+1 <2)
7. 承认不断改进软件工程实践的必要性

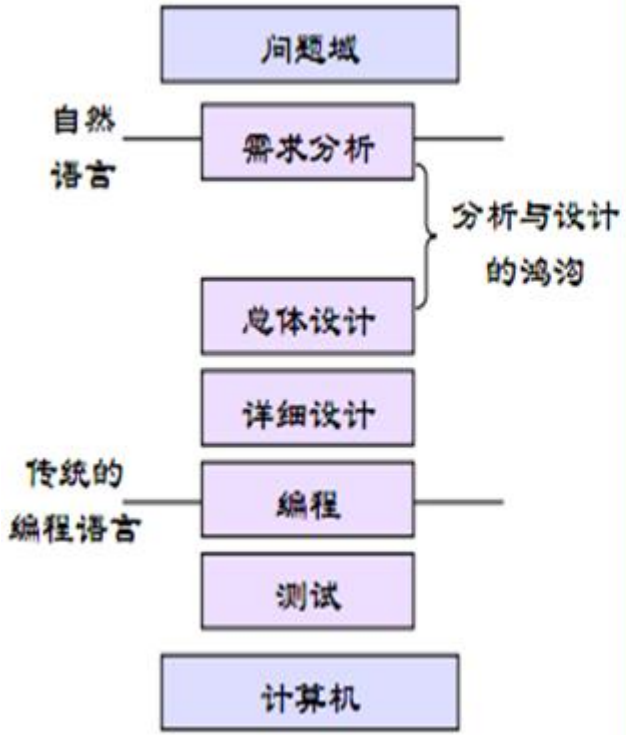
第一章 软件危机与软件工程

1.2 软件工程-

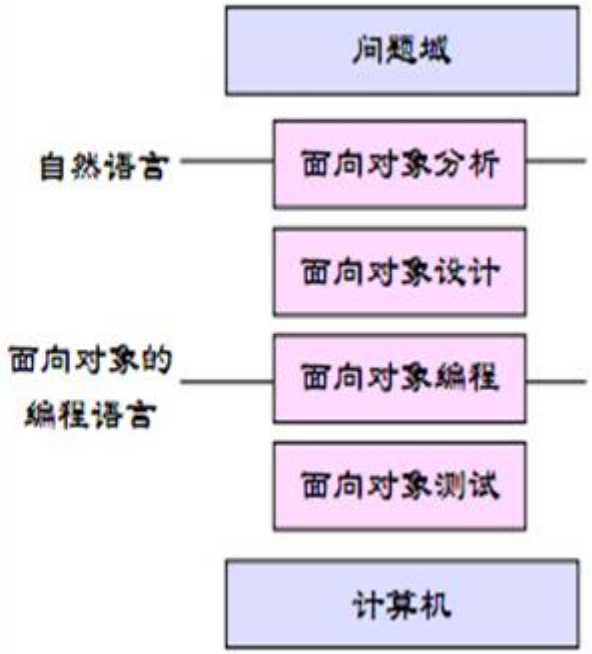
1.2.3 软件工程方法学

目前使用得最广泛的两种软件工程方法学：

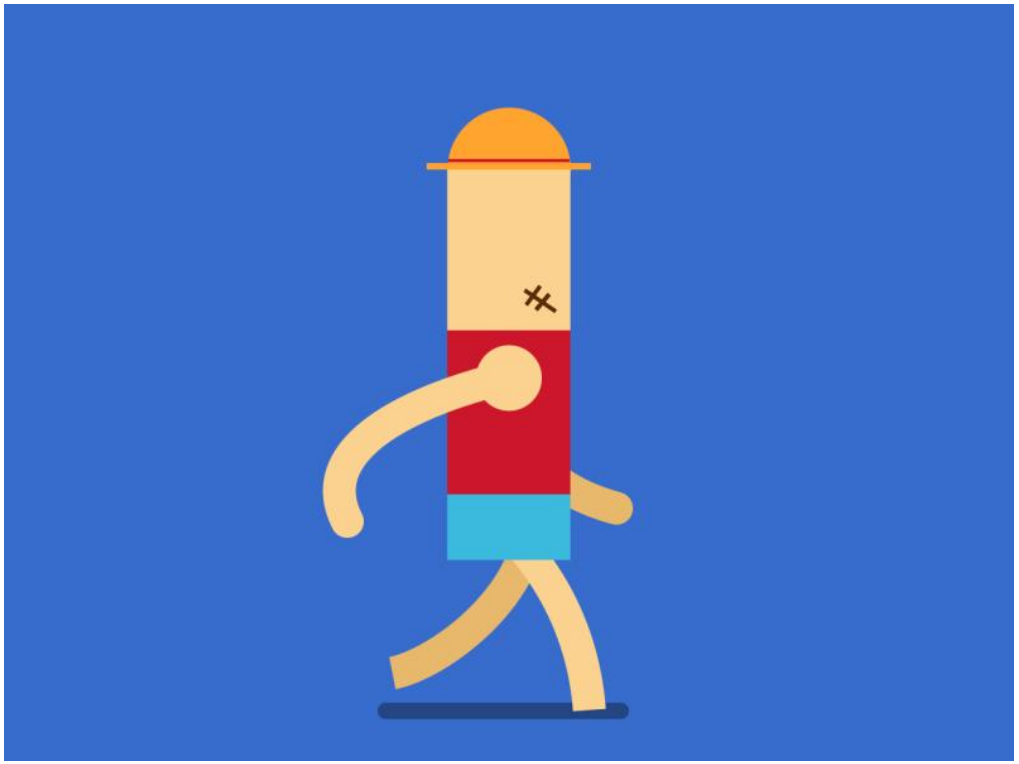
- 传统的方法学
  （生命周期方法学或结构化范型）



- 面向对象方法学



做个小练习



填空题 3分

设置

使用最广泛的两种程序设计方法，分别是：

A 结构化程序设计方法；

B 面向对象的程序设计方法；

提交

单选题 1分

设置

（　　）是属于面向对象的程序设计语言。

A C

B Pascal

C C++

D Fortran

思考：你目前学习过的语言是面向对象的还是面向过程的？

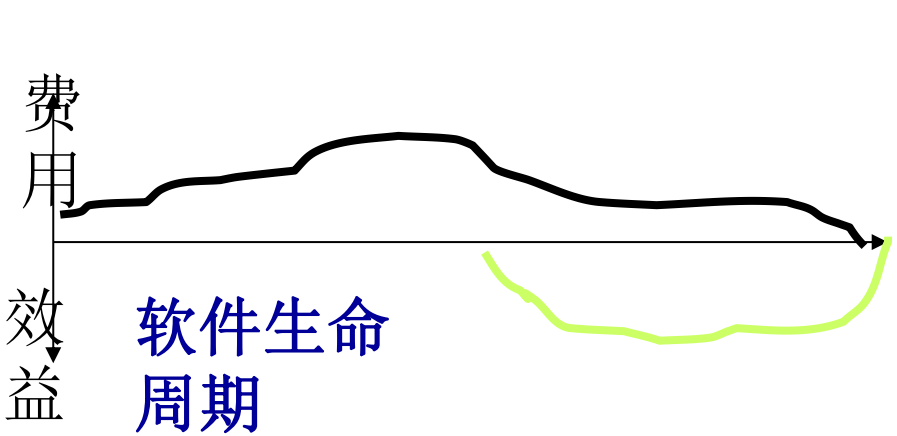
提交

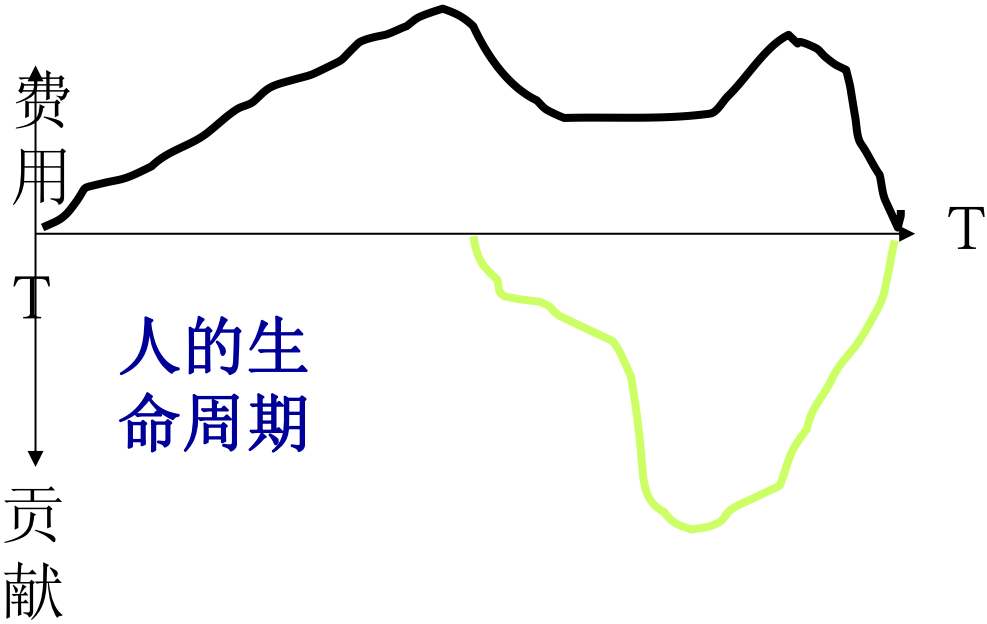
1.3 软件生命周期

采用生命周期的方法分阶段完成

软件和软件生命期模型（life cycle）

- 一个软件从被提出开始研制至软件最终被废弃不再使用为止的全过程，称为软件生命期





软件和软件生命期模型

生命型模型的优点：

- 每个阶段的工作结果用书面形式描述出来，使得“不可见”的软件变成了“可见”的文档资料。
- 使得开发过程分阶段按步骤进行，以交付某种特定规格的文档作为标志某个阶段完成的里程碑，使“难于管理的思考过程”变为“可以管理的生产过程”。

