软 件 测 试

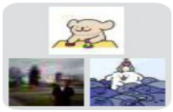
聂长海

changhainie@nju.edu.cn

南京大学计算机科学与技术系

课程微信群

- 请大家拿出手机，扫码入群
- 群名设为“2025软件测试课程-智软院”
- 请大家实名：学号+姓名



群聊： 2025软件测试课程--智软院

该二维码7天内(8月27日前)有效，重新进入将更新

教材及教学内容

南京大学 聂长海编著

2013年5月

清华大学出版社

第1章 软件测试概论

第2章 白盒测试和黑盒测试

第3章 开发过程中的测试

第4章 软件特性及方面的测试

第5章 特殊的软件测试技术

第6章 特定应用软件(X-软件)的测试

考核方式

- 平时成绩25分
  - 出勤率少1次扣1分，全勤18次计10分，现场交作业点名
  - 演讲报告10分，现场点评现场打分
  - 研究论文（实验报告）5分，我们会认真阅读
- 期中考试25分，安排在第9周，10月28日
- 期末考试50分，严格把关
- 具体执行方式会根据具体情况进行调整

教学方式

- 课堂教学（2+1模式）
- 实验教学（手机APP、web、Xunit）
- 实验交流（测试具体软件，交流体会）
- 论文实验报告写作（每人一个题目，汇报实践环节）
  - 工具的开发与使用
- 人工智能工具的探索与实践
  - 智能化时代，共同探索课堂新形势
  - 人人都具备的条件：手机，笔记本电脑

学生课堂演讲

- 选择演讲题目
  - 从软件测试方法列表中选择，研究一种方法
  - **提前或现场制作5分钟个人演讲视频作品**
  - **接受老师和同学们的质询**
- 充分使用大模型工具搜集组织材料
  - 写一个完整的研究报告及**实验报告**
  - 精心准备ppt
- 收获：讲得好，相互学习
  - 上课认真听，多提问，学问学问边学边问，大模型时代，学会提问
  - 教师讲解和同学讲解既是互相补充，又是温习巩固

实践1 用好一种方法

- 将选择的方法用于多个软件
  - 例如组合测试方法测试浏览器、手机和机顶盒等
- **写研究与实验报告**
  - **用真实的软件**
  - **选具体的方法**
  - **做真实的实验（操作）**
  - **写真实的体验**
- 可以借助大模型
- 但必须亲手做一遍

实践2：测好一个软件

- 选择一个软件
  - 系统地使用各种方法进行测试，例如微信、支付宝，鸿蒙系统
- 写实验报告
  - **用真实的软件**
  - **选具体的方法**
  - **做真实的实验（操作）**
  - **写真实的体验**
- 深入了解、解刨一个软件系统
- 综合使用LLM各种工具辅助

基 本 现 状

- 在学校，一门课程 （100多本教科书，很多学校都开）
- 工业界，一个职业 （数以万计的从业人员，引用盖茨语）(软件世界的医疗行业)
- 学术界，一个很热的研究领域 （1/3－ICSE paper）
- Facebook，微软的测试正在发生颠覆性变化，测试该往何处去？？？

基本理念
领域专家是怎样炼成的？

- 10000小时投入
- 学者，作家，研究员，教师，实践者
- become a worker in software testing
  - reader，writer，listener，speaker，researcher and practitioner

Position of our course

- Information technique
- Computer science
- Computer engineering
- Software engineering
- 人工智能
- 网络安全
- 大数据
- 区块链

Position of our course

- Requirement
- Design
- Coding
- Testing
- Maintenance

软件测试的位置

尽管软件工程的发展为高可信软件的开发提供了许多方法、技术和工具环境等支持。

- 软件的分析与理解
- 形式化验证
- 软件过程管理（第一关：预防错误）
- 软件测试（第二关：检查错误）
- 容错（第三关：容错计算）
- 错误预测（第四关：软件可靠性）

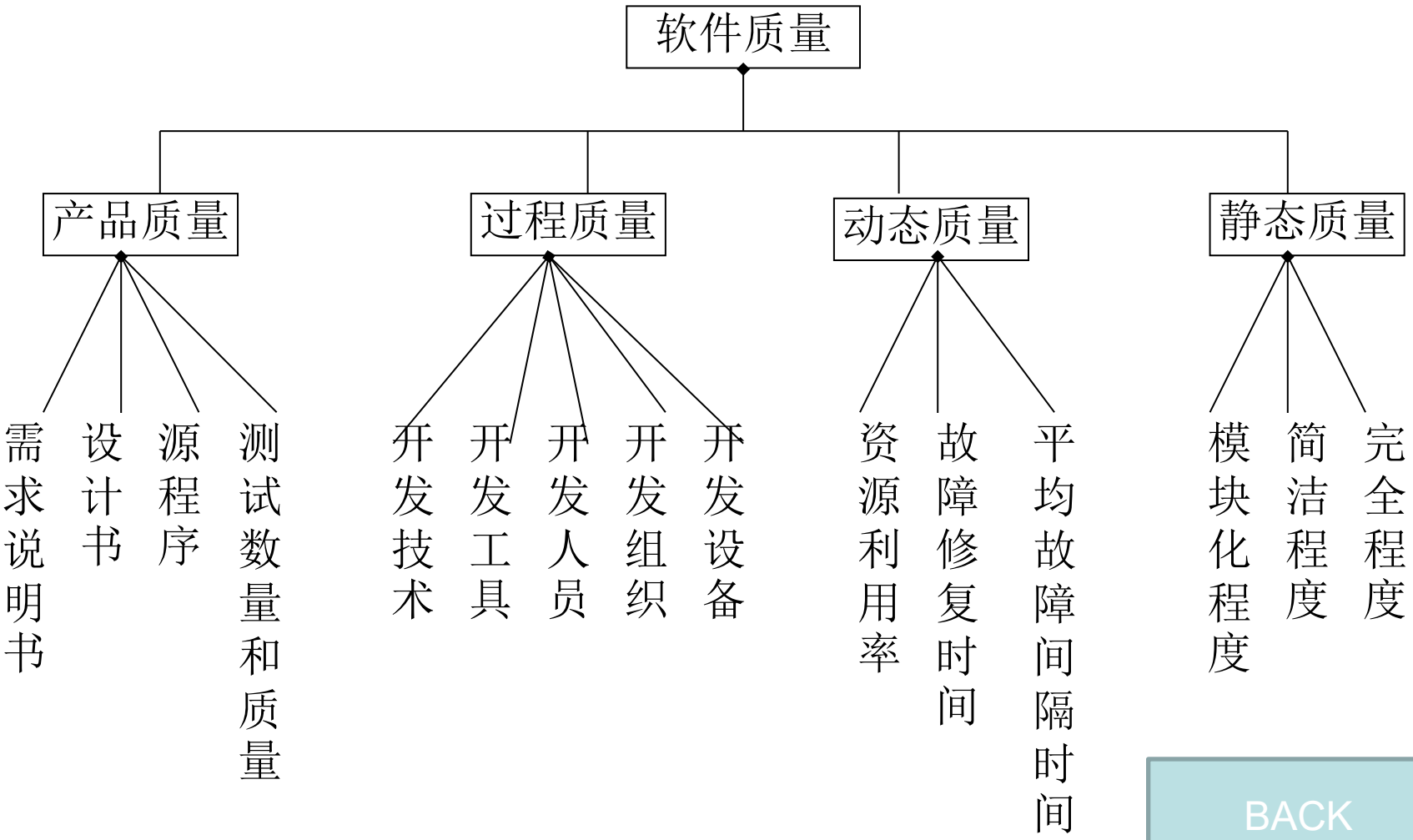
软件测试概论

- 1 软件的质量
- 2 软件缺陷及其危害与管理
- 3 软件测试的概念
- 4 测试的普遍观
- 5 软件测试的关键问题及方法分类
- 6 软件测试的对象、目的、意义和原则
- 7 软件测试的流程
- 8 软件测试在软件工程中的位置（在若干开发模型中）
- 9 软件测试的模型
- 10软件测试的研究
- 11软件测试作为一种职业和作为一门科学
- 12软件测试的工具
- 13软件测试的管理
- 14软件测试的历史和未来
- 15 软件测试标准

1 软件质量

- 软件测试的核心目的就是要度量和提高软件质量，那么什么是软件质量？
- 软件质量是一个**包含多个属性的，可度量的多维度的量**，其中有些属性比另外一些属性重要。根据软件质量的度量方式可以分为静态质量属性和动态质量属性
- 静态质量属性是指实际代码和相关文档的质量，包括代码的结构化、可维护性、可测试性，文档的完整性和可读性
- 动态质量属性是指软件使用过程中质量属性，包括软件的可靠性、正确性、完备性、可用性、可维护性、可信性和性能等。
- 由于软件是一种不可见的复杂的逻辑体，而且软件开发过程中环节较多，测试工作上的人力、物力是有限的，不可能做穷举测试，所以，客观上给软件质量留下很多隐患。再加上我们在技术，以上也存在很多局限性，例如软件测试工具本身也需要测试、软件测试工具对软件的理解程度达不到人的理解程度、软件技术的发展使得软件的测试方法需要不断地扩展新的测试技术及软件质量没有统一指标来评价等因素，在度量和提高软件质量方面，我们还有很长的路要走。

1 软件质量特性类图



BACK

2 错误/缺陷无处不在

- 错误是我们生活的一个组成部分，不仅在思想上、行动上会犯错误，而且这些错误可以表现到人们生产出来的产品上，可以说，错误处处都可以发生。
- 人们可以犯语言错误、观察错误、处方错误、手术错误、驾驶错误、运动错误、恋爱错误等等各种各样的错误，当然程序员在软件开发的过程中也就可能犯类似的错误。
- 错误的后果有时可能微不足道，有时却可能导致重大的灾难，特别是软件开发过程中，人们的错误往往会给软件留下隐患。
- 软件工程就是一门与缺陷做斗争的工程学科

2 软件缺陷和软件故障案例

- 案例1　美国迪斯尼公司的狮子王游戏软件bug
  兼容性问题
- 案例2　美国航天局火星登陆事故
  系统测试　衔接问题
- 案例3　跨世纪“千年虫”问题
- 案例4　爱国者导弹防御系统炸死自家人
  系统时钟误差积累
- 案例5　Windows 2000 中文输入法漏洞
- 案例6　金山词霸bug　屏幕取词时 explore.exe出错
- 案例7　飞机场事件

上述所有实例中的软件问题在软件工程或软件测试中都被称为软件缺陷或软件故障。

2 软件缺陷是什么

2.1、描述软件失败的术语

要想成为软件测试员，就要使用各种术语描述软件失败时的现象。常用的术语为：

缺点（defect）　　偏差（variance）
谬误（fault）　　失败（failure）
问题（problem）　　矛盾（inconsistency）
错误（error）　　特殊（feature）
毛病（incident）　　缺陷（bug）
异常（anomaly）

将所有的软件问题通称微缺陷，不管它是大的、小的、有意的、无意的，因为它们都会制造障碍。

2 软件缺陷与故障（续）

2.2、软件缺陷的定义

（1）软件未达到产品说明书中已经标明的功能；

（2）软件出现了产品说明书中指明不会出现的错误；

（3）软件未达到产品说明书中虽未指出但应当达到的目标；

（4）软件功能超出了产品说明书中指明的范围；

（5）软件测试人员认为软件难以理解、不易使用，或者最终用户认为该软件使用效果不良。

- 举例讨论：手机的嵌入式软件

2 软件缺陷与故障（续）

2.3、软件缺陷的特征

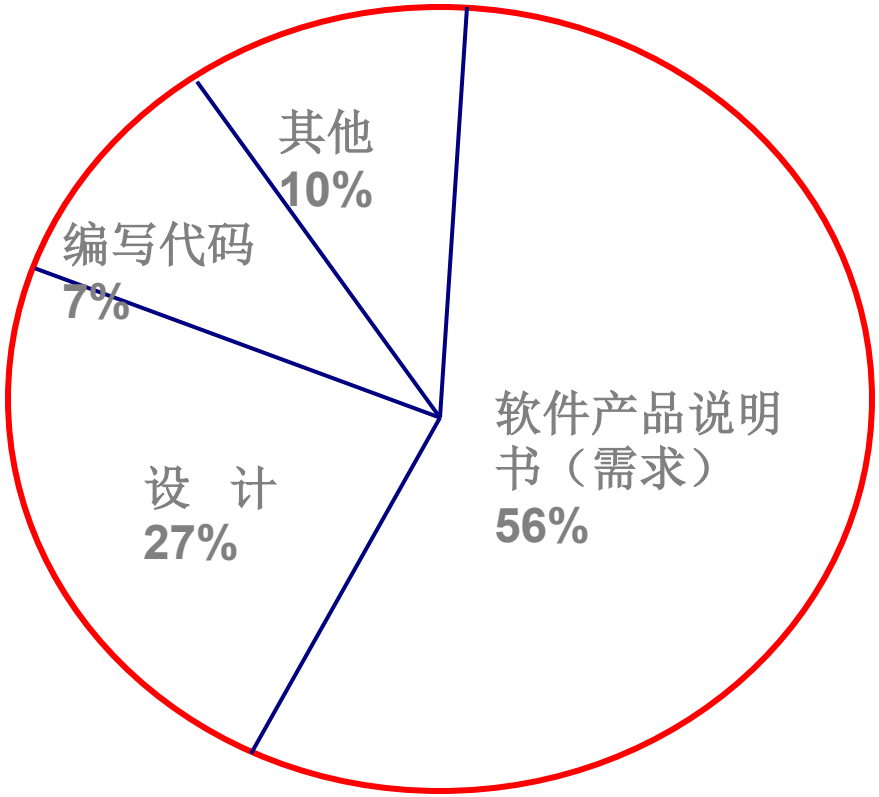
- “看不到”

  ——软件的特殊性决定了缺陷不易看到

- “看到但是抓不到”

  ——发现了缺陷，但不易找到问题发生的原因所在

2 软件缺陷产生的原因



软件缺陷产生的原因分布

2 软件测试和缺陷修复的代价

- 软件在从需求、设计、编码、测试一直到交付用户公开使用后的过程中，都有可能产生和发现缺陷。随着整个开发过程的时间推移，更正缺陷或修复问题的费用呈几何级数增长。

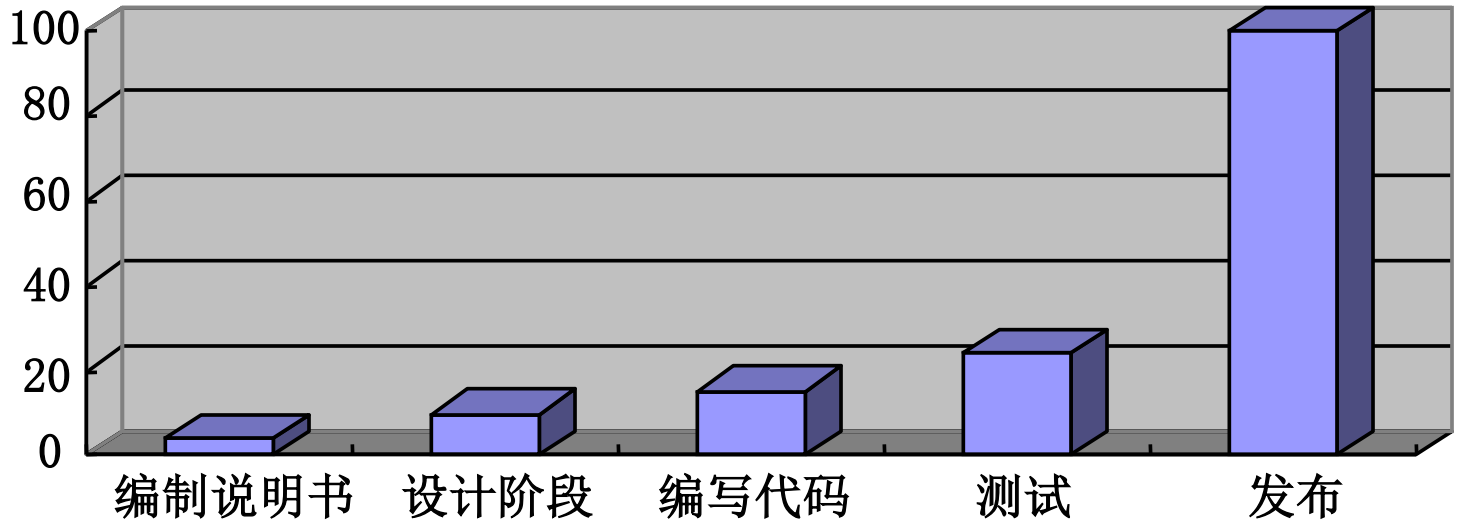


图1-4 软件缺陷在不同阶段发现时修复的费用示意图

2 软件缺陷管理

- 缺陷预防
- 缺陷发现（软件测试的目的）
- 缺陷记录和报告
- 缺陷分类和跟踪
- 缺陷处理
- 缺陷预测

BACK

小测试

- 三角形分类程序
- 输入：三个正整数
- 输出：判断三角形是等腰三角形、等边三角形还是普通三角形。
- 要求：记录时间，写好程序员姓名学号、评审员姓名学号及发现的问题
- 考察点：提前体验白盒测试与黑盒测试、同行评审等知识点

简单程序的现场编写与互查

```
#include <iostream>
using namespace std;
int main() {
    int a, b, c;
    cin >> "请输入三条边" >> a >> b >> c;
    if(a+b<=c || a+c<=b || b+c<=a)
        cout<<"这三条边不能够成三角形";
    else if (a==b==c)
        cout<<"这是一个等边三角形";
    else if (a==b || b==c || c==a)
        cout<<"这是一个等腰三角形";
    else cout<<"这是一个普通的三角形";
    return 0;
}
```

没有double，没有判断是否为正

修改人：李海鹏

08122102 郭良宾

3 软件测试的定义

1、软件测试的定义

软件测试就是在软件投入运行前，对软件需求分析、设计规格说明和编码实现的最终审查，它是软件质量保证的关键步骤。通常对软件测试的定义有两种描述：

- **定义1：** 软件测试是为了发现错误而执行程序的过程。
- **定义2：** 软件测试是根据软件开发各阶段的规格说明和程序的内部结构而精心设计的一批测试用例，并利用这些测试用例运行程序以及发现错误的过程，即执行测试步骤。

3 软件测试的定义

- **定义3：**使用人工或自动的手段来运行或测定某个软件系统的过程，其目的在于检验软件是否满足规定的需求或弄清预期结果和实际结果之间的差别。

3 软件测试的定义（续）

- **定义4：** 测试是以评价一个程序或者系统的属性为目标的任何一种活动，测试是对软件质量的度量。
- **定义5：** 软件测试是一种作为主体的人通过各种手段对客体软件的某种固有属性进行的一种以认知和改造为目的的活动。
- **定义6：** 测试是为了度量和提高被测试软件的质量，对测试软件进行工程设计，使用和维护的并发生命周期活动

3 软件测试的定义7（续）

- **测试：**所谓测试的含义，首先是一项活动，在这项活动中某个系统或组成的部分将在特定的条件下运行，结果将被观察和记录，并对系统或组成部分进行评价。测试活动有两种结果：找出缺陷和故障，或显示软件执行正确。测试是一个或多个测试用例的集合。
- **测试用例：**所谓测试用例是为特定的目的而设计的一组测试输入、执行条件和预期的结果；测试用例是执行测试的最小实体。
- **测试步骤：**测试步骤详细规定了如何设置、执行、评估特定的测试用例。

