软件质量保证与测试

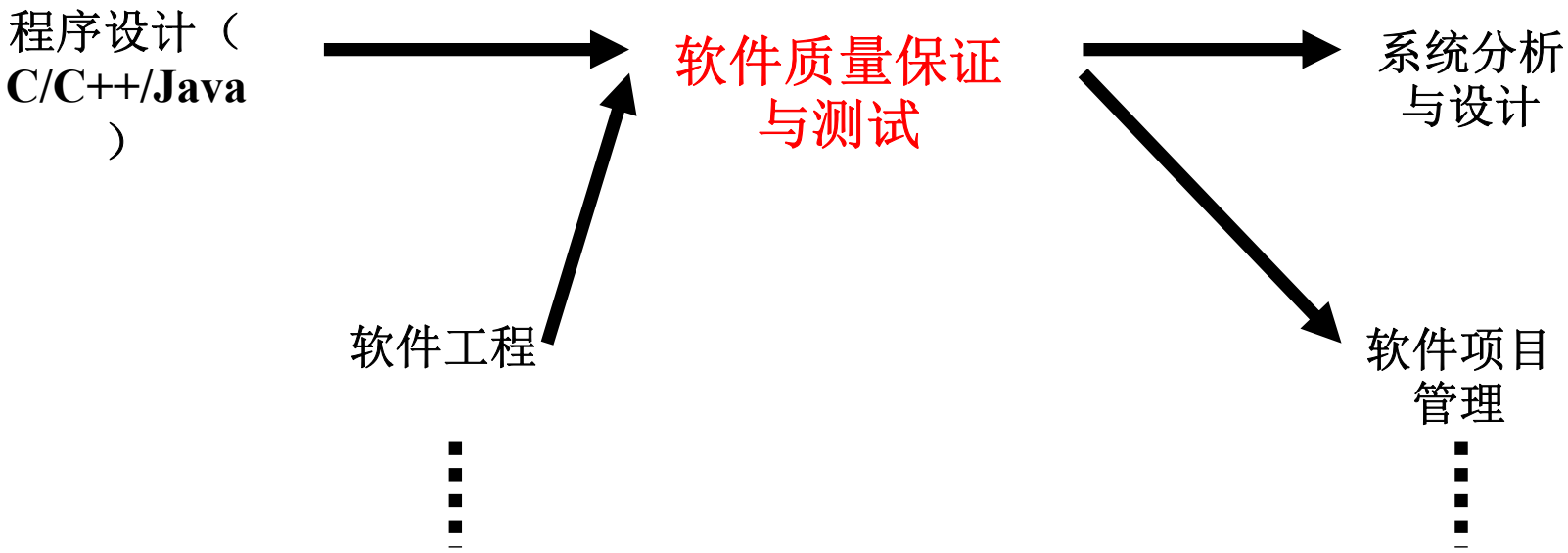
——软件测试、调试、重构、Design Patterns基础

User : testing_software@163.com
Password: testing

任课教师：侯鲲

blueblоodhk@163.com

课程体系



关于课程名称

软件质量保证——Software Quality Assurance (SQA)
一系列用于监控软件工程过程的方法，以保证软件的质量。

软件测试——Software Testing
一种用来促进被测软件的正确性、完整性、安全性、和品质的过程。

关于课程名称

软件质量保证与软件测试的关系

- 软件质量保证是建立达到良好软件质量的标准及开发过程，检查和评价当前软件开发过程，并设法达到防止软件出现错误的目标。
- 软件测试是软件质量保证的关键步骤。
- 软件质量保证与测试的关系就像“健康、长寿秘诀”与“到医院体检及看病”的关系。

关于课程名称

软件质量保证和测试的关系

QA + Testing = good software

质量保证+测试=好的软件

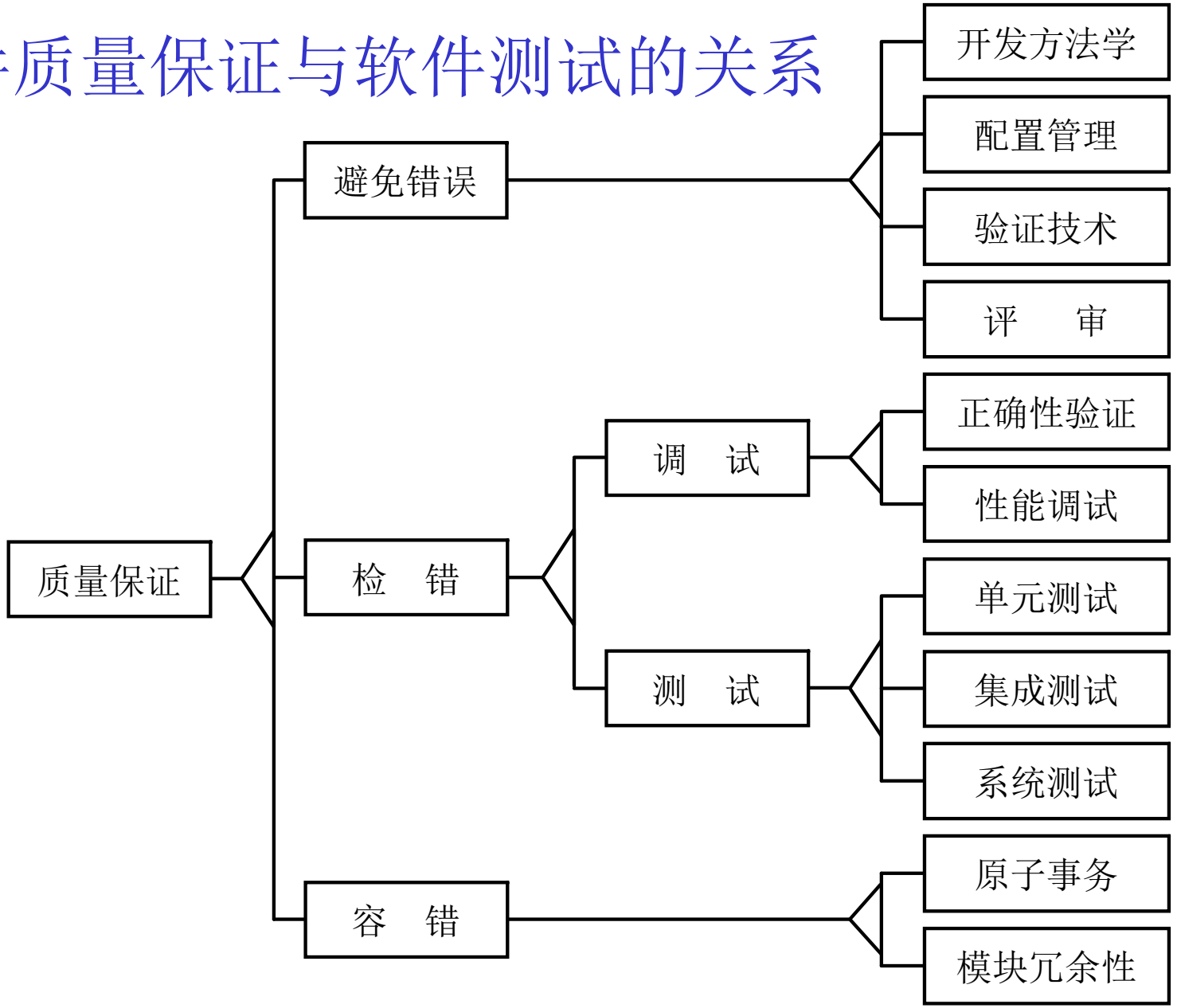
软件质量的特征，是每个人在某种条件之下需要它，每个人都觉得自己理解它，却又不愿意解释它。

关于课程名称

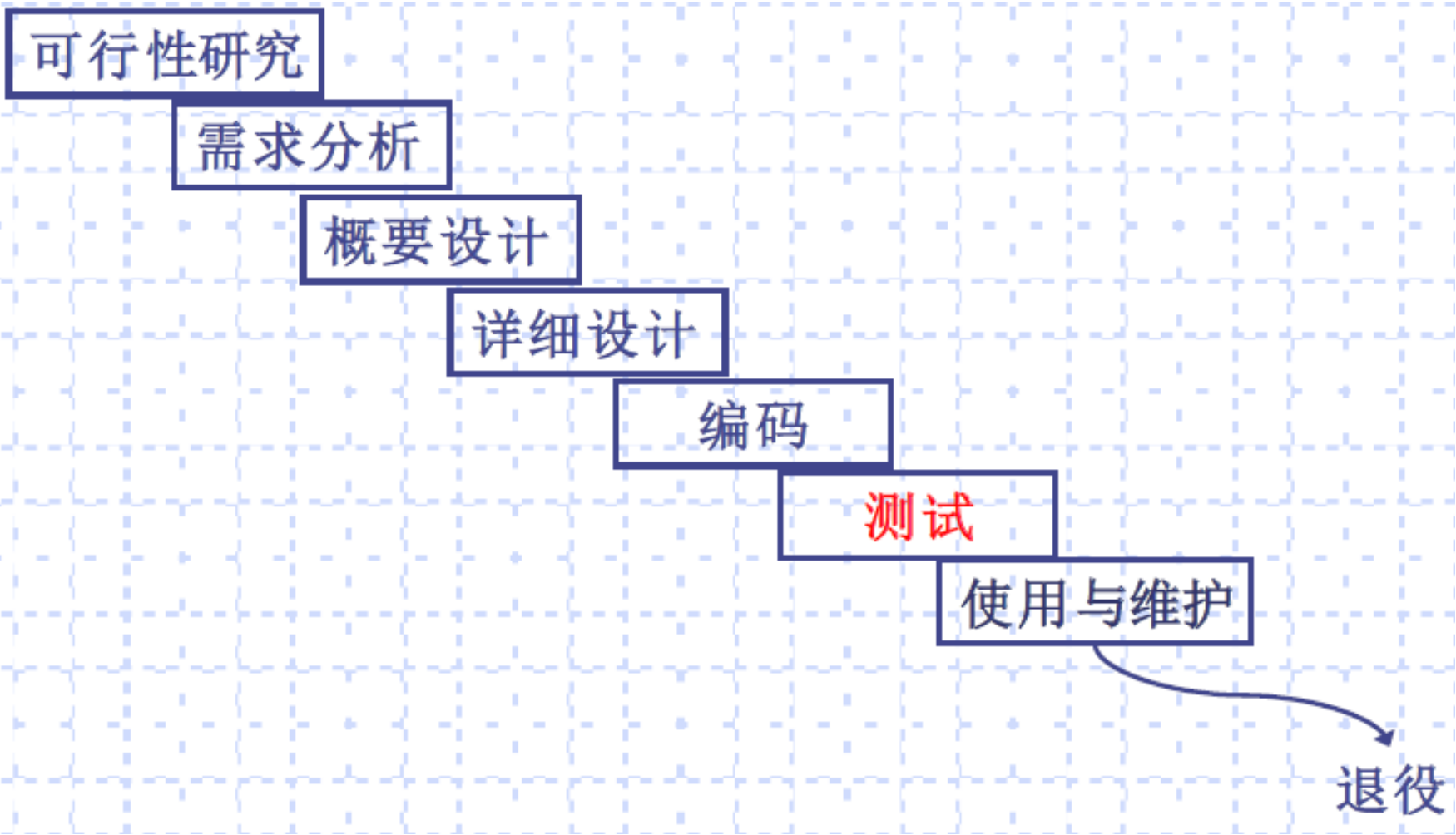
质量保证和质量控制

关于课程名称

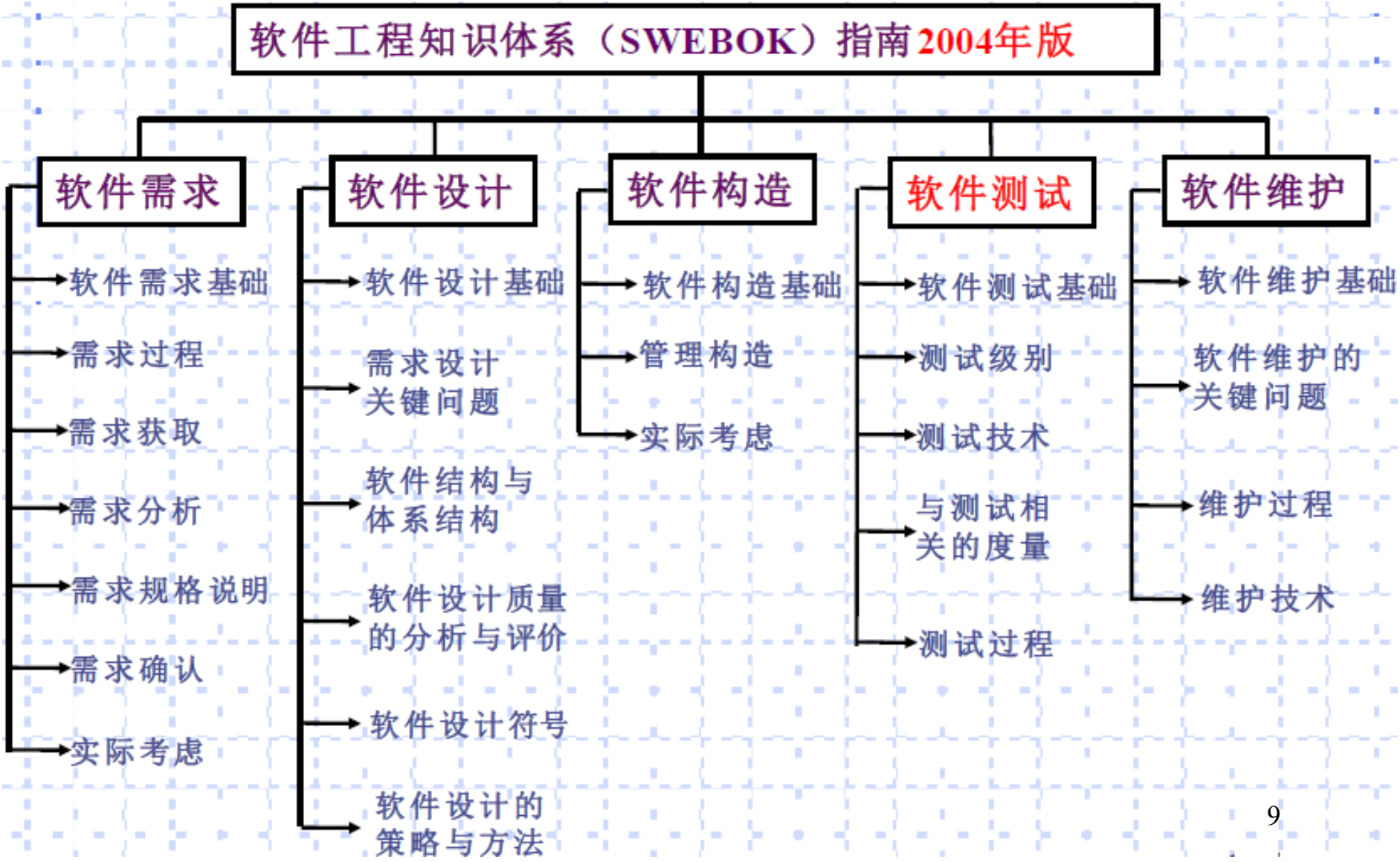
软件质量保证与软件测试的关系



软件测试在软件工程中的地位



软件测试在软件工程中的地位

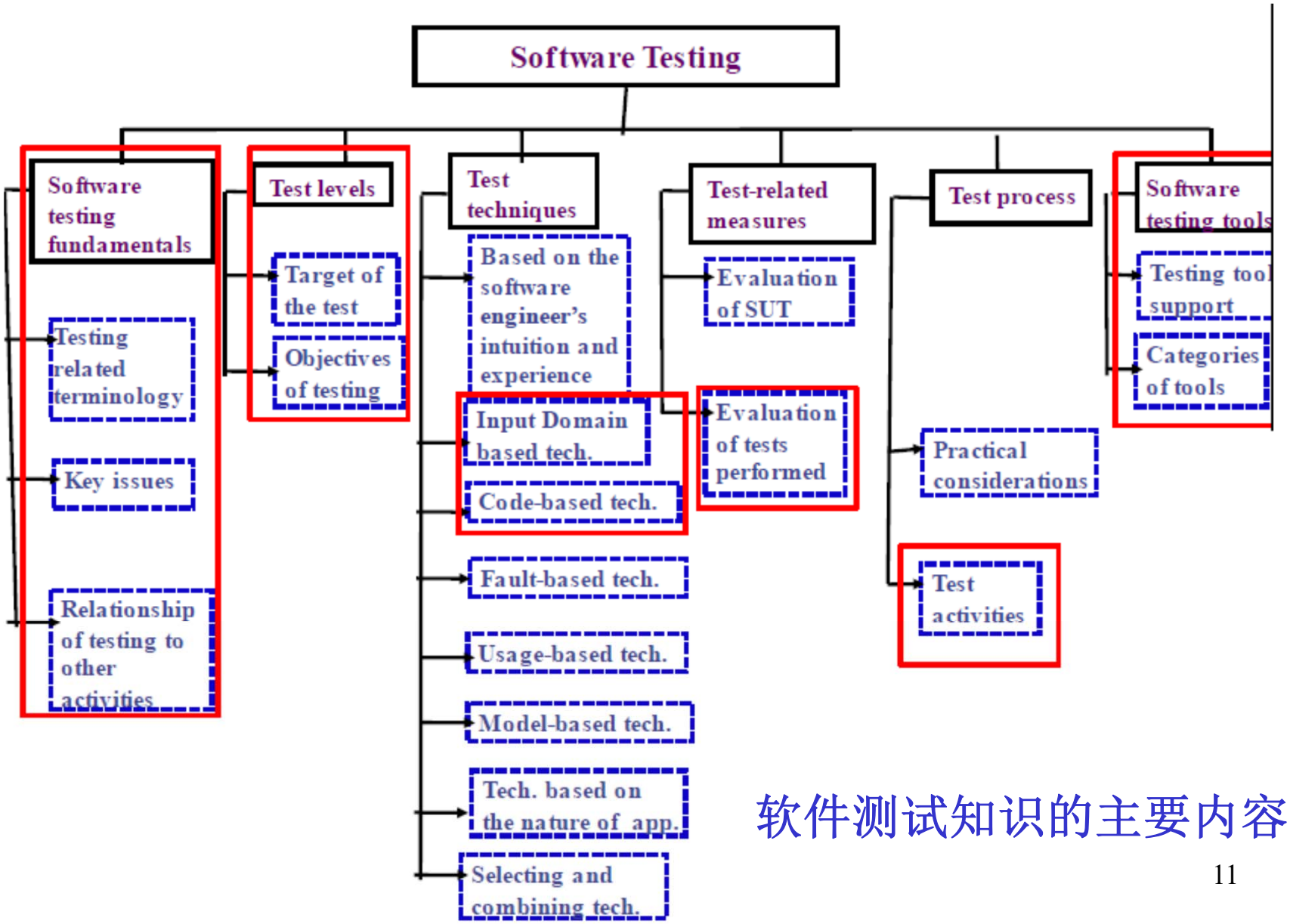


软件测试在软件工程中的地位

软件工程知识体系15个知识点（SWEBOK指南2014年版）

Software Requirements
Software Design
Software Construction
Software Testing
Software Maintenance
Software Configuration Management
Software Engineering Management
Software Engineering Process
Software Engineering Models and Methods
Software Quality
Software Engineering Professional Practice
Software Engineering Economics
Computing Foundations
Mathematical Foundations
Engineering Foundations

软件测试在软件工程中的地位



软件测试知识的主要内容

课程主要内容

- 软件质量保证
  - 软件质量管理
  - 软件质量标准和度量
  - 软件评审
  - 高质量编程*
- 软件测试基础
  - 黑盒测试与白盒测试*
  - 代码评审和静态分析
  - 各级别测试
  - 自动测试和测试工具*
  - 测试管理

课程目标

1. 了解软件质量保证和测试的基本概念
2. 了解软件测试的完整知识体系，包括测试的概念、原则，各种测试技术、测试级别、测试应用、测试工具、测试计划、测试过程和测试结果分析等
3. 掌握软件测试的各种技术，静态、动态测试，白盒、黑盒测试等
4. 了解（使用）常用的测试工具。