软件是程序以及开发、使用和维护程序所需要的所有文档。软件=程序+文档+数据

第一个时期：软件定义时期

- （1）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。

- （2）可行性研究：

  任务是为前一阶段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。

第二个时期：软件开发时期

- （1）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。

- （2）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。

- （3）详细设计：

  针对单个模块的设计。

- （4）编码：

  按照选定的语言，把模块的过程性描述翻译为源程序。

- （5）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。

第三个时期：软件运行时期

- （1）软件安装运行
- （2）维护

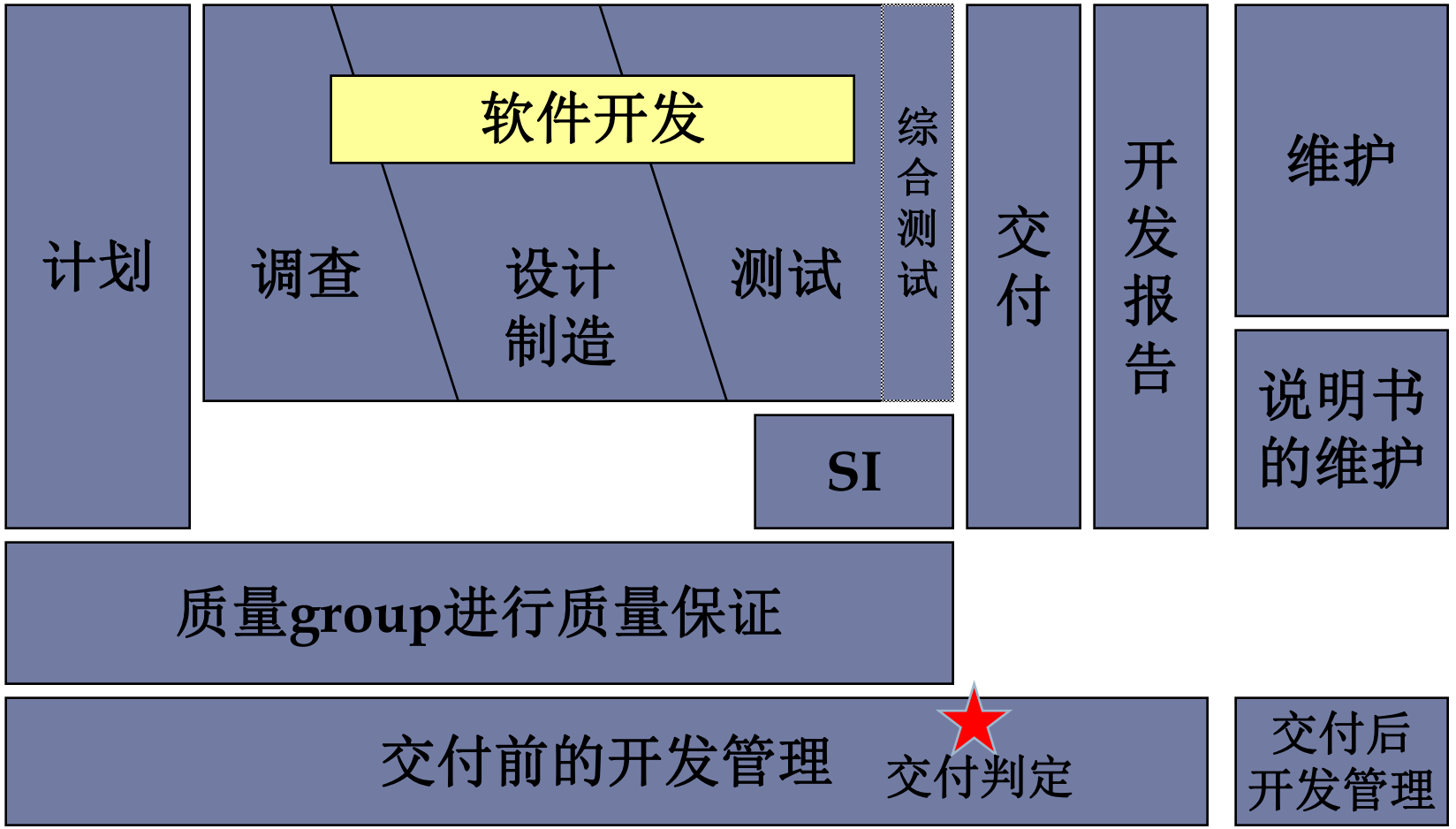
结构分析设计过程（8个阶段）小结

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。

结构分析设计过程小结

阶段	关键问题	结束标准
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

开发活动的生命期全貌


计划
软件开发
调查
设计
制造
测试
综合测试
SI
交付
开发报告
维护
说明书的维护
质量group进行质量保证
交付前的开发管理
交付判定
交付后开发管理