3 软件测试的定义7（续）

2、软件测试的基本问题

- 软件生命周期：一个软件生命周期包括制定计划、需求分析定义、软件设计、程序编码、软件测试、软件运行、软件维护、软件停用等8个阶段。
- 软件测试的对象：

  ——软件测试不等于程序测试。

  ——软件测试贯串于软件定义和开发的整个过程。

  ——软件开发过程中所产生的需求规格说明、概要设计规格说明、详细设计规格说明以及源程序都是软件测试的对象。

3 软件测试的定义7（续）

2、软件测试的基本问题（续）

- 软件测试在软件生命周期中横跨两个阶段：

  第一个阶段：单元测试阶段，即在每个模块编写出以后所做的必要测试。

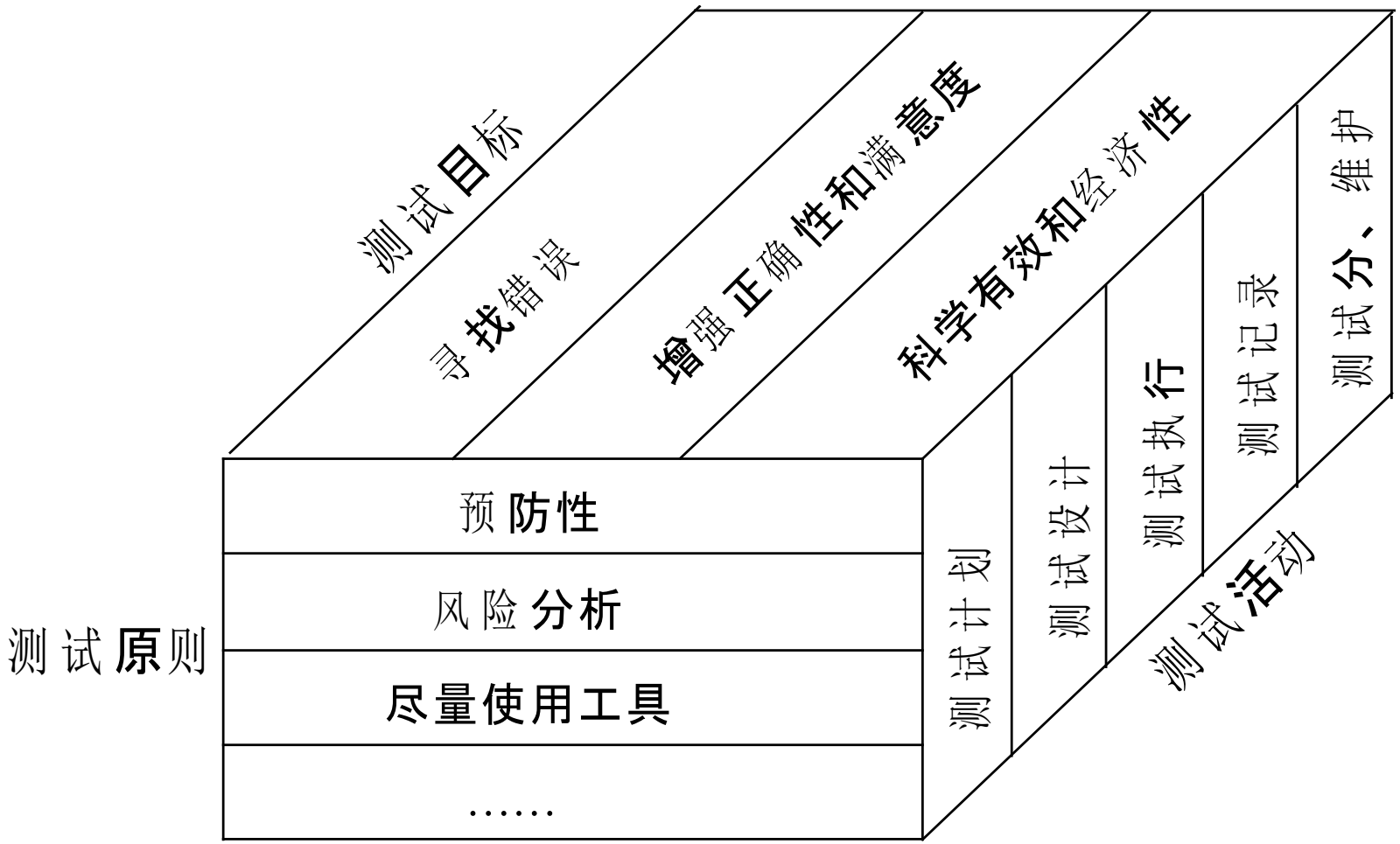
  第二个阶段：综合测试阶段，即在完成单元测试后进行的测试，如集成测试、系统测试、验收测试。

- 软件测试涉及的关键问题包括四个方面：

  （1）测试由谁来执行。　（2）测试什么。

  （3）什么时候进行测试。　（4）怎样进行测试。

3 软件测试的定义8



软件测试的目标、活动和原则

3 Definition 1 [Myers79]

- Software Testing is the process of executing a program or system with the intent of finding errors.

3 Definition 2 [Hetzel88]

- it involves any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results.

3 Definition 3 [Wikipedia]

- **Software testing** is an investigation conducted to provide stakeholders with information about the quality of the product or service under test.
- Software testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks at implementation of the software.
- Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs.

3 Definition 4 [worldIQ.com]

- **Software testing** is a process used to identify the correctness, completeness and quality of developed computer software.
- Actually, testing can never establish the correctness of computer software, as this can only be done by formal verification. It can only find defects, not prove that there are none.
- There are a number of different testing approaches that are used to do this ranging from the most informal *ad hoc* testing, to formally specified and controlled methods such as automated testing.

3 Definition 5(SWEBOK2014)

- Software testing consists of the *dynamic verification* that a program provides *expected behaviors* on a *finite set of test cases, suitably selected from* the usually infinite execution domain.

BACK

4 测试的普遍性：Testing exists everywhere

- Shopping
- Education
- Produce
- Social communication
- Academic
- Industrial

4 Everyone is a tester

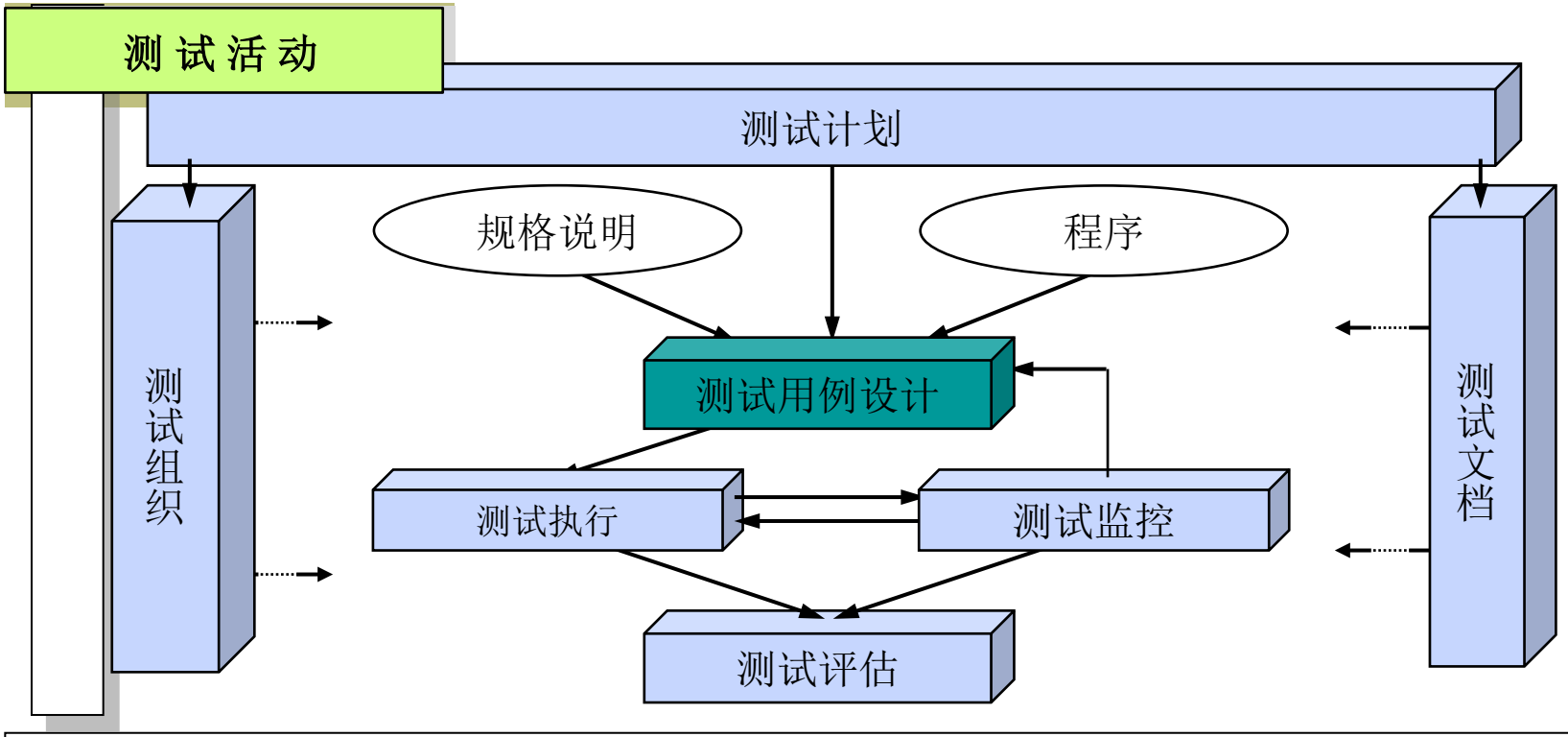
- Make friends
- Buy something
- Work

4 他山之石,可以功玉（讨论）

- 教育
- 建筑
- 制造业
- 软件
- 农业
- 各行各业
- … …

BACK

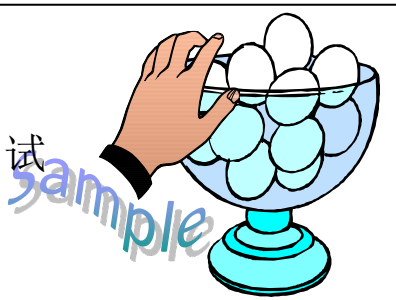
5 软件测试的关键问题--测试活动



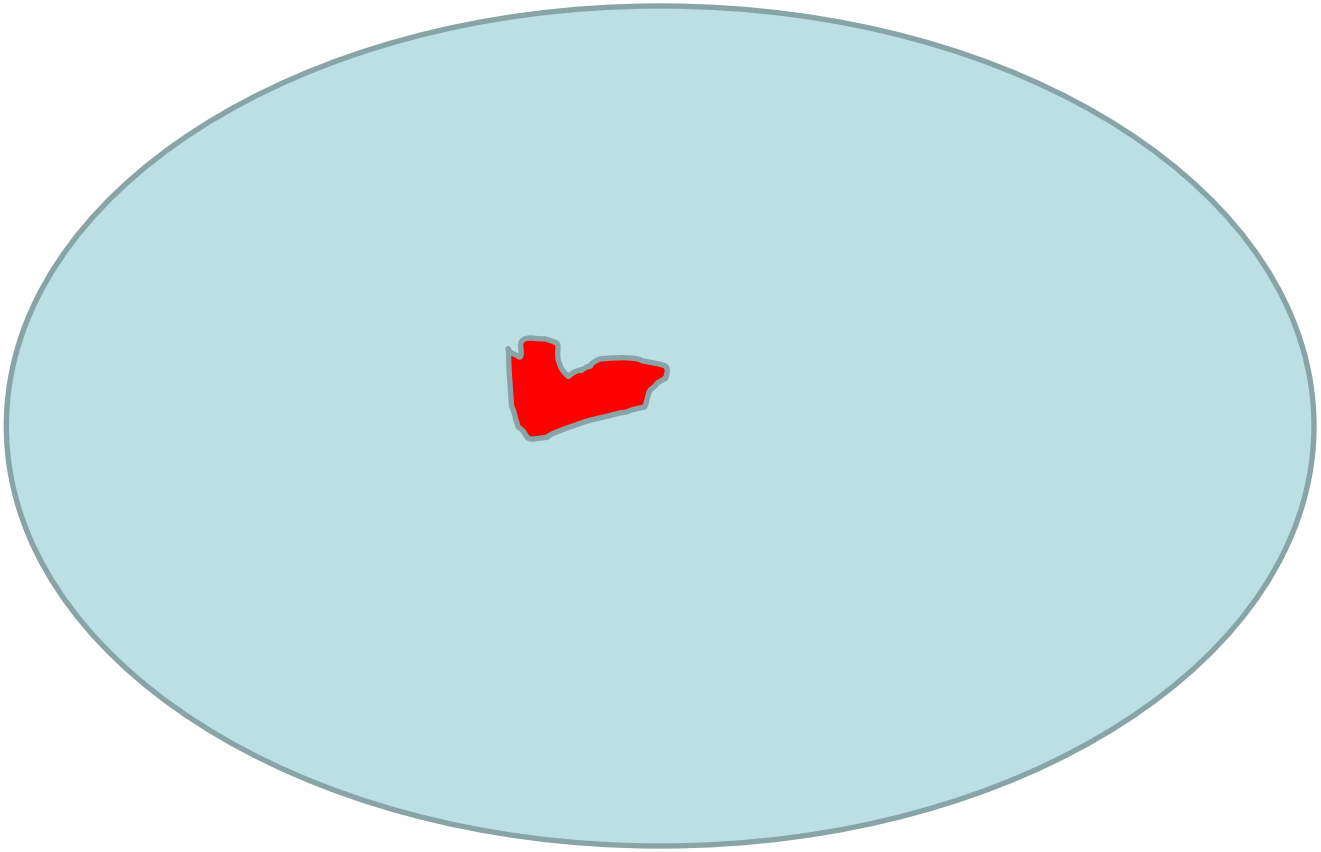
穷尽测试通常是不可能的（Key issue）

测试数据必须根据某些测试标准进行选择（最小的可靠测试用例集合）

各种测试方法

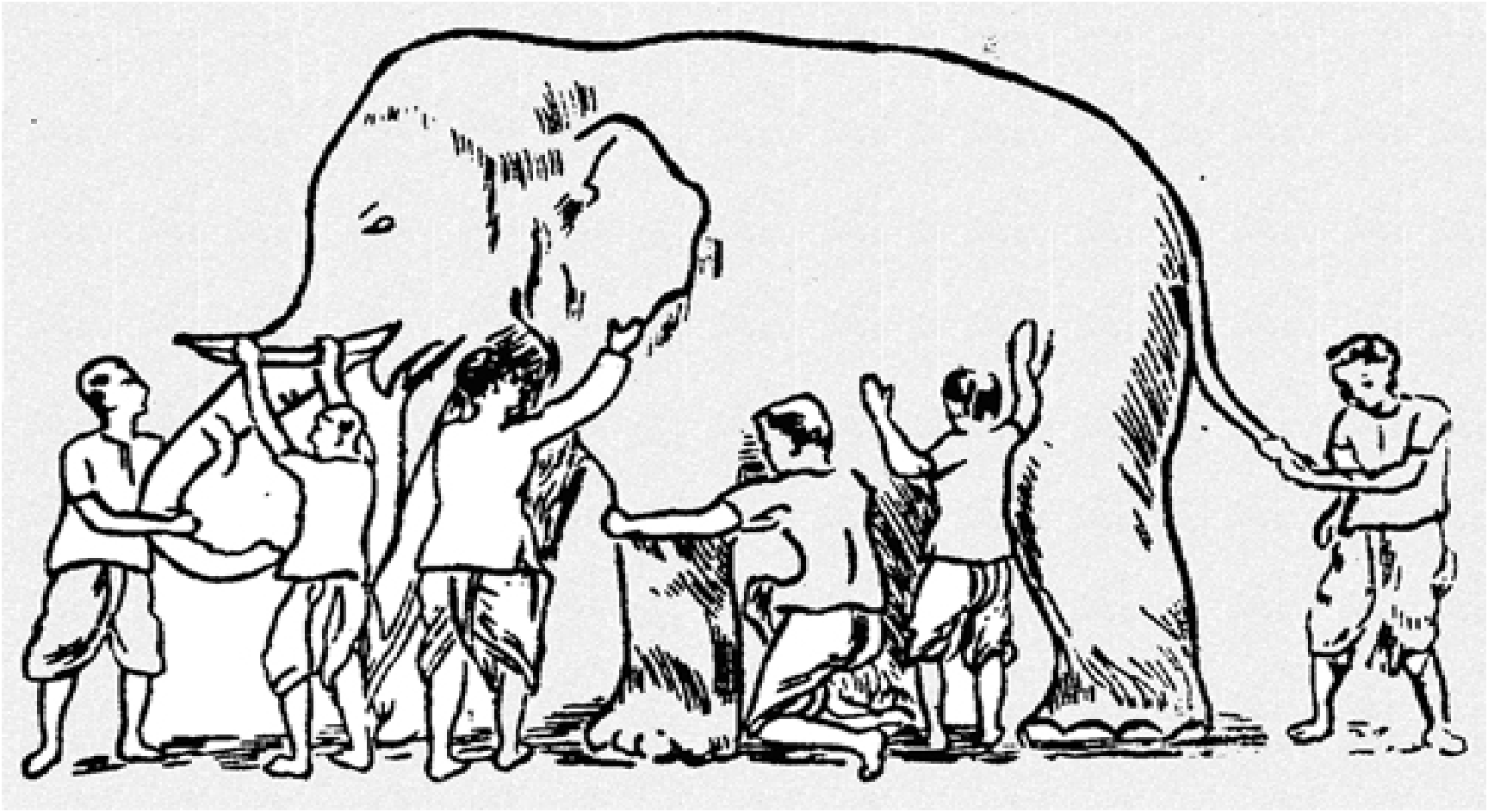


5 寻找最小有效测试用例集



Tall, 软件所有可以使用的测试用例（输入或其他事件等）

5 真理面前，我们都是盲人



软件测试方法体系

基于代码
code

基于用户
user

基于故障
fault

基于输入
input

基于属性
attribute

基于性质
property

基于开发过程
stage

基于规格
specification

基于软件开发
方法及特性
Nature inspired

基于模型
model

基于专家经验
Intuition&Experience

？Future？
更多新方法探索
更完整的理论

5 软件测试方法

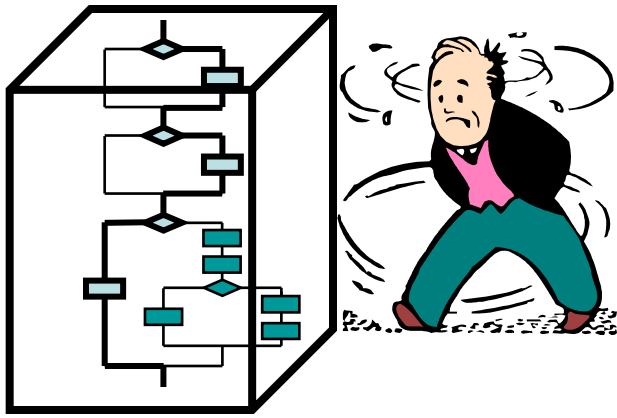

针对不同开发方式和应用场景的软件测试方法
面向对象软件测试
面向方面软件测试
面向服务软件测试
基于构件软件测试
嵌入式软件测试
普适环境软件测试
云计算软件测试
Web应用软件测试
网构软件测试
其他新型软件测试
针对软件不同特性和方面的软件测试方法
负载测试
压力测试
性能测试
安全性测试
安装测试
可用性测试
稳定性测试
配置测试
文档测试
兼容性测试
针对不同开发阶段的软件测试方法
单元测试
集成测试
系统测试
验收测试
回归测试
验证测试
确认测试
Alpha测试
Beta测试
Gamma测试
不同特殊的软件测试方法
组合测试
蜕变测试
变异测试
演化测试
FUZZ测试
基于性质的测试
基于故障的测试
基于模型的测试
统计测试
逻辑测试
软件测试的基本方法
静态测试
桌面检查
代码审查
代码走查
右下方三行属于动态测试
动态测试
黑盒测试
等价类划分
边际值分析
因果图分析
错误猜测
状态转换测试
白盒测试
语句覆盖
判定覆盖
条件覆盖
路径覆盖
条件组合覆盖

