软件测试入门

内容概要

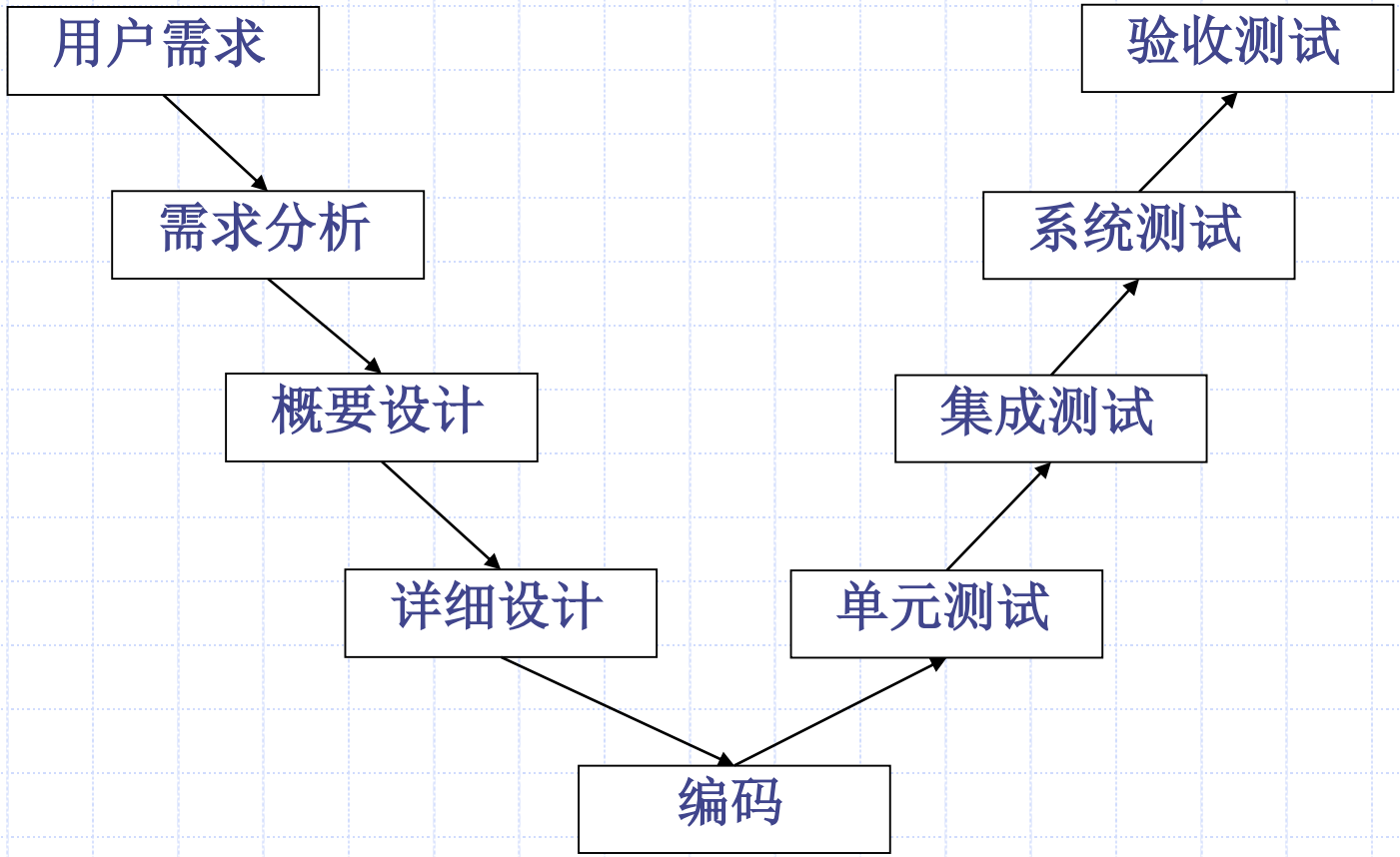
1. 软件测试的模型
2. 软件测试的原则
3. 软件测试的术语
4. 软件测试基本过程
5. 完整的软件测试系统