用“测试意识”知道编程和软件开发！

教学与考核方式

- 时间：周二 （5-6上课）

  周学时：2

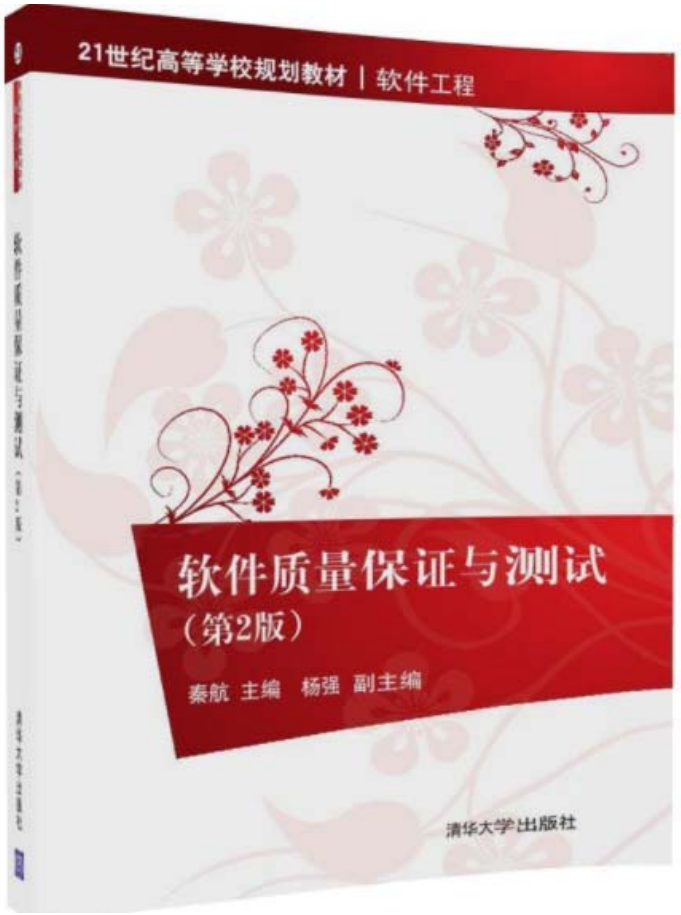
  总学时：<= 36

  方式：　课堂讲授，随堂练习

- 考核：平时成绩（作业、出勤、课程设计）占40%，期末考试占60%

  说明：作业+出勤20分，课程设计20分。

参考书

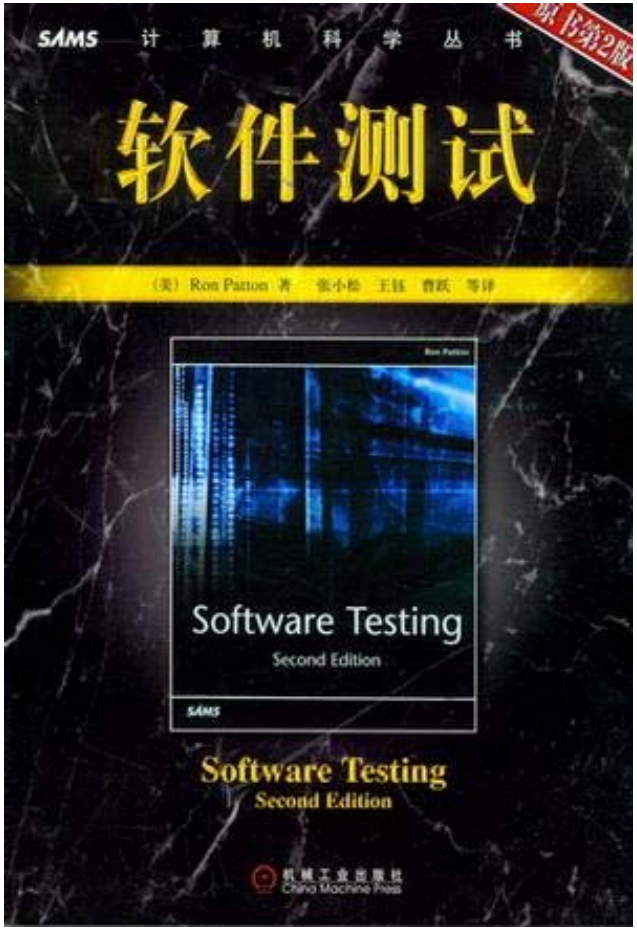


书名： **软件质量保证与测试(2nd)**

作者： 秦航

出版社：清华大学出版社

参考书



原书名： **Software Testing (2nd)**

原出版社： Sams

作者： （美）Ron Patton

出版社：机械工业出版社

参考书

软件测试-第4版

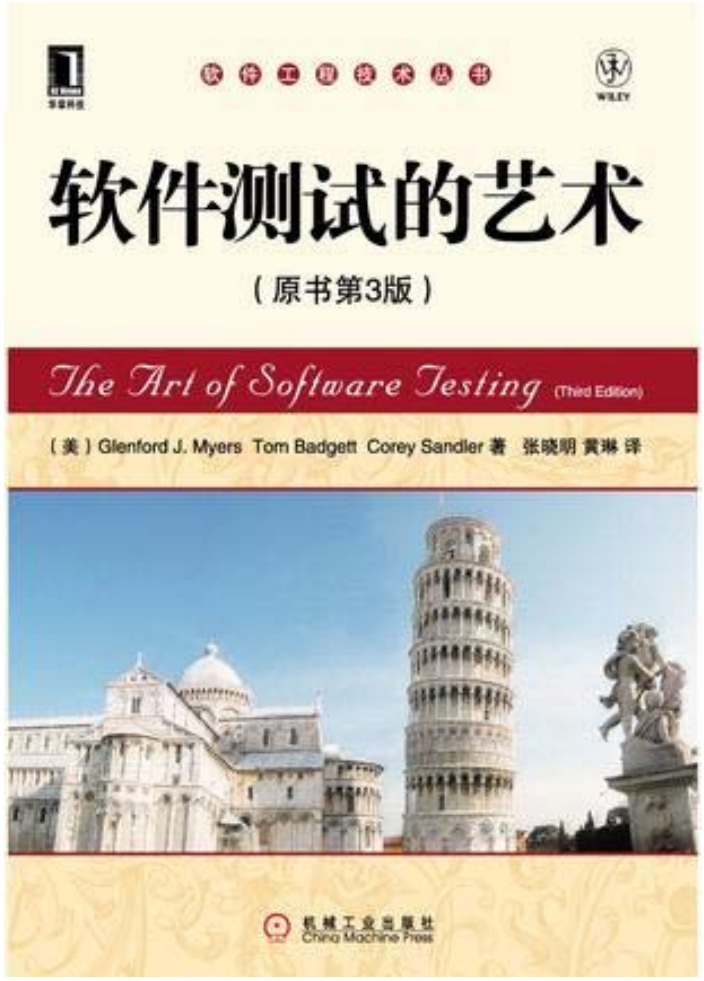
原书名： Software Testing: A Craftsman's Approach, Third Edition

作者： (美)Paul C. Jorgensen

出版社：机械工业出版社

本书曾是由ACM和IEEE计算机学会（www.swebok.org）联合编制的“软件工程知识体系”软件测试标准的主要参考文献之一。

参考书



软件测试的艺术-第3版

原书名： Software Testing: A Craftsman's Approach, Third Edition

作者： (美)Paul C. Jorgensen

出版社：机械工业出版社

本书曾是由ACM和IEEE计算机学会（www.swebok.org）联合编制的“软件工程知识体系”软件测试标准的主要参考文献之一。

参考书

其他参考书籍：

- 《软件测试与质量保证-理论与实践》，**Kshirasagar Naik**，电子工业出版社
- 《软件质量和测试》，傅兵，清华大学出版社
- 《软件测试实用教程——方法与实践(第2版)》，武剑洁，电子工业出版社
- 《质量·软件·管理》，Gerald M. Weinberg，清华大学出版社

引子

你是怎么测试你的软件的？
你发现过自己设计软件中的问题吗？

大家都应该编过程序，但是都测试过程序吗？
也许你在编程序的过程中遇到过问题，然后又找到了问题——这是测试吗？

一个有瑕疵的二分查找

```
public static int buggyBinarySearch(int[] a, int target) {
    int low = 0;
    int high = a.length - 1;
    while (low <= high) {
        int mid = (low + high) / 2;
        int midVal = a[mid];
        if (midVal < target)    low = mid + 1;
        else if (midVal > target)   high = mid - 1;
             else   return mid;
    }
    return -1;
}
```

一个有瑕疵的二分查找：修正

```
public static int buggyBinarySearch(int[] a, int target) {
    int low = 0;
    int high = a.length - 1;
    while (low <= high) {
        int mid = (low + high) >>>2;
        int midVal = a[mid];
        if (midVal < target)    low = mid + 1;
        else if (midVal > target)   high = mid - 1;
             else   return mid;
    }
    return -1;
}
```

int mid = low + ((high - low) / 2);

SQL注入（Web）

.Net平台下有如下代码（查询字符串）

String queryStr =
“**select *form Accounts where username =‘ ”** +
Request.QueryString[“username”] + “ **’and password= ‘ ”** +
Request.QueryString[“password”] + “ **’ ”** ;

目的是想让用户输入他们的信息并登陆：
select *form Accounts where username=‘HouKun**’ and password=‘**Ilovethisgame**’**
但是…….

如果用户输入：
用户名：**haha**
密码：**‘; drop table Accounts**

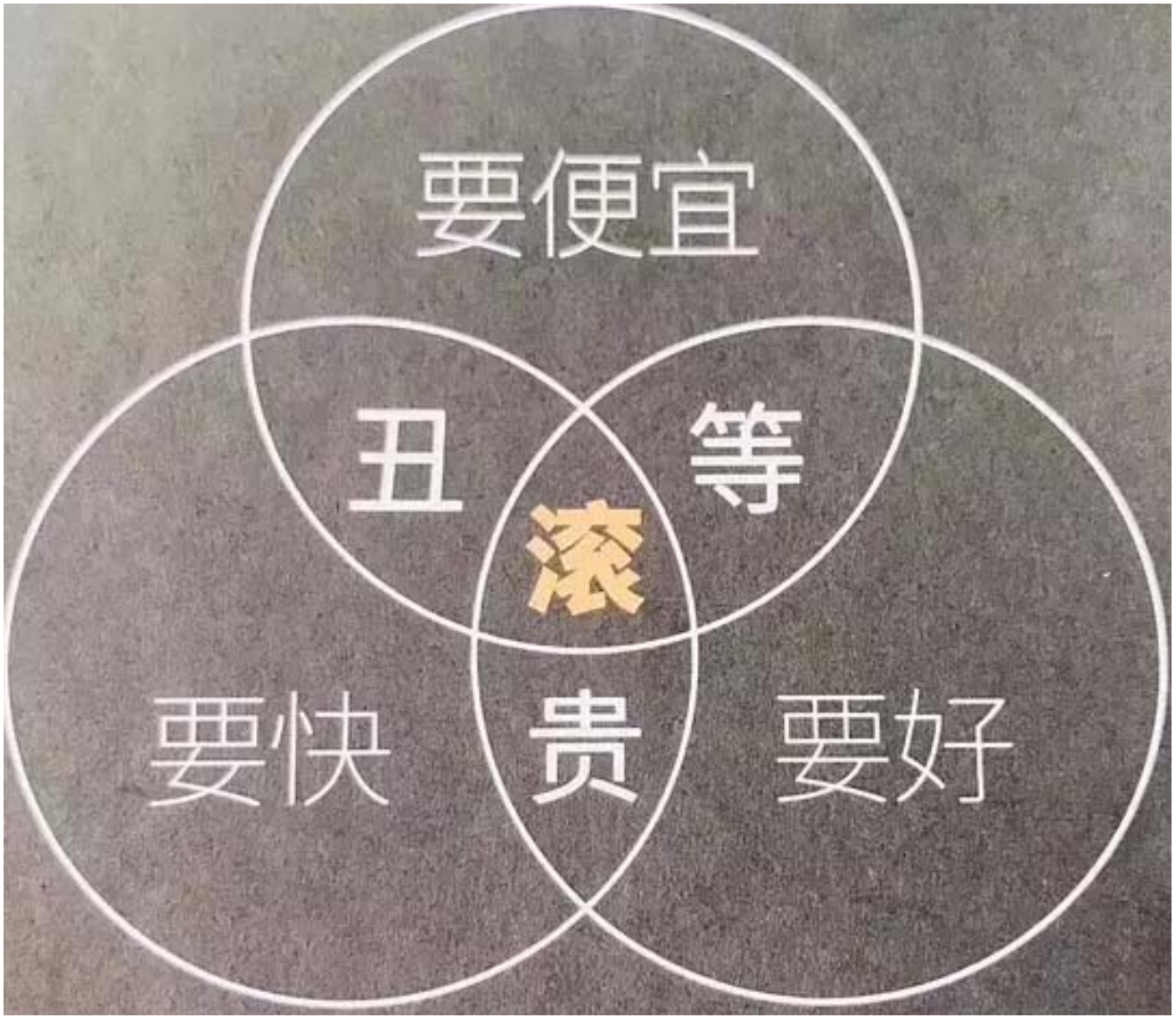
第一条进行无用查找，第二条破坏表

另外，密码输入“ **’ ; delete from Accounts** ”

select *form Accounts where username=‘haha’ and password=‘ ’; drop table Accounts

注：
“ ‘ ” ：打开和关闭数据库字符串-字符串的开始和结束
“ ; ” ：SQL语句结束符
“ -- ”：SQL注释，忽略之后内容

软件测试的背景
——软件工程和软件过程

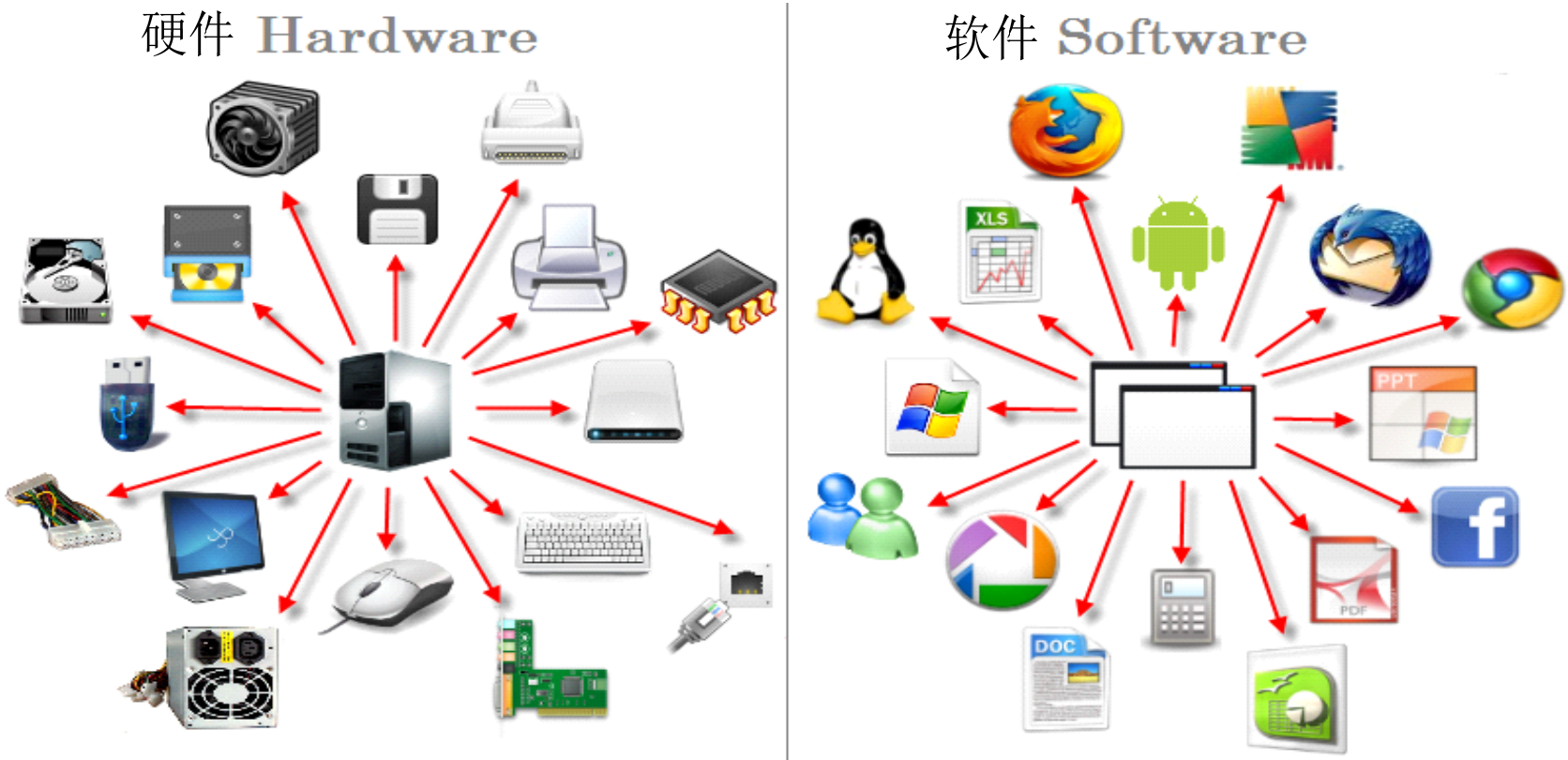

要便宜
丑
等
滚
要快
贵
要好

软件特征与软件工程

- 要理解软件的含义并全面地理解软件工程，要明确软件的特征，并据此知道软件与人类建造的其它事物之间的区别。
- IEEE定义对软件的定义如下：
  - 软件是计算机程序、规程以及可能的相关文档和运行计算机系统需要的数据。
  - 软件包含计算机程序、规程、文档和软件系统运行所必需的数据四个部分。

计算机硬件vs计算机软件

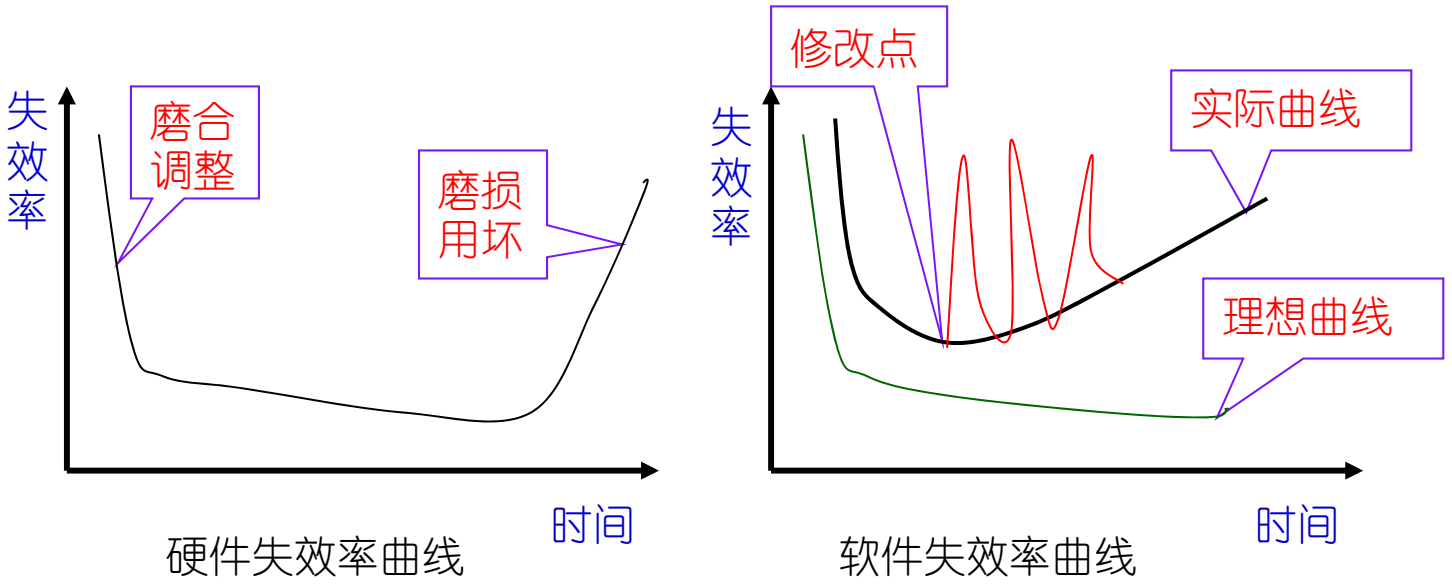
- 软件是逻辑产品，而不是物理产品，所以，软件具有和硬件完全不同的特征



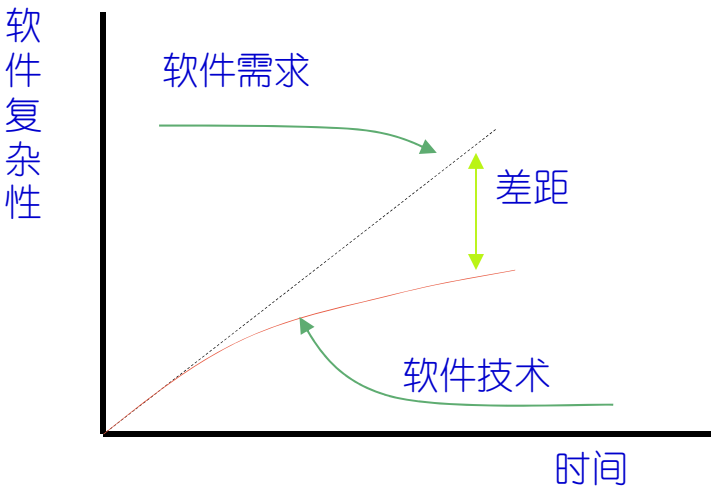
软件的概念与特点

- 软件是一种逻辑实体，而不是具体的物理实体
- 软件的生产与硬件不同
  - 在软件的运行和使用期间，没有硬件那样的机械磨损，老化问题

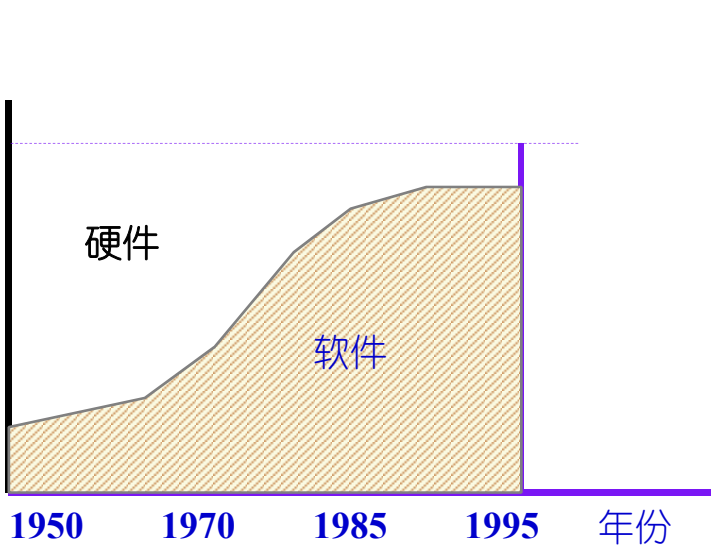
“浴缸曲线”