1.4　软件过程

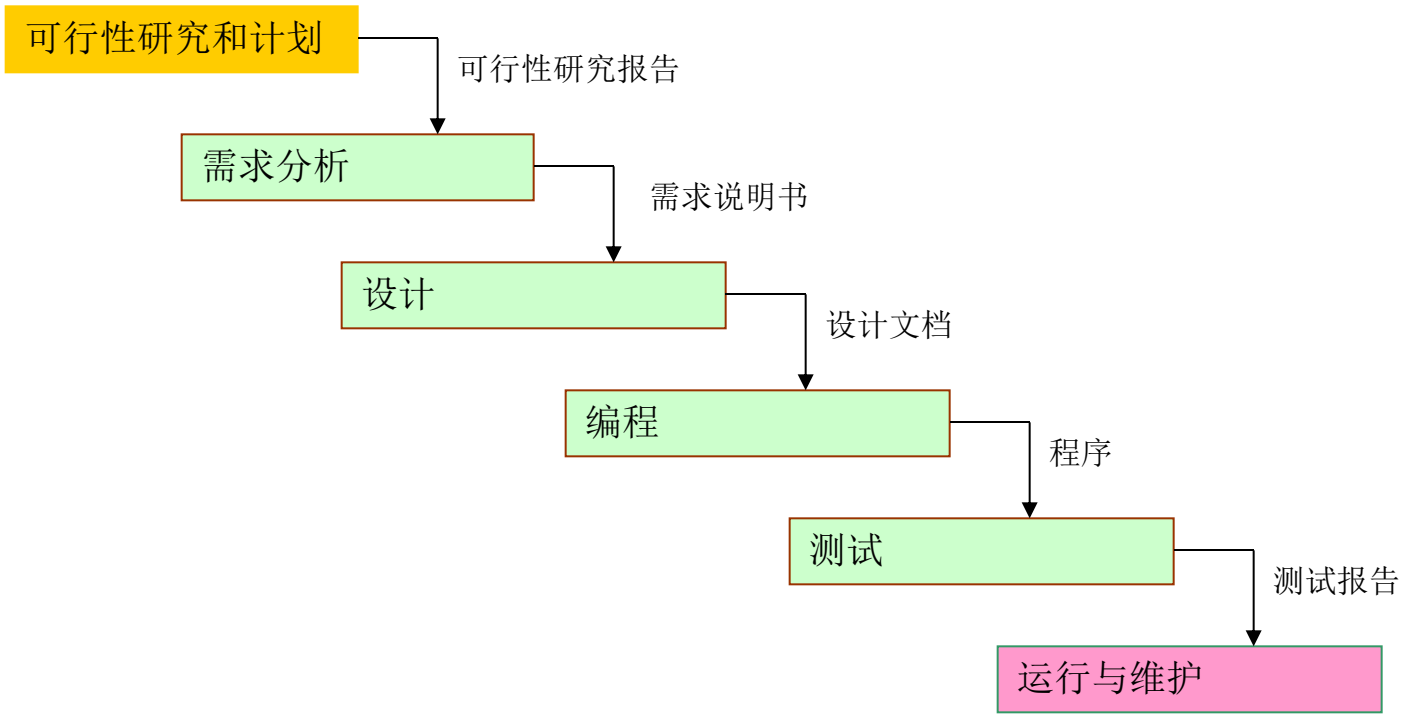
软件开发过程生命周期模型

软件生命期模型

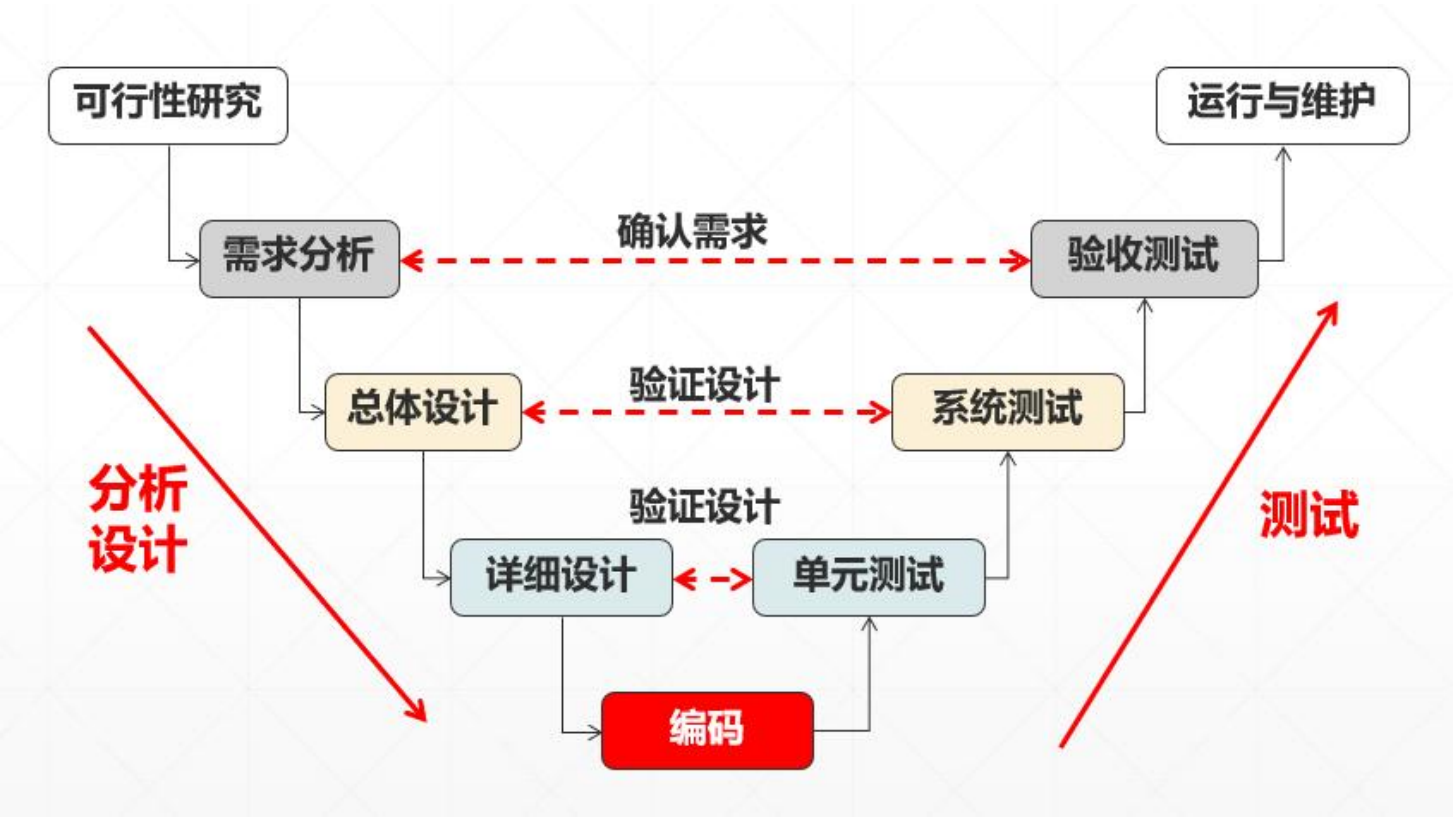
主要有以下几种模型：

- 1.瀑布模型 (waterfall model)
- 2.快速原型 (Rapid application)
- 3.增量模型 (Incremental model)
- 4.螺旋模型 (spiral model)
- 5.敏捷开发 (Agile Development)

1.4.1 瀑布模型（ waterfall model ）



瀑布模型的变体 – V模型



V模型描述了：质量保证动作同早期构建相关的动作之间的关系。

瀑布模型特点

- 瀑布模型将软件生命周期的各项活动规定为依固定顺序联接的若干阶段工作，形如瀑布流水，最终得到软件产品。

- **瀑布模型的特点：**
  - 阶段间具有顺序性和依赖性。
  - 推迟程序的物理实现。
  - 质量保证：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，及早改正错误。
  - 易于组织，易于管理：因为你可以预先完成所有计划。
  - 是一种严格线性的、按阶段顺序的、逐步细化的过程模型（开发模式）。

瀑布模型 的优缺点

- 优点：

  a. 强调开发的阶段性：阶段间具有顺序性和依赖性
  b. 强调早期计划及需求调查：推迟实现的观点
  c. 强调评审，强调产品测试：质量保证的观点

- 缺点：

  a. 依赖于早期进行的唯一一次需求调查，不能适应需求的变化；
  b. 由于是单一流程，开发中的经验教训不能反馈应用于本产品的过程；
  c. 风险往往迟至后期的开发阶段才显露，因而失去及早纠正的机会。
  d. 文档驱动的，这对于非专业的用户来说是难以阅读和理解的。

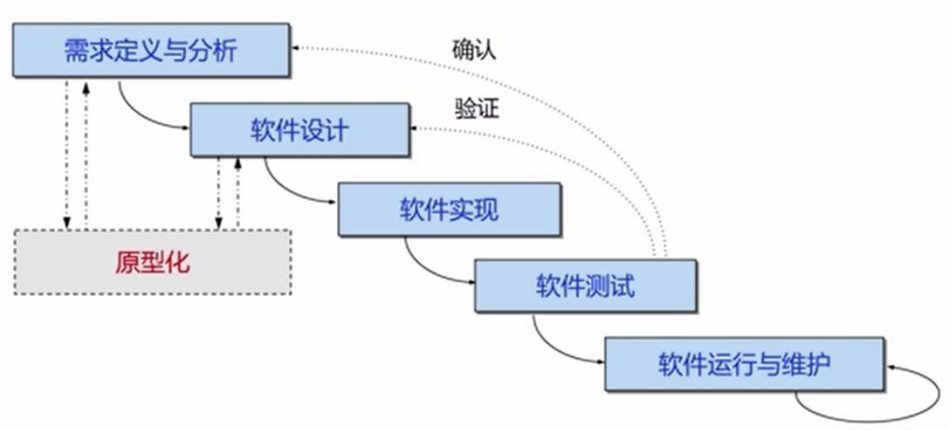
- 这是最早存在的开发模型，并且现在使用的也比较多。

瀑布模型的使用

- 当需求很确定，有一个稳定的产品定义和很容易被理解的技术解决方案时，纯瀑布模型特别合适。
- 当你对一个定义得很好的版本进行维护或将一个产品移植到一个新的平台上，瀑布模型也特别合适。
- 对于那些容易理解但很复杂的项目，采用纯瀑布模型比较合适，因为可以用顺序方法处理问题。
- 在质量需求高于成本需求和进度需求的时候，它尤为出色。
- 当开发队伍的技术力量比较弱或者缺乏经验时，瀑布模型更为适合。

1.4.2 快速原型(Rapid application)