5 软件测试的分类

软件测试按照不同的划分方法，有不同的分类：

- 按照软件测试用例的设计方法而论，软件测试可以分为白盒测试法(10)和黑盒测试法(6)。

- test case design is performed on the basis of the internal structure
- common test approach (included in many standards)
- not possible to check whether all requirements have been implemented
- difficult to automate (limits of symbolic execution)
- full coverage often not achievable



白盒测试法(10)和黑盒测试法(6)

1 黑盒测试
2 白盒测试
3 静态测试
4 动态测试
5 文档审查
6 同行评审
7桌面评审
8 代码走查
9 代码审查
10语句覆盖
11判定覆盖
12条件覆盖
13判定/条件覆盖
14条件组合覆盖
15 修改决策条件覆盖
16 路径覆盖
17 LCSAJ 覆盖
18数据流测试（定义/引用对覆盖）
19 等价类划分
20 边界值分析
21 因果图与决策表
22 错误猜测
23 状态转换测试
24 语法测试

5 软件测试的分类

- 按照软件测试的策略和过程来分类，软件测试可分为单元测试、集成测试、系统测试、验证测试和确认测试。

5 不同阶段的测试(9)

- Unit testing
- Integration testing
- Smoke testing
- System testing
- Regression testing
- Acceptance testing
- Alpha testing
- Beta testing
- Gamma testing

5 不同应用软件的测试（20）

- 6.1面向对象软件的测试(Object Oriented Software Testing)
- 6.2面向方面的软件测试(Aspect Oriented Software Testing)
- 6.3面向服务的软件测试(Service Oriented Software Testing)
- 6.4构件软件测试(Component Based Software Testing)
- 6.5 Web应用软件测试(WEB Testing)
- 6.6普适计算环境下的软件测试(Pervasive Computing Software Testing)
- 6.7云测试(Cloud Testing)　云原生
- 6.8物联网环境下的软件测试(Software Testing for Internet of Things)
- 6.9并行并发软件测试(Concurrent Software Testing)
- 6.10嵌入式软件测试(Embedded Software Testing)
- 6.11高可信软件测试(High Confidence Software Testing)
- 6.12网构软件测试（Internetware testing）
- 6.13移动应用软件测试（app testing）
- 6.14 人工智能软件测试（Artificial intelligence testing）
- 6.15 区块链软件系统测试(blockchain software testing)
- 6.16 元宇宙测试(metaverse testing)
- 6.17 大模型测试（LLM testing）
- 6.18 开源系统测试（opensource software testing）
- 6.19 实时系统测试（Realtime system testing）
- 6.20 量子软件测试（quantum software testing）

5 不同测试技术（30+）

5.1 组合测试（Combinatorial Testing） 127
5.2 蜕变测试（Metamorphic Testing） 131
5.3 基于规格说明的软件测试(Specification Based Software Testing) 133
5.4 基于模型的软件测试（model based testing） 134
5.5 基于错误的软件测试(fault based testing) 136
5.6 基于搜索的软件测试(Search Based Testing) 137
5.7 统计测试(Statistics Testing) 140
5.8 基于操作剖面的测试(Operational Profile Based Testing) 144
5.9 变异测试（Mutation Testing） 145
5.10 冒烟测试（Smoke Testing） 147
5.11 基于性质的软件测试方法(Property-Based Testing) 148
5.12 极限测试(Extreme Testing) 150
5.13 模糊测试（Fuzzing Testing） 152
5.14 软件测试的控制论方法(Cybernetics Based Testing)-自适应测试（Adaptive Testing） 154
5.15 导向性随机测试（Concolic Testing） 159
5.16 图形用户界面测试（GUI Testing） 161
5.17 随机测试(Random Testing) 165
5.18 自适应随机测试(Adaptive Random Testing) 167
5.19 反随机测试（Antirandom Testing） 169
5.20 结对测试(Pair Testing) 170
5.21 在线测试（Online Testing） 171
5.22 探索性测试（Exploratory Testing） 172
5.23 反模型测试（Anti-model Testing） 173
5.24 成分测试（Compositional Testing） 174
5.25 有限状态机测试（FSM Testing） 175
5.26 基于 Petri 网的测试（Petri Net based Testing） 177
5.27 基于模型检查的测试（Model Checking based Testing） 178
5.28 TTCN 测试（TTCN Testing） 179
5.29 布尔规格测试（Boolean Specification Testing） 180
5.30 基于统一建模语言测试（UML Based Testing） 181

5.1组合测试（Combinatorial Testing）
5.2蜕变测试（Metamorphic Testing）
5.3基于规格说明的软件测试(Specification Based Software Testing)

5.4基于模型的软件测试（model based testing）
5.5基于错误的软件测试(fault based testing)
5.6基于搜索的软件测试(Search Based Testing)
5.7统计测试(Statistics Testing)
5.8基于操作剖面的测试(Operational Profile Based Testing)
5.9变异测试（Mutation Testing）
5.10脆弱性测试（flaky Testing）
5.11基于性质的软件测试方法(Property-Based Testing)
5.12极限测试(Extreme Testing)
5.13模糊测试（Fuzzing Testing）
5.14软件测试的控制论方法(Cybernetics Based Testing)-自适应测试（Adaptive Testing）
5.15导向性随机测试（Concolic Testing）
5.16图形用户界面测试（GUI Testing）
5.17随机测试(Random Testing)
5.18自适应随机测试(Adaptive Random Testing)
5.19反随机测试（Antirandom Testing）
5.20结对测试(Pair Testing)
5.21在线测试（Online Testing）
5.22探索性测试（Exploratory Testing）

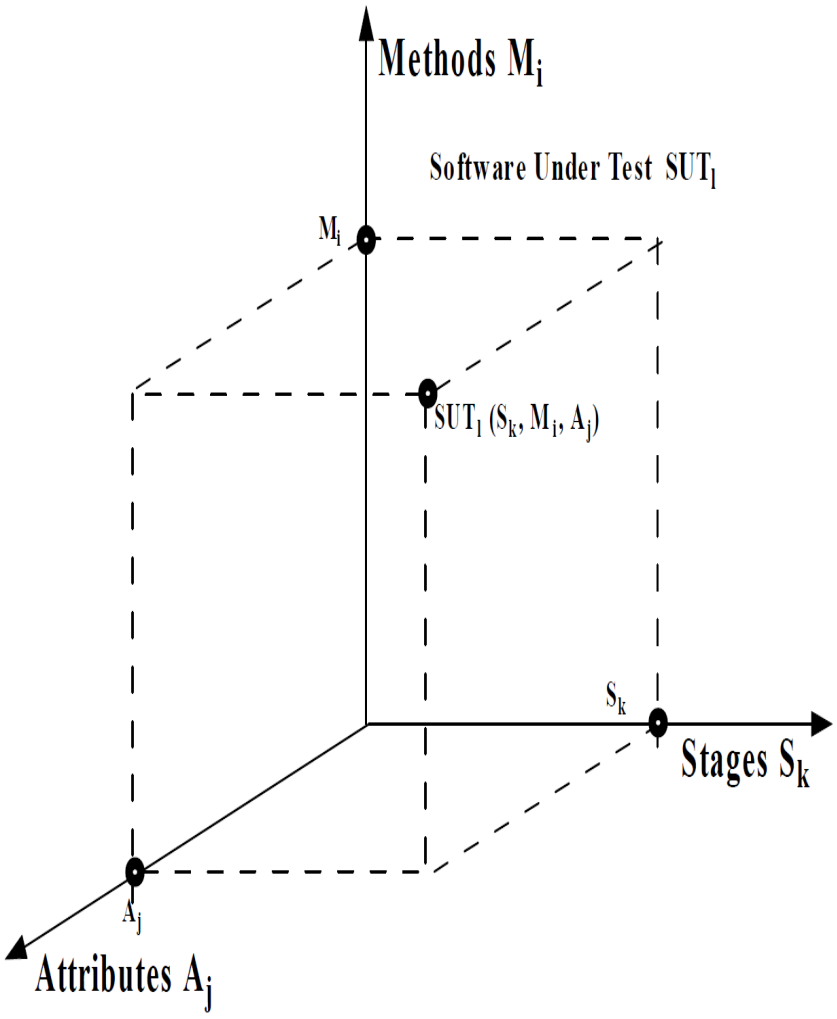
5.23反模型测试（Anti-model Testing）
5.24成分测试（Compositional Testing）
5.25 有限状态机测试（FSM Testing）
5.26 基于Petri网的测试（Petri Net based Testing）
5.27 基于模型检查的测试（Model Checking based Testing）
5.28 TTCN测试（TTCN Testing）
5.29 布尔规格测试（Boolean Specification Testing）
5.30 基于统一建模语言测试（UML Based Testing）
5.31 差分测试（differential testing）
5.32故障注入测试（Fault Injection Testing）
5.33 图覆盖测试（graph cover testing）
5.34 A/B测试（A/B testing）

5 不同方面测试（30+）

4.1 负载测试（Load Testing）.....80
4.2 压力测试（Stress Testing）.....83
4.3 性能测试（Performance Testing）.....84
4.4 可靠性测试（Reliability Testing）.....86
4.5 容量测试（Volume Testing）.....89
4.6 安全性测试（Security Testing）.....90
4.7 安装测试（Installation Testing）.....93
4.8 可用性测试（Usability Testing）.....97
4.9 稳定性测试（Stability Testing）.....98
4.10 本地化和国际化测试(Localization and Internationalization Testing).....100
4.11 可访问性测试（Accessibility Testing）.....101
4.12 授权测试（Authorization Testing）.....102
4.13 容错性测试（Fault Tolerance Testing）.....103
4.14 一致性测试（Conformance Testing）.....104
4.15 配置测试（Configuration Testing）.....105
4.16 文档测试(Document Testing).....109
4.17 兼容性测试(Compatibility Testing).....110
4.18 Playtest.....112
4.19 可恢复性测试（Recovery Testing）.....113
4.20 卸载测试（Uninstall Testing）.....114
4.21 能力测试(Facility Testing).....115
4.22 健壮性测试（Robustness Testing）.....116
4.23 穿越测试（By-pass Testing）.....117
4.24 在线帮助测试（Online Help Testing）.....118
4.25 数据转换测试（Data Conversion Testing）.....119
4.26 备份测试（Backup Testing）.....120
4.27 接口测试（Interface Testing）.....122
4.28 人机交互界面测试（User Interface Testing）.....123
4.29 余量测试（Remainder Testing）.....124
4.30 协议测试（Protocol Testing）.....125
4.31 内存泄漏测试（Memory Leak Testing）.....126

4.1负载测试（Load Testing）
4.2压力测试（Stress Testing）
4.3性能测试（Performance Testing）
4.4可靠性测试（Reliability Testing）
4.5 容量测试（Volume Testing）
4.6安全性测试（Security Testing）
4.7安装测试（Installation Testing）
4.8可用性测试（Usability Testing）
4.9稳定性测试（Stability Testing）
4.10 本地化和国际化测试(Localization and Internationalization
4.11 可访问性测试（Accessibility Testing）
4.12授权测试（Authorization Testing）
4.13容错性测试（Fault Tolerance Testing）
4.14一致性测试（Conformance Testing）
4.15配置测试（Configuration Testing）
4.16文档测试(Document Testing)
4.17兼容性测试(Compatibility Testing)
4.18 试玩 （Playtest）
4.19 可恢复性测试（Recovery Testing）
4.20 卸载测试（Uninstall Testing）
4.21 能力测试(Facility Testing)
4.22 健壮性测试（Robustness Testing）
4.23 穿越测试（By-pass Testing）
4.24 在线帮助测试（Online Help Testing）
4.25 数据转换测试（Data Conversion Testing）
4.26 备份测试（Backup Testing）
4.27接口测试（Interface Testing）
4.28 人机交互界面测试（User Interface Testing）
4.29 余量测试（Remainder Testing）
4.30 协议测试（Protocol Testing）
4.31 内存泄漏测试（Memory Leak Testing）
4.32 存储测试 (Storage Testing)
4.33 软件老化（software aging）
4.34 不稳定性测试（flaky testing）
4.35 工作流测试（workflow testing）
4.36 领域测试（domain testing）
4.37 输入测试（input testing）
4.38 使用测试（usage testing）
4.39 数据库测试（database testing）

Changhai Nie, Hareton Leung. **A Framework of Understanding Software Testing**.



Various Special Testing Methods M_i, $1 \le i \le 40$							
No.	Testing Method	No.	Testing Method	No.	Testing Method	No.	Testing Method
M_1	*White Box* [26]	M_2	*Black Box* [26]	M_3	*Static* [26]	M_4	*Dynamic* [27]
M_5	*Model Based** [16]	M_6	*Search Based** [22]	M_7	*Fault Based* [21]	M_8	*Property* [12]
M_9	*Metamorphic* [37]	M_{10}	*Operation Profile* [7]	M_{11}	*Statistics* [31]	M_{12}	*Combinatorial** [28]
M_{13}	*Mutation** [17]	M_{14}	*Specification* [34]	M_{15}	*Adaptive* [8]	M_{16}	*Random* [11]
M_{17}	*Anti-random* [39]	M_{18}	*Adaptive Random* [9]	M_{19}	*Concolic* [18]	M_{20}	*Anti-model* [6]
M_{21}	*Fuzzing* [35]	M_{22}	*Compositional* [5]	M_{23}	*GUI* [25]	M_{24}	*Pair* [38]
M_{25}	*Online* [36]	M_{26}	*Ad Hoc* [7]	M_{27}	*Exploratory* [2]	M_{28}	*Extreme* [26]
M_{29}	*Syntax* [30]	M_{30}	*FSM* [19]	M_{31}	*TTCN* [14]	M_{32}	*UML based* [23]
M_{33}	*Model Checking* [21]	M_{34}	*Playtest* [10]	M_{35}	*Petri-net* [41]	M_{36}	*Back-to-Back* [7]
M_{37}	*Error Guessing* [26]	M_{38}	*Boolean Spec.* [3]	M_{39}	*State Trans.* [33]	M_{40}	*Graph coverage* [30]
Testing Various Attributes and Aspects A_j, $1 \le j \le 36$ [21, 27]							
A_1	*Performance*	A_2	*Stress*	A_3	*Load*	A_4	*Volume*
A_5	*Reliability*	A_6	*Security**	A_7	*Recovery*	A_8	*Internationalization*
A_9	*Compatibility*	A_{10}	*Installation*	A_{11}	*Protocol*	A_{12}	*Fault Tolerance*
A_{13}	*Authorization*	A_{14}	*Uninstallation*	A_{15}	*Backup*	A_{16}	*Online Help*
A_{17}	*Robustness*	A_{18}	*Usability*	A_{19}	*Facility*	A_{20}	*Data Conversion*
A_{21}	*Configuration*	A_{22}	*Accessibility*	A_{23}	*Interface*	A_{24}	*Memory Leak*
A_{25}	*Conformance*	A_{26}	*Stability*	A_{27}	*Predicate*	A_{28}	*By-pass, Penetration*
A_{29}	*Remainder*	A_{30}	*Document*	A_{31}	*Workflow*	A_{32}	*Data Base*
A_{33}	*Input*	A_{34}	*Logic*	A_{35}	*Domain*	A_{36}	*Usage*
Testing Various Development Stages S_k, $1 \le k \le 8$ [27]							
S_1	*Unit*	S_2	*Smoke(daily build)*	S_3	*Integration*	S_4	*System*
S_5	*Acceptance*	S_6	*Regression**	S_7	*Alpha*	S_8	*Beta*
Testing Various Softwares SUT_l, $1 \le l \le 16$ [21, 7, 27]							
SUT_1	*Pervasive Computing*	SUT_2	*Aspect-oriented**	SUT_3	*Component-based*	SUT_4	*Concurrent**
SUT_5	*High Confidence*	SUT_6	*Object-oriented*	SUT_7	*Protocol-based*	SUT_8	*Real-time*
SUT_9	*Internet of Things**	SUT_{10}	*Service-oriented*	SUT_{11}	*Web-based*	SUT_{12}	*Internetware** [24]
SUT_{13}	*Cloud Computing** [13]	SUT_{14}	*Safty-critical*	SUT_{15}	*Open-source*	SUT_{16}	*Embedded* [32]