软件的概念与特点



软件技术的发展落后于需求



硬、软件成本比例的变化

软件的成本相当昂贵

软件危机

软件规模

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**

Windows95有1000万行代码
Windows2000有5000万行代码
项目经理约250人
开发人员约1700人
测试人员约3200人

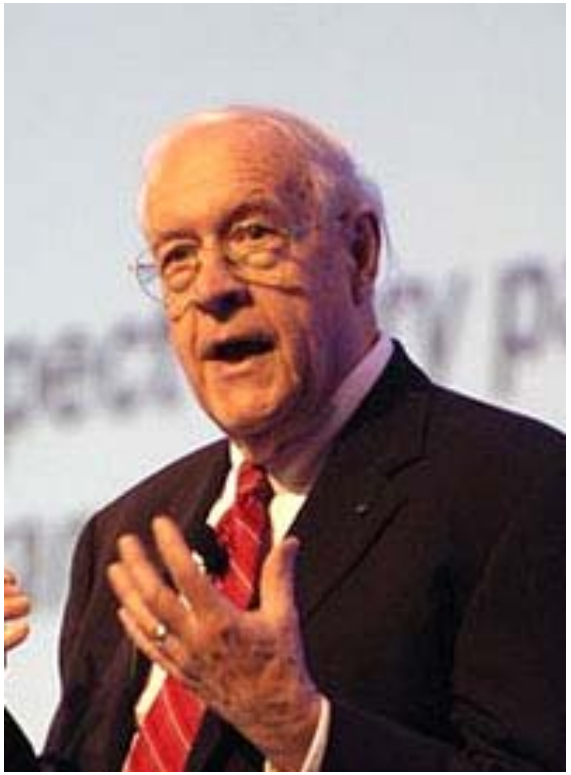
大型软件的命运-IBM System/360

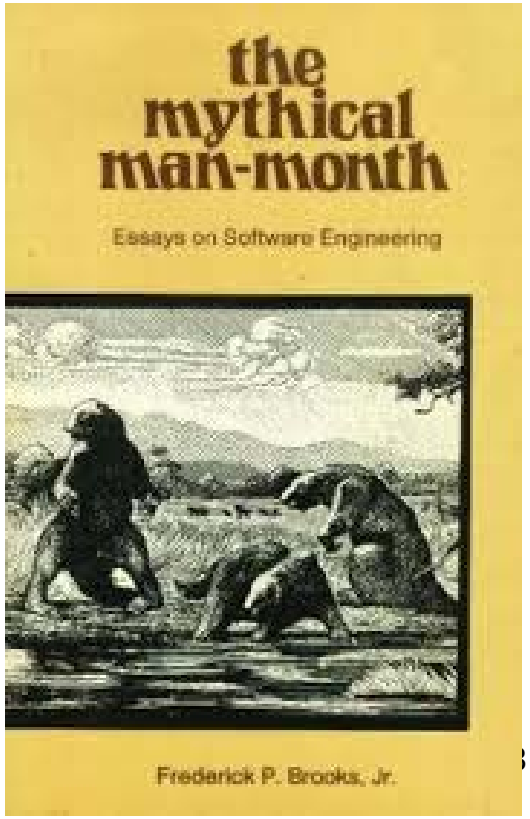
- 1964年4月7日，IBM推出了System/360系列大型主机，这个划时代的创新，改变了商业界、科学界、政府、IT界本身

因为IBM System/360的工作，佛瑞德·布鲁克斯（Frederick P. Brooks）于1999年获得图灵奖

大型软件的命运-IBM System/360

- Frederick P. Brooks把在IBM公司任System计算机系列以及其庞大的软件系统OS项目经理时的实践经验，总结成《人月神话》一书，





失败-大型软件项目的宿命？！

没有别的场景比巨兽们在焦油坑中垂死挣扎的场面更令人震撼

软件危机

软件危机的原因

- 客观：软件本身特点
- 主观：不正确的开发方法
  - 忽视需求分析
  - 错误认为：软件开发 = 程序编写
  - 轻视软件维护

软件危机

克服软件危机的途径

- 消除错误的概念和做法
- 推广使用成功的开发技术和方法
- 使用软件工具和***软件工程***支持环境
- 加强软件管理

软件工程的概念及范畴

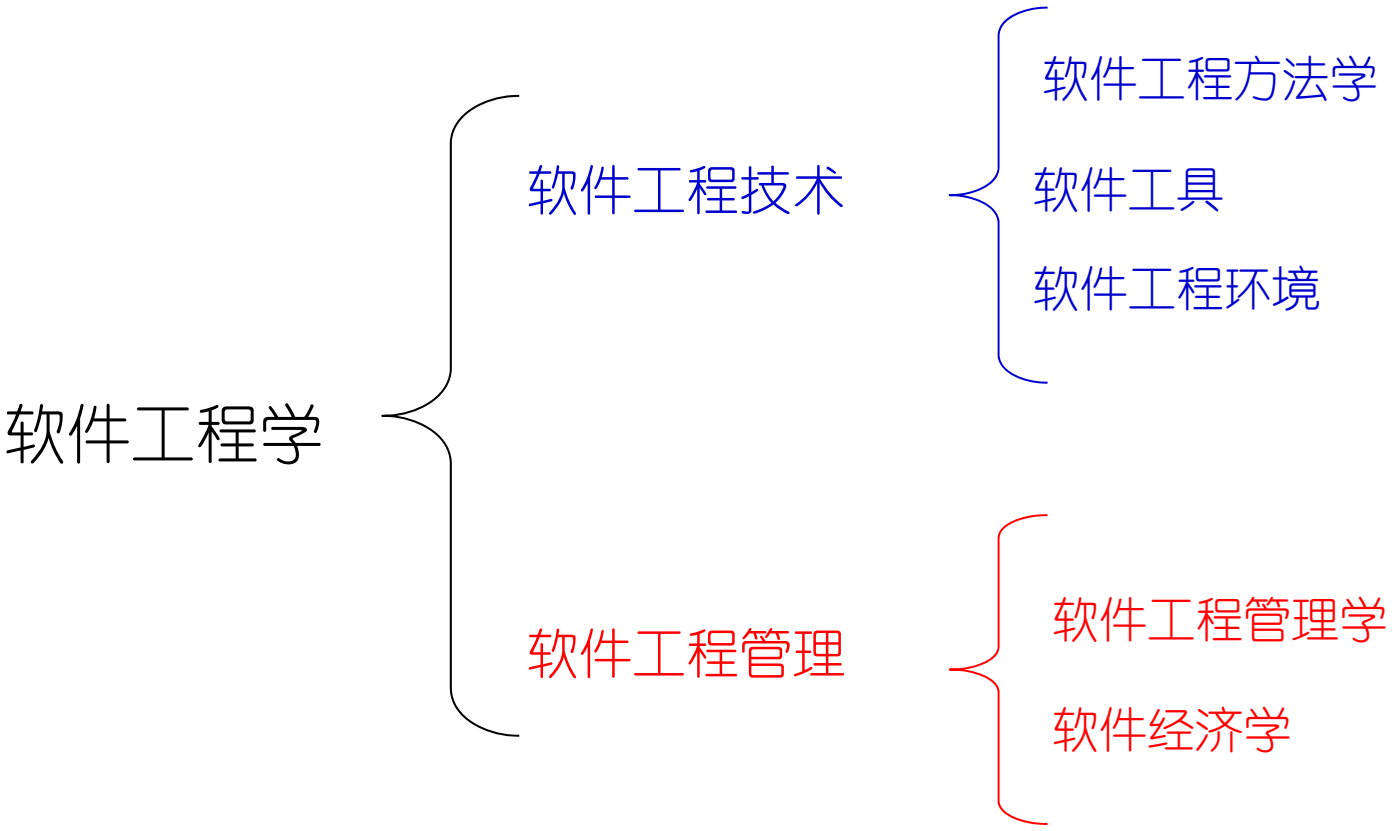
“软件工程”　----Software Engineering

于1968年 NATO 组织在德国召开的一次会议上提出

是把软件当作一种工业产品，要求 “采用工程化的　原理与方法对软件进行计划、开发和维护”。

软件工程的概念及范畴

软件工程学的范畴



软件工程的概念及范畴

软件工程方法学包含3个要素

- 方法

  完成软件开发的各项任务的技术方法。

- 工具

  工具是为运用方法而提供的自动的或半自动的软件工程支撑环境。

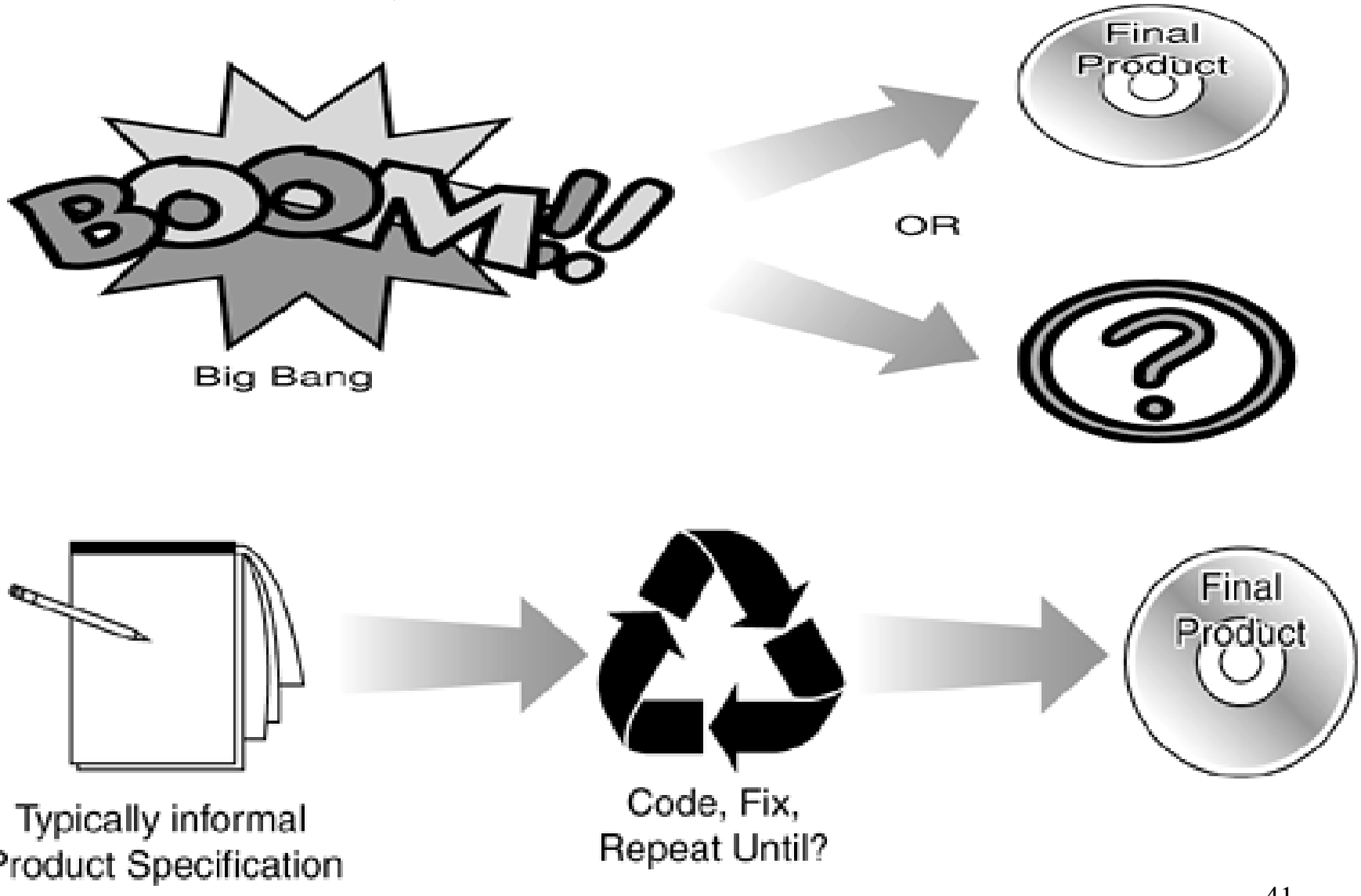
- 过程（软件过程）

  为了获得高质量的软件所需要完成的一系列任务的框架，它规定了完成各项任务的工作步骤。

软件过程（软件生命周期）

- **软件过程**：是为了获得高质量软件所需要完成的一系列任务的框架，它规定了完成各项任务的工作步骤。
- 通常使用**生命周期模型**简洁地描述软件过程。生命周期模型规定了把生命周期划分成哪些阶段及各个阶段的执行顺序，因此也称为**过程模型**。
- 软件过程本身就是软件
  软件过程是一种被由人构成的虚拟机执行的软件。

“大爆炸”和“编写边改”


BOOM!!
Big Bang
Final Product
OR
Typically informal Product Specification
Code, Fix, Repeat Until?
Final Product

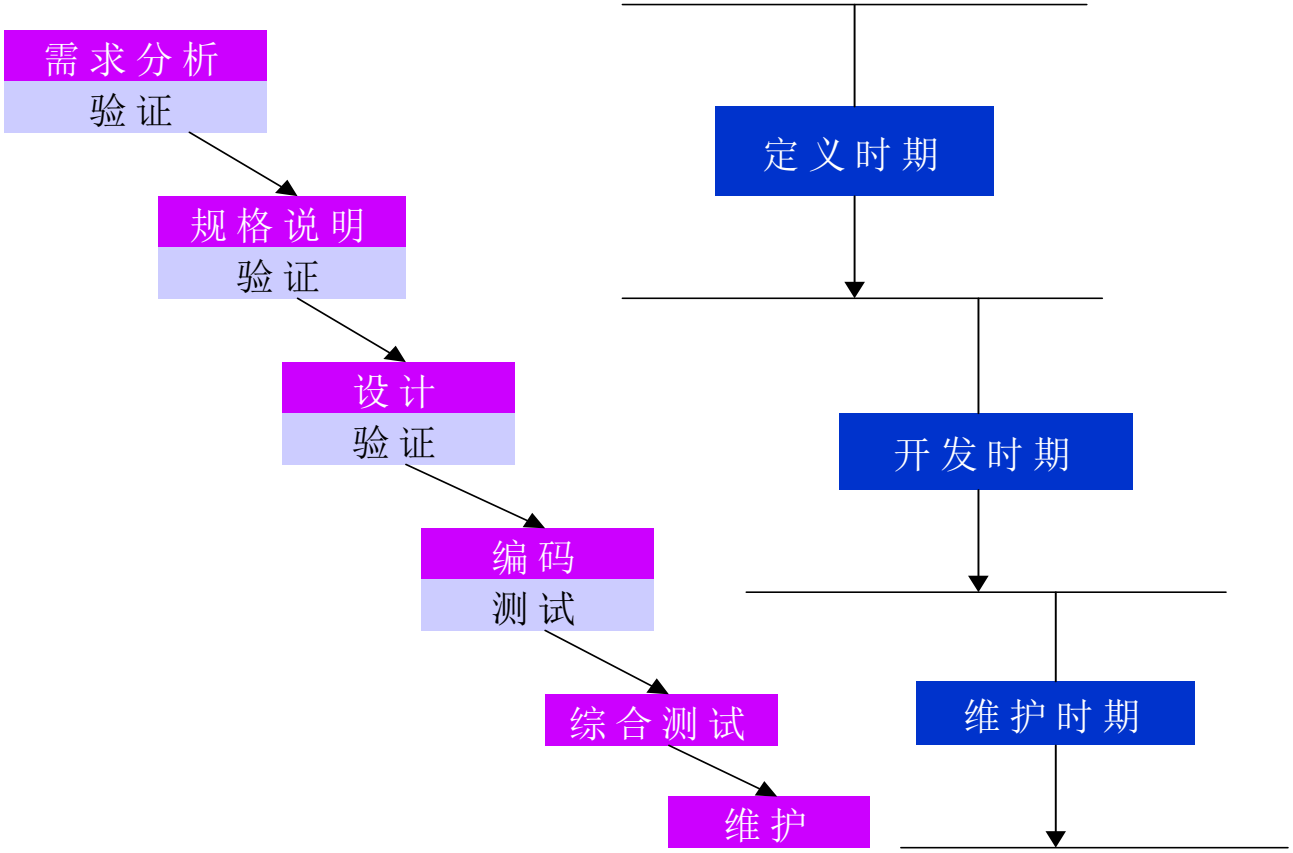
软件开发过程

基本的计划性软件过程（软件生命周期）

- 分析：（需求获取、分析/软件分析）
- 设计：
- 编码：（实现）
- 测试：
- 部署：（发布）
- 维护：

瀑布模型

瀑布模型 — 传统的瀑布模型



基于活动分解项目

瀑布模型

瀑布模型基本思想

- 将软件开发过程划分为分析、设计、编码、测试等阶段。
- 软件开发要遵循过程规律，按次序进行。
- 每个阶段均有里程碑和提交物。
- 工作以线性方式进行，上一阶段的输出是下一阶段的输入。

瀑布模型

瀑布模型的优点

- 简单、易懂、易用。
- 为项目提供了按阶段划分的检查点，项目管理比较规范。
- 每个阶段必须提供文档，而且要求每个阶段的所有产品必须进行正式、严格的技术审查。