- 用户不能标示出详细的输入，处理和输出需求。
- 开发者不能确定算法的有效性，操作的适应性或人机交互的形式。

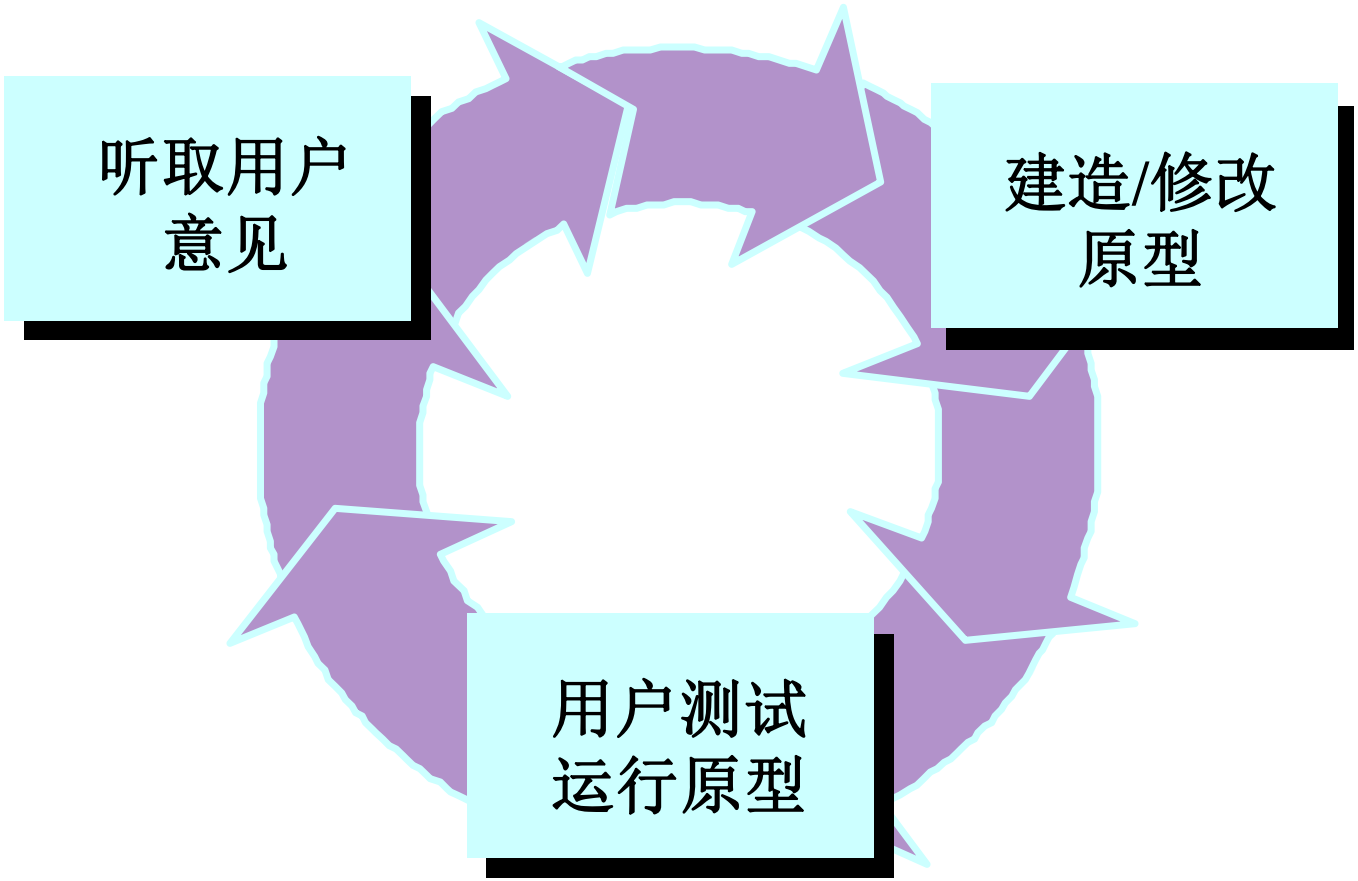


原型被建造仅是为了定义需求，之后就被抛弃了（或至少部分被抛弃）

原形仅仅是为了演示，之后会抛弃

快速原型的模型:演化模型的一种

Prototyping : Evolutionary Models



快速原型的分类

原型模型分类

- **抛弃型原型**：用于试验某些概念，试验完系统将无用处
- **进化型原型**：原型系统不断被开发和被修正，最终它变为一个真正的系统。

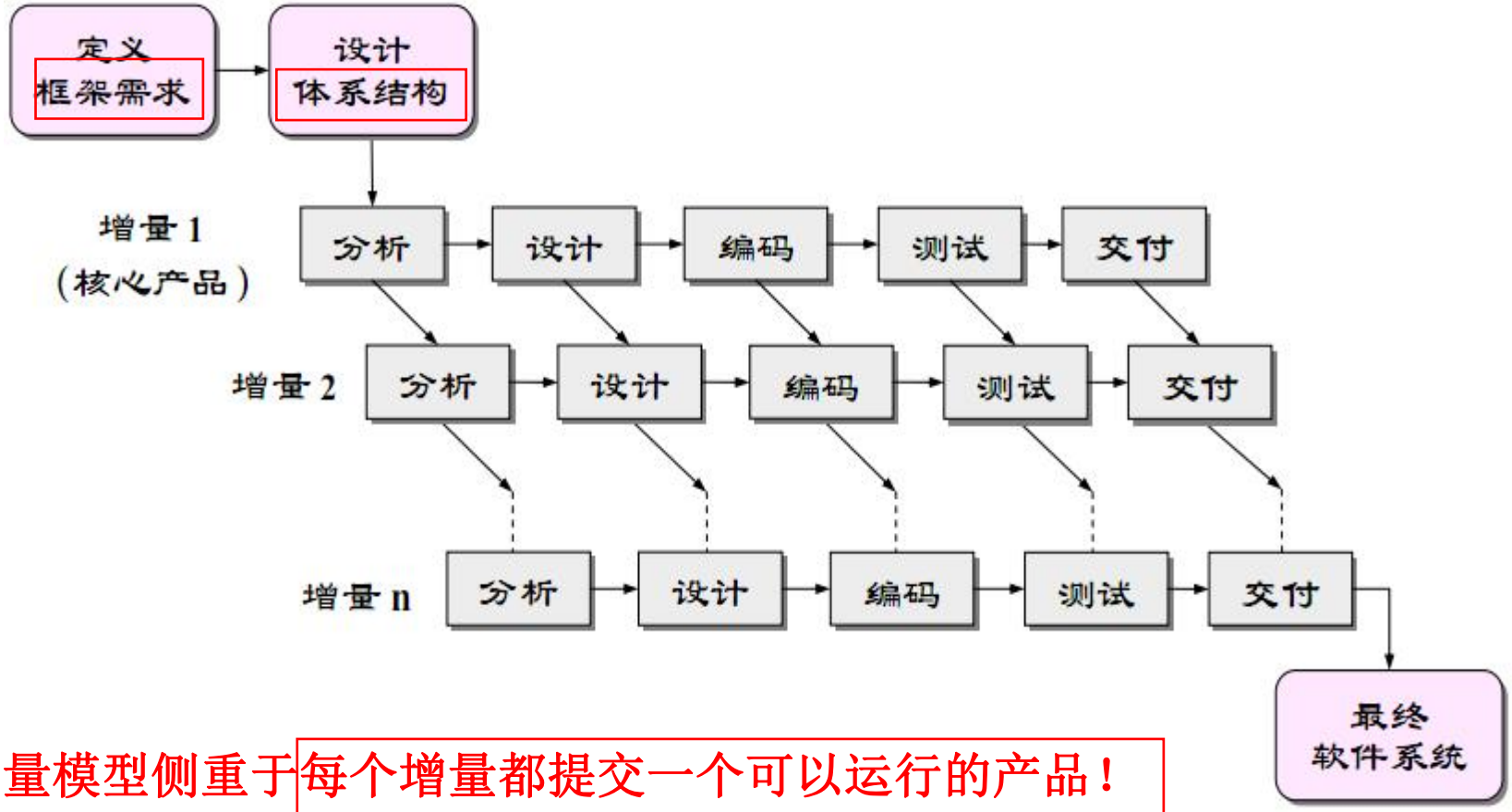
快速原型的优缺点

- 适用于用户驱动的系统（即需求模糊或随时间变化的系统）。
- 优点：
  - 从实践中学习(Learning by doing)
  - 改善通信
  - 改善用户参与
  - 使部分已知需求清晰化
  - 展示描述的一致性和完整性
  - 提高系统的实用性、可维护性
  - 节省开发的投入、缩短整个软件的开发周期

快速原型的优缺点

- 缺点
  - 用户有时误解了原型的角色，例如他们可能误解原型应该和真实系统一样可靠。
  - 缺少项目标准，进化原型方法有点像编码修正。
  - 缺少控制，由于用户可能不断提出新要求，因而原型迭代的周期很难控制。
  - 额外的花费：研究结果表明构造一个原型可能需要10%额外花费。
  - 为了尽快实现原型，采用了不合适的技术，运行效率可能会受影响。
  - 原型法要求开发者与用户密切接触，有时这是不可能的。例如外包软件。

1.4.3 增量模型（The Incremental Model）



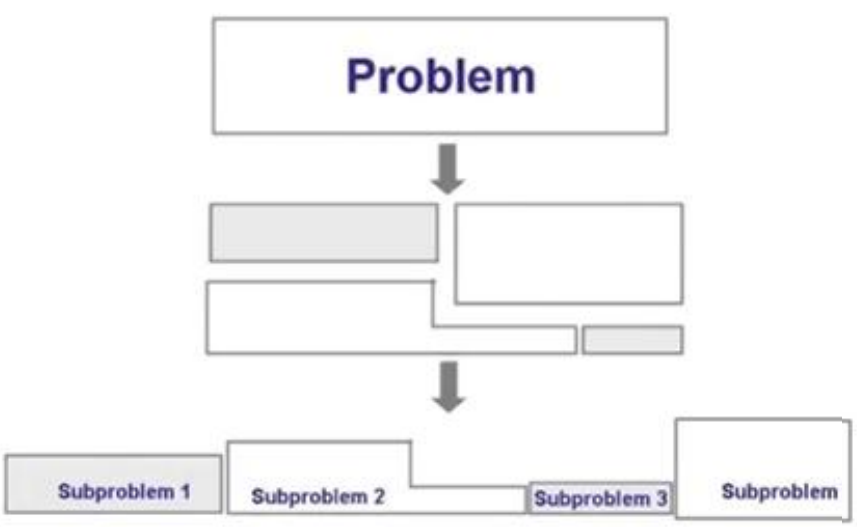
增量模型侧重于每个增量都提交一个可以运行的产品！

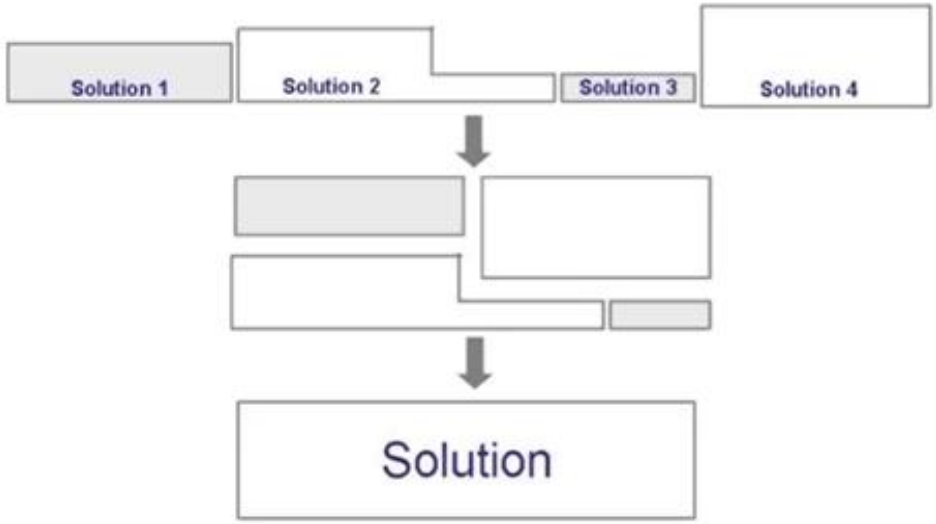
增量模型的优缺点

- 优点
  - 整个产品被分解成若干个构件逐步交付，用户可以不断地看到所开发软件的可运行中间版本
  - 将早期增量作为原型有助于明确后期增量的需求
  - 降低开发风险
  - 重要功能被首先交付，从而使其得到最多的测试
- 缺点
  - 需要软件具备开放式的体系结构
  - 需求难以在增量实现之前详细定义，因此增量与需求的准确映射以及所有增量的有效集成可能会比较困难
  - 容易退化为边做边改方式，使软件过程的控制失去整体性