Our road map

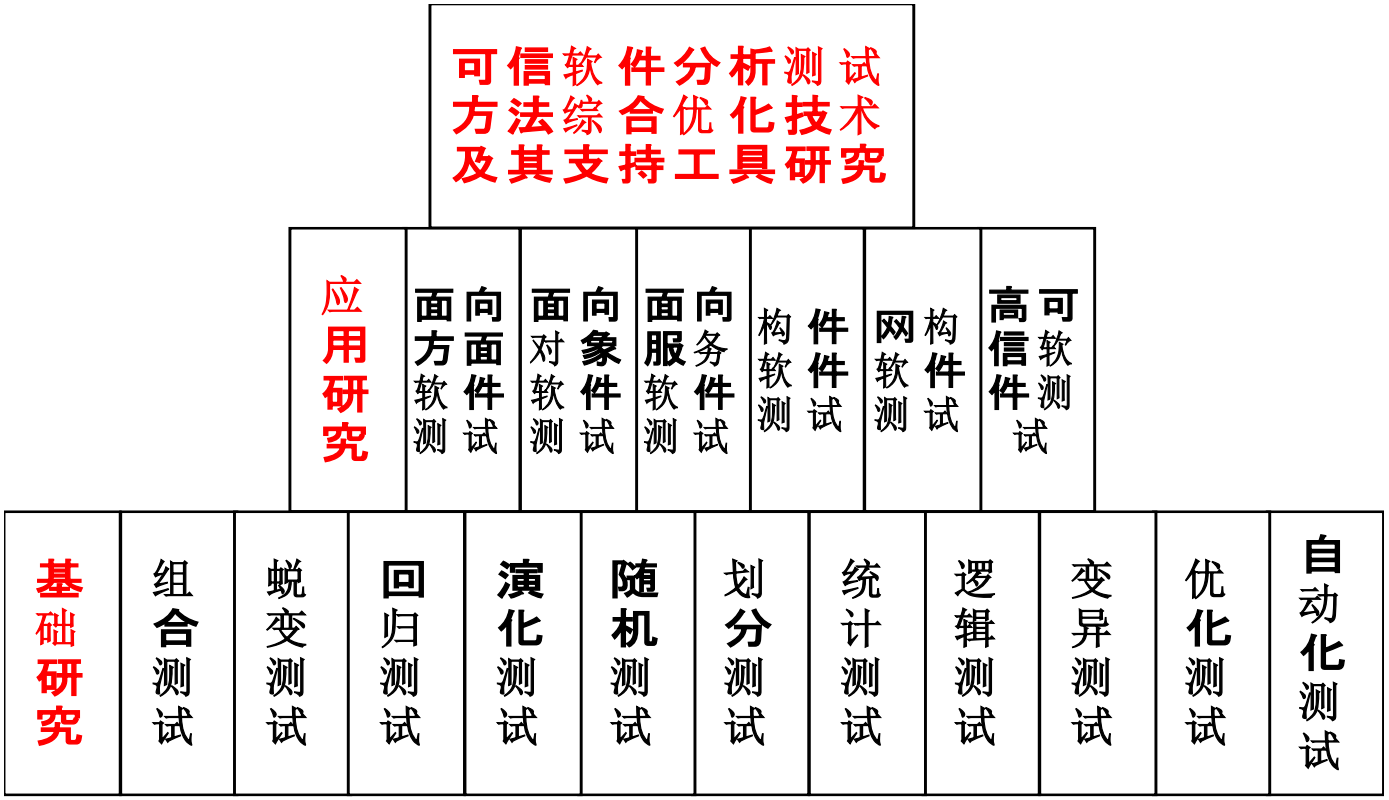


图 1： 可信软件测试方法和技术研究路线图

Our road map

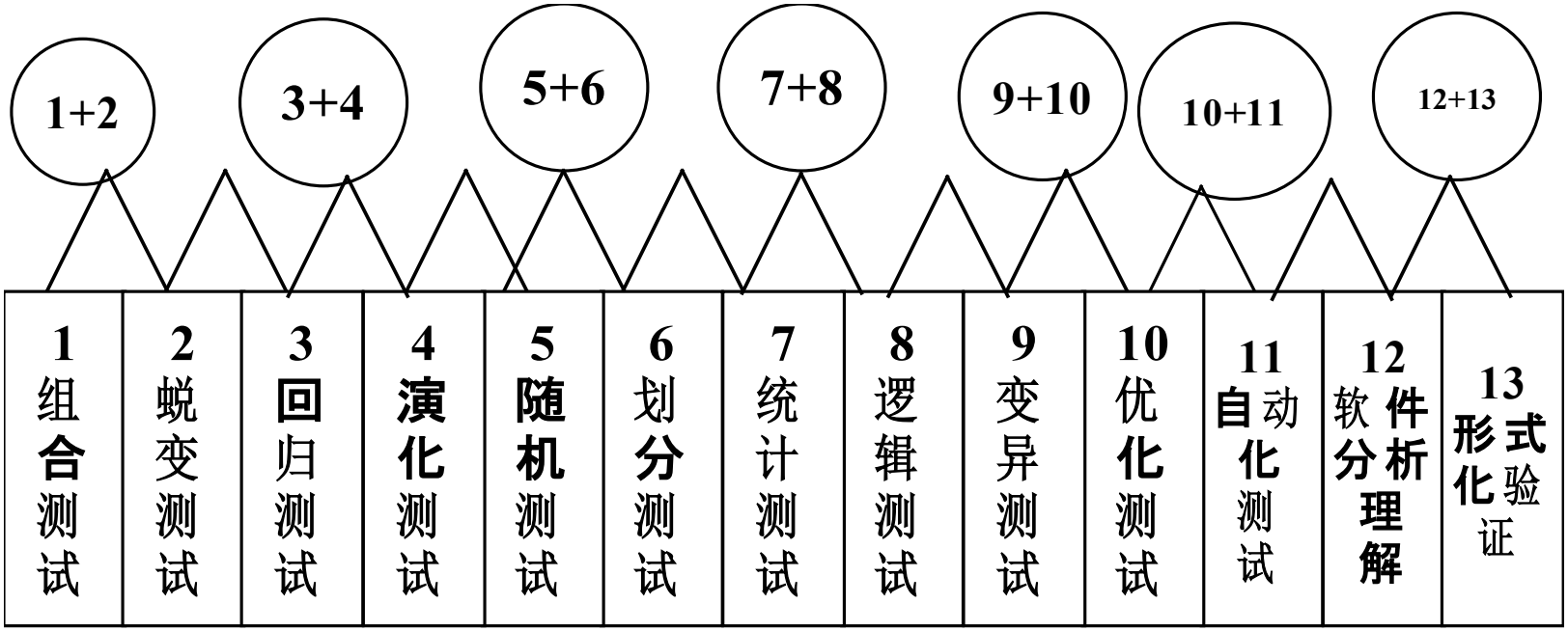
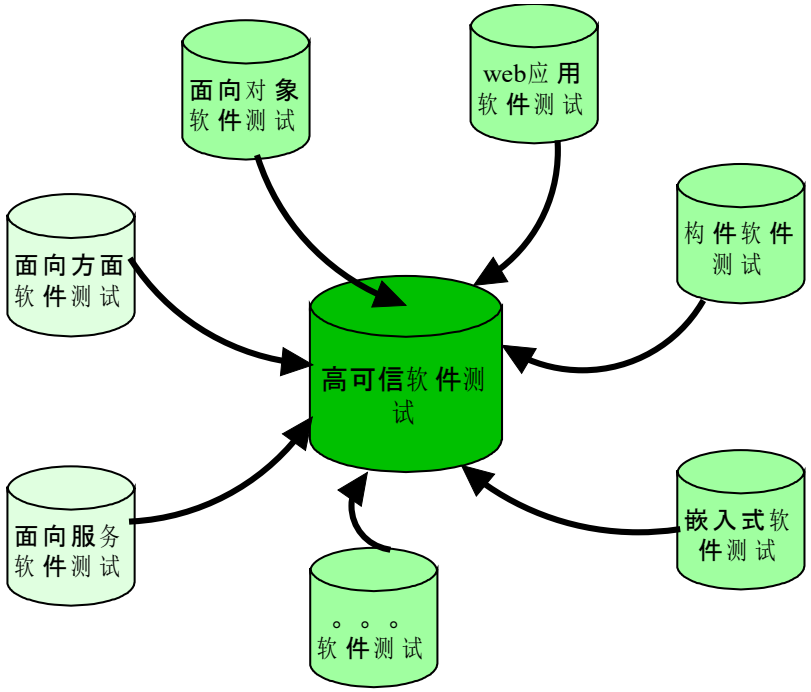


图 2：各种方法的两两组合研究 $C_n^2, C_{13}^2 = 78$

Our road map



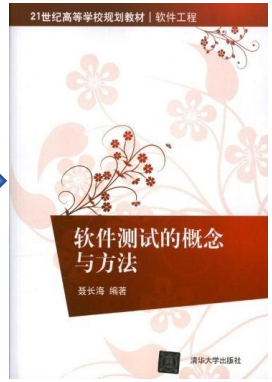
相互借鉴，**相互**启发，**相互**补**充**，**相互促**进

Our road map

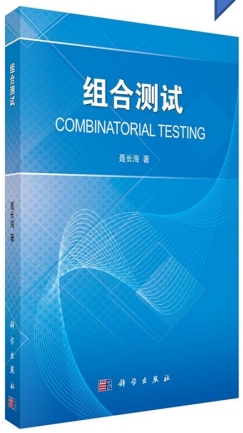
科学研究的 "深" 与 "跟"

从 “组合测试” 到 “软件测试” 再到 “智能化软件质量保证”
由点到面到体的教学与科研体系 （2000 -- 2020）跨越20年

百分之一

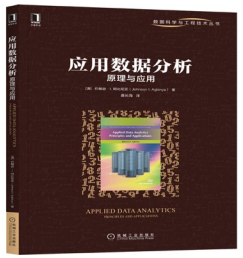


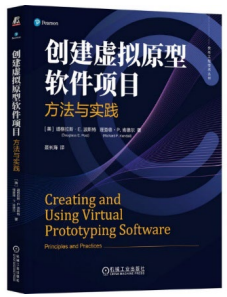
百分之一



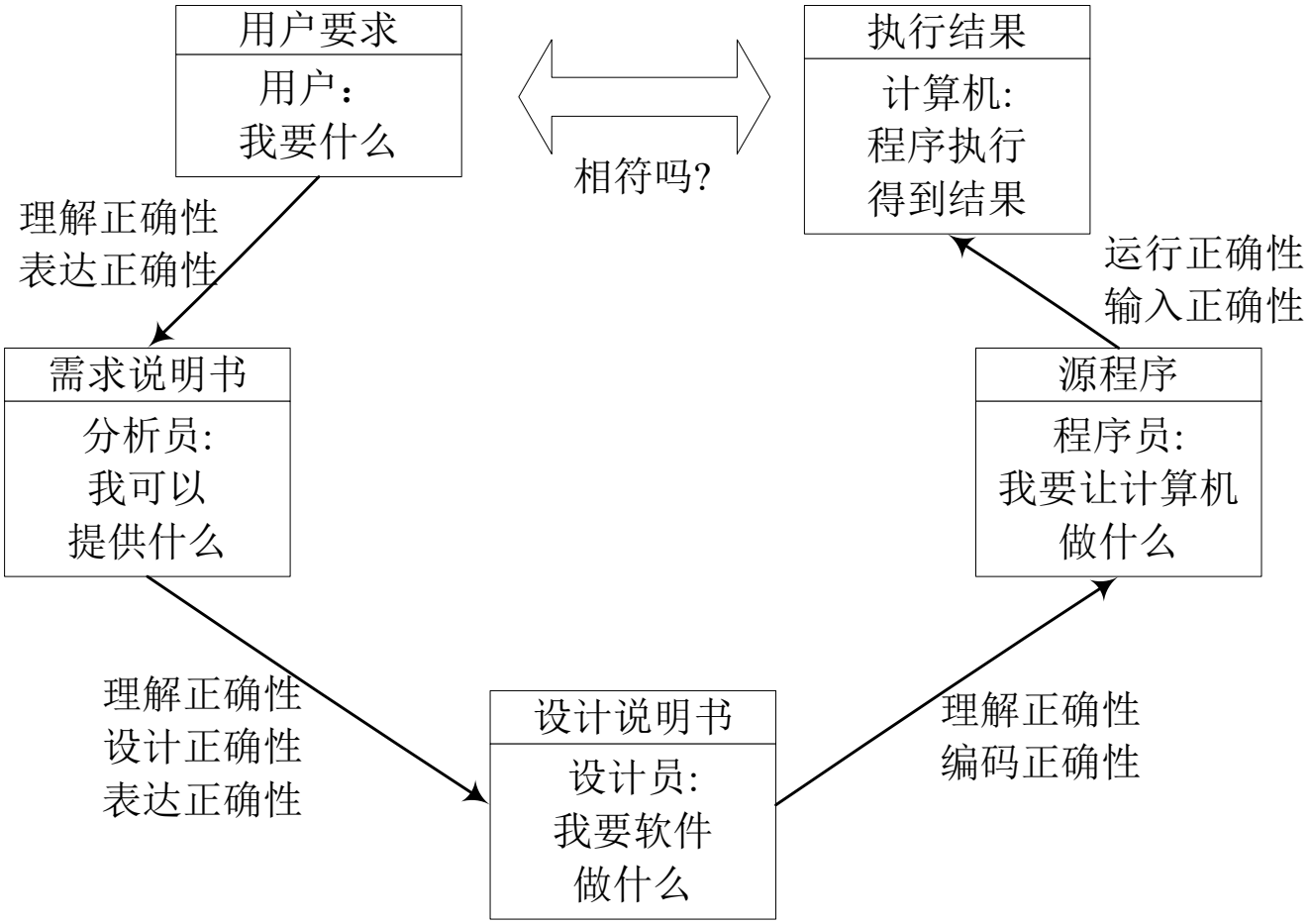
既要 “深度挺进”，又要 “广度发展”，还要 “与时俱进”
否则会肤浅，　　不然会狭隘，　　照样会过时

从人工智能到区块链再到元宇宙
紧跟软件发展步伐





6-1 软件测试的对象


用户要求
用户：
我要什么
执行结果
计算机:
程序执行
得到结果
相符吗?
理解正确性
表达正确性
运行正确性
输入正确性
需求说明书
分析员:
我可以
提供什么
源程序
程序员:
我要让计算机
做什么
理解正确性
设计正确性
表达正确性
设计说明书
设计员:
我要软件
做什么
理解正确性
编码正确性

6-2 软件测试的目的

（1）测试是程序的执行过程，目的在于发现错误；不能证明程序的正确性，除非仅处理有限种情况。

（2）检查系统是否满足需求也是测试的期望目标。

（3）一个好的测试用例在于发现了还未曾发现的错误；一次成功的测试则是发现了错误的测试。

注意：测试无法说明错误不存在，只能说明软件错误已出现。

Purpose of testing

- Testing usually serves more than one purpose. Typical purposes include, but are not restricted to:
- a) detecting defects - this allows for their subsequent removal thus increasing software quality;
- b) gathering information on the test item - testing generates information; this information can serve different purposes, such as:
  - developers can use the information to remove defects, increase the code quality or learn to create better code in the future;
  - testers can use the information to create better test cases;
  - managers can use the information to decide when to stop testing;
  - users eventually benefit from a higher product quality.
- c) creating confidence and taking decisions - by providing evidence that the test item performs correctly under specific circumstances, the stakeholders' confidence that the test item will perform correctly operationally increases; with sufficient confidence, stakeholders can decide to release the test item.
- Testing may be performed for some or all of the above purposes; and additional purposes not listed may also exist; these purposes should be identified and agreed as a starting point to any testing activity.

6-3 软件测试的意义

- 对消费者而言
- 对生产者而言
- 微软公司中一半人做测试，而其余的人则一半时间都在做测试

6-4 软件测试的原则

（1）尽早地和及时地测试；
（2）测试用例应当由测试数据和与之对应的预期结果这两部分组成；
（3）在程序提交测试后，应当由专门的测试人员进行测试；
（4）测试用例应包括合理的输入条件和不合理的输入条件；
（5）严格执行测试计划，排除测试的随意性；
（6）充分注意测试当中的群体现象；
（7）应对每一个测试结果做全面的检查；
（8）保存测试计划、测试用例、出错统计和最终分析报告，为维护工作提供充分的资料。
（9）进行风险分析，确定计划风险
（10）制定测试策略
（11）使用测试清单
（12）使用测试工具
（13）度量测试有效性
（14）不断进行培训
（15）宣传测试思想
（16）软件测试是一种服务
（17）软件测试是一项极富创造性、极具智力挑战性的工作
（18）选择一种合适的方法
（19）增量和分级测试
（20）分析缺陷趋势和模式

7 测试信息流程

测试信息流程如图1-2所示。测试过程中需要三类输入：软件配置、测试配置和测试工具。

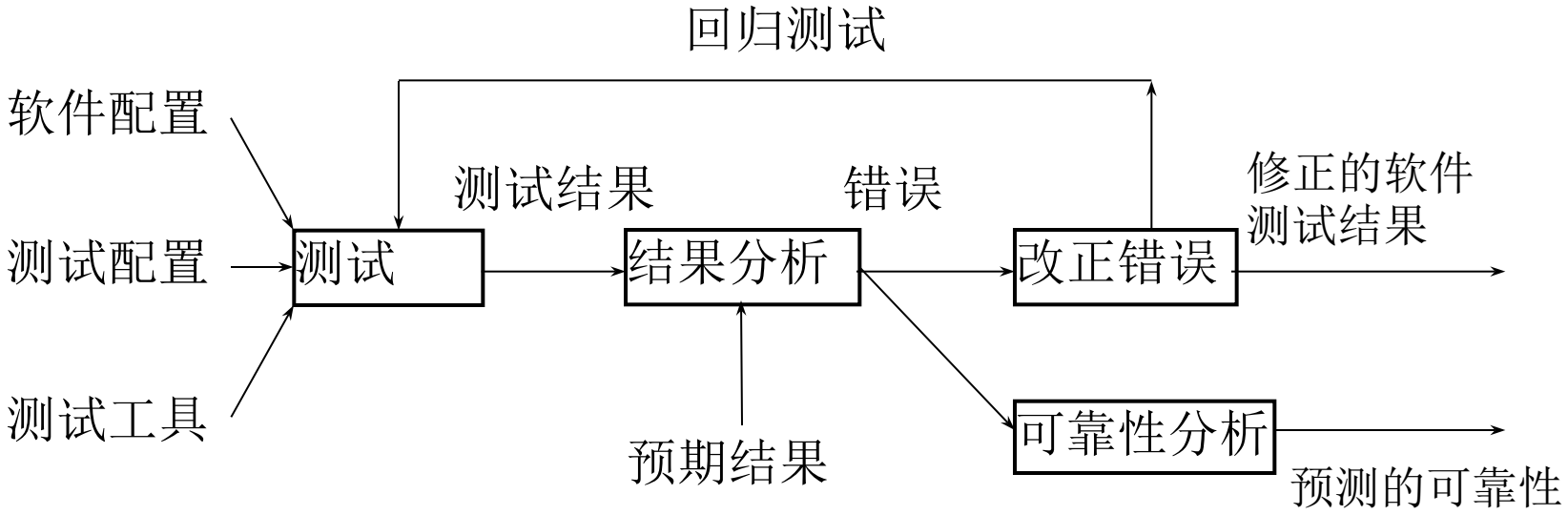


图1-2 测试信息流程

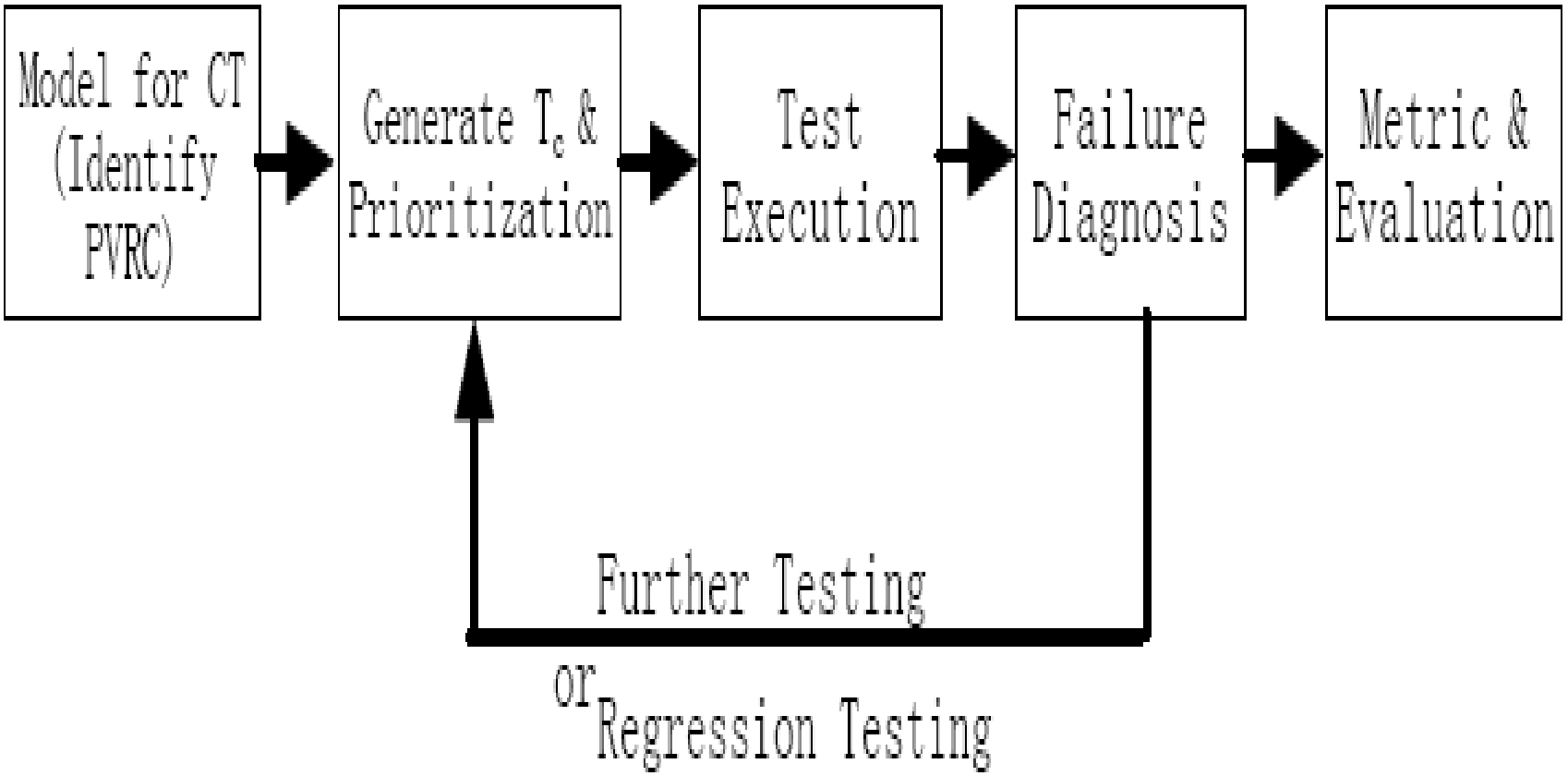
测试信息流

(1) 软件配置：是指测试对象，它包括软件需求规格说明、软件设计说明、和被测试的源程序清单。

(2) 测试配置：包括测试计划，测试用例，测试驱动程序，实际上，在整个软件工程中，测试配置只是软件配置的一个子集。

(3) 测试工具：为提高软件测试效率，可使用测试工具支持测试。其作用就是为测试的实施提供某种服务，以减轻人们完成测试任务中的手工劳动。例如：测试数据自动生成程序、静态分析程序、动态分析程序、测试结果分析程序、以及驱动测试的数据库等等。