瀑布模型

运用瀑布模型遇到的问题

- 实际的项目很少遵守瀑布模型提出的顺序。
- 客户通常难以清楚地描述所有的需求。
- 客户必须要有耐心，只有在项目接近尾声的时候，他们才能得到可执行的程序。

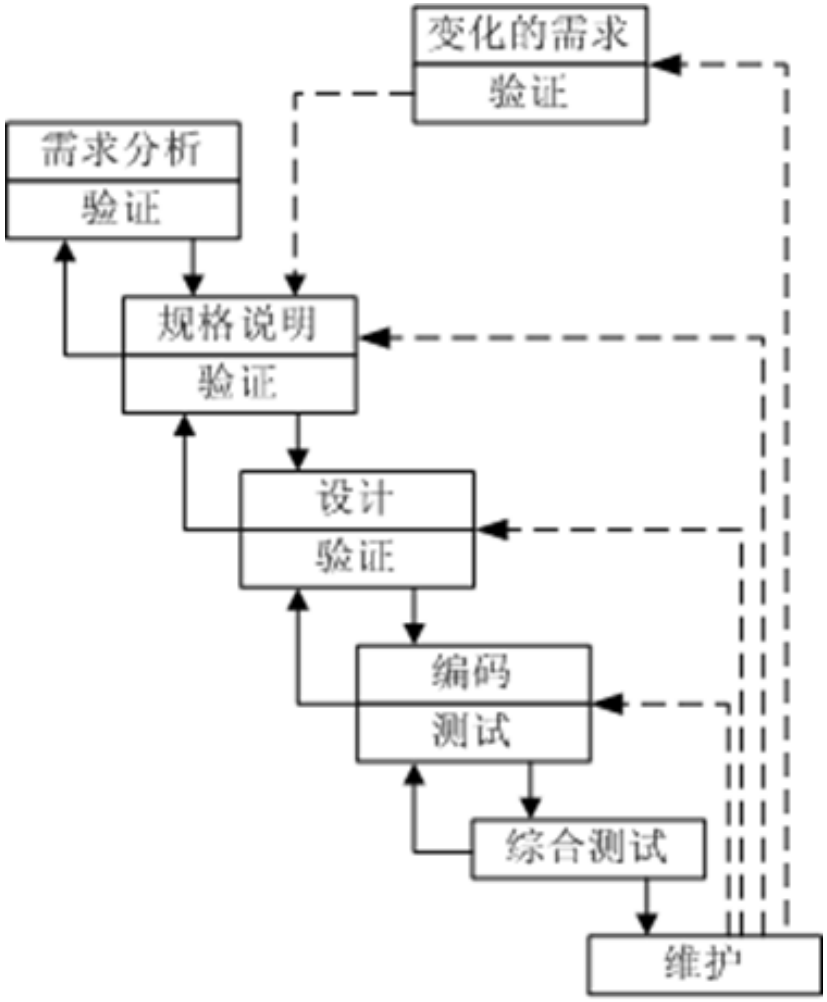
瀑布模型

瀑布模型的适用场合

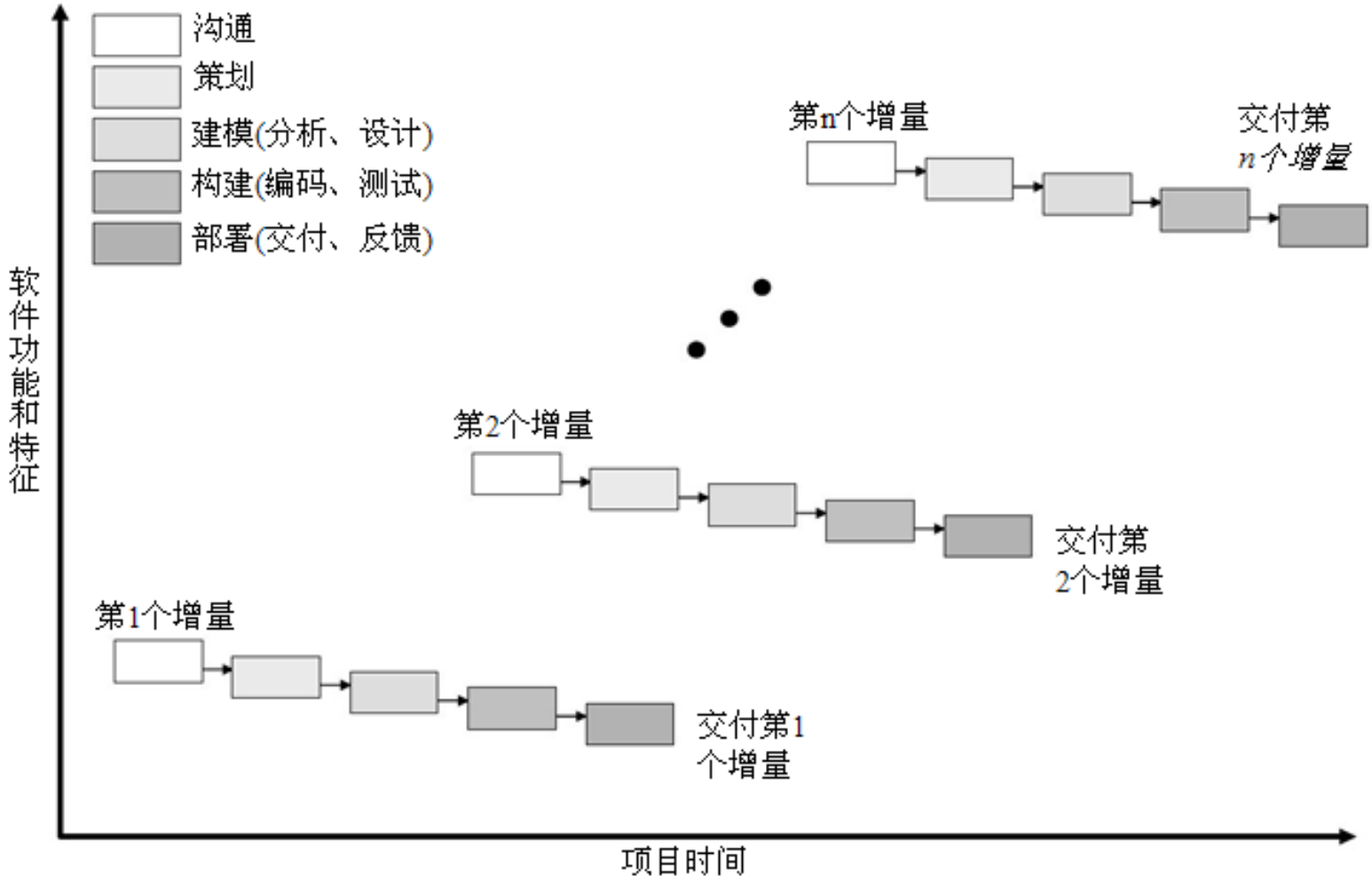
- 需求相当稳定，客户需求被全面的了解风险管理。
- 开发团队对于这一应用领域非常熟悉。
- 外部环境的不可控因素很少。
- 小型清晰的项目或长周期的项目。

瀑布模型

瀑布模型——实际的瀑布模型：



增量模型


沟通
策划
建模(分析、设计)
构建(编码、测试)
部署(交付、反馈)
第n个增量
交付第
n个增量
第2个增量
交付第
2个增量
第1个增量
交付第1
个增量
软件功能和特征
项目时间

增量模型

增量模型的使用方法

- 软件被作为一系列的增量来进行开发，每一个增量都提交一个可以操作的产品，可供用户评估。
- 第一个增量往往是核心产品：满足了基本的需求，但是缺少附加的特性。
- 客户使用上一个增量的提交物并进行自己评价，制定下一个增量计划，说明需要增加的特性和功能。
- 重复上述过程，直到最终产品产生为止。

增量模型

增量模型应用举例

- 应用举例：开发一个类似于Word的字处理软件。
  - 增量1：提供基本的文件管理、编辑和文档生成功能。
  - 增量2：提供高级的文档编辑功能。
  - 增量3：实现拼写和语法检查功能。
  - 增量4：完成高级的页面排版功能。

增量模型

增量模型的优点

- 提高对用户需求的响应：用户看到可操作的早期版本后会提出一些建议和需求，可以在后续增量中调整。
- 人员分配灵活：如果找不到足够的开发人员，可采用增量模型，早期的增量由少量人员实现，如果客户反响较好，则在下一个增量中投入更多的人力。
- 可规避技术风险：不确定的功能放在后面开发。

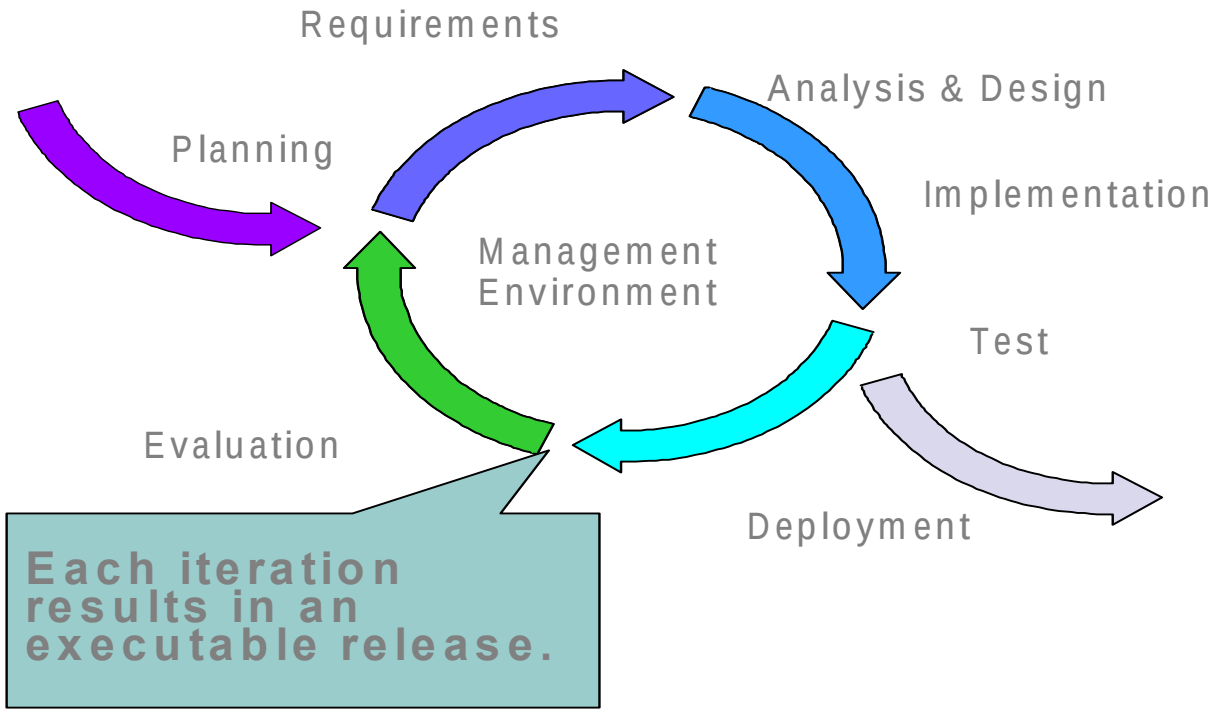
增量模型

增量模型存在的问题

- 每个附加的增量并入现有的软件时，必须不破坏原来已构造好的东西。
- 加入新增量时应简单、方便 ——该类软件的体系结构应当是开放的。
- 仍然无法处理需求发生变更的情况。
- 管理人员须有足够的技术能力来协调好各增量之间的关系。

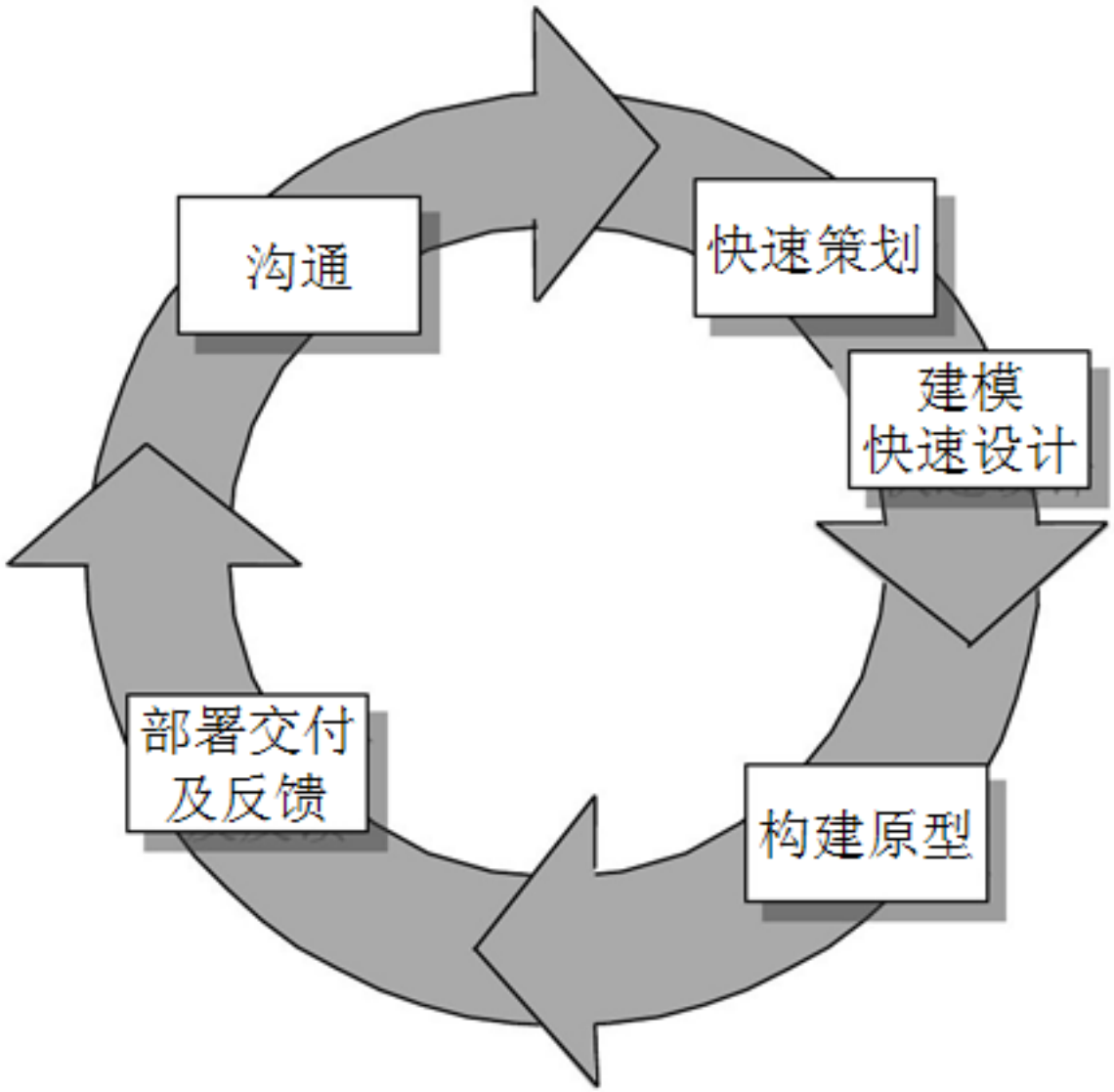
迭代模型

迭代模型(Cont.) ——一般形式



（快速）原型模型

一种迭代模型



（快速）原型模型

原型模型的使用方法

- 使用方法一：
  - 步骤1：开发人员与客户双方通过沟通，明确已知的需求，并大致勾画出以后再进一步定义的东西。
  - 步骤 2：迅速策划一个原型开发迭代并进行建模，主要集中于那些最终用户
  - 所能够看到的方面，如人机接口布局或者输出显示格式等；
  - 步骤3：快速设计产生原型，对原型进行部署，由客户和用户进行评价；
  - 步骤4：根据反馈，抛弃掉不合适的部分，进一步细化需求并调整原型；
  - 步骤5：原型系统不断调整以逼近用户需求。
- 使用方法二：把原型系统作为需求分析的工具，明确需求后，原型系统被抛弃。

（快速）原型模型

原型开发应用举例

- 应用举例：开发一个教务管理系统。
  - 第一次迭代：完成基本的学籍管理、选课和成绩管理功能。（6周）
  - 客户反馈：基本满意，但是对大数据量运行速度慢效率，不需要学生自己维护学籍的功能等。
  - 第二次迭代：修改细节，提高成绩统计和报表执行效率（2周）。
  - 客户反馈：需要严格的权限控制，报表打印格式不符合要求。
  - 第三次迭代：完善打印和权限控制功能。（2周）
  - 客户反馈：可以进行正式应用验证。

（快速）原型模型

原型开发的优点

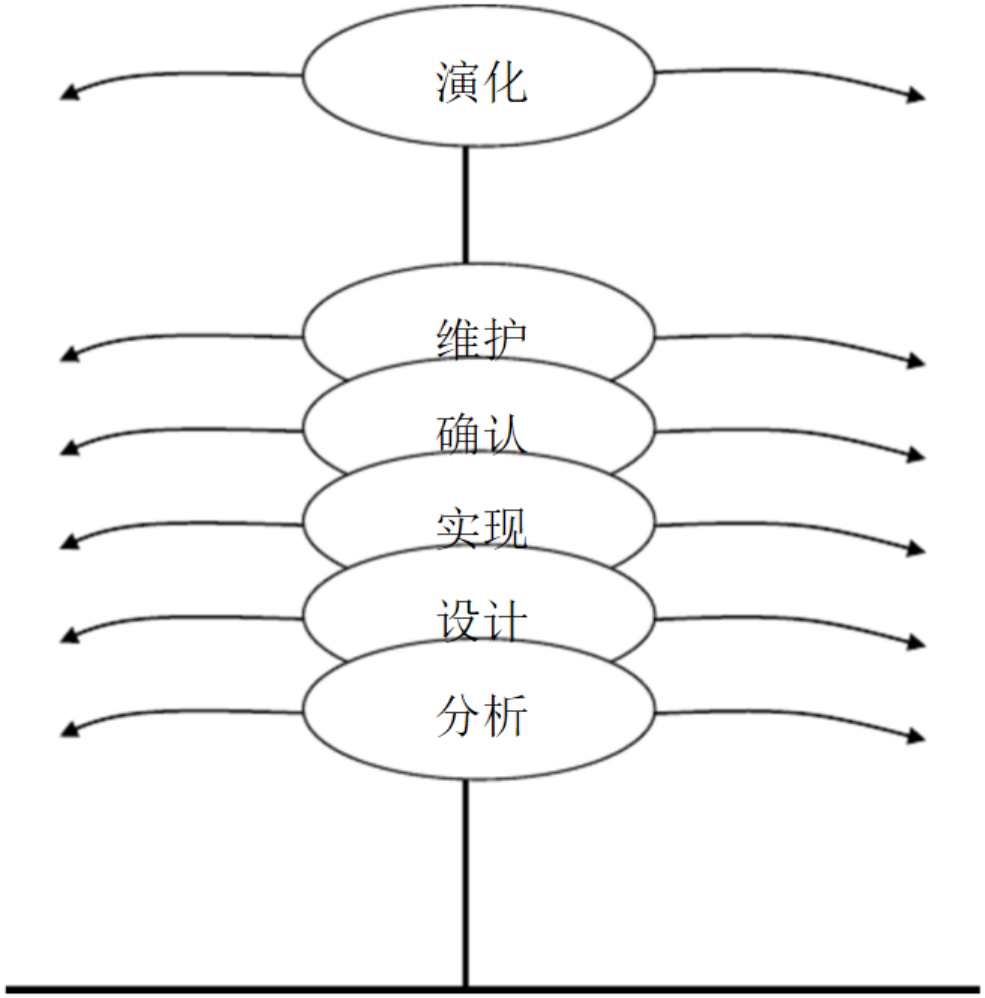
- 快速开发出可以演示的系统，方便了客户沟通。
- 采用迭代技术能够使开发者逐步弄清客户的需求。

（快速）原型模型

原型开发存在的问题

- 为了尽快完成原型，开发者没有考虑整体软件的质量和长期的可维护性，系统结构通常较差。
- 用户可能混淆原型系统和最终系统，原型系统在完全满足用户需求之后可能会被直接交付给客户使用。