软件开发的基本策略:分而治之

软件工程是一项解决问题的工程活动，通过对问题进行研究分析，将一个复杂问题分解成可以理解并能够处理的若干小问题，然后再逐个解决。





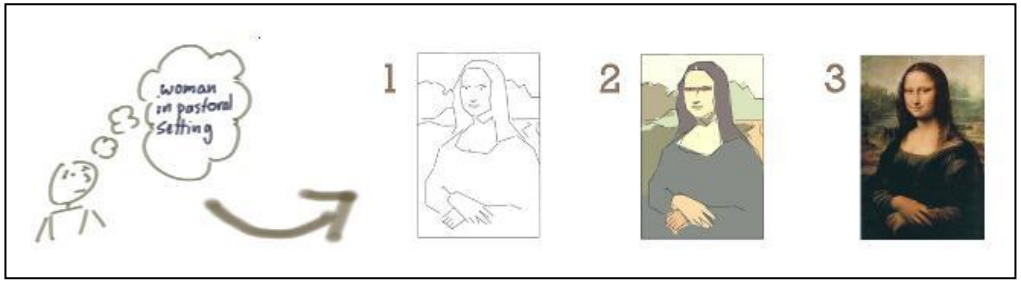
迭代增量开发

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。

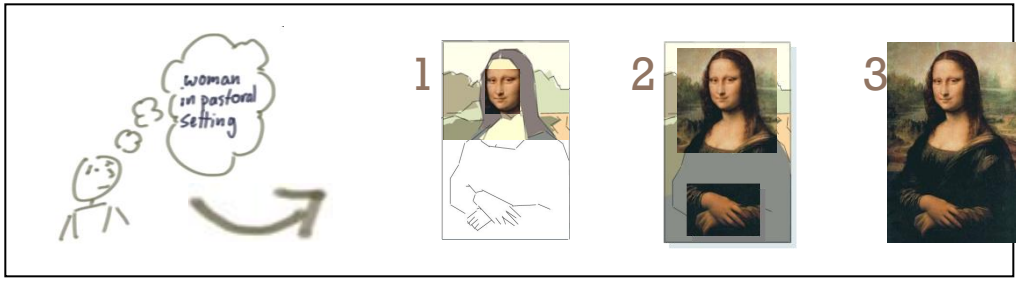
迭代开发

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

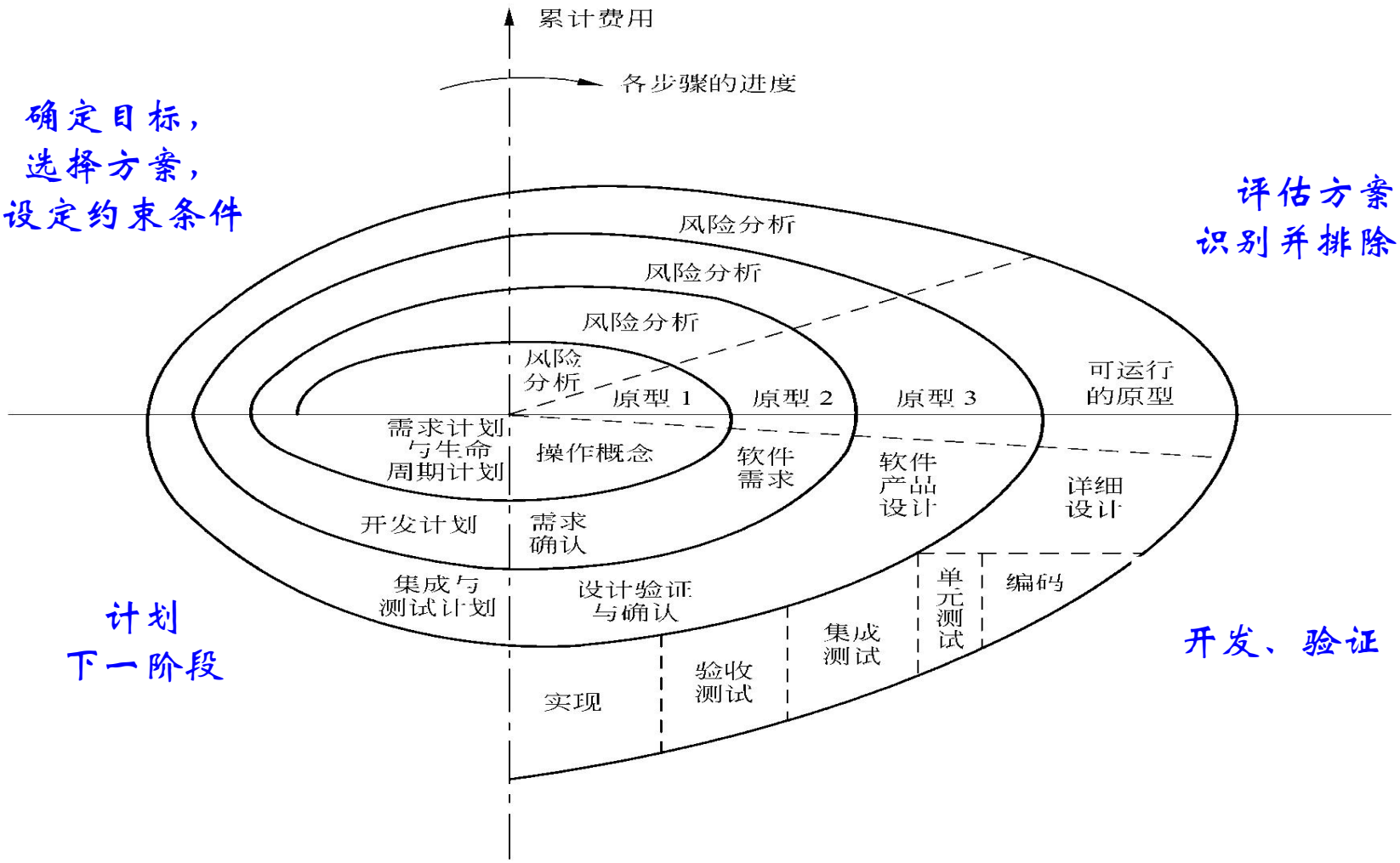
结合迭代与增量，每一次都可以有整个方案的一部分可执行的程序，并不断的提炼和详细化。



软件开发的基本策略:逐步演进

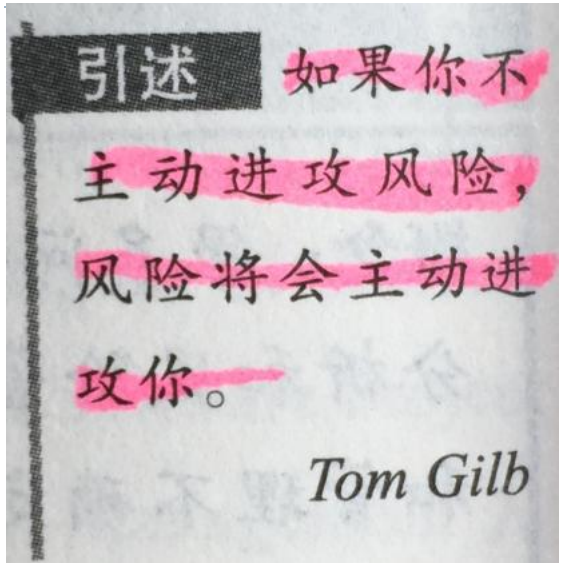
软件更像一个活着的植物，其生长是一个逐步有序的过程。软件开发应该遵循软件的客观规律，不断进行迭代式增量开发，最终交付符合客户价值的产品。

1.4.4 螺旋模型（The Spiral）



螺旋模型的特点

- 螺旋模型基本的做法是在“瀑布模型”的每一个开发阶段之前，引入非常严格的风险识别、风险分析和风险控制。直到采取了消除风险的措施之后，才开始计划下一阶段的开发工作。否则，项目就很可能被取消。
- 综合原型实现模型模型的迭代特征和顺序模型的控制和系统化的优点。



- 每个回线表示开发过程的一个阶段，增加了风险分析，是以风险为导向的生命期模型。通常用来指导大型软件项目的开发。
- 分为4个步骤：确定目标、风险分析、开发和验证、评价结果。

螺旋模型的优缺点

- 优点：

  a. 强调严格的全过程风险管理。
  b. 强调各开发阶段的质量。
  c. 提供机会检讨项目是否有价值继续下去。

- 缺点：

  a. 引入非常严格的风险识别，风险分析，和风险控制，这对风险管理的技能水平提出了很高的要求。
  b. 这需要人员，资金，和时间的投入比较复杂，需要相当的风险评估技术，且成功依赖于这种技术。

- 如果有充足的把握判断遗留的风险已降低到一定的程度，项目管理人员可作出决定让余下的开发工作采用另外的生命周期模型，如“瀑布模型”，或自定的混合模型。

1.4.5 敏捷过程与极限编程（XP）

产生背景：

软件开发的新挑战

- 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

传统的软件开发方法

- 强调过程和文档
- 对变化的适应能力偏弱

概念：

敏捷过程为了使软件开发团队具有高效工作和快速响应变化的能力，17位著名的软件专家于2001年2月联合起草了敏捷软件开发宣言。敏捷软件开发宣言由下述4个简单的价值观声明组成。