A typical procedure for CT



软件测试的周期性与并行性

软件测试的周期性是“测试->改错->再测试->再改错”这样一个循环过程，如下图1-3所示。

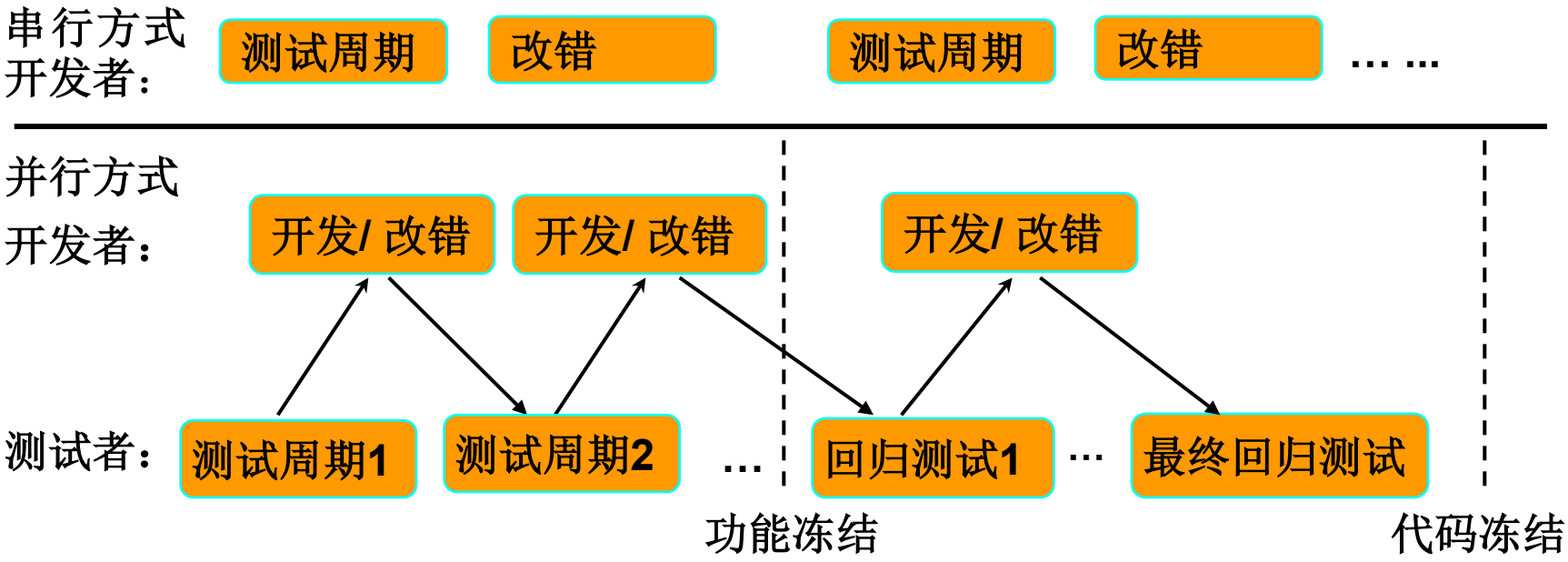


图1-3 软件测试的周期性

软件测试的特性

- “Program testing can be used to show the presence of bugs, but never to show their absence.” ----A Famous Dijkstra Quote
- 不完全的软件测试无法证明软件的正确性，只能证明软件的不正确性。尽管软件测试是保证软件质量的一个重要手段，但它的能力是有限的；经过软件测试的软件不一定就完全可靠。【证错不证对性】
- 软件测试是一项技术性工作，同时涉及经济学和心理学的一些重要因素，甚至涉及管理学的很多内容，所以，软件测试是一门实践性很强的综合学科。【综合性或多方求证性】
- 一般情况下，软件的输入空间非常庞大，程序中可以执行的路径也可能无法穷尽，而且软件的规格说明有着不同标准，可能存在不完整，不准确等问题，这就决定了软件测试是一种不完全测试，部分测试或者抽样性测试。【不完全性或抽样性】

测试停止的依据（标准）

- 第一类标准：测试超过了预定时间，则停止测试。
- 第二类标准：执行了所有的测试用例，但并没有发现故障，则停止测试。
- 第三类标准：使用特定的测试用例设计方案作为判断测试停止的基础。
- 第四类标准：正面指出停止测试的具体要求，即停止测试的标准可定义为查出某一预订数目的故障。
- 第五类标准：根据单位时间内查出故障的数量决定是否停止测试。

8 软件开发模型中的软件测试

- 1、大棒开发法
- 2、边写边改法
- 3、瀑布法
- 4、快速原型法
- 5、螺旋模式法
- 6、敏捷开发模式

8 软件开发模式

1、大棒开发法

- 源于能量爆发创造宇宙，万物都由能量和物质积聚而成的理论，但如果不是遵循某种正确的排列和组合，形成的将不是预先期望的事物。大棒模式与上述理论一样：一大堆能量（这里指开发软件所需的人力和物力）放在一起，巨大的能量进行释放，通常的结果可能是产生了优秀的软件产品或成为一堆“废品”（不成功的软件）。
- 优点：思路简单， 通常可能是开发者的“突发奇想”
- 缺点：开发过程是非工程化的，随意性大
- 关于测试：有的较简单，有的则非常困难

8 软件开发模式（续）

2、边写边改法

- 采用边写边改法的软件开发通常只是有了比较粗略的想法就开始进行简单的设计、然后进行较长的反复编写、测试与修复这样一个循环的过程。在认为无法更精细的描述软件产品要求时，就发布产品。
- 优点：能够较为迅速的展现成果，适合需要快速制作而且用完就扔的小项目，如示范程序、演示程序等。
- 缺点：其编码和测试可能将是长期的循环往复的过程。

8 软件开发模式（续）

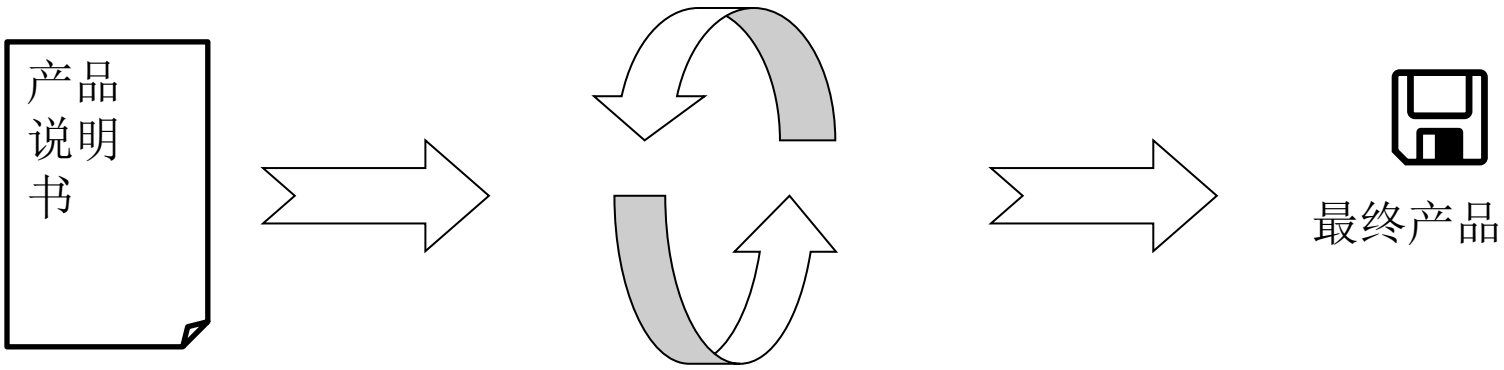


图1-6 边写边改开发模式

8 软件开发模式（续）

3、瀑布法

- 瀑布模式是将软件生命周期的各项活动，规定为按照固定顺序相连的若干个阶段性工作，形如瀑布流水，最终得到软件产品。
- 优点：易于理解；调研开发的阶段性；强调早期计划及需求调查；确定何时能够交付产品及何时进行评审与测试。
- 缺点：需求调查分析只进行一次，不能适应需求变化；顺序的开发流程，使得开发中的经验教训不能反馈到该项目的开发中去；不能反映出软件开发过程的反复与迭代性；没有包含任何类型的风险评估；开发中出现的问题直到开发后期才能够显露，因此失去及早纠正的机会。

8 软件开发模式（续）

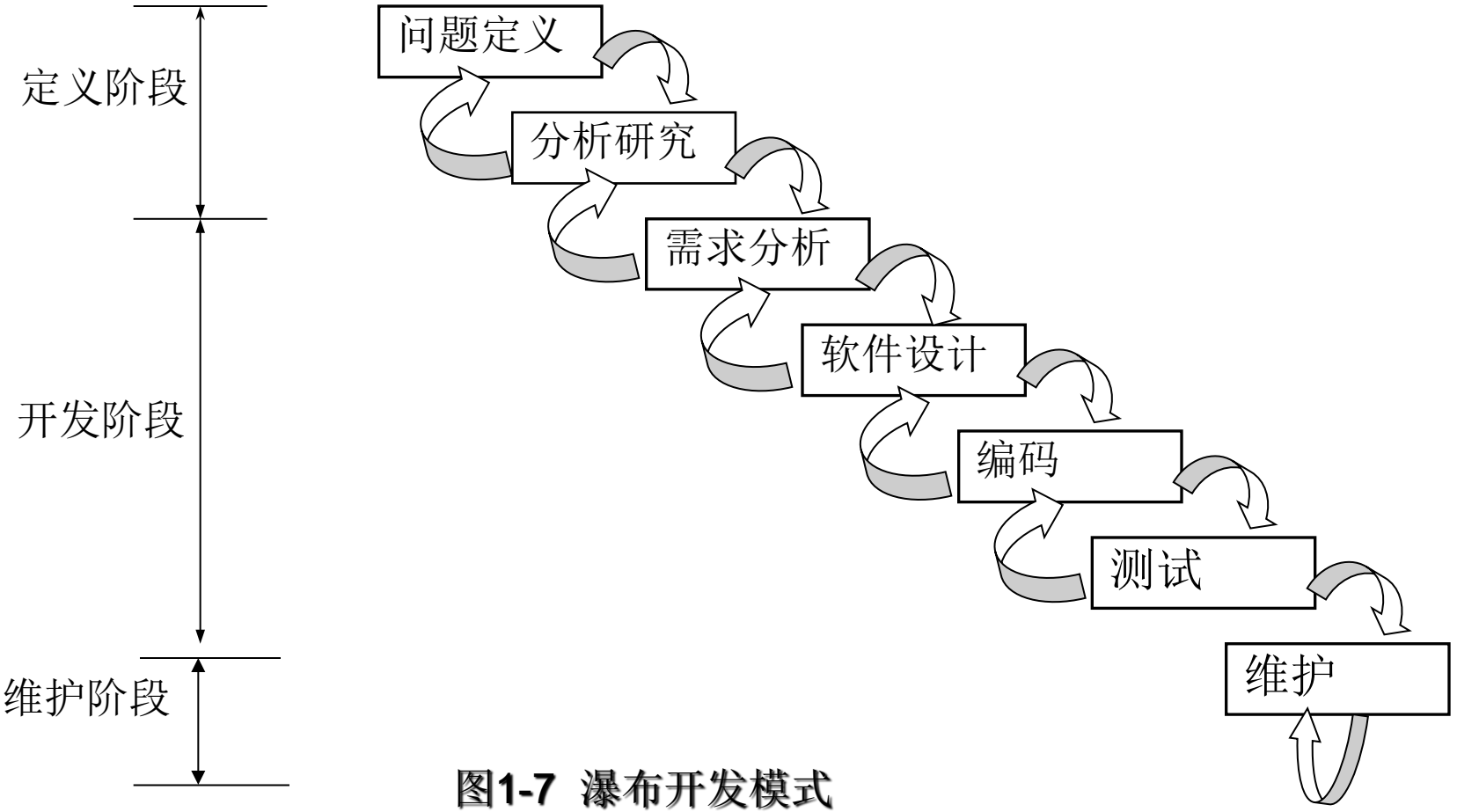


图1-7 瀑布开发模式

8 软件开发模式（续）

4、快速原型法

根据客户需求在较短的时间内解决用户最迫切解决的问题，完成可演示的产品。这个产品只实现最重要功能，在得到用户的更加明确的需求之后，原型将丢弃。

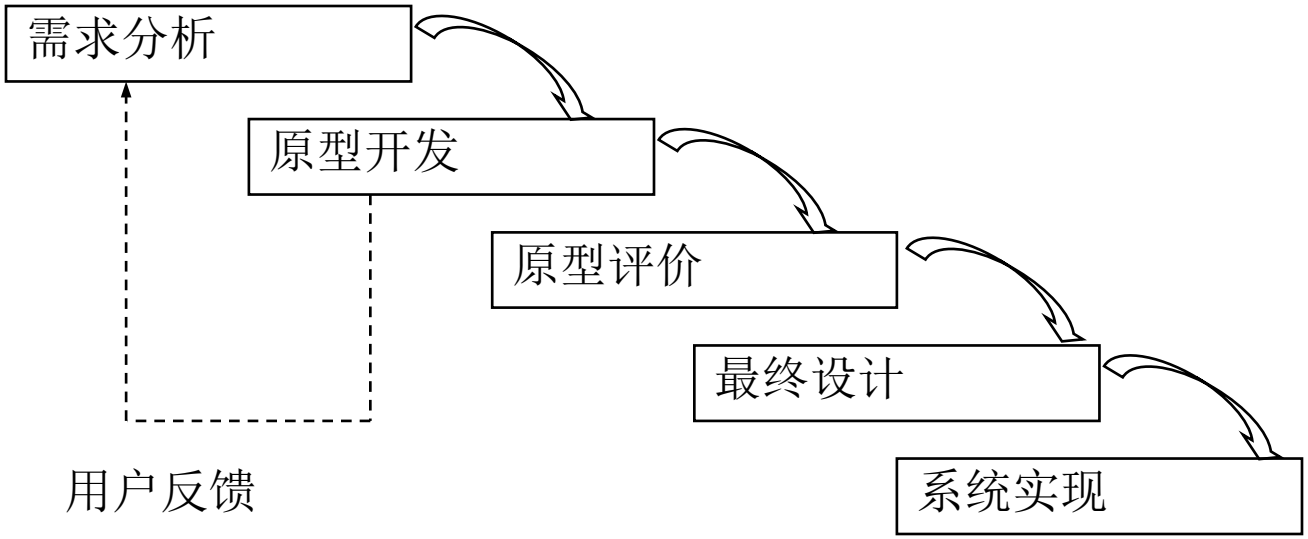


图1-8 快速原型开发模式

8 软件开发模式（续）

5、螺旋模式法

- 螺旋模式是瀑布模式与边写边改演化模式相结合，并加入风险评估所建立的软件开发模式。
- 主要思想是在开始时不必详细定义所有细节，而是从小开始，定义重要功能，尽量实现，接受客户反馈，进入下一阶段，并重复上述过程，直到获得最终产品。
- 每一螺旋（开发阶段）包括5个步骤：①确定目标，选择方案和限制条件。 ②对方案风险进行评估，并能解决风险。 ③进行本阶段的开发和测试。 ④计划下一阶段。 ⑤确定进入下阶段的方法。
- 优点：严格的全过程风险管理；强调各开发阶段的质量；提供机会评估项目是否有价值继续下去。

8 软件开发模式（续）

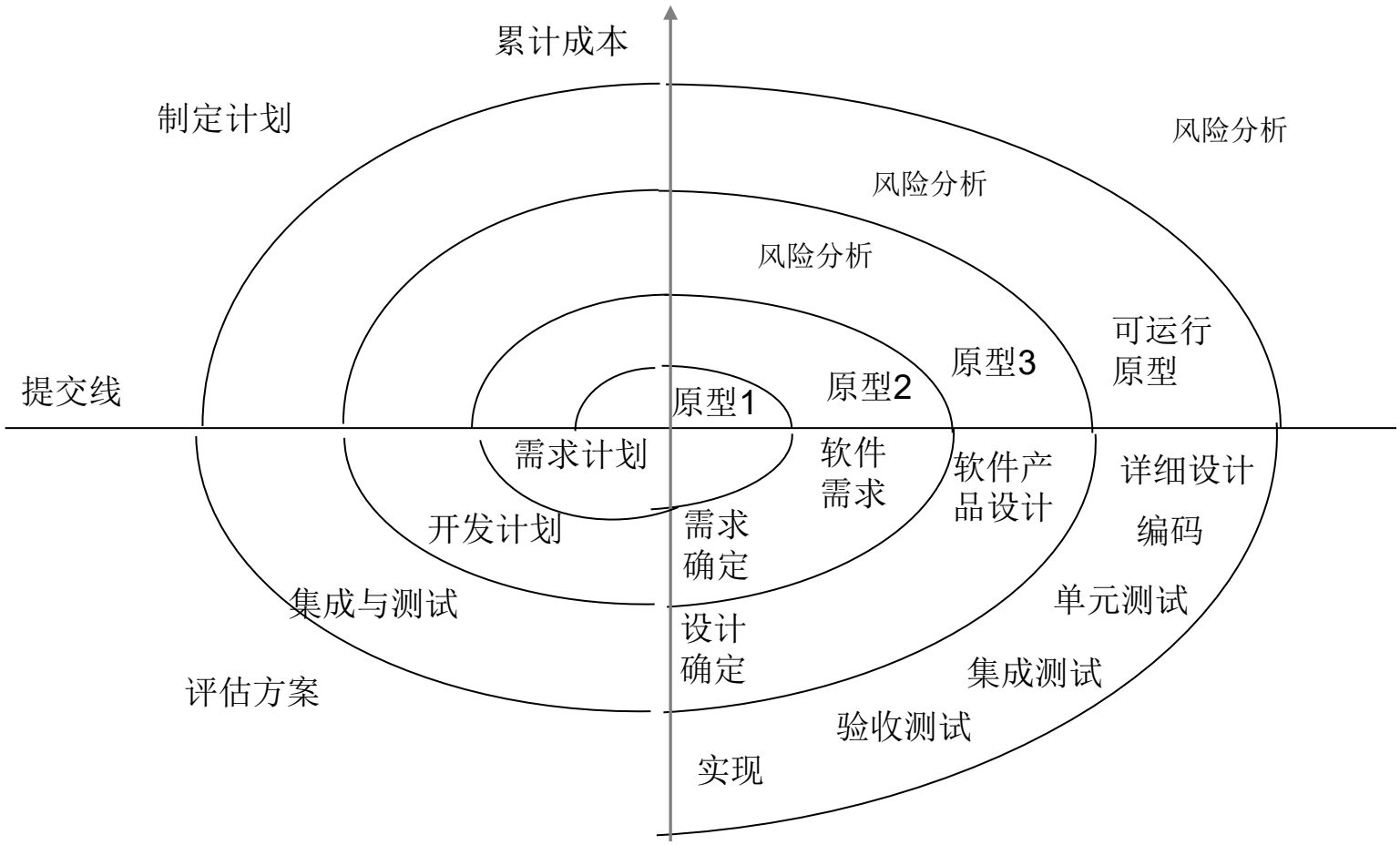


图1-9 螺旋开发模式

8-6 敏捷方法之极限编程

- 最简单的可能就是最有效的
- 极限编程适合
  - 小团队 (2-10 programmers)
  - “高风险”
  - 快速变化或不稳定的需求
  - 强调可测试性
- 格言
  - “沟通、简化、反馈、激励”