喷泉模型



一种迭代模型

喷泉模型

喷泉模型的优点

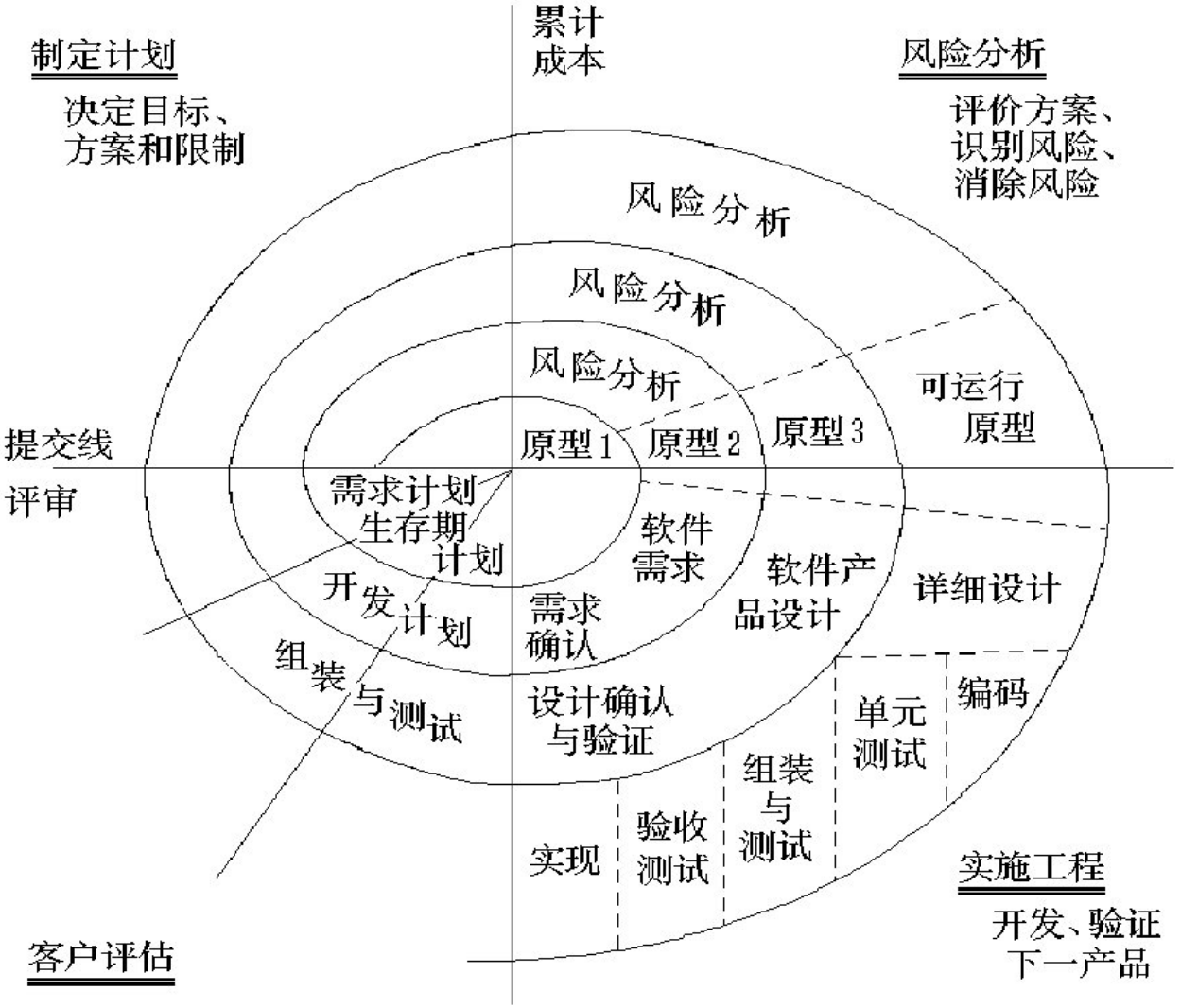
- 该模型的各个阶段没有明显的界限，开发人员可以同步进行开发。可以提高软件项目开发效率，节省开发时间，适应于面向对象的软件开发过程。

喷泉模型

喷泉模型的缺点

- 由于喷泉模型在各个开发阶段是重叠的，因此在开发过程中需要大量的开发人员，因此不利于项目的管理。此外这种模型要求严格管理文档，使得审核的难度加大，尤其是面对可能随时加入各种信息、需求与资料的情况。

螺旋模型


累计
成本
制定计划
决定目标、
方案和限制
风险分析
评价方案、
识别风险、
消除风险
风险分析
风险分析
风险分析
原型1
原型2
原型3
可运行
原型
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
编码
设计确认
与验证
单元
测试
组装
与
测试
验收
测试
实现
客户评估
实施工程
开发、验证
下一产品

一种迭代模型

螺旋模型

螺旋模型的优点

- 结合了原型的迭代性质与瀑布模型的系统性和可控性，是一种风险驱动型的过程模型。
- 采用循环的方式逐步加深系统定义和实现的深度，同时更好地理解、应对和降低风险。
- 确定一系列里程碑，确保各方都得到可行的系统解决方案。
- 始终保持可操作性，直到软件生命周期的结束。
- 由风险驱动，支持现有软件的复用。

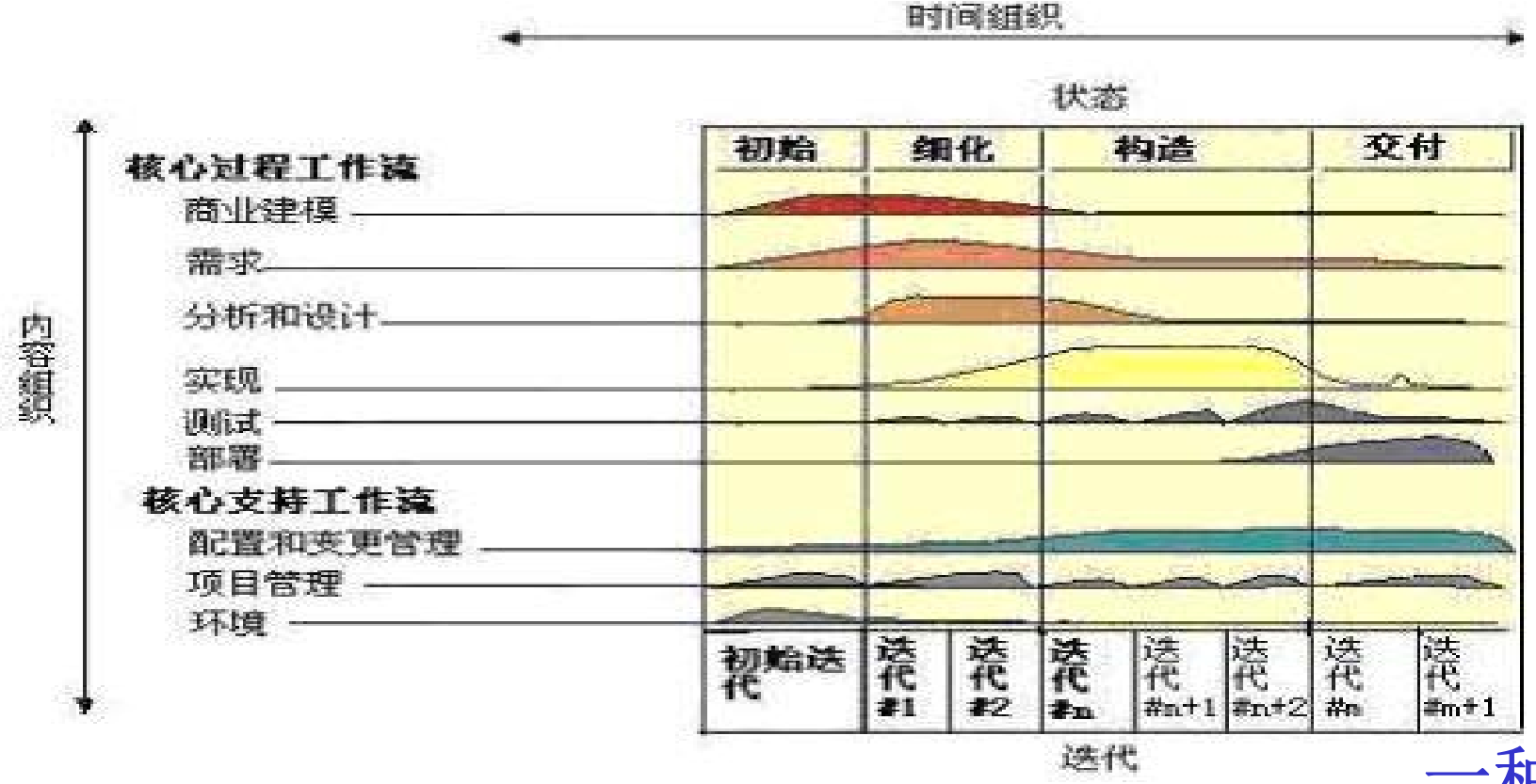
螺旋模型

螺旋模型存在的问题

- 螺旋模型依赖大量的风险评估专家来保证成功。如果有较大的风险没有被发现和管理，肯定会发生问题。
- 软件开发人员应该擅长寻找可能的风险，准确的分析风险，否则将会带来更大的风险。

统一过程模型(RUP-Rational Unified Process)

RUP软件开发生命周期是一个二维的软件开发模型。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(Cycle)、阶段(Phase)、迭代(Iteration)和里程碑(Milestone)；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



参考：《 The Rational Unified Process : An Introduction》

一种迭代模型

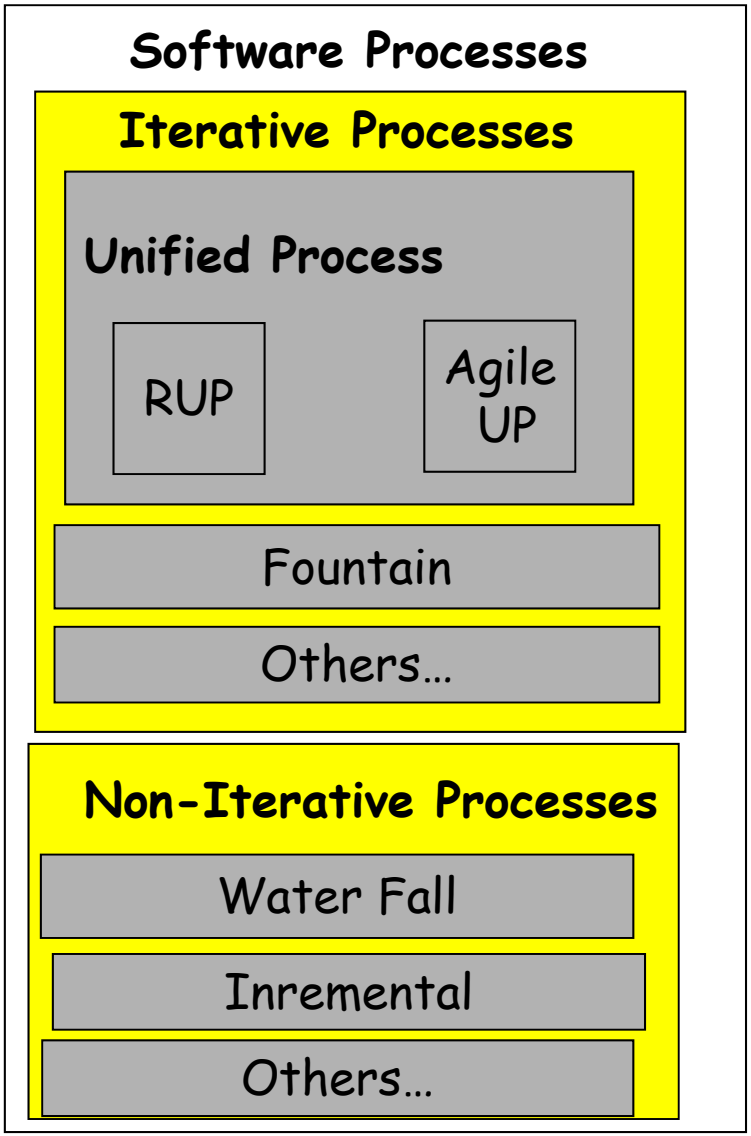
敏捷开发模型（Agile ）

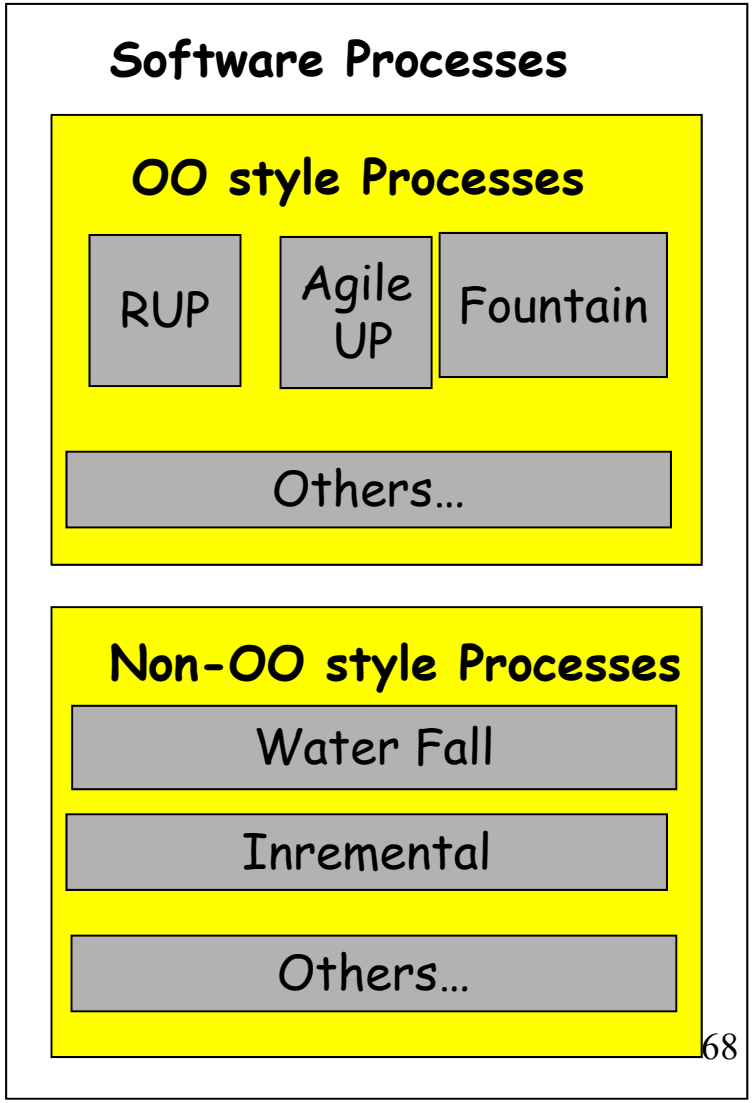
一种应对快速变化的需求的一种软件开发能力。它们的具体名称、理念、过程、术语都不尽相同，相对于“非敏捷”，更强调程序员团队与业务专家之间的紧密协作、面对面的沟通（认为比书面的文档更有效）、频繁交付新的软件版本、紧凑而自我组织型的团队、能够很好地适应需求变化的代码编写和团队组织方法，也更注重软件开发过程中人的作用。

个体和互动：高于 流程和工具。
工作的软件：高于 详尽的文档。
客户合作：高于 合同谈判。
响应变化：高于 遵循计划。

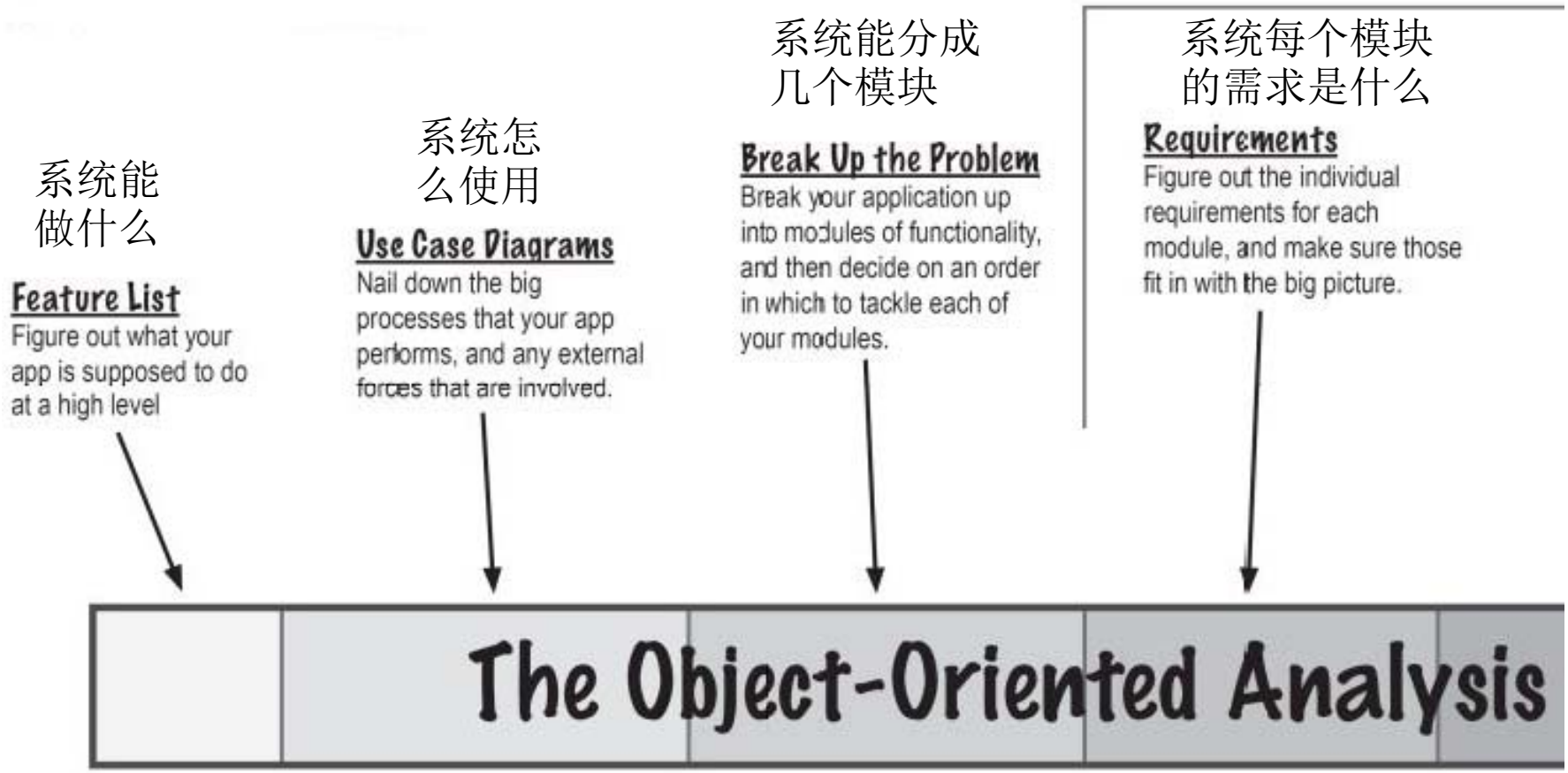
一种迭代模型

软件过程的分类

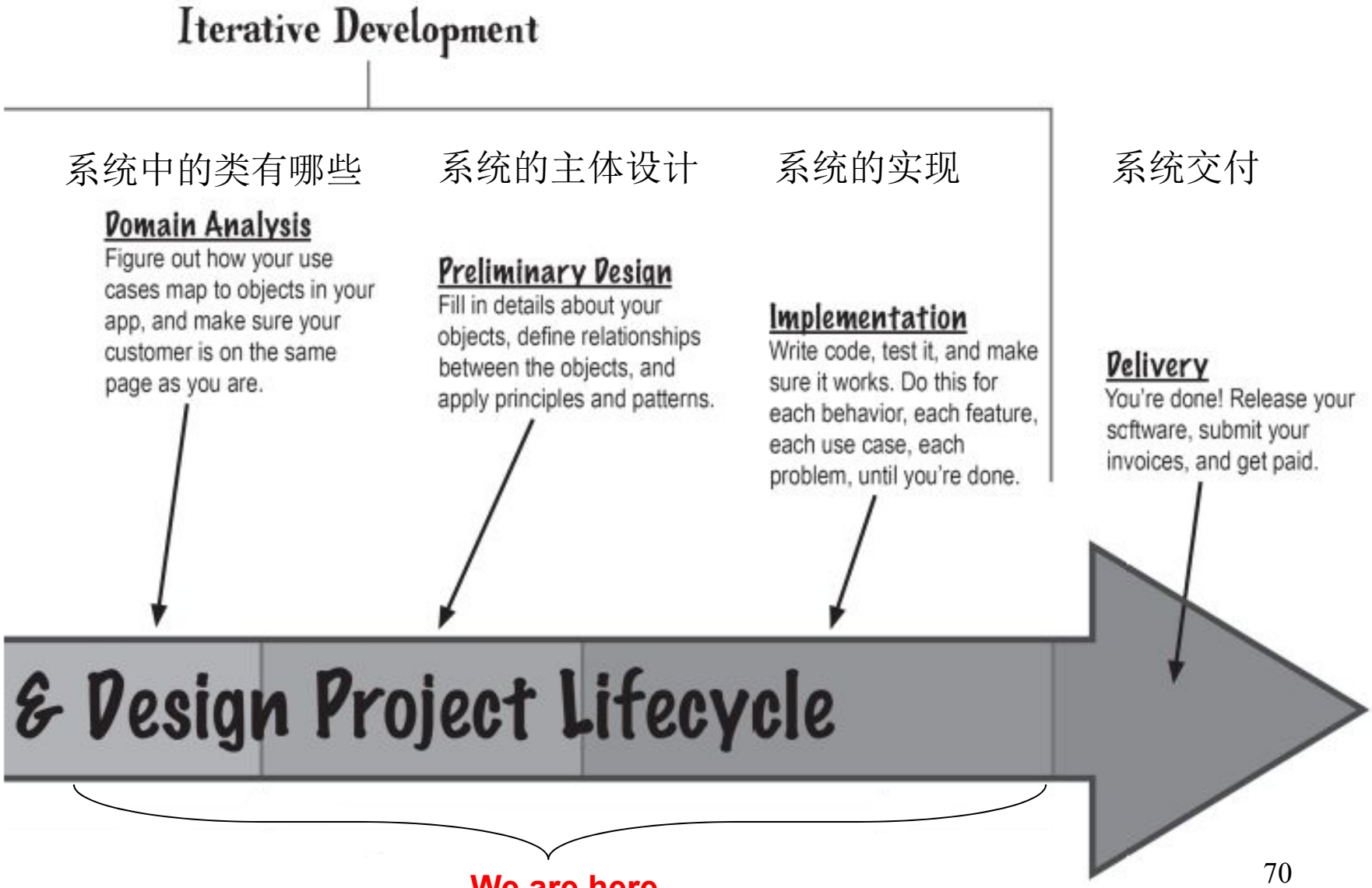

Software Processes
Iterative Processes
Unified Process
RUP
Agile UP
Fountain
Others…
Non-Iterative Processes
Water Fall
Inremental
Others…