敏捷开发过程的价值观：

1. 个体和交互胜过过程和工具

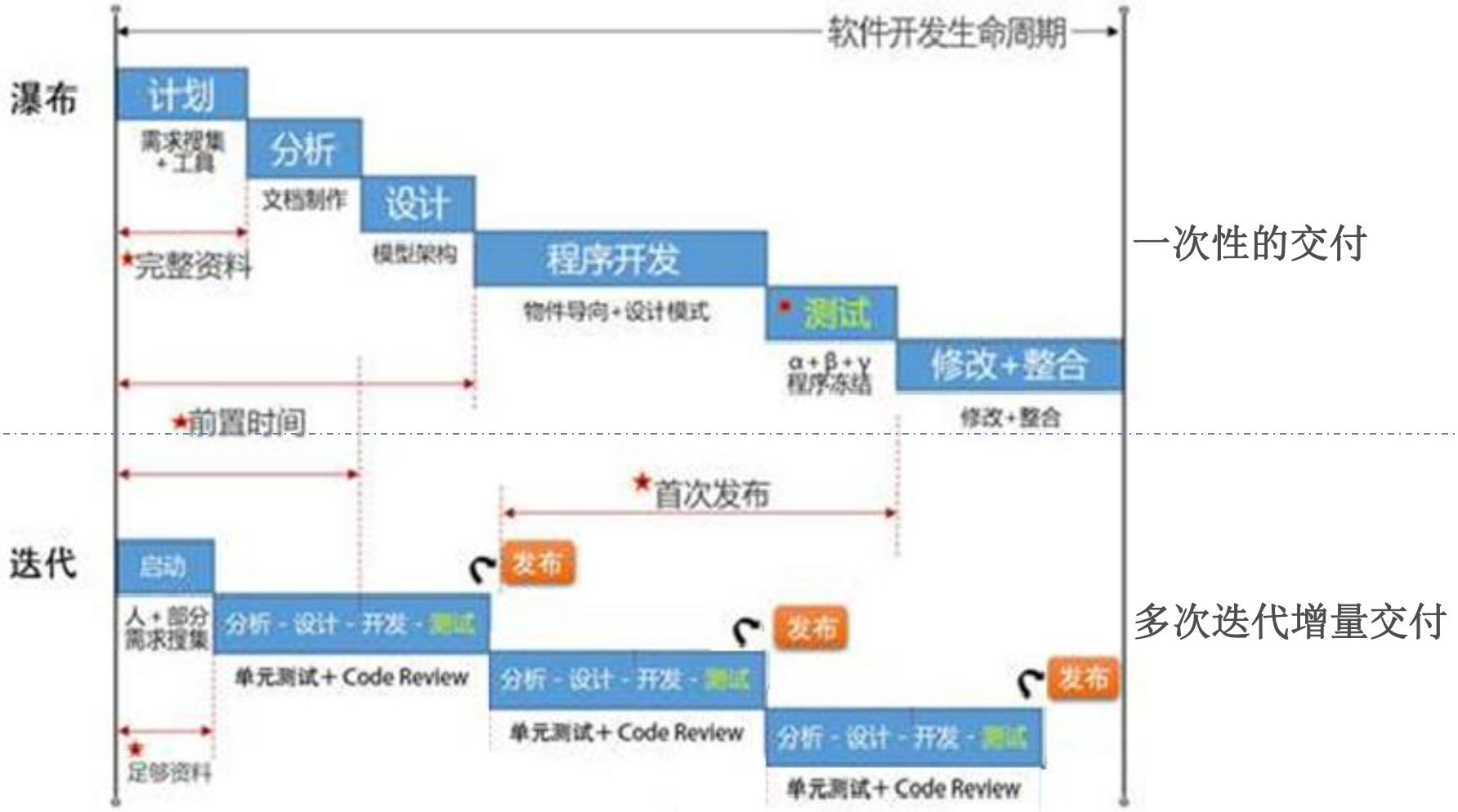
2. 可以工作的软件胜过面面俱到的文档

3. 客户合作胜过合同谈判

4. 响应变化胜过遵循计划

敏捷开发的最大特点是高度迭代，有周期性，并且能够及时、持续地响应客户的频繁反馈。短周期迭代交付，可视化，自适应调整，开放式及时沟通。不再强调传统测试过程中严格的测试阶段。

传统与敏捷


软件开发生命周期
瀑布
计划
需求搜集+工具
分析
文档制作
设计
模型架构
程序开发
物件导向+设计模式
测试
α+β+γ
程序冻结
修改+整合
修改+整合
完整资料
前置时间
首次发布
迭代
启动
人+部分需求搜集
分析 - 设计 - 开发 - 测试
单元测试+Code Review
发布
分析 - 设计 - 开发 - 测试
单元测试+Code Review
发布
分析 - 设计 - 开发 - 测试
单元测试+Code Review
发布
足够资料
一次性的交付
多次迭代增量交付

敏捷过程与极限编程（XP）

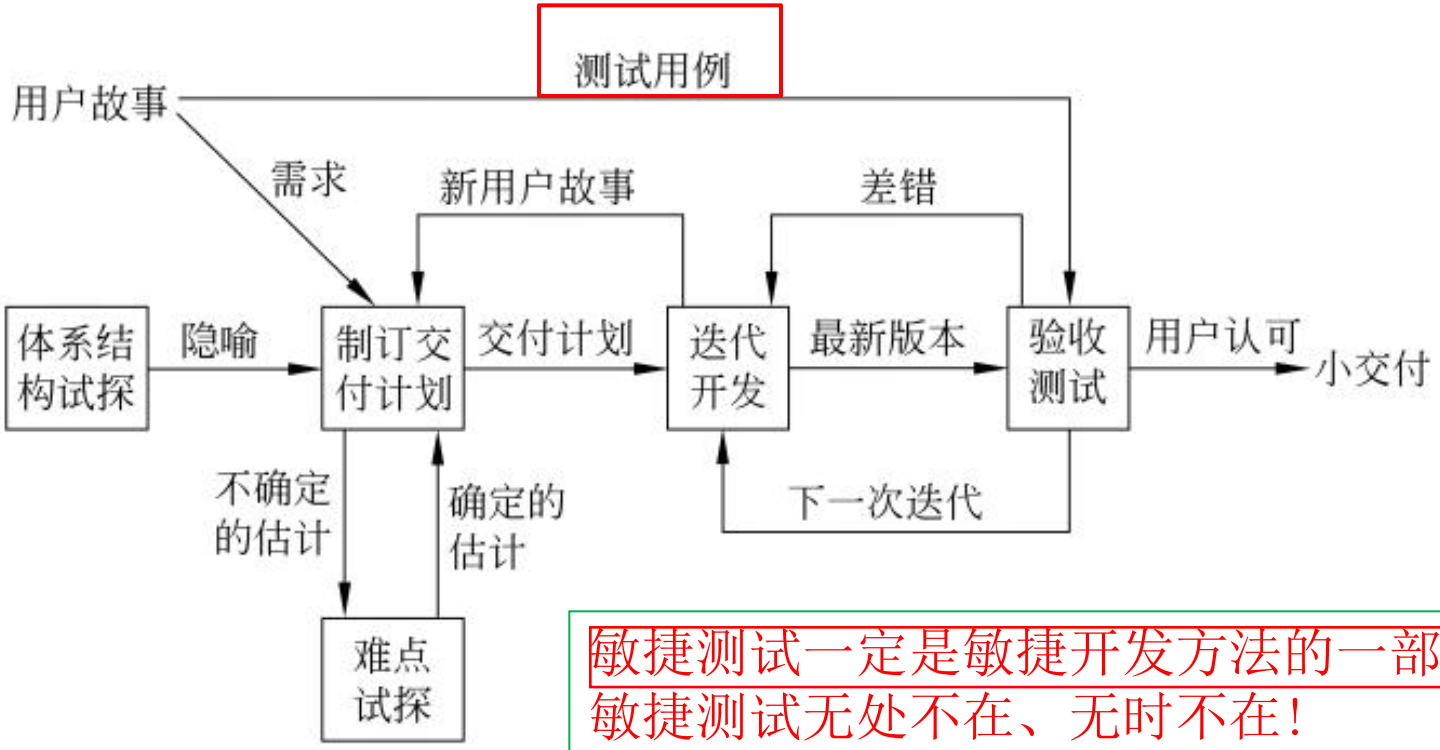
极限编程是什么：

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。

敏捷过程与极限编程（XP）

极限编程的整体开发过程：



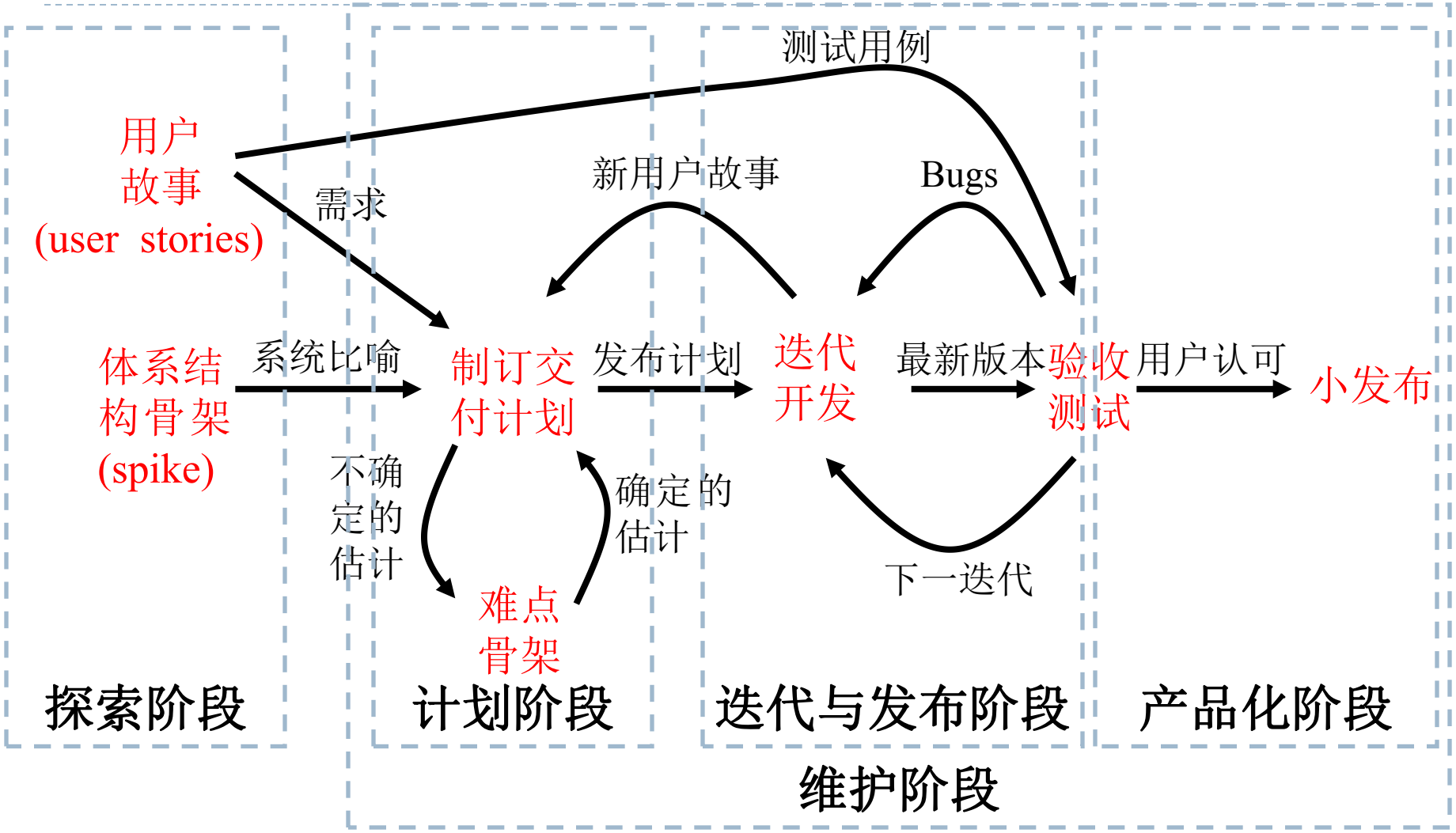
敏捷测试一定是敏捷开发方法的一部分。
敏捷测试无处不在、无时不在！

用户故事

- 用户故事在软件开发过程中被作为描述需求的一种表达形式。
- 为了规范用户故事的表达，便于沟通，用户故事通常的表达格式为：作为一个<用户角色>，我想要<完成活动>，以便于<实现价值>。