XP-eXtreme Programming

Kent Beck

8-6 XP基本思想和原则

个体和交互	胜过	过程和工具
可以工作的软件	胜过	面面俱到的文档
客户合作	胜过	合同谈判
响应变化	胜过	遵循计划

- 最优先要做的是通过尽早的、持续的交付有价值的软件来使客户满意。
- 敏捷过程提倡可持续的开发速度，责任人、开发者和用户应该能够保持一个长期稳定的开发速度。
- 即使到了开发的后期，需求改变还是受欢迎。
- 每隔一定时间，团队会进行反省，然后相应地对自己的行为进行调整。
- ……

9 测试模型

- V模型
- W模型
- X模型

9 软件开发与软件测试的关系

1、测试与开发各阶段的关系

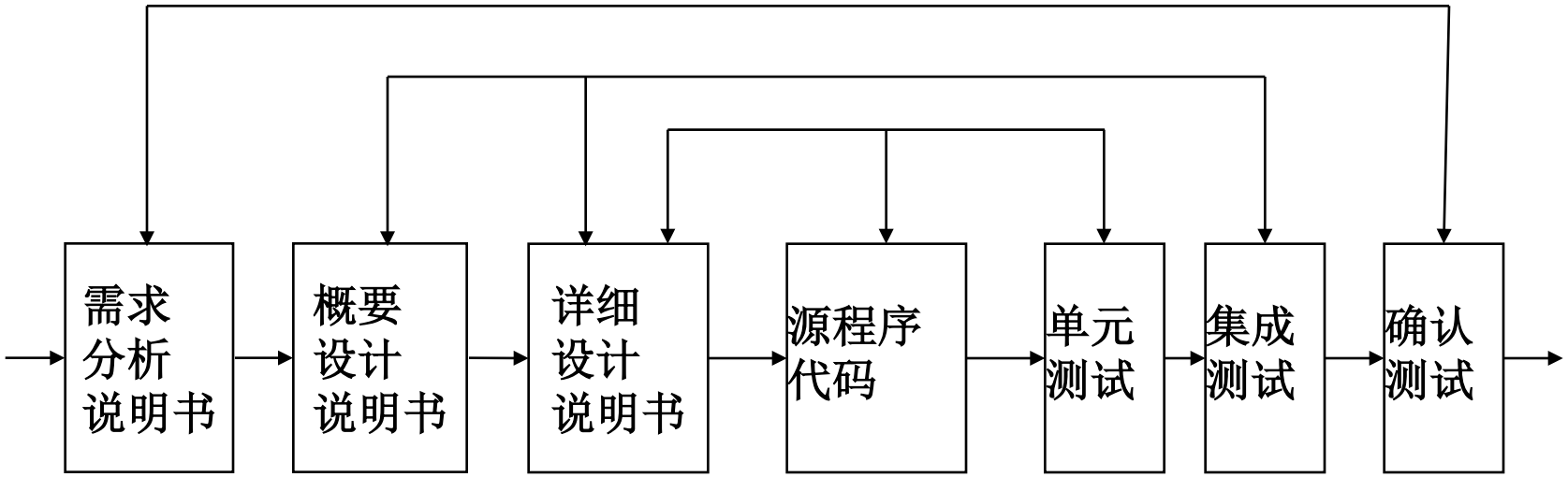
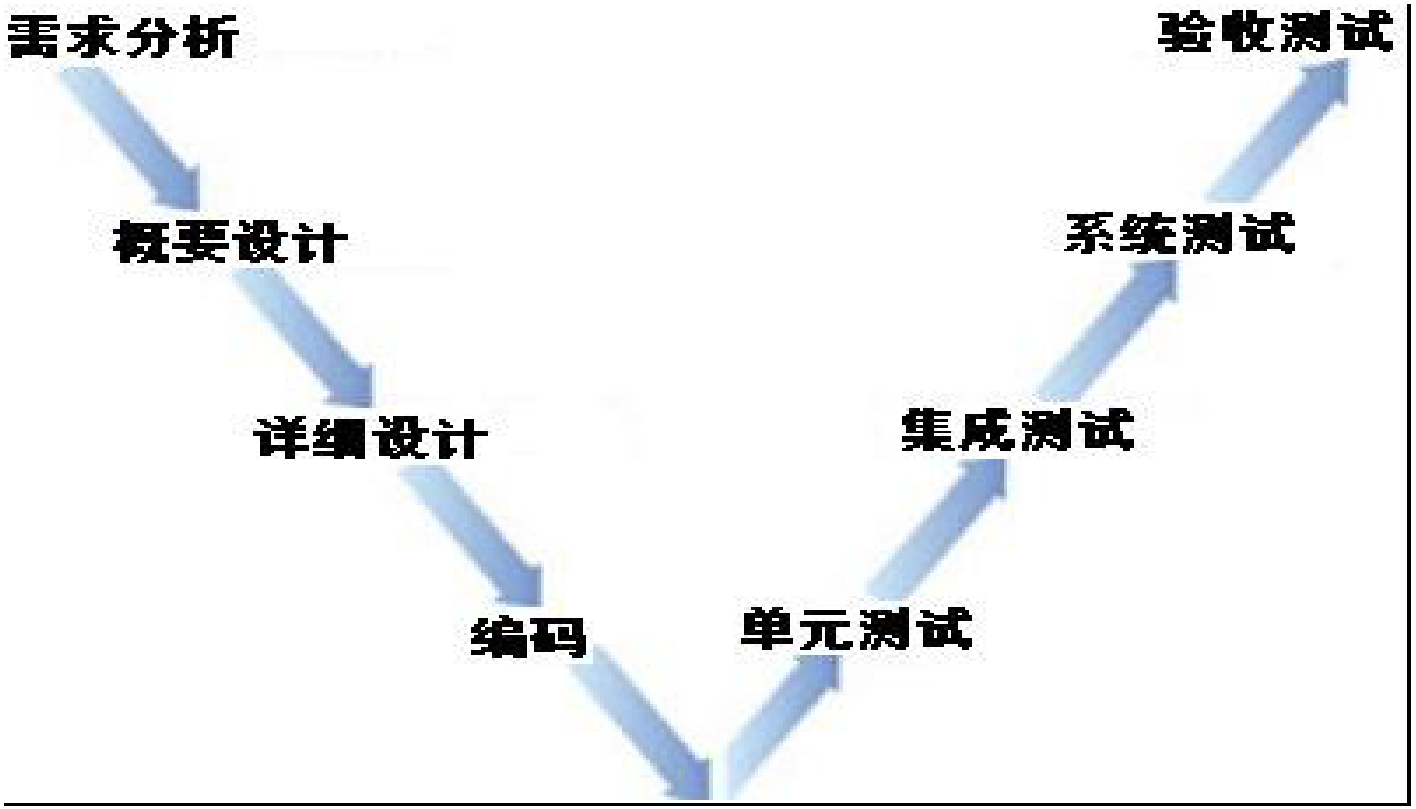
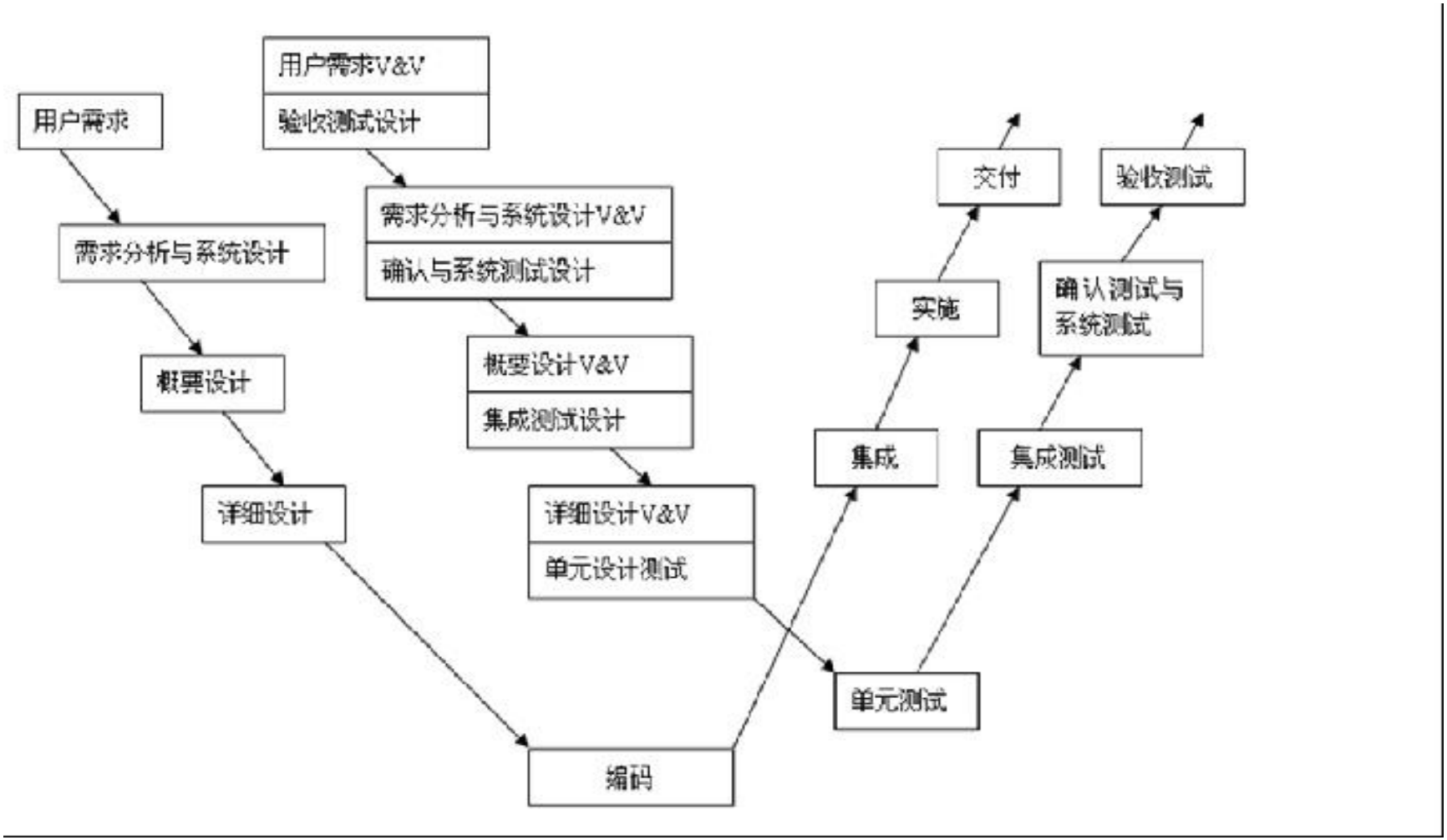


图1-10　软件测试与软件开发过程的关系

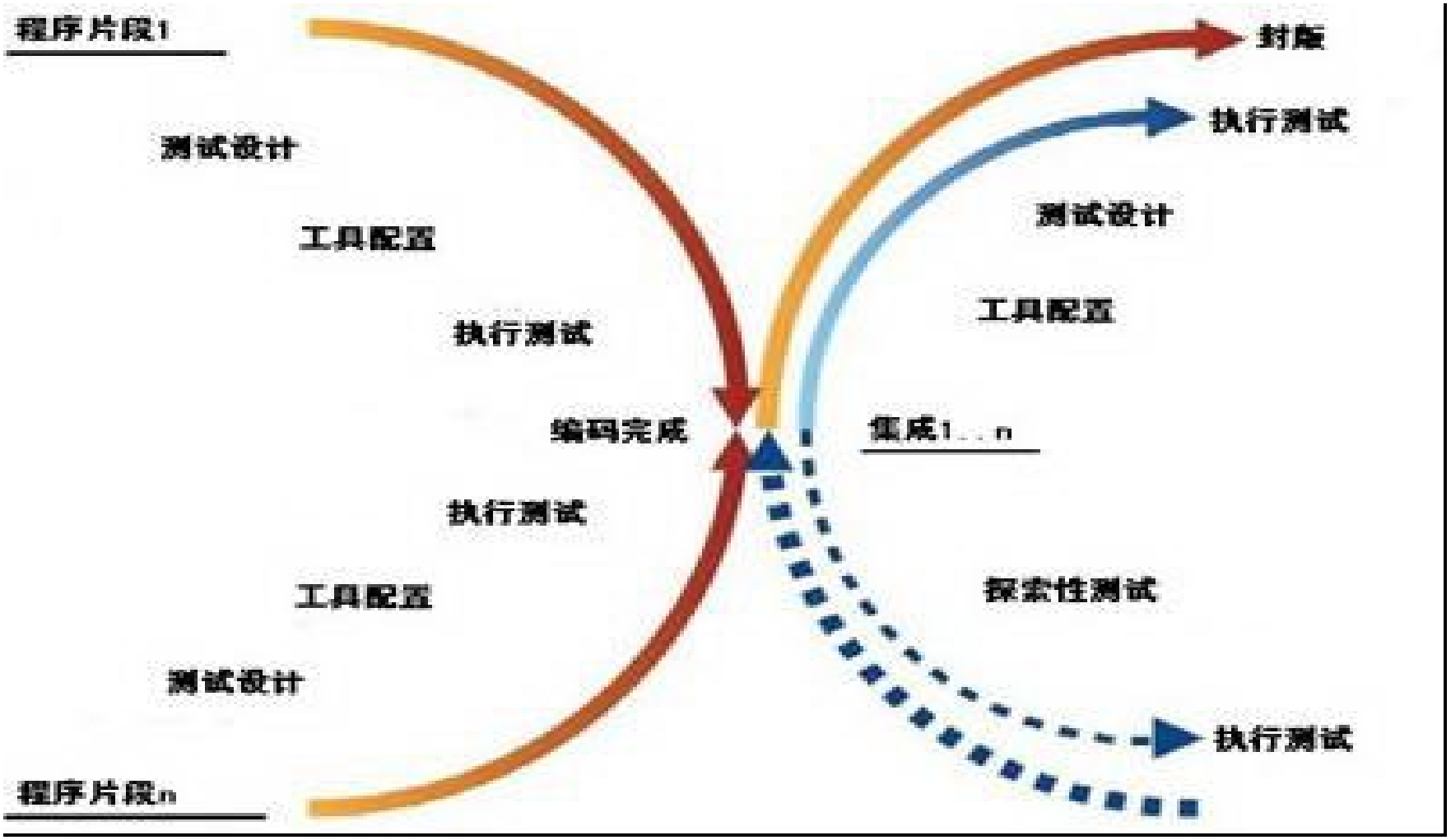
9 V模型



9 W模型


用户需求
用户需求V&V
验收测试设计
需求分析与系统设计
需求分析与系统设计V&V
确认与系统测试设计
概要设计
概要设计V&V
集成测试设计
详细设计
详细设计V&V
单元设计测试
编码
单元测试
集成
集成测试
实施
确认测试与系统测试
交付
验收测试

9 X模型


程序片段1
测试设计
工具配置
执行测试
编码完成
执行测试
工具配置
测试设计
程序片段n
封版
执行测试
测试设计
工具配置
集成1 . . n
探索性测试
执行测试

9 软件开发与软件测试的关系（续）

测试在开发阶段的作用：

- 项目规划阶段：负责从单元测试到系统测试的整个测试阶段的监控。
- 需求分析阶段：确定测试需求分析、系统测试计划的制定，评审后成为管理项目。
- 详细设计和概要设计阶段：确保集成测试计划和单元测试计划完成。
- 编码阶段：由开发人员进行自己负责部分的测试代码。在项目较大时，由专人进行编码阶段的测试任务。
- 测试阶段（单元、集成、系统测试）：依据测试代码进行测试，并提交相应的测试状态报告和测试结束报告。