Software Processes
OO style Processes
RUP
Agile UP
Fountain
Others…
Non-OO style Processes
Water Fall
Inremental
Others…

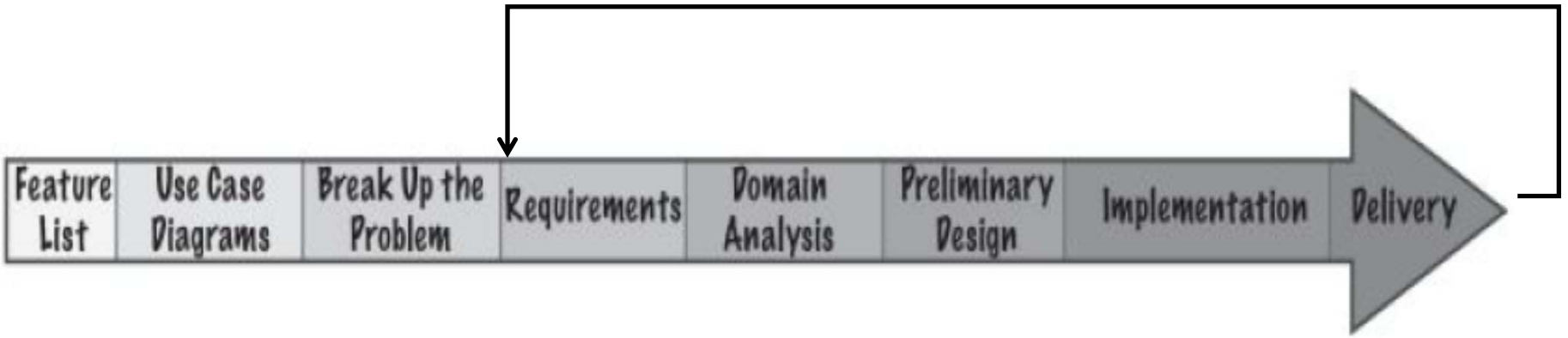
OOA&D Style Process



OOA&D Style Process



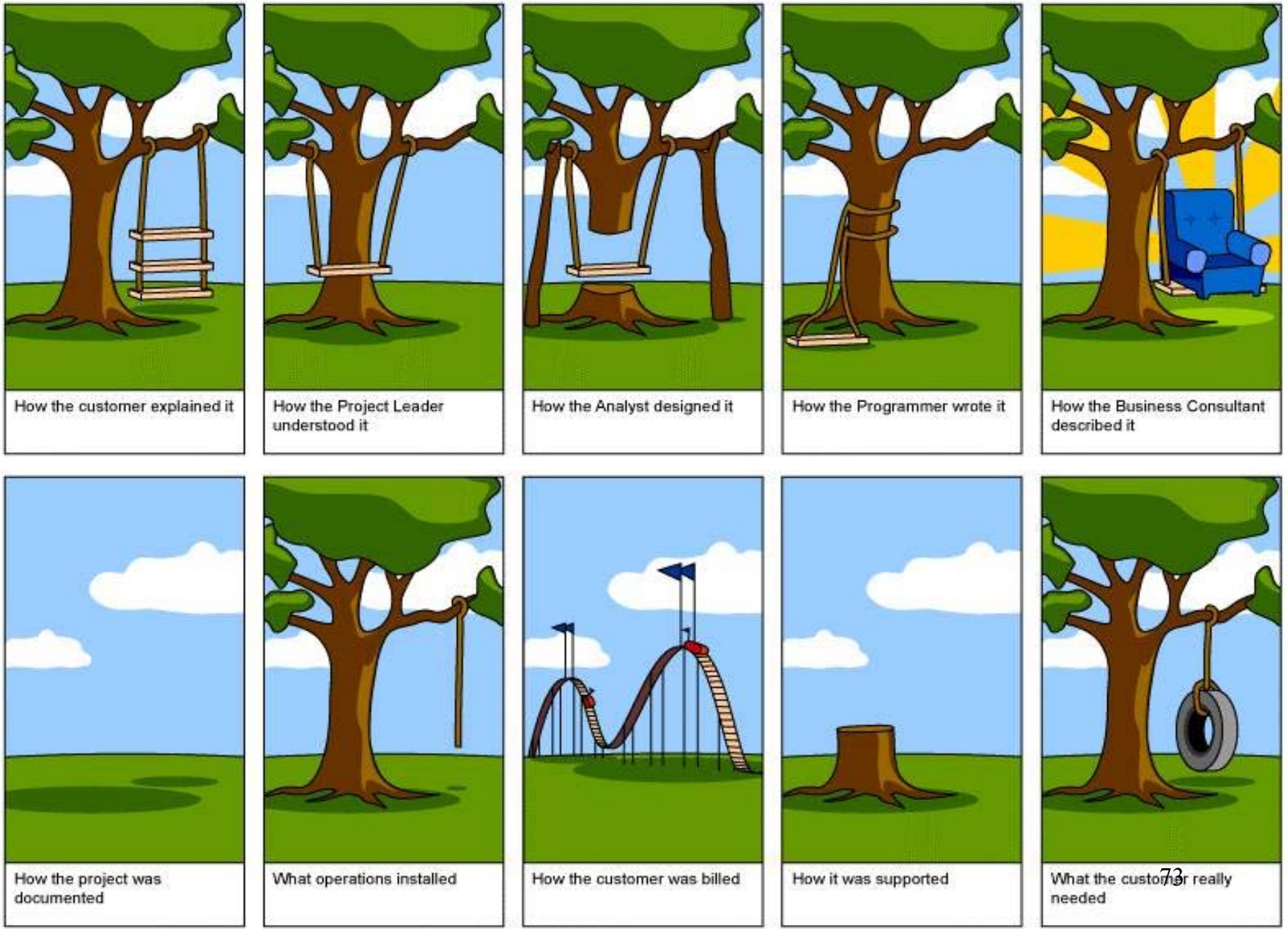
OOA&D Style Process



参考：

《Head First Object-oriented Analysis and Design》

chapter 10


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

软件质量和测试概述

软件缺陷定义

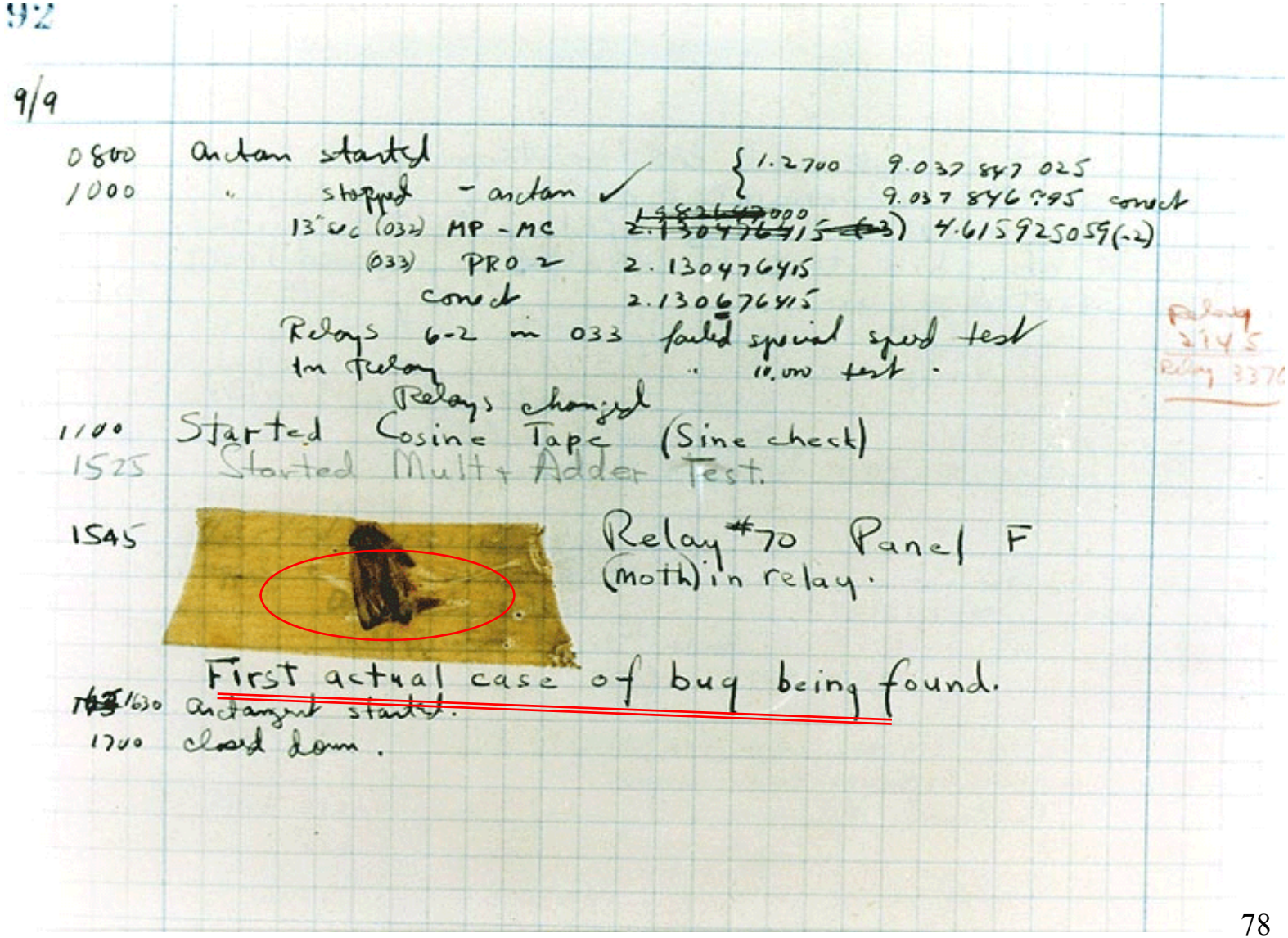
软件缺陷的正式定义

- 软件未达到产品说明书标明的功能
- 软件出现了产品说明书指明不应有的错误
- 软件功能超出产品说明书指明的范围
- 软件未达到产品说明书虽未指出但应达到的目标
- 软件测试员认为软件难以理解、不易使用、运行速度缓慢，或者最终用户认为不好

“BUG”的由来

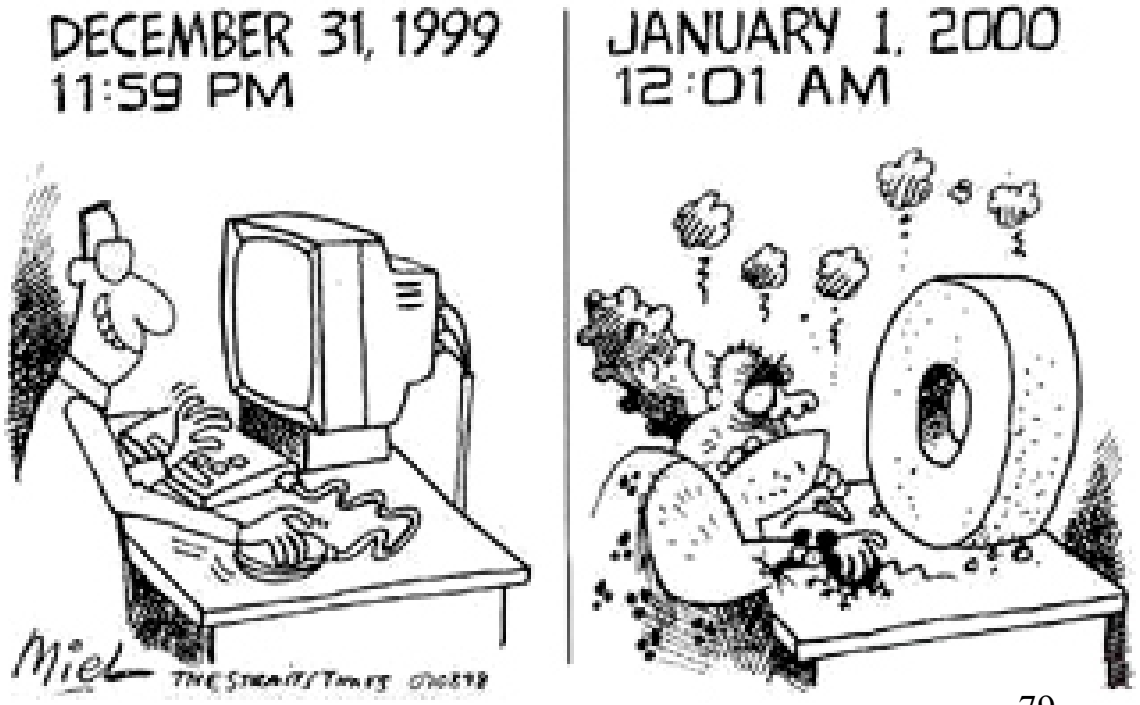
故事发生在1945年9月9日，一个炎热的下午。当时的机房是一间第一次世界大战时建造的老建筑，没有空调，所有窗户都敞开着。Grace Hopper正领导着一个研究小组夜以继日地工作，研制一台称为“MARK II”的计算机，它使用了大量的继电器（电子机械装置，那时还没有使用晶体管），一台不是纯粹的电子计算机。突然，MARK II死机了 ……

“BUG”的由来


9/9
0800 Arctan started
1000 " stopped - arctan ✓
1.2700 9.037 847 025
9.037 846 795 correct
13"UC (032) MP - MC 4.615925059(-2)
(033) PRO 2 2.130476415
correct 2.130676415
Relays 6-2 in 033 failed special speed test
In relay " 10.00 test.
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Multy Adder Test.
1545 Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
1700 closed down.

著名的软件错误案例

- The Y2K (Year 2000) Bug, circa 1974
- Patriot Missile Defense System, 1991
- Disney's Lion K
- Intel Pentium F
- NASA Mars Pc



著名的软件错误案例

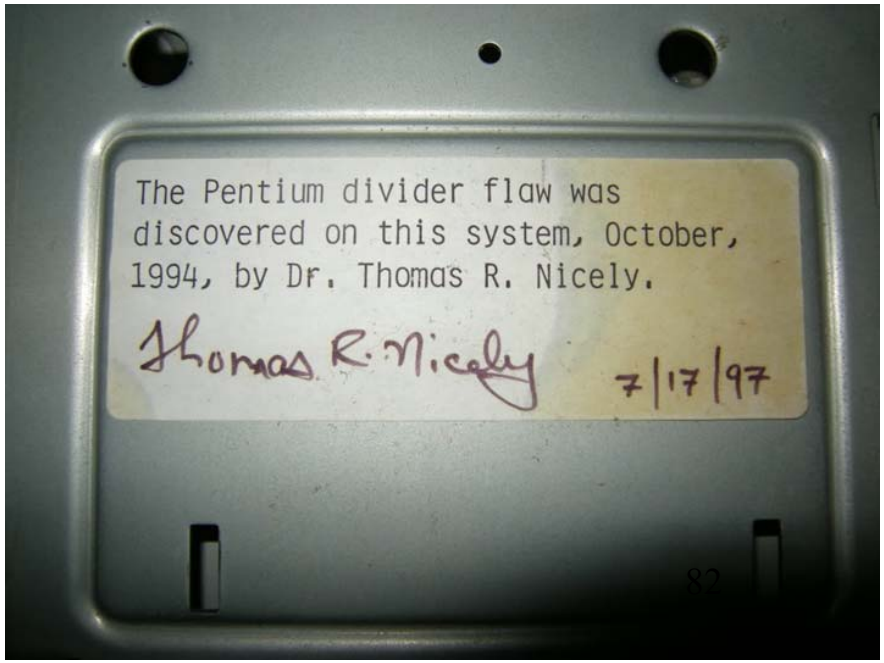
- The Y2K (Year 2000) Bug, circa 1974
- Patriot Missile Defense System, 1991
- Disney's Lion King, 1994-1995
- Intel Pentium Floating-Point Division Bug, 1994
- NASA Mars Polar Lander, 1999

著名的软件错误案例

- The Y2K (Year 2000) Bug, circa 1974
- Patriot Missile Defense System, 1991
- Disney's Lion King, 1994-1995
- Intel Pentium Floating-Point Division Bug, 1994
- NASA Mars Polar Lander, 1999

著名的软件错误案例

- The Y2K (Year 2000) Bug, circa 1974
- Patriot Missile Defense System, 1991
- Disney's Lion King, 1994-1995
- Intel Pentium Floating-Point Division Bug, 1994
- NASA Mars Polar Lander, 1999



著名的软件错误案例

- The Y2K (Year 2000) Bug, circa 1974
- Patriot Missile Defense System, 1991
- Disney's Lion King, 1994-1995
- Intel Pentium Floating-Point Divis
- NASA Mars Polar Lander, 1999

其他的软件错误案例

- 伦敦救护车服务调度系统

  1992年11月26、27日，系统调度发生严重错误

  多辆救护车被分配到同一事故地点

  没有将最近的救护车分配到事故地点

  造成至少20人死亡

- 可以自动吐钞的ATM机

  2010年，西雅图信息安全专家杰克在“黑帽”(Black Hat)计算机安全会议演示破解ATM机：

  通过ATM机的接口连接并运行他编写的破解程序，让ATM吐钞

  或通过网络联机，入侵厂商的远程管理软件，遥控命令让ATM机吐钞票

- “甬温线”动车追尾事故

  2011年7月23日，造成40人死亡、172人受伤。事故的主要原因之一是列控中心设备中的自检模块软件存在严重设计缺陷，未将系统导向安全侧；

……

软件缺陷是什么

Defect	Incident	Inconsistency
Fault	Anomaly	Feature
Problem	Variance	Bug
Error	Failure	

Error ⟶ defect ⟶ Fault ⟶ Failure

```
public class Test {
    public static void main(String args[]) {
        int x = 10;
        double y = 0.5;
        int z = (int) y;
        int m = x/z;
        System.out.println("x/z="+m);
    }
}
```

应该为 int z = Math.abs((int)y + 1)

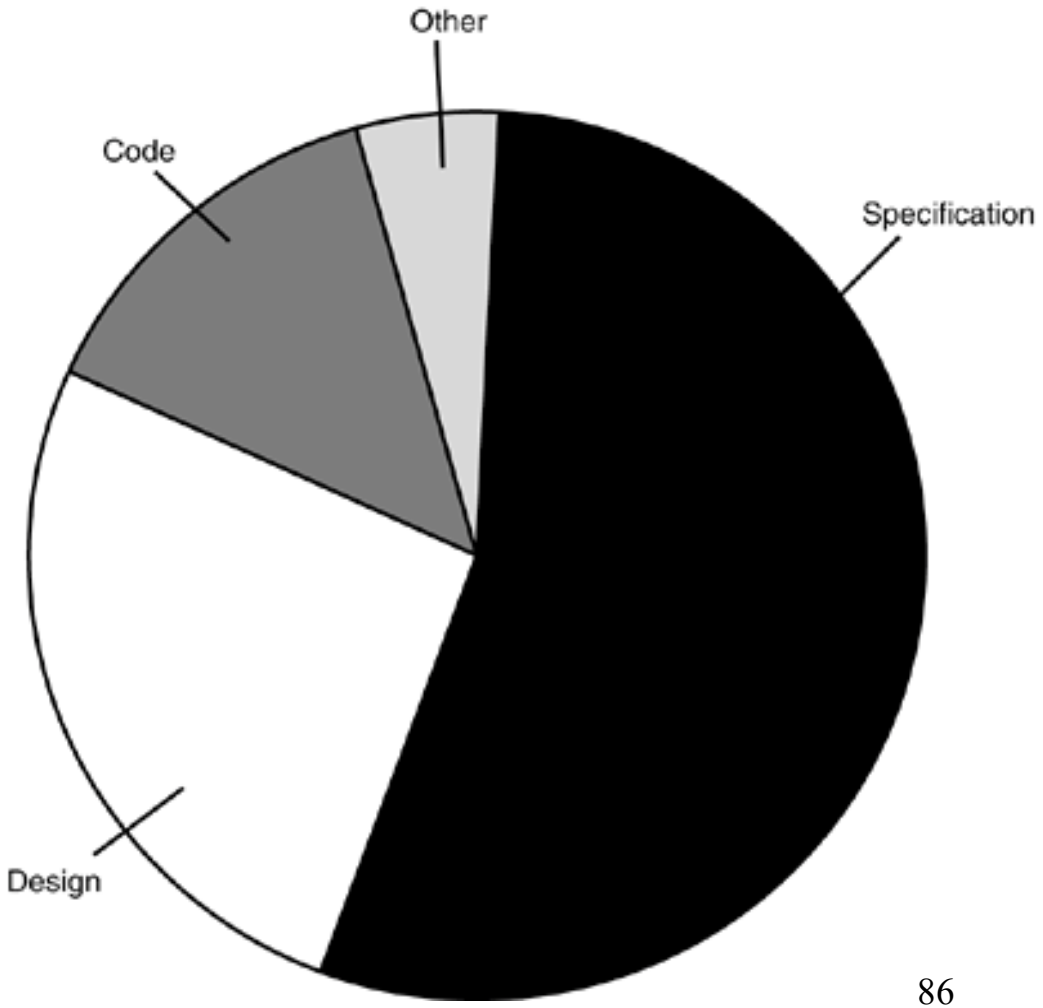
软件缺陷产生的原因

编码 - 复杂度?