内容概要

1. 软件测试的模型
2. 软件测试的原则
3. 软件测试的术语
4. 软件测试基本过程
5. 完整的软件测试系统

1.1 软件测试的V模型

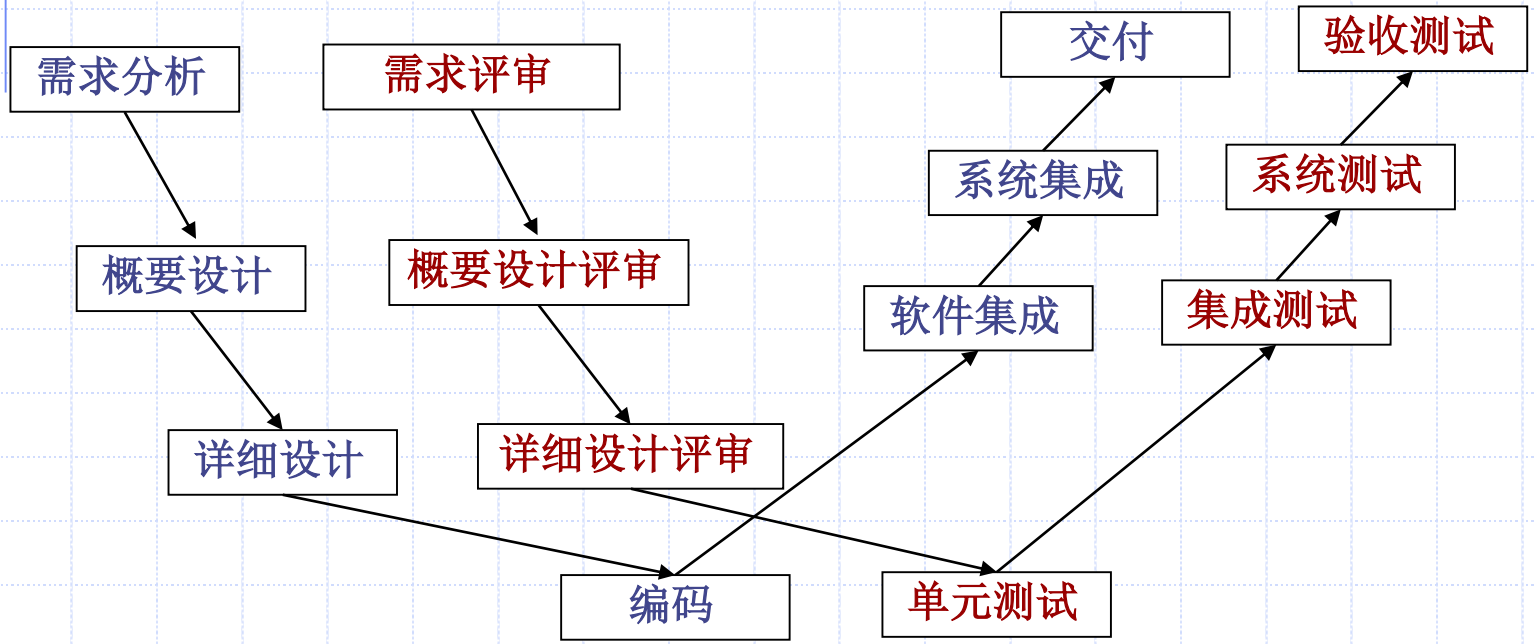
- V模型是将软件开发过程和软件测试过程对应起来，最早由Paul.Rook提出。