9 软件开发与软件测试的关系（续）

2、测试与开发的并行性

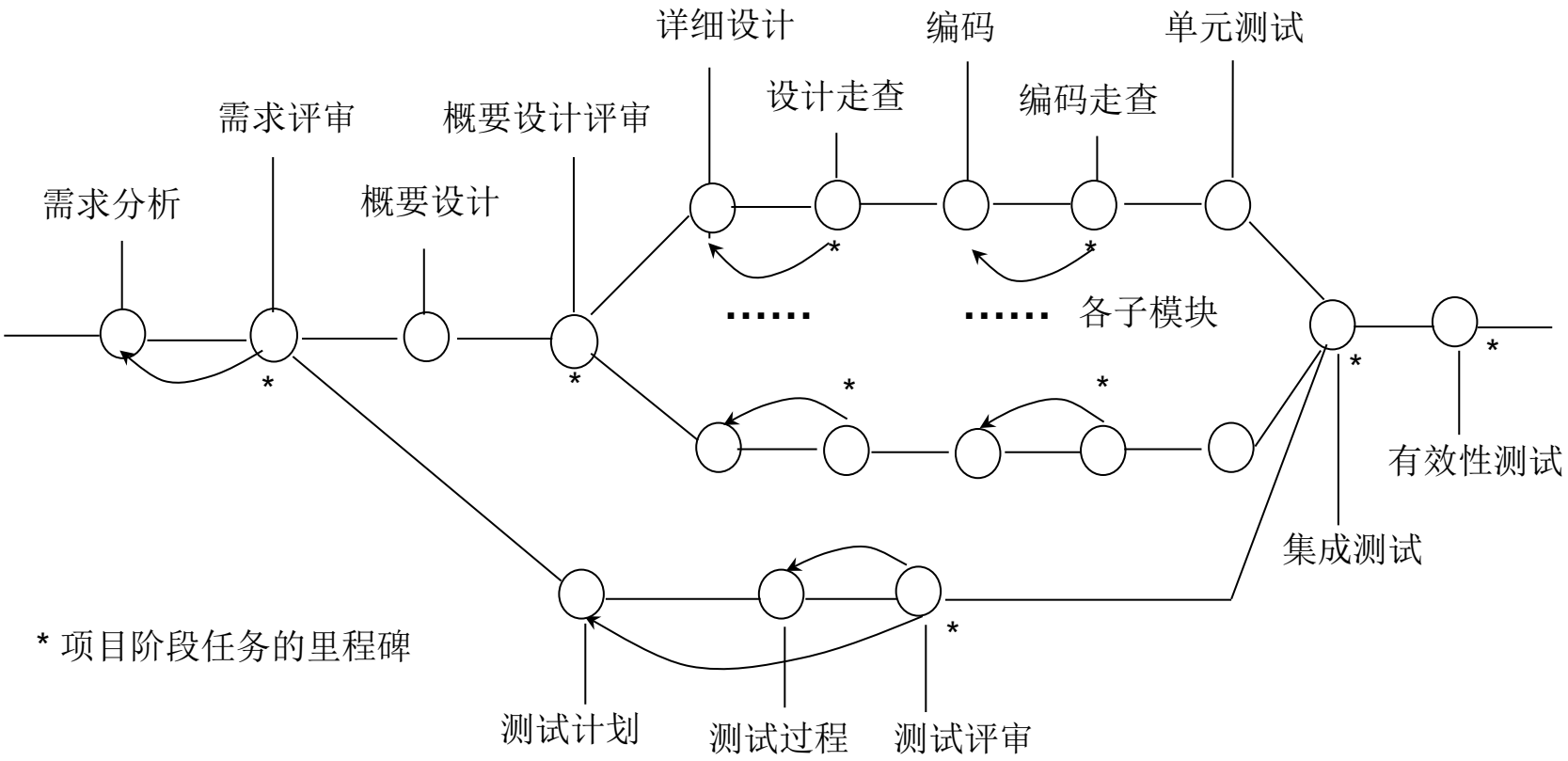
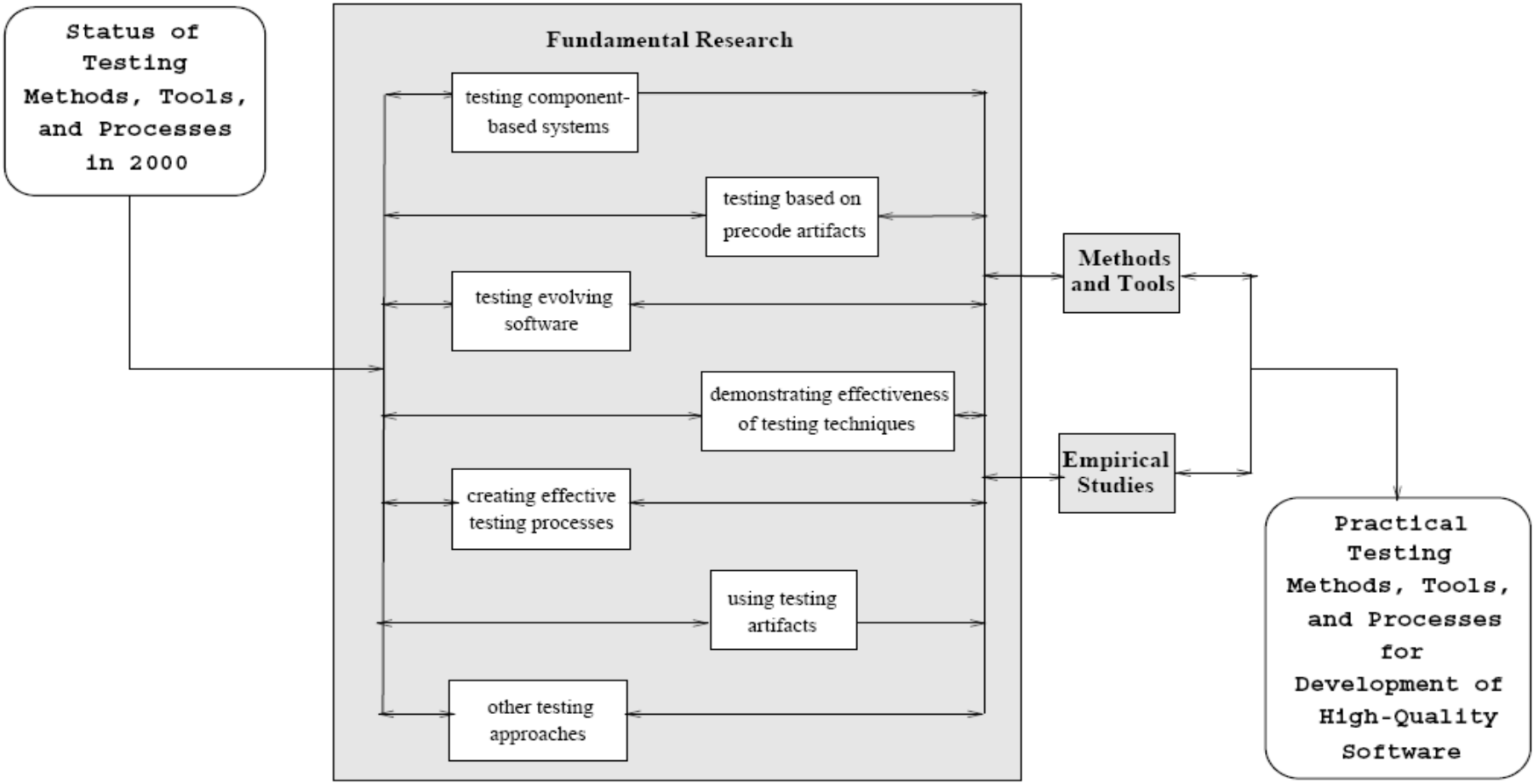


图1-11 软件测试与软件开发的并行性

10 软件测试的研究

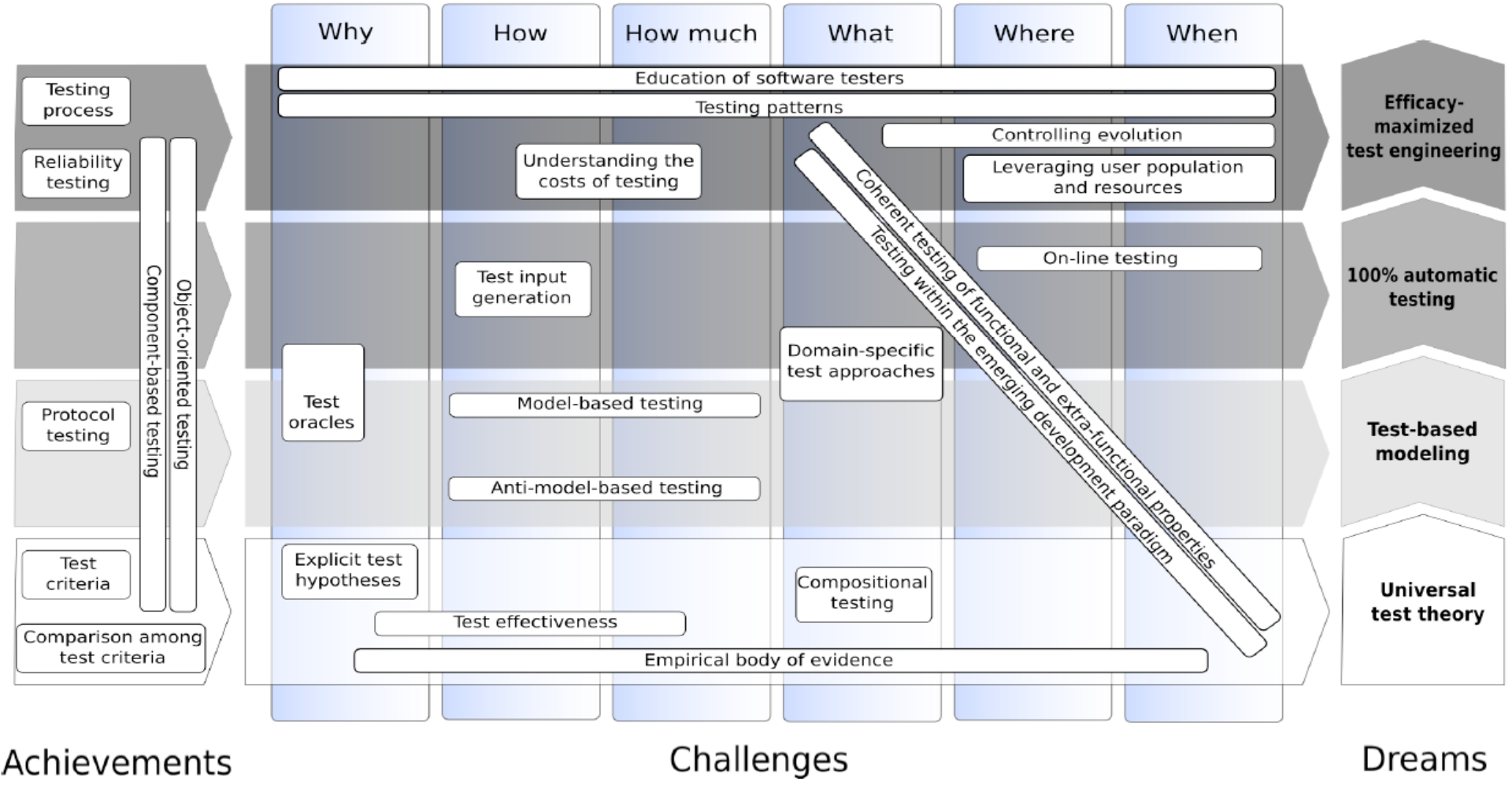
- 2000 FSE
- 2007 FSE
- 2014 Changhai Nie

2000 FSE, Mary Jean Harrold

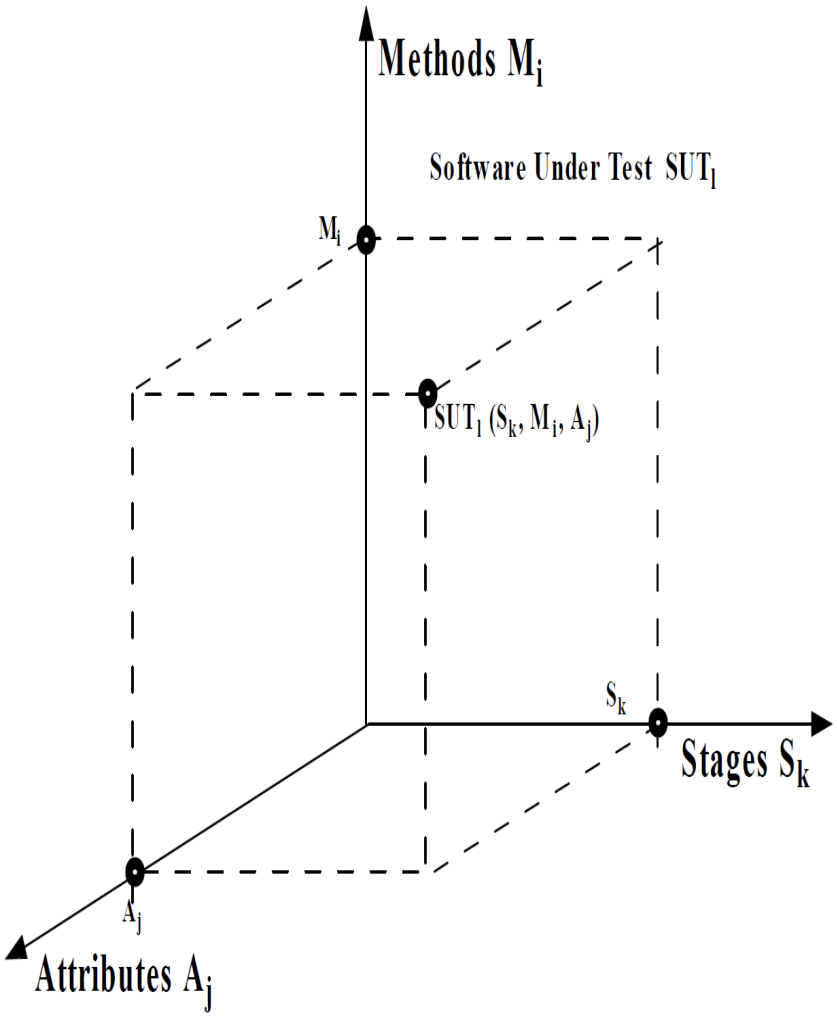


2007 FSE, Antonia Bertolino

Software testing research roadmap

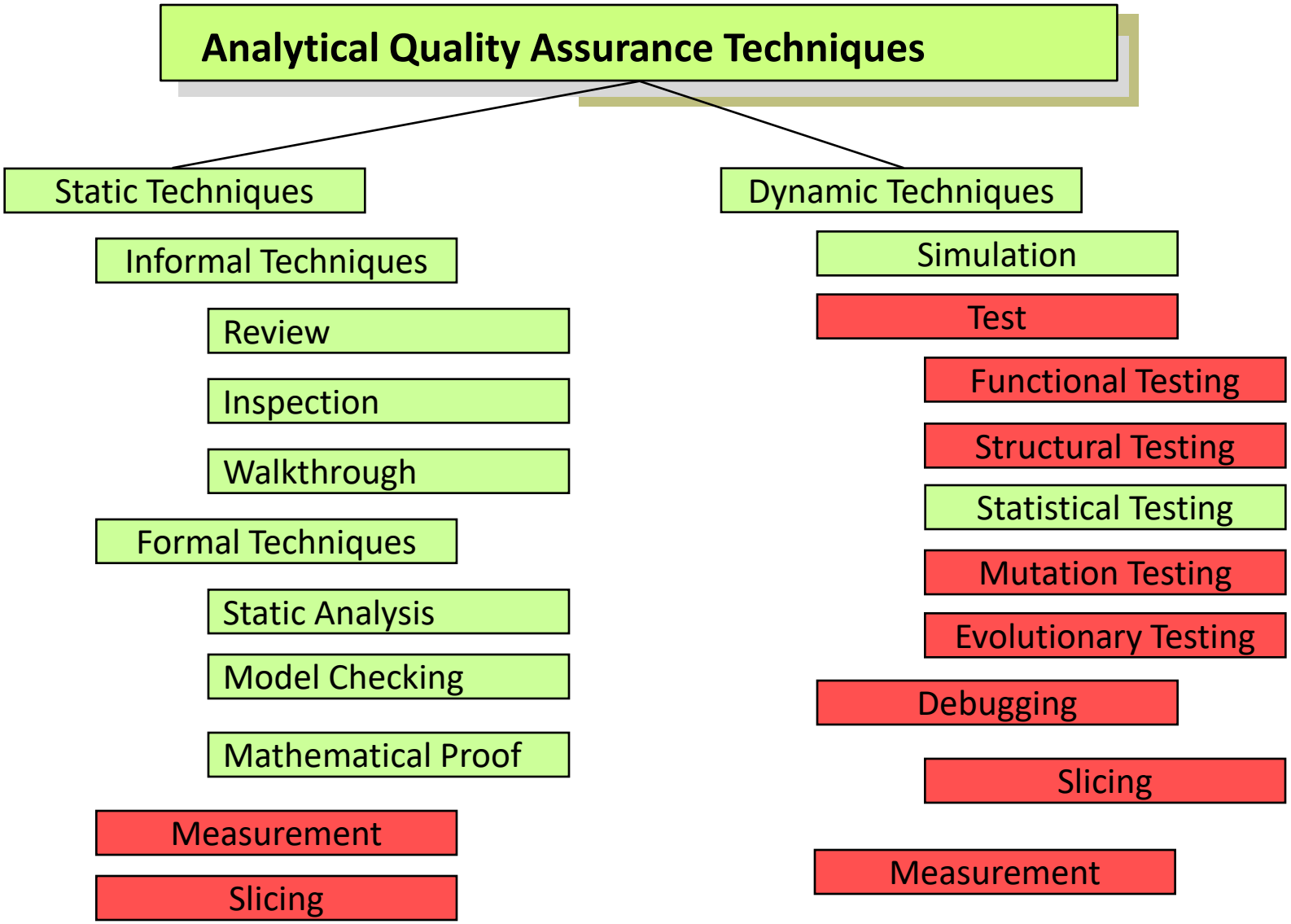


Changhai Nie, Hareton Leung. **A Framework of Understanding Software Testing**. 2014



Various Special Testing Methods M_i, $1 \le i \le 40$							
No.	Testing Method	No.	Testing Method	No.	Testing Method	No.	Testing Method
M_1	*White Box* [26]	M_2	*Black Box* [26]	M_3	*Static* [26]	M_4	*Dynamic* [27]
M_5	*Model Based** [16]	M_6	*Search Based** [22]	M_7	*Fault Based* [21]	M_8	*Property* [12]
M_9	*Metamorphic* [37]	M_{10}	*Operation Profile* [7]	M_{11}	*Statistics* [31]	M_{12}	*Combinatorial** [28]
M_{13}	*Mutation** [17]	M_{14}	*Specification* [34]	M_{15}	*Adaptive* [8]	M_{16}	*Random* [11]
M_{17}	*Anti-random* [39]	M_{18}	*Adaptive Random* [9]	M_{19}	*Concolic* [18]	M_{20}	*Anti-model* [6]
M_{21}	*Fuzzing* [35]	M_{22}	*Compositional* [5]	M_{23}	*GUI* [25]	M_{24}	*Pair* [38]
M_{25}	*Online* [36]	M_{26}	*Ad Hoc* [7]	M_{27}	*Exploratory* [2]	M_{28}	*Extreme* [26]
M_{29}	*Syntax* [30]	M_{30}	*FSM* [19]	M_{31}	*TTCN* [14]	M_{32}	*UML based* [23]
M_{33}	*Model Checking* [21]	M_{34}	*Playtest* [10]	M_{35}	*Petri-net* [41]	M_{36}	*Back-to-Back* [7]
M_{37}	*Error Guessing* [26]	M_{38}	*Boolean Spec.* [3]	M_{39}	*State Trans.* [33]	M_{40}	*Graph coverage* [30]
Testing Various Attributes and Aspects A_j, $1 \le j \le 36$ [21, 27]							
A_1	*Performance*	A_2	*Stress*	A_3	*Load*	A_4	*Volume*
A_5	*Reliability*	A_6	*Security**	A_7	*Recovery*	A_8	*Internationalization*
A_9	*Compatibility*	A_{10}	*Installation*	A_{11}	*Protocol*	A_{12}	*Fault Tolerance*
A_{13}	*Authorization*	A_{14}	*Uninstallation*	A_{15}	*Backup*	A_{16}	*Online Help*
A_{17}	*Robustness*	A_{18}	*Usability*	A_{19}	*Facility*	A_{20}	*Data Conversion*
A_{21}	*Configuration*	A_{22}	*Accessibility*	A_{23}	*Interface*	A_{24}	*Memory Leak*
A_{25}	*Conformance*	A_{26}	*Stability*	A_{27}	*Predicate*	A_{28}	*By-pass, Penetration*
A_{29}	*Remainder*	A_{30}	*Document*	A_{31}	*Workflow*	A_{32}	*Data Base*
A_{33}	*Input*	A_{34}	*Logic*	A_{35}	*Domain*	A_{36}	*Usage*
Testing Various Development Stages S_k, $1 \le k \le 8$ [27]							
S_1	*Unit*	S_2	*Smoke(daily build)*	S_3	*Integration*	S_4	*System*
S_5	*Acceptance*	S_6	*Regression**	S_7	*Alpha*	S_8	*Beta*
Testing Various Softwares SUT_l, $1 \le l \le 16$ [21, 7, 27]							
SUT_1	*Pervasive Computing*	SUT_2	*Aspect-oriented**	SUT_3	*Component-based*	SUT_4	*Concurrent**
SUT_5	*High Confidence*	SUT_6	*Object-oriented*	SUT_7	*Protocol-based*	SUT_8	*Real-time*
SUT_9	*Internet of Things**	SUT_{10}	*Service-oriented*	SUT_{11}	*Web-based*	SUT_{12}	*Internetware** [24]
SUT_{13}	*Cloud Computing** [13]	SUT_{14}	*Safty-critical*	SUT_{15}	*Open-source*	SUT_{16}	*Embedded* [32]