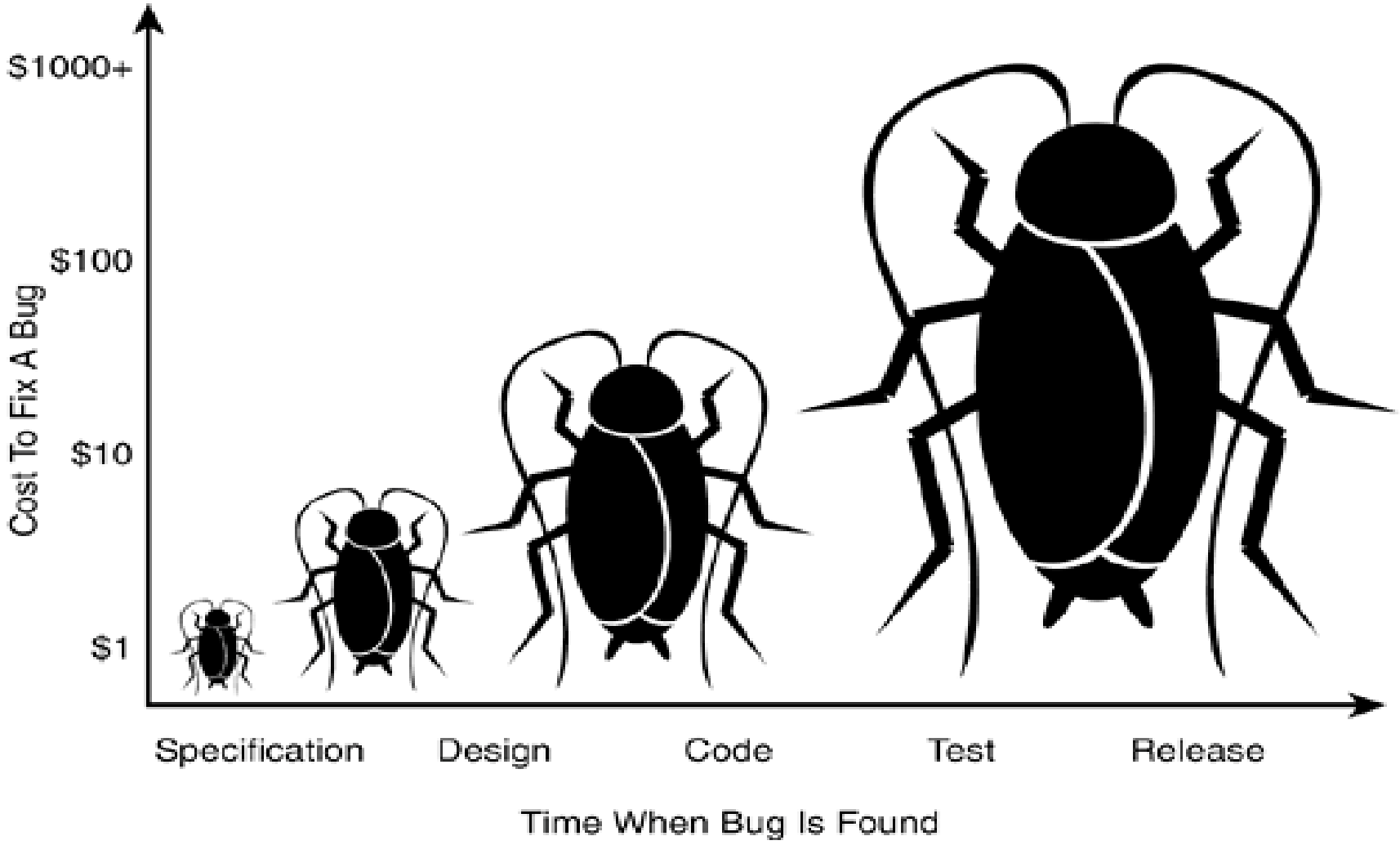
设计

说明书

其他



软件缺陷的修复费用



扼杀在摇篮里！

基本事实

- 世界上不存在没有缺陷的软件
- 可以通过两种途径开发出没有错误的软件：
  - 在一开始就防止引入错误
  - 识别潜藏在代码中的错误，找到并消灭它们
- 软件设计故障和计算机硬件设计故障比例大约是：10:1 !!!
- 运行软件的驻留故障密度（每千行代码的故障数）
  - 要求很高的关键财务或财产软件为：每千行代码**1-10**个故障
  - 关键的生命软件为：每千行代码**0.01-1**个故障

如何保证软件产品的质量？

软件质量

- 有些软件开发者仍然相信软件质量是在编码之后才应该开始担心的事情。
- 这是误解，因为软件质量保证（Software Quality Assurance，SQA）是一种应用于整个软件过程的保护性活动，包括：
  - 一种质量管理方法，
  - 有效的软件工程技术（方法和工具）
  - 在整个软件过程中采用的正式技术复审
  - 一种多层次的测试策略
  - 对软件文档及其修改的控制
  - 保证软件遵从软件开发标准的规程
  - 度量和报告机制

软件质量

应从以下几个方面考虑软件质量：

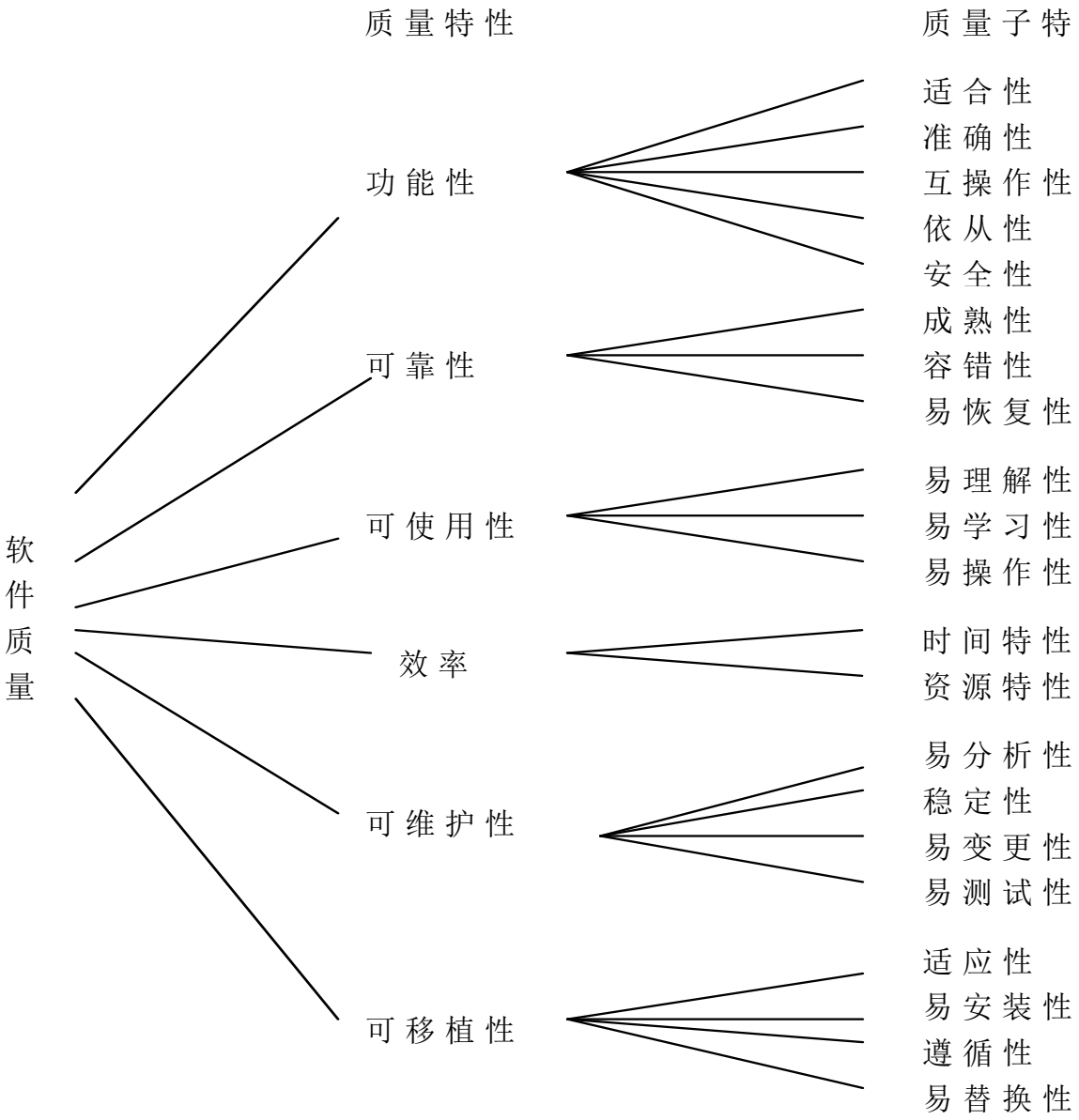
- 软件结构方面
- 功能与性能方面
- 开发标准与文档方面

软件系统规模和复杂性的增加，使得软件开发成本和软件故障而造成的经济损失也在增加，软件质量问题，正成为制约计算机发展的关键因素。

软件质量概念

- IEEE关于软件质量的定义：软件质量是
  - 系统、部件或者过程满足规定需求的程度。
  - 系统、部件或者过程满足顾客或者用户需要或期望的程度。
  - 该定义相对客观，强调了产品（或服务）和客户/社会需求的一致性。
- ANSI关于软件质量的定义：按照ANSI（美国国家标准学会）在1983年的标准陈述，软件质量定义为“与软件产品满足规定的和隐含的需求的能力有关的特征和特性的全体”
  - 软件产品中能满足用户给定需求的全部特性的集合，
  - 软件具有所期望的各种属性组合的程度，
  - 用户主观得出的软件是否满足其综合期望的程度，
  - 决定所用软件在使用中将满足其综合期望程度的软件合成特性。

6个主要特征



度量

度量由使用单位自行决定

软件测试概述——定义

- 1983年，IEEE在提出的软件测试文档标准（IEEE Standard For Software Test Document），即IEEE 829-1983中对软件测试进行了准确的定义：
  - 软件测试是使用人工或自动手段来运行或测定某个系统的过程，检验它是否满足规定的需求或者弄清预期结果与实际结果之间的差别。
- IEEE在1990年颁布的软件工程标准术语集中沿用了这一概念，该概念非常明确的提出了软件测试以检验是否满足需求为目标。
- 其次，G. J. Myers在其经典论著《软件测试的艺术》中对软件测试提出如下观点：
  - 测试是程序的执行过程，目的在于发现错误，
  - 一个好的测试用例可以发现至今尚未发现的错误，
  - 一个成功的测试能发现至今未发现的错误。

软件测试的发展历程

Hetzel将软件测试的历史按照时间分为5个阶段

1. 1956年之前，面向调试的测试
2. 1957～1978，面向证明的测试
3. 1979～1982，面向查错的测试
4. 1982～1987，面向评估的测试
5. 1988～2000，面向预防的测试

面向调试的测试

- 1956年之前，发现软件缺陷和修改缺陷的是同样的编程人员，那时是不区分软件调试和测试的，既没有系统的测试理论出现，也没有专人从事发现软件缺陷的测试工作。
- 软件设计人员仅在机器出现不正常状态时去找问题，然后修复它。

面向证明的测试

- 1957-1978，开始出现了软件测试的理论。
- 但是，此时软件测试的主要工作是证明软件可以使用，测试还处于一种软件开发的辅助状态；
- 测试人员主要使用可以正常运行的合法数据来检验软件是否能够正常完成其工作。

面向证明的测试

例 计算银行利息的函数，不加保护的程序

```
double Get_Interest( double fPrincipal, double fBack_Rate, int iYear)
{
    double fTotal = fPrincipal * ( 1 + fBack_Rate ) iYear
    return ( fTotal - fPrincipal );
}
```

面向查错的测试

- 1979-1982，设计和开发程序的逻辑越来越严密，不仅考虑程序正常状态下的运行情况，也要考虑程序在各种错误操作和数据下的承受能力，从这个意义上说，软件测试促进了程序质量的提高；
- 这一阶段对于软件测试的理解并不太成熟，往往过分强调找到软件中的错误。

面向查错的测试

例 计算银行利息的函数，加保护的程序

```
double Get_Interest( double fPrincipal, double fBack_Rate, int iYear)
{
    if（fPrincipal < 0.0 ） /// 对本金加保护，必须大于等于0
        return ( 0 );
    if（fBack_Rate < 0.0 ）// 对利率加保护，必须大于等于0
        return ( 0 );
    if（iYear <= 0 ） /// 对年加保护，必须大于0
        return ( 0 );
    double fTotal = fPrincipal× ( 1 + fBack_Rate ) iYear
    return ( fTotal - fPrincipal );
}
```

面向评估的测试

- 1983-1987，测试的目的不是为了证明软件的对错，而是将可观察到的软件缺陷减少到一个可以接受的程度；
- 测试员为程序员提供软件缺陷的信息，为管理员提供如果发布软件系统会造成负面影响的评估报告；
- 在这个阶段，软件测试不仅得到了蓬勃地发展，而且软件测试的目的变得客观成熟。

面向预防的测试

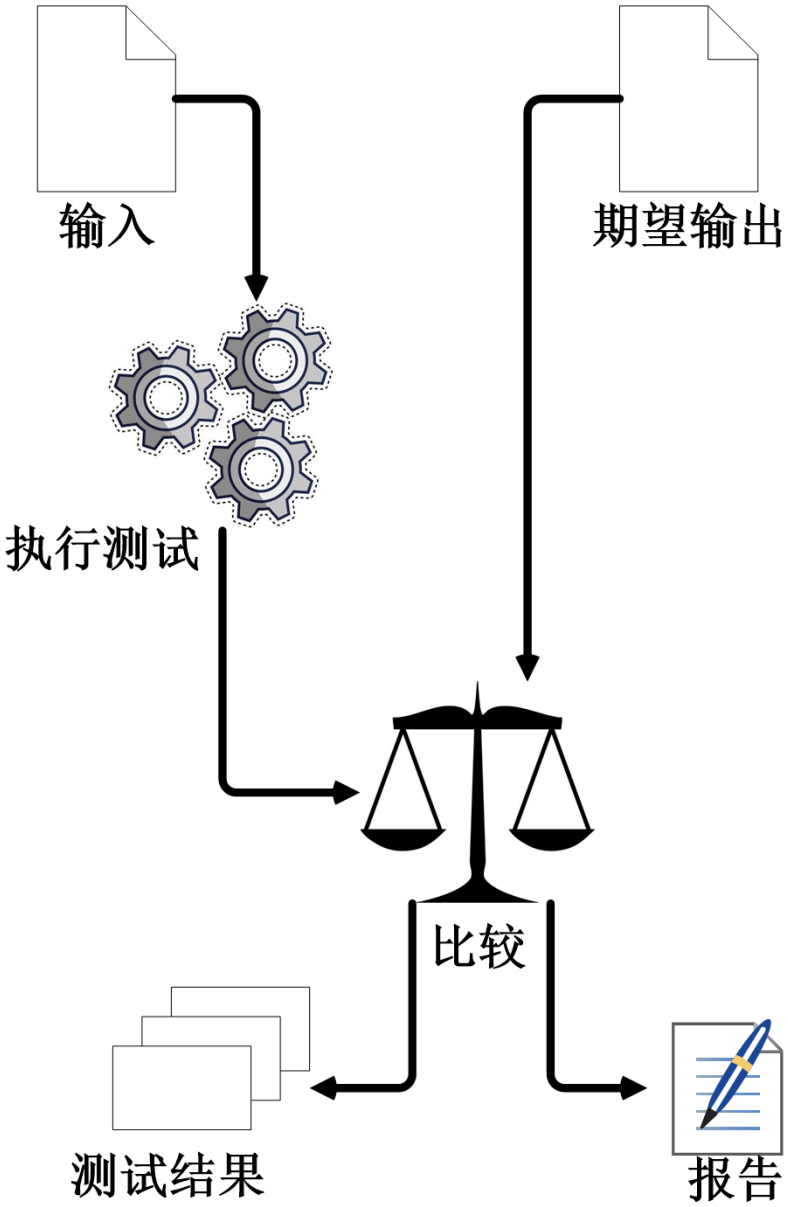
- 1988-2000，软件测试不仅是操作，而是一种智力训练，它将导致即使进行较少地软件测试也使软件是低风险的；
- 在这种成熟的软件测试情况下，致力于从软件开发初期就是可测试的。

面向预防的测试

例计算银行利息的函数，具有自诊断能力的代码。

```
double Get_Interest( double fPrincipal, double fBack_Rate, int iYear)
{   if（fPrincipal < 0.0 ） { // 本金必须大于等于0
        # ifdef _DEBUG /// 自诊断能力的加入，便于错误定位
        printf( “ Principal can’t less 0. Please reinput！ ” );
        # endif
        return ( 0 ); }
    if（fBack_Rate < 0.0 ） { //利率必须大于等于0
        # ifdef _DEBUG
        printf( “ Back rate can’t less 0. Please reinput！ ” );
        # endif
        return ( 0 ); }
    if（iYear <= 0 ） { //年必须大于0
        # ifdef _DEBUG
        printf( “ y ear can’t less 0. Please reinput！ ” );
        # endif
        return ( 0 ); }
    double fTotal = fPrincipal× ( 1 + fBack_Rate ) iYear;
    return ( fTotal - fPrincipal );
}
```

测试是什么



测试是什么

- 测试是通过运行程序来发现错误的过程；
- 测试的目的是发现程序中的错误，是为了证明程序有错，而不是证明程序无错；
- 测试可以说明软件存在错误，但不能说明它不存在错误；
- 理想的目标：用相对少的测试尽可能多地找到程序中的缺陷。
- “好软件是做出来的，不是改出来的。”