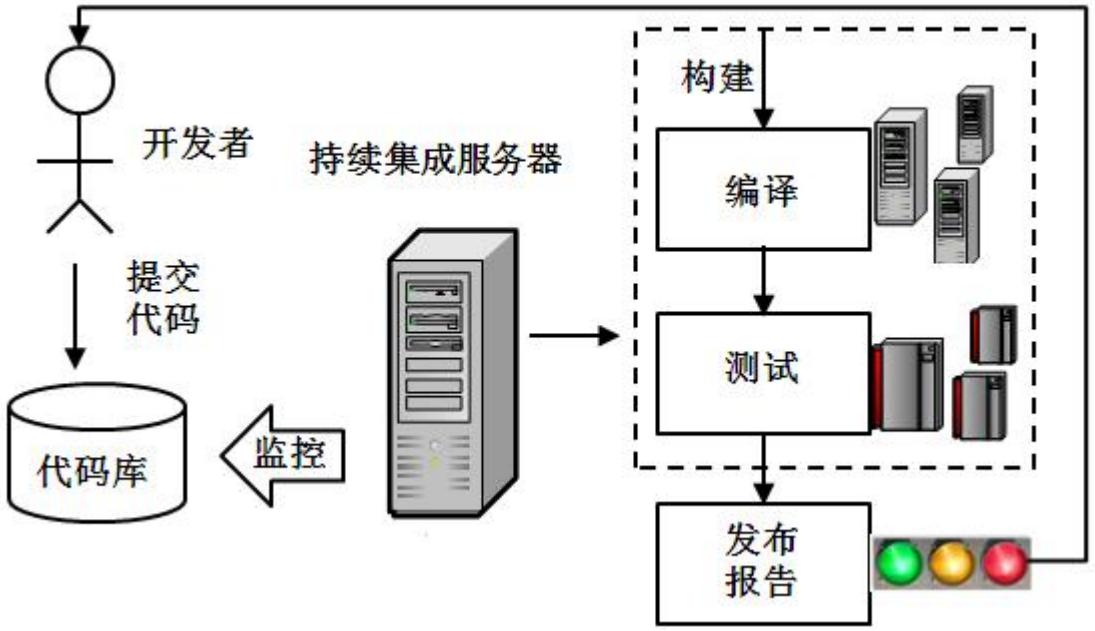
- 例如：腾讯课堂的用户故事：
  - 学生登陆 - 查找需要学习的课程 - 进入课堂 - 听课
  - 教师登陆 - 创建一门课程 - 将课程分享给学生
  - 教师登陆 - 进入课堂（课程） - 授课 - 考勤 - 讲解ppt - 提问抽答
  - 教师登陆 - 进入课堂（课程） - 查看上次课程学生回答的问题情况
  - 学生登陆 - 查找需要学习的课程 - 进入课堂 - 选择课堂练习 - 提交练习答案 - 查看练习结果

另外一种画法


测试用例
用户
故事
(user stories)
需求
新用户故事
Bugs
体系结
构骨架
(spike)
系统比喻
制订交
付计划
发布计划
迭代
开发
最新版本
验收
测试
用户认可
小发布
不确
定的
估计
确定的
估计
难点
骨架
下一迭代
探索阶段
计划阶段
迭代与发布阶段
产品化阶段
维护阶段

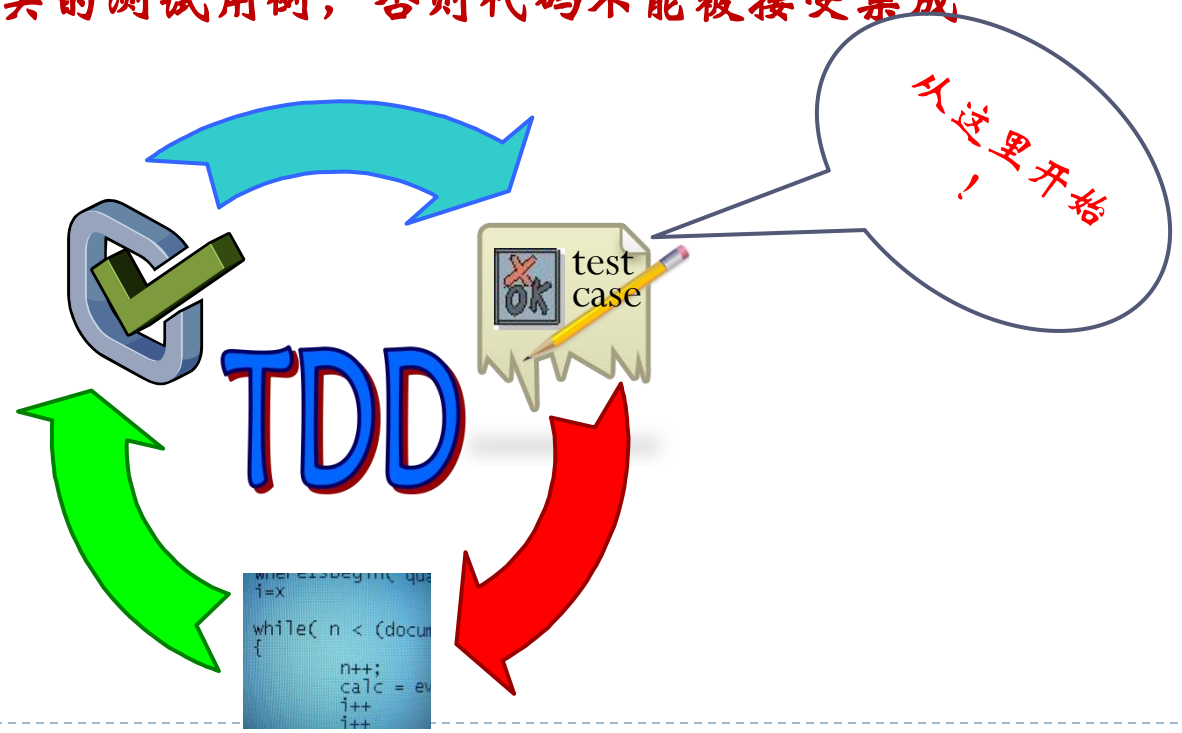
XP核心实践：持续集成（CI）

关键词：用户故事、简单设计、测试驱动开发（TDD）、持续集成、重构、结对编程



测试驱动的开发（TDD）

- 一种不同于传统的开发方式:
  - 首先设计测试用例；
  - 然后编写能通过测试的代码；
  - 维护一组详尽的回归测试用例；
  - 除非有相关的测试用例，否则代码不能被接受集成