Introduction and Motivation



11-1 讨论1

- 具备哪些要素可以构成一种**职业**
- 独特的技能
- 技术含量
- 门槛

测试员应具备的素质

(1) 探索精神

软件测试员不会害怕进入陌生环境。他们喜欢拿到新软件，安装在自己的机器上，观看结果。

(2) 故障排除能手

软件测试员善于发现问题的结症，他们喜欢猜谜。

(3) 不懈努力

软件测试员总是不停尝试。他们可能会碰到转瞬即逝或难于重建的软件缺陷。他们不会心存侥幸，而是尽一切可能去寻找。

(4) 创造性

测试显而易见的事实，那不是软件测试员。他们的工作是想出富有创意甚至超常的手段来寻找缺陷。

测试员应具备的素质

(5) 追求完美

他们力求完美，但是知道某些无法企及时，不去苛求，而是尽力接近目标。

(6) 判断准确

软件测试员要决定测试内容、测试时间，以及看到的问题是否作真正的缺陷。

(7) 老练稳重

软件测试员不害怕坏消息。他们必须告诉程序员，你的孩子（程序）很丑。优秀的软件测试员知道怎样老练地处理这些问题，和不够冷静的程序员怎样合作。

(8) 说服力

软件测试员找出的软件缺陷有时被认为不重要。不用修复。测试员要善于表达观点，表明软件缺陷必须修复，并通过实际演示力陈观点。

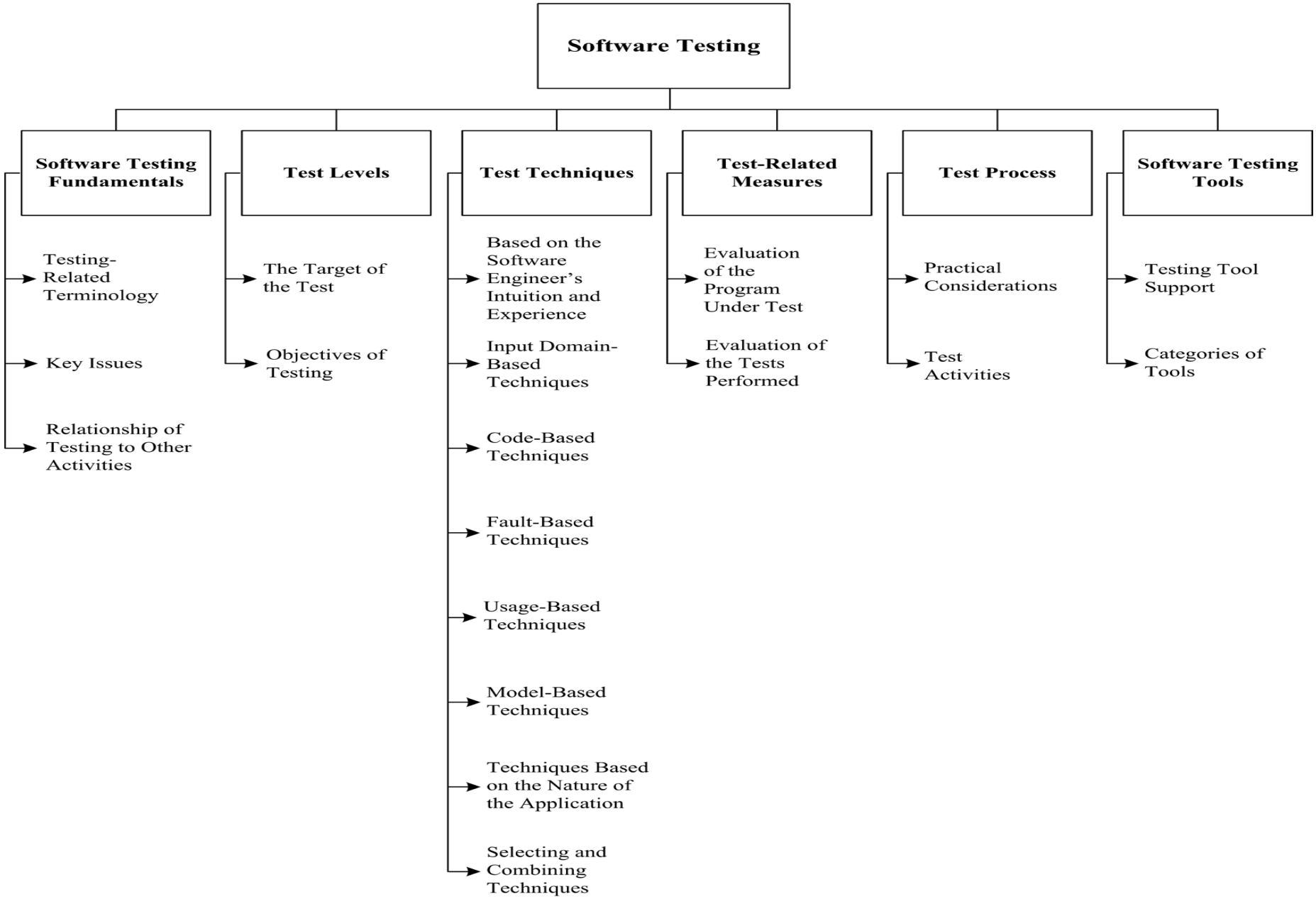
软件测试知识体

- 软件测试重要性和概念
- 软件测试的目的和原则
- 软件测试用例设计
- 软件测试策略
- 软件测试种类
- 软件测试标准
- 软件测试的工具
- 软件测试的管理

11-2 讨论2

- 具备哪些因素可以形成一门科学
- 科学问题有哪些
- 测试计划
- 测试策略
- 测试生成
- 测试执行与观察
- 故障诊断
- 回归测试

2014 Guide to SEKA



12 软件测试的工具

- 测试管理工具：用于管理测试的整个工作过程以及过程中产生的各种相关文档、数据、记录和报告等。常用的管理工具有TestDirector和TestManager等
- 自动化功能测试工具：用于自动化执行功能测试脚本，一般采用基于录制回放的机制，通过录制操作过程，产生基本的测试脚本，然后在脚本编辑器中进一步完善测试脚本，在通过回放脚本的方式执行测试脚本的步骤，从而实现功能测试的自动化。常用的测试自动化工具有QTP(Quick Test Professional), Rational Robot, TestCoplete等。
- 性能测试工具：用于性能测试过程中的通信协议模拟、并发用户模拟及性能参数监控等方面的测试工具，如LoadRunner，SilkPerformer等。
- 单元测试工具：用于单元测试的测试框架，这些测试工具提供单元测试的一些接口，管理单元测试的执行，例如XUnit系列、MSTest等。
- 白盒测试工具：用于测试程序内部逻辑和错误的测试工具，又可以细分为代码标准检查工具、代码效率检查工具和内存泄露检查工具等，如TEST、AQTime和BundsChecker等。
- 测试用例设计工具：用于辅助测试用例的设计或测试数据生成的工具，一般常用的有CTE XL、AETG和PICT等。
- 测试工具还可以根据收费方式分为商业测试工具、开源测试工具和免费测试工具。

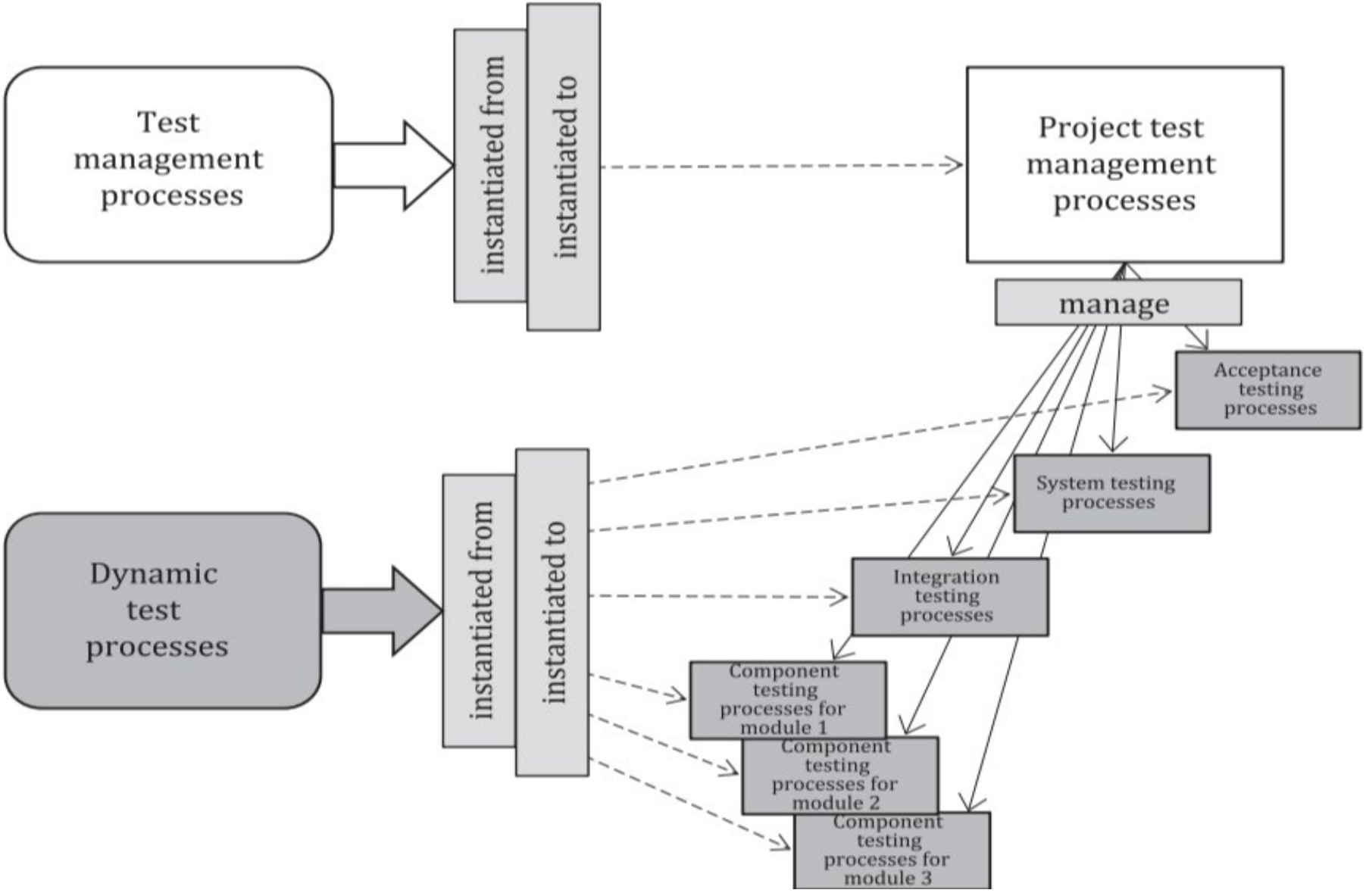
Tool support

- Tool support (usually referred to as test tools) is available for many of the tasks and activities described in the test management and dynamic testing processes defined in ISO/IEC/IEEE 29119-2, as well as aspects of the testing techniques described in ISO/IEC/IEEE 29119-4. The following list provides examples of some of the areas covered by test tools:
  - — test case management;
  - — test monitoring and control;
  - — test data generation;
  - — static analysis;
  - — test case generation;
  - — test case execution;
  - — test environment implementation and maintenance.
- There are a wide variety of test automation tools available. They can be developed in-house, obtained commercially, or obtained from the open-source community

13 软件测试的管理

- 时间维：全过程管理，对软件测试项目的全过程进行控制，具体包括：测试计划管理，测试设计管理，测试执行管理，测试结果管理等。
- 空间维：全方位管理，对与软件质量有关的关键因素实施全方位管理，具体包括：缺陷管理、文档管理、配置管理、评审管理、质量管理和回归管理等。
- 组织维：人员管理，构建从测试人员、测试小组到测试机构的多层次的组织管理模式。

— Hierarchy of instantiated test management processes



14-1 软件测试概念发展简史

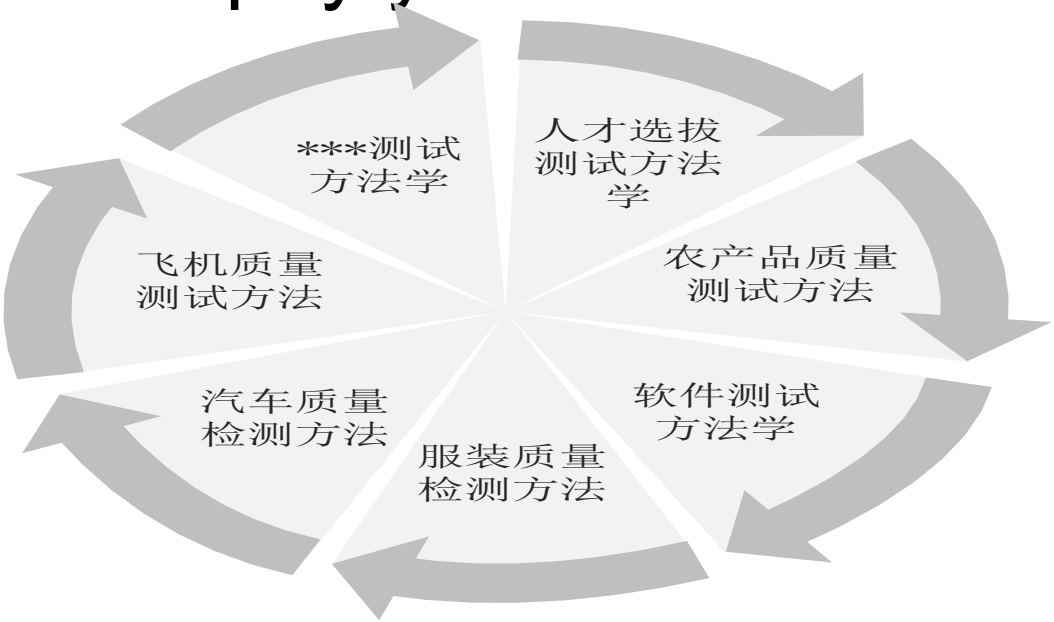
- Debugging oriented (1950年左右)
- Demonstration oriented (1960年左右)
- Destruction oriented (1970年左右)
- Evaluation oriented (1980年左右)
- Prevention oriented (1990年以后)
- Professional、education and research （2000年以后）

14-2软件测试的发展趋势

- 软件测试自动化
- 测试技术智能化
- 测试方法工具化
- 测试工具服务化
- 测试理论系统化
- 测试手段多样化
- 测试过程标准化
- 测试人员专业化
- 测试部门独立化
- 测试管理全面化
- 测试对象精细化

14-3 Ultimate goal of testing workers **Elaine Jessica Weyuker**

- We should be able to make a living by it
- As you can find bugs while they can not, they should pay you.



20 Standards on Testing

- **IEEE Test Standard:**
  https://standards.ieee.org/search/?q=Software%20Systems&type=Standard
- ISO/IEC/IEEE International Standard - Software and systems engineering - Software testing -- Part 2: Test processes
- IEEE/ISO/IEC 29119-1-2021 ISO/IEC/IEEE International Standard - Software and systems engineering --Software testing --Part 1:General concepts
- ◦ ◦ ◦ Part 3: Test documentation
- ◦ ◦ ◦ Part 4: Test technique

Standards on Testing

- ISO 9001 on Testing, http://www.iso.org/
- a) Software testing includes test plan preparation and review, test data preparation and review, and review of test results.
- b) Corrective actions to fix causes of defects.
- c) Reassessment of tools, techniques and methodologies used in software production.
- d) Programming standards (including testing standards) that describe approved practice and list any prohibited practice.
- e) Evaluation of customer-supplied software products and purchased products.
- f) Check that test processes are adhered to.
- g) Test processes improved where required.

Standards on Testing

- SW-CMM on Testing, http://www.sei.cmu.edu/
- See ‘Software Product Engineering’ KPA, and ‘Training Program’, ‘Technology Change Management’, ‘Process Change Management’ KPA.
- a) Require 4 testing levels: unit, integration, system and acceptance testing.
- b) Regression testing should be done to ensure changes are correct.
- c) The test plan should be reviewed.
- d) Proper training of testers should be conducted for better job performance.
- e) Software process standards must be maintained.
- f) A system test group should be responsible for performing an independent system test.
- g) Test processes should be continually improved.

Standards on Testing

Test Standards		IEEE	CMM	ISO9001
Category	*Sub-category*			
Test phases	unit test	R	R	
	integration test	R	R	
	system test	R	R	
	acceptance test	R	R	
Test activities	test planning	R	R	R
	test scheduling	R	R	
	test case dev.	R	R	R
	test execution	R	R	R
	test result reporting	R	R	R
	regression test	R	R	
Test Management	test plan review	R	R	R
	test staff training	R	R	R
	test documentation	R	R	

Definitions1

- Test Phases: A ‘test phase’ is a level of testing in the software life cycle where certain components of the software system are tested to check for compliance with requirements.
- Test Activities: Test planning involves specifying the general approach, objectives, scope, resources and schedule of testing, as well as identifying functions and features to be tested, test tasks, and personnel for each task.
- The test result reporting activity records the defects detected during the test execution and other data for analysis. In addition, it reports on the test comprehensiveness and produces a summary of the testing activities.

Definitions 2

- Test Management: Test management consists of activities such as
  - the test plan review,
  - training of test staff,
  - establishing a test standard,
  - ensuring the test documentation follows a standard format and
  - organizing an independent test group.
- Test standards and policies are essential for improving testing and to ensure consistency in testing. Guidelines on test techniques and strategies are defined in the test standard.

Mature (or “Good”) Test Process has

- A set of defined testing polices
- A test planning process
- A test lifecycle
- An independent test group
- A set of test-related metrics
- Appropriate tools and equipment
- Controlling and tracking
- Product quality control

21 软件测试的PIE模型

- PIE模型对于软件测试日常笼统说的“Bug”一词做了细分：
  - 1、Fault：软件中存在的静态错误。
  - 2、Error：由于Fault导致的内部状态的错误。
  - 3、Failure：不满于规格说明，用户可见的外部错误。

PIE模型

- 我们要观察到错误的存在，必须经过三个步骤，也就是“**PIE**”名字的由来：
  - 1、**Execution**（执行）：错误代码必须要被执行到。
  - 2、**Infection**（感染）：触发了错误的中间状态。
  - 3、**Propagation**（传播）：错误的中间状态必须可以传播到最后的输出使得可以被观测到。
- 错误代码被执行到，未必会触发错误的中间状态；错误的中间状态，未必会导致错误的输出。

```
public static void CSta (int [ ] numbers)
{
    int length = numbers.length;
    double mean, sum;

    sum = 0.0;
    for (int i = 1; i < length; i++)//i=0
    {
        sum += numbers [ i ];
    }
    mean = sum / (double) length;

    System.out.println ("mean:   " + mean);
}
```

- 比如上面一个计算数组中数字之和的程序，其中for循环从i=1开始，遗漏掉了i=0的情况，这显然是一个Fault。
- 如果我们测试的数组是[0，5，3]，由于遗漏掉的i=0的情况刚好为0，所以sum计算的结果为8，与预期相符。此时，Fault并未触发错误的中间状态。
- 但如果换成[4，5，3]，就可以触发错误的中间状态。

```
public static void CSta (int [ ] numbers)
{
    int length = numbers.length-1;
    double mean, sum;

    sum = 0.0;
    for (int i = 0; i < length; i++)
    {
        sum += numbers [ i ];
    }
    mean = sum / (double) length;

    System.out.println ("mean:   " + mean);
}
```

- 再看另一个例子，计算数组的平均值时，错误地将数组长度减一，这也是一个Fault。
- 当测试的数组是[3，5，4]时，sum只计算了前两个数的总和，造成了一个错误的中间状态。
- 但很巧合的是，(3+5) / 2的结果 跟 (3+5+4) / 3的结果是一样的，也就是说，虽然存在一个错误的中间状态，但并没有导致一个错误的输出结果。
- 因此，我们可以得出一个结论，如果要发现一个Bug，必须要满足PIE模型。PIE模型对于我们如何提升软件质量有一定的指导借鉴意义。

思考题

- 1 什么是软件质量
- 2 什么是软件缺陷，它具有哪些危害，如何管理？
- 3 什么是软件测试
- 4 测试的普遍观
- 5 软件测试的种类
- 6 软件测试的关键问题是什么？
- 7 软件测试的对象是什么？
- 8 软件测试的停止准则有哪些？
- 9 软件测试的周期性和并行性
- 10 软件测试的目的、意义和原则
- 11 软件测试的未来发展趋势
- 12 软件测试在软件工程中的位置（在若干开发模型中）
- 13 软件测试的模型
- 14 软件测试的过程（测试信息流程）
- 15 软件测试的研究
- 16 软件测试作为一种职业和作为一门科学
- 17 软件测试的工具
- 18 软件测试的管理
- 19 软件测试的历史
- 20 软件测试的标准
- 21 解释软件的故障模型PIE。