局限性： 把测试作为编码之后的一个活动，需求分析等前期产生的错误直到后期的验收测试才能发现。无法体现“尽早地和不断地进行软件测试”的原则

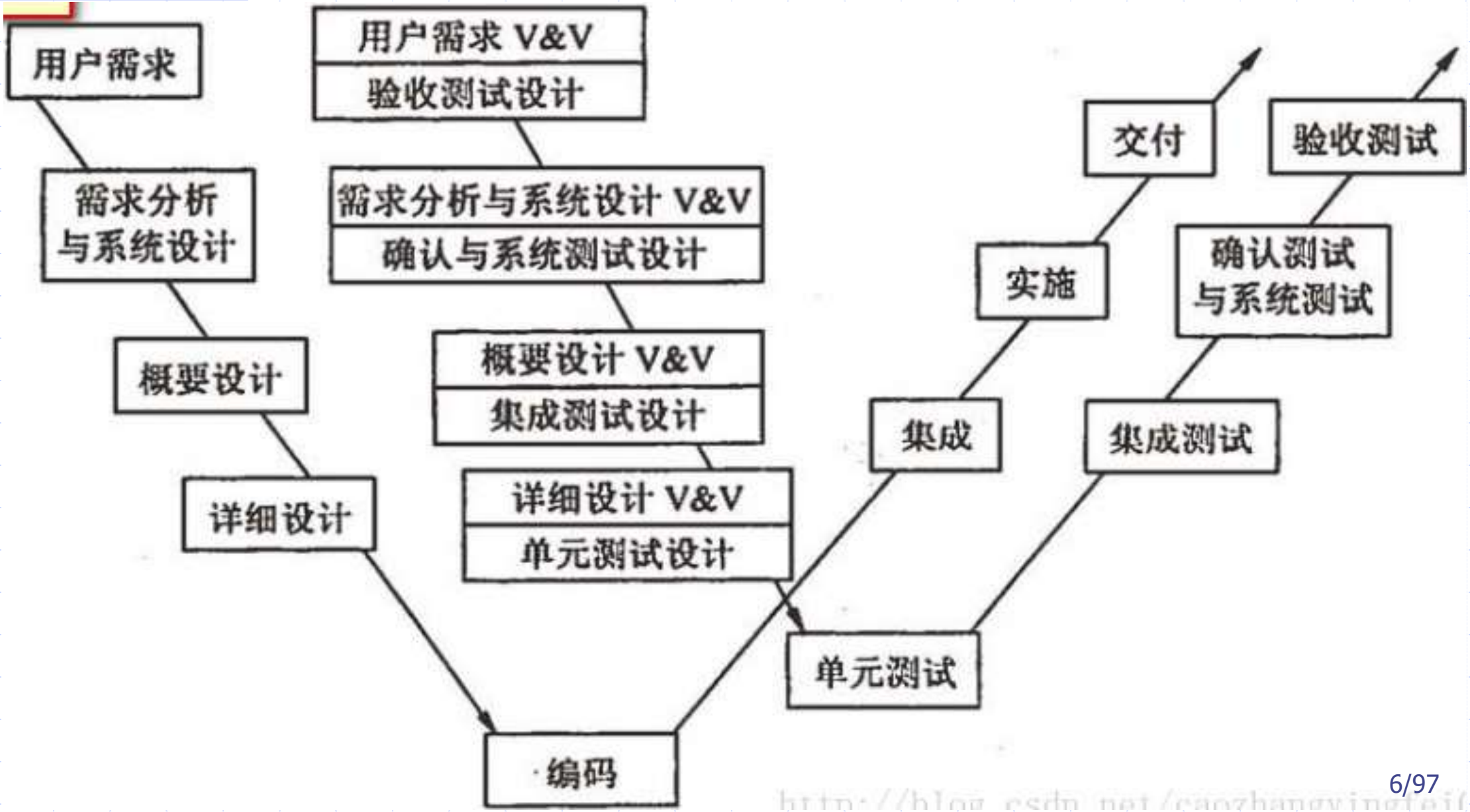
1.2 软件测试的W模型

- **W模型由Evolutif公司提出，相对于V模型而言，W模型增加了软件开发各个阶段的验证和确认活动，避免将最初的错误设计带入到代码中之后再进行验证，造成程序难于回溯。**