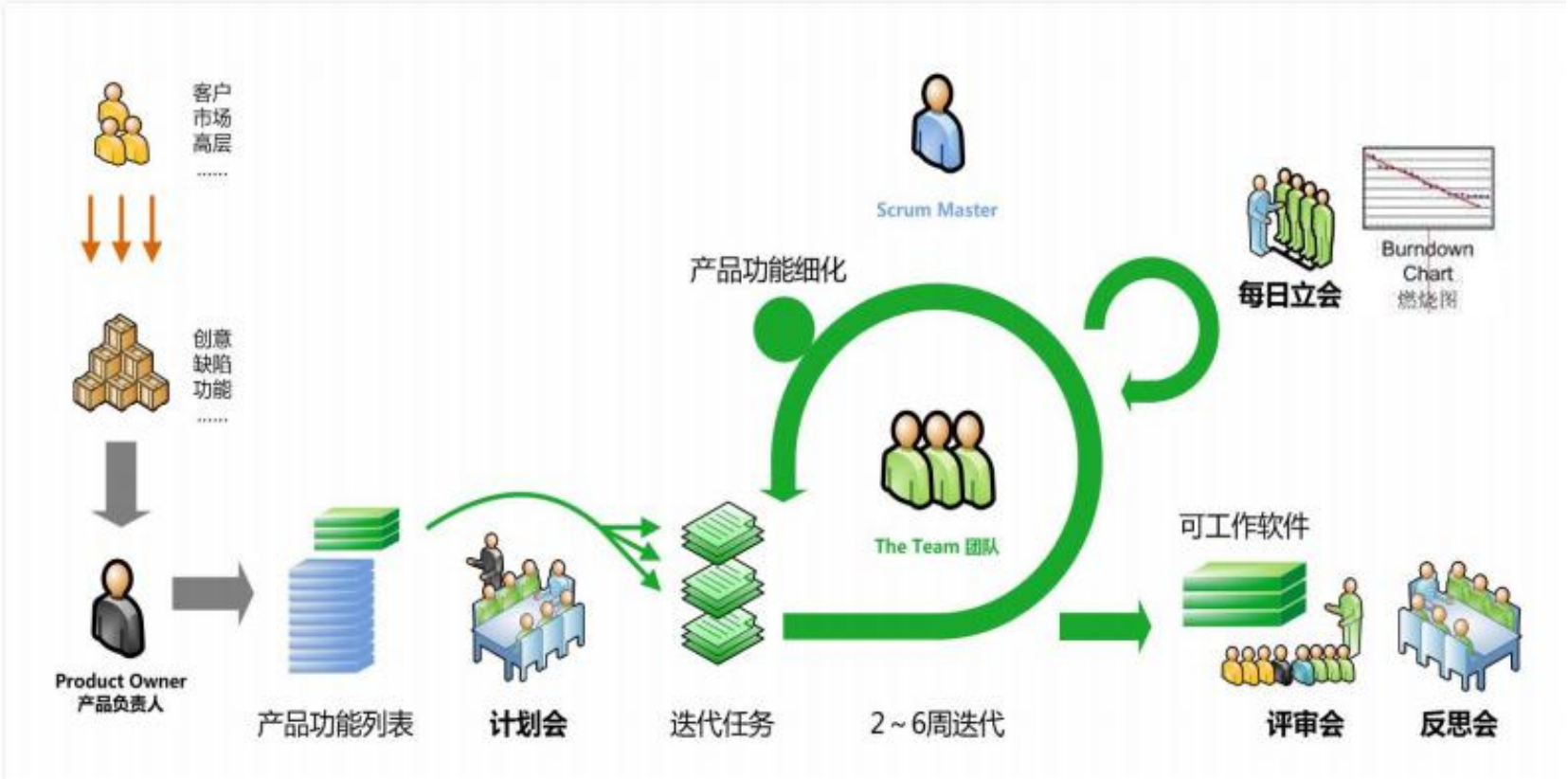
敏捷过程与Scrum

- Scrum是迭代式增量软件开发过程。Scrum将整个开发过程拆分成若干短的迭代周期，一个短的迭代周期称为一个Sprint，每个Sprint的建议长度是2到4周（互联网产品可使用1周的Sprint）。
- 在Scrum中，使用产品Backlog来管理产品的需求，是一个按照商业价值排序的需求列表，体现形式通常为用户故事。Scrum团队总是先开发具有对用户具有较高价值的需求。
- 挑选的需求在Sprint计划会议上经过讨论、分析和估算得到相应的开发列表，称为Sprint Backlog。
- 在每个迭代结束时，Scrum团队提交可交付的产品增量。

Scrum流程

Scrum由一组迭代周期组成，每个迭代(sprint)为2-4周，都依次有：planning（计划），daily standup meeting(每日站会)，review(评审)和retrospective（回顾)会议组成。

scrum的优点：快速迭代，这对于快速发展的互联网产品来说事十分必要的。



每日立会

会议目的：

- 团队在会议中做计划，协调其每日活动，还可以报告和讨论遇到的障碍。
- 任务板帮助团队聚焦于每日活动上，应在这个时候更新任务板和燃尽图。

每个团队成员需要回答三个问题：

- 上次例会后完成了什么？
- 遇到了什么困难（或障碍）？
- 下次例会前计划做什么？

*特点：简短，一般不超过**15~20**分钟

Scrum的特点

- 团队自组织；
- 通过迭代增量交付可工作的软件；
- 项目进展对所有人公开可见；
- 交付过程中持续的检视和调整；
- 商业价值驱动，以客户为导向；
- 时刻关注质量，降低长期成本；

Scrum框架

三个角色

产品负责人

Scrum Master

Scrum团队成员

六个时间箱

Sprint

发布计划会议

迭代计划会议

每日站立会议

Sprint评审会议

迭代回顾会议

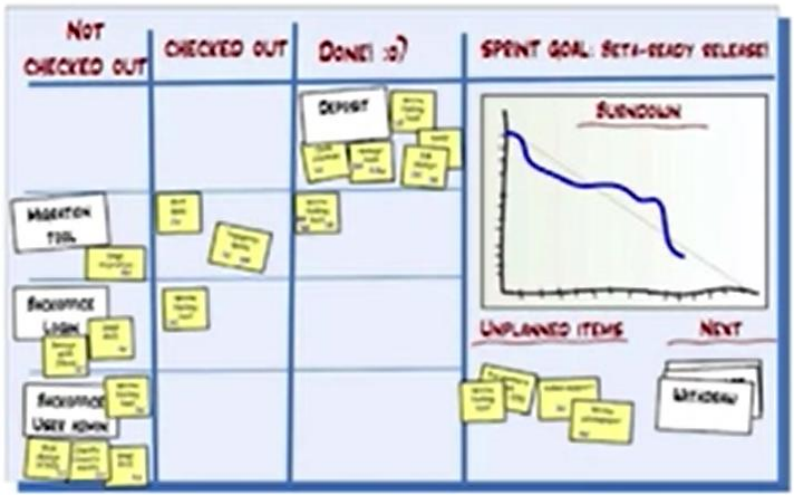
四个工作

产品Backlog

发布燃尽图

Sprint Backlog

Sprint燃尽图



什么是敏捷方法（来自知乎）

首先，敏捷开发是一种过程控制论，通俗的说，就是一种做事情的方法。

1. 它适用于软件，因为软件是软的，可以改。要是硬件，改起来就没那么方便了

2. 它适用于客户不知道自己要啥的情况，其实，这样的客户占绝大多数。因为客户不知道要啥，所以你需要不断帮客户弄明白他到底想要啥。。。换句话说，你需要和客户沟通，合作，倾听反馈，持续改进。。。

3. 它适用于竞争激烈的市场，这样的情况下，赶在竞争对手前交付一个不完美但至少能用的产品非常重要。

4. 它适用于快速变化的市场，你在埋头造一辆汽车的时候，客户已经想开飞机满天飞了，这就需要你能一步步的把汽车改成飞机，还能按时交付。

5. 它适用于在一个地方办公的小团队，一般10个人以内。这样能使敏捷中主要的沟通方式“Face to Face”是可行的。

其次，敏捷开发是一套工具集，里面有形形色色的工具，你可以不搞敏捷，但可以用那么一两个来提高工作效率。

比如：

1. 站会：三个问题，简洁有效的小团队沟通方式
2. 看板：直观反映工作进度，反映流程遵守情况，反映流程缺陷
3. 演示，计划，反思会：适合于小团队的协作和优化反馈方式
4. 用户故事：站在用户的角度讲需求
5. 持续集成：随时高质量交付的基础，有利于应对变化剧烈的市场

什么是敏捷方法（来自知乎）

再其次，敏捷开发是一种企业管理方式
比如：
1. 一线员工可以同时是架构师，Scrum Master，开发工程师，测试工程师，发挥了他的主观能动性，有利于创新和效率
2. 敏捷不专注于敏捷团队中个人的绩效考核，而更多的侧重于整个团队的绩效，更好的避免了KPI驱动模式。
3. 把大项目拆分成小项目去做（每个Sprint都是一个迭代，需要输出一个高质量的版本，相当于完成一个小项目），把bug的生存期控制在一个迭代以内，降低了风险，也减少了后期改bug的工作量。
4. 把数十人的大team 分成几个敏捷团队，这几个敏捷团队的Scrum Master/PO再组成一个更高一级的敏捷团队，利用站会，反思，看板等等敏捷元素，可以避免数十份邮件也不能解决一个小问题，大家互相踢皮球，沟通不畅的大企业病。
5. 老板可以是最大的PO（ Product owner 产品经理），他给下面的高管讲idea(User Story)，定期检查Demo，把控产品用户体验，负责和外界的沟通合作-----比如乔布斯，360的周鸿祎等。

第一章小结

- 什么是软件？软件的特点是什么？本质特征是什么？
- 软件危机的现象是什么？
- 深刻理解软件危机产生的原因。
- 软件工程基本概念，关注焦点是什么？
- 软件的生命周期模型。

课堂练习

进一步理解概念

思考以下问题如何解决：

- 一个医疗诊断仪器的实时控制系统，计划在9个月内推向市场。
- 在进行了仔细的估算和风险分析之后，软件项目管理者的得到的结论是在现有人员条件下，需要14个月的时间才能完成。
- 项目管理者下一步该怎么办？

如果市场时间错过了，软件项目自身可能会变得毫无意义！

你的对策：

思考题1

- 你以前开发过多少行程序？软件的过程是什么？你认为正确的软件开发过程是怎样的？

- 你对以下知识了解多少？

  计算机的基础知识：

  （内存管理，文件系统知识，进程控制等 OS的知识）

  Unix/Linux（操作系统）　TCP/IP的知识（网络）

  面向对象的概念，语言（Java/C++、python等）
  各种脚本语言（csh、awk、perl等）。

  客户/服务器体系结构等。

  找到方向，多写代码多练手。

思考题2

- 通过本次课程纠正了你哪些误解？
- 介绍某个前沿的软件领域的最新进展。推荐可选的领域包括：人工智能，虚拟现实，人工神经网络，高级人机界面，云计算，大数据等。

第一次作业

- 阅读第一章课外资料，了解中国软件的发展历史以及现状，特别关注国家信创产业的发展情况。
- 查阅资料，调查目前中国软件发展的优势有哪些方面？短板（卡脖子领域）有哪些？。
- 希望自己将来在哪个子领域获得发展？为了达成这个目标，大致的行动计划是什么？

- 根据以上调查结果，完成小报告；
- 3人以内自由组成小组， 提交电子版到以下邮箱地址：11857697@qq.com
- 提交时间：群内说明；
- 组队的情况，要说明个人分工；

选作作业：
一个稍微复杂一些的HelloWorld程序：

- **根据输入的用户名，以及当前时间，打印相应的message。**
- **用户名通过命令行输入，当前时间通过系统函数获得。**

- 例如对于用户：Maggie ，
- 当当前时间位于6 ~12之间：打印“Hi, Maggie, Good morning!”
- 当当前时间位于12 ~19之间：打印“Hi, Maggie, Good afternoon!”
- 当当前时间位于19 ~第二天6点之间：打印“Hi, Maggie, Good night! ”

- **注意边界值，深色字体标示的时间表示包含这个边界。**
- 编码时注意对软件质量的把控！

需要考虑将来可能的功能扩展：

- 当要增加“Good evening!”，代码是否具有扩展性。
- 再增加午夜问候“Good midnight!”时？

- 除此之外，用户希望在一些特殊的节日，如新年问候“Happy new year!”、情人节问候“Happy valentine's day!”、三八妇女节问候“Happy women's day!”，等等。
- 除了已经列出的节日，他们还希望临时添加一些特殊的日子，因此问候语需要形成一个库，并支持动态添加。

谢谢!

END