软件测试的对象

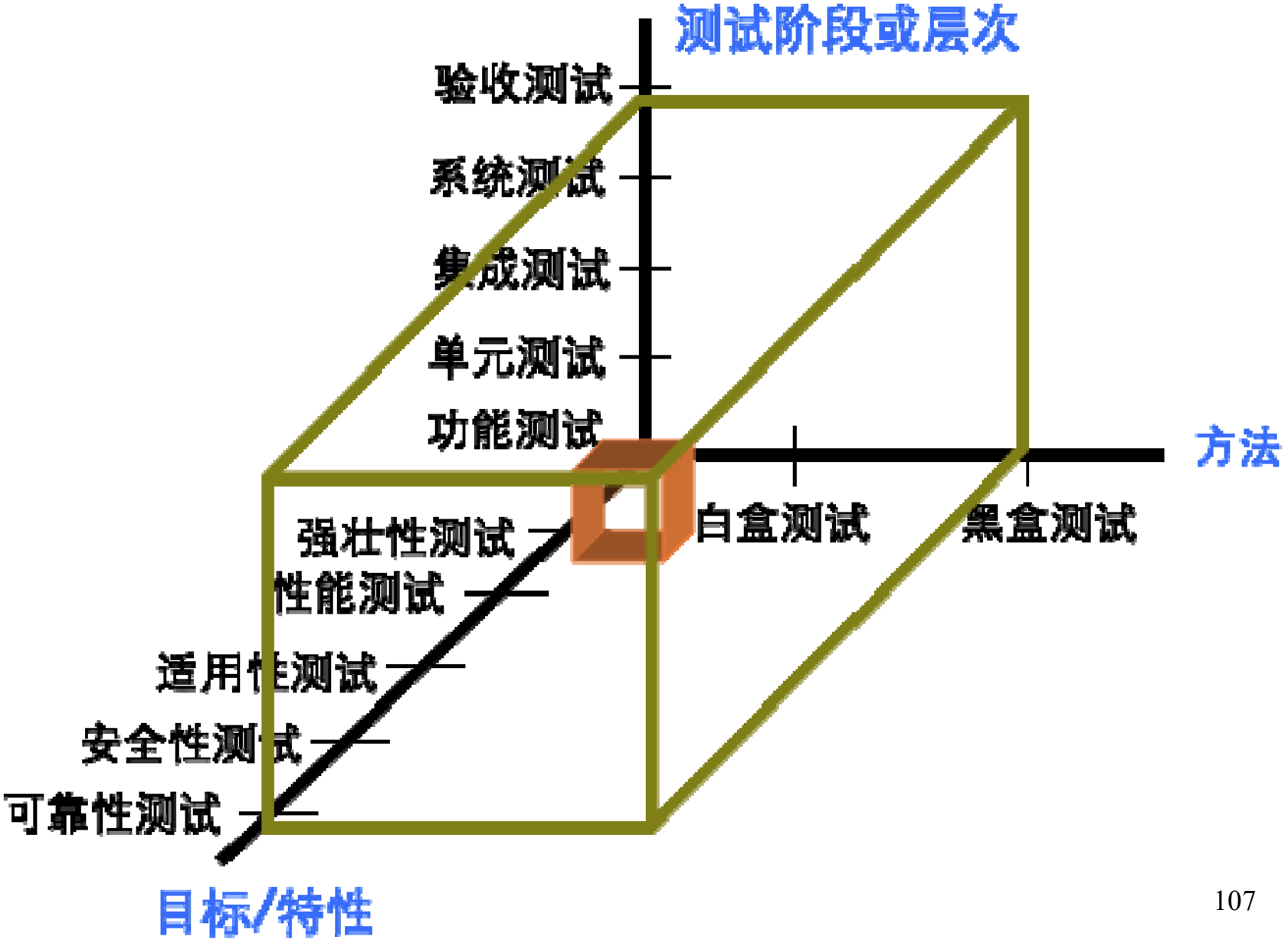
- 软件测试并不等于程序测试。软件测试应贯穿于软件定义与开发的整个期间。
- 测试对象：软件开发过程中各阶段的人工制品。
  - 需求分析、概要设计、详细设计以及程序编码等各阶段所得到的文档，包括需求规格说明、概要设计说明、详细设计说明以及最终的系统，都是软件测试的对象。

软件测试的挑战

- 软件结构越来越复杂
- 团队规模越来越大
- 开发成本与风险越来越高
- 用户对应用程序的质量要求越来越严格

- 软件类型繁多
  - OO软件系统
  - 基于组件的系统
  - 并发系统
  - 分布式系统
  - GUI系统
  - Web系统
- 测试目的不同
  - 可用性
  - 安全性
  - 正确性
  - 性能…

软件测试的分类


测试阶段或层次
验收测试
系统测试
集成测试
单元测试
功能测试
方法
白盒测试
黑盒测试
强壮性测试
性能测试
适用性测试
安全性测试
可靠性测试
目标/特性

自动测试 Vs 手工测试

手工模拟用户操作

对软件测试工作的误解

- 软件测试技术要求不高，至少比编程容易多了。
- 软件测试随便找一个能力差的人就能做。
- 有时间就多测试一些，来不及就少测试一些。
- 软件测试是测试人员的事，与开发人员无关。
- 设计－实现－测试，软件测试事开发后期的一个阶段。软件开发完成后才能进行测试。
- 软件测试是没有前途的工作，只有程序员才是程序高手。

软件工程师和测试工程师的区别

Developer 程序员

Understands the system
but, will test "gently" and, is
driven by "delivery"

理解系统，但是，只进行“温柔”测试，并由“交付”驱动

Independent tester 独立的测试员

Must learn about the system,
but, will attempt to break it
and, is driven by quality

必须向系统学习，但是，会尽力破坏系统，并由质量驱动

软件测试员做什么？

软件测试员的目标是

- 发现软件缺陷。
- 尽可能早地发现软件缺陷。
- 尽可能早地发现软件缺陷，并确保其得以修复。

软件测试人员应具备下列专业素质：

1. 对于系统测试，把握需求是第一位的。
   要有很强的需求理解能力显得很重要
2. 测试基础
3. 测试方案的分析设计能力、测试案例的设计能力
4. 测试工具的使用能力
5. 编程能力
6. 判断准确
7. 团队协作能力，与各小组之间的沟通能力
8. 测试管理，管理决定了工作质量
9. 追求完美

软件测试职业和职位

- ✓ 软件测试员
- ✓ 软件测试工程师/程序分析员
- ✓ 高级软件测试工程师/程序分析员
- ✓ 软件测试组负责人
- ✓ 软件测试/编程负责人
- ✓ 软件测试/质量保证/项目经理

软件测试人才的现状

在欧美，软件测试的工作量要占到项目总工作量的**40%**，测试费用要占到项目总经费的**30%**。在微软，开发人员和测试人员的比例为**1：1.5**。

	Exchange 2000	Windows 2000
项目经理	25	约250
开发人员	140	约1700
测试人员	350	约3200
测试人员与开发人员比例	2.5	1.9

软件测试人才的现状

软件测试目前已经是一门非常紧缺的职业

国外大型软件公司

微软，测试:开发人员大约为1:1，Win2000团队中甚至是2:1

IBM，测试:开发人员大约为1:3，但是正在向1:1方向发展

Google，测试:开发人员 大约为1:10

根据不同的系统类型

Web系统，测试:开发 大约为1:5-10

安全性要求高的系统，测试:开发 大约为5:1

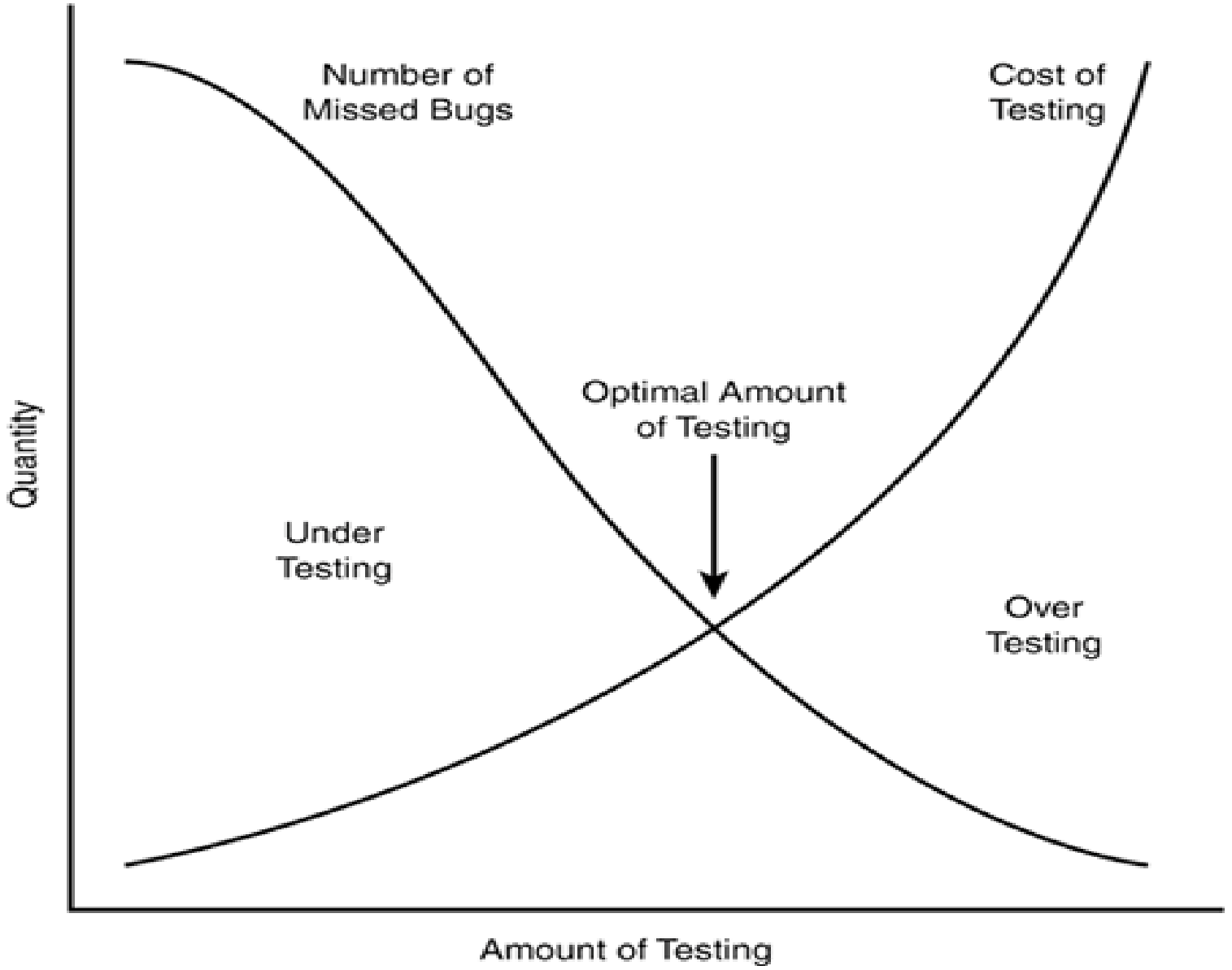
软件测试的实质

一、测试原则：

- 完全彻底的测试是不可能的（P23）

- The number of possible inputs is very large.
- The number of possible outputs is very large.
- The number of paths through the software is very large.
- The software specification is subjective. You might say that a bug is in the eye of the beholder.

●软件测试是有风险的行为



- 测试无法发现潜在的软件缺陷

正确性　VS　完备性

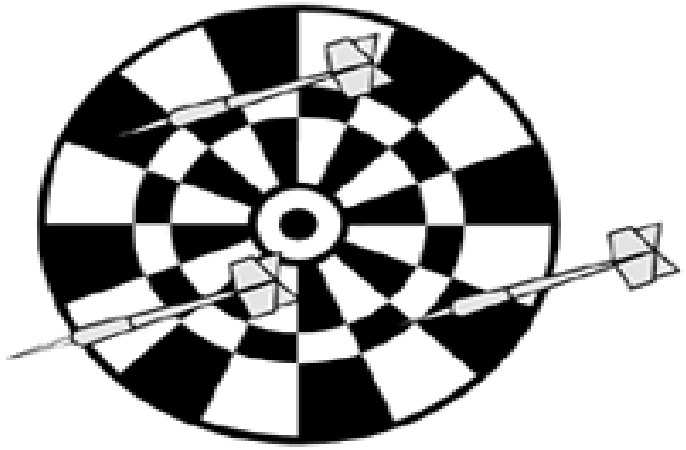
- 找到的软件缺陷越多，则实际存在的缺陷肯定越多

  祸不单行！！

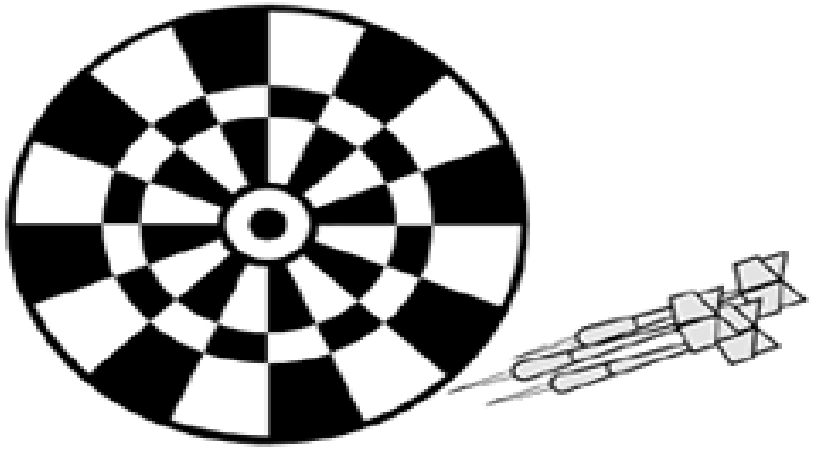
- “抗药性”
- 并非所有的软件缺陷都能修复 （P27）
- 难以说清的软件缺陷
- 软件说明书（需求）一直在变化
- 软件测试人员在开发团队里“讨人嫌” （P29）

二、软件测试的术语和相关定义

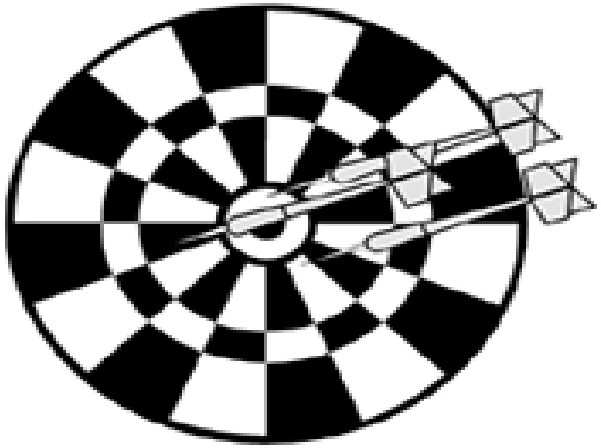
●“精确”和“准确”（Precision and Accuracy）



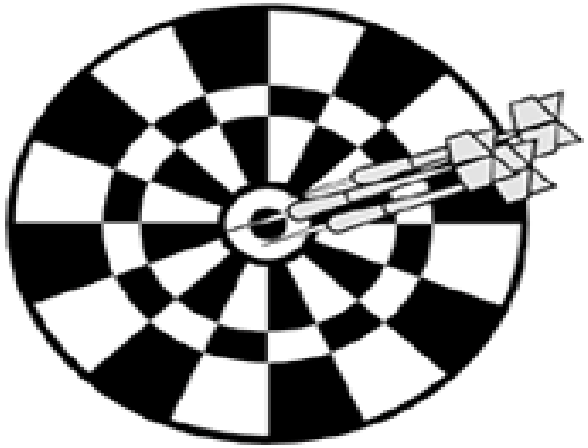
Neither Accurate nor Precise



Precise, but not Accurate



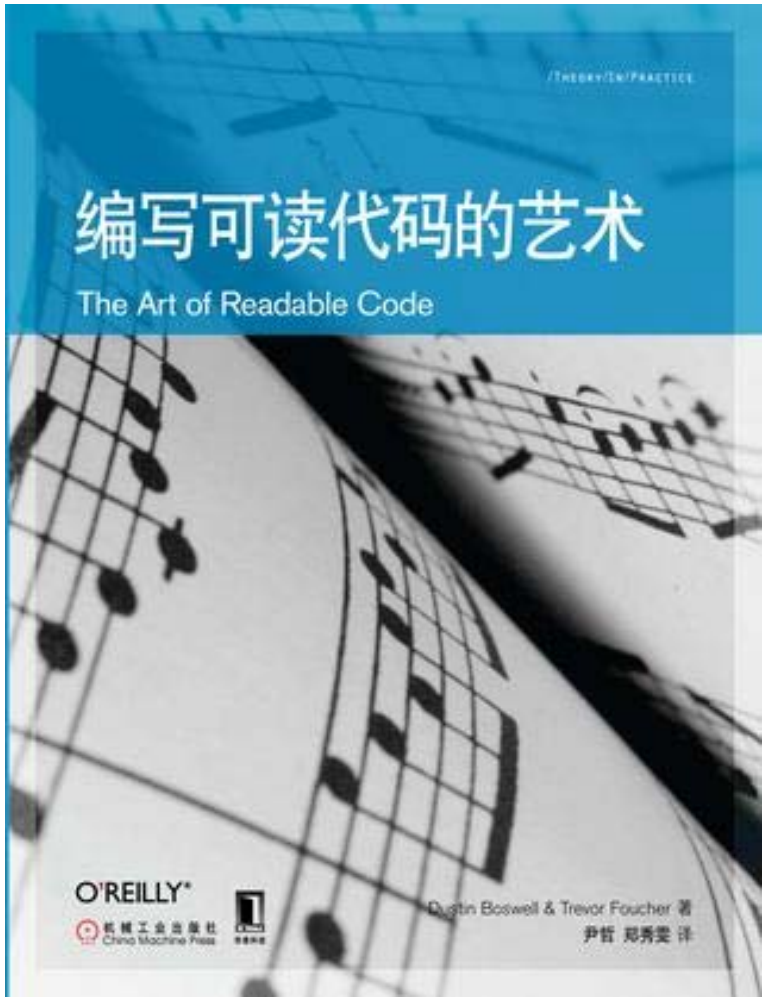
Accurate, but not Precise

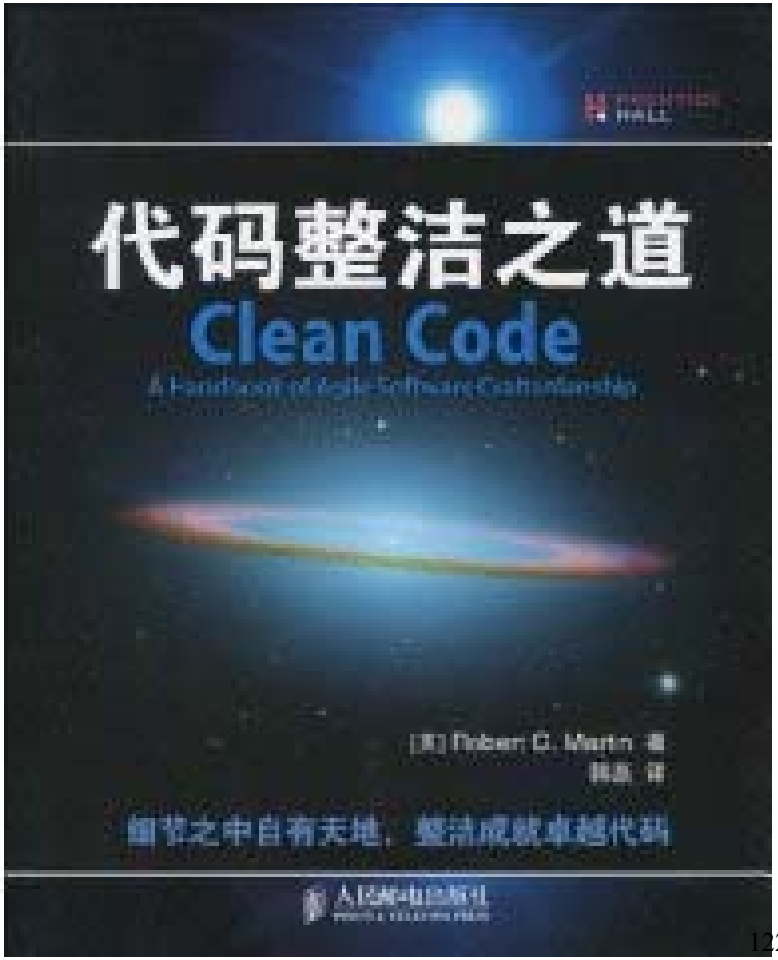


Accurate and Precise

- “验证”和“合法性检查”（Verification and Validation）
- “质量”和“可靠性”（Quality and Reliability）
- “测试”和“质量保证”（Testing and Quality Assurance

推荐阅读


编写可读代码的艺术
The Art of Readable Code
O'REILLY
Dustin Boswell & Trevor Foucher 著
尹哲 郑秀雯 译


代码整洁之道
Clean Code