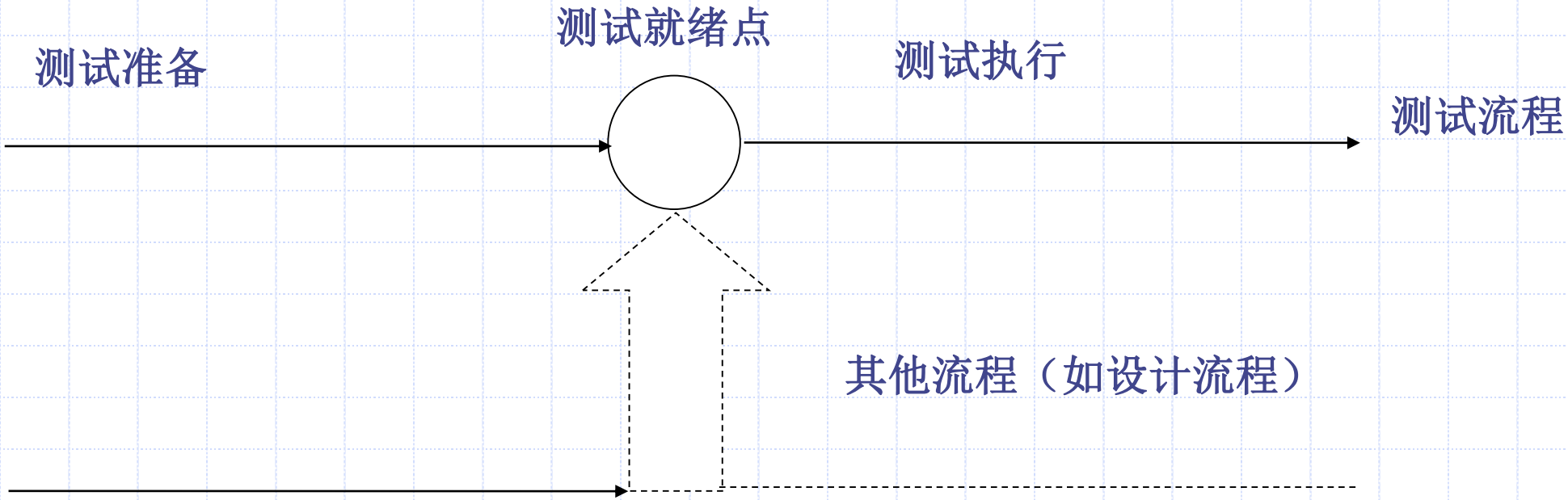
局限性：在W模型中，需求、设计、编码等活动被视为串行的，上一阶段完全结束，才可正式开始下一个阶段工作，无法支持迭代的开发模型。

软件测试的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
单元测试设计
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

1.3 软件测试的H模型

- 在H模型中，软件测试是一个独立的流程，贯穿于整个产品周期，与其他流程并发进行，当某个测试时间点就绪时，即从测试准备阶段进入测试执行阶段。



内容概要

1. 软件测试的模型
2. 软件测试的原则
3. 软件测试的术语
4. 软件测试基本过程
5. 完整的软件测试系统

软件测试的10大原则

- 测试的独立性原则（心理学）
- 测试应尽早介入（经济学）
- 完全测试程序是不可能的
- 软件测试是有风险的行为
- 找到的缺陷越多，就说明软件缺陷越多
- 软件缺陷有免疫力
- 并非所有软件缺陷都能修复
- 产品说明书不断变化
- 软件测试员在产品小组中不受欢迎
- 软件测试需要技术

2.1 测试的独立性原则（心理学）

- **程序员**通常不能高效测试自己的程序，因为很难适应这种心理，即尽量暴露自己的错误，这种掩饰行为降低了测试的效率；
- **测试员**的工作则是暴露程序的缺陷，他们担心的是找不到错误，在这种心理下，测试员会想尽办法找出程序的错误。
- 测试的独立性原则包含两个方面的意思：
  - 程序员应该避开测试自己的程序
  - 程序开发组织应避开测试自己开发的程序

2.2 测试应尽早介入（经济学）

- 早期的缺陷在其后的开发和发布过程中会不断被放大，需要很高的修复费用，还可能因为会触及到整个软件的构架而无法修复；
- 美国软件质量安全中心2000年对美国100家知名的软件厂商统计，软件缺陷在**开发前期**发现比开发后期发现在资金、人力上节约90%；软件缺陷在**推向市场前**发现比在推出后发现，在资金、人力上节约90%。

2.2 测试应尽早介入（经济学）

- 一个例子：1994年，Intel奔腾CPU芯片出现浮点运算缺陷
  - 计算(4195835/3145727)*3145727-4195835，其结果却不为0 ☹☠
- 实际上，在芯片发布之前，Intel的软件测试工程师已经发现这个问题，但管理层认为不需要修正
- 最终，Intel支付了4亿多美元进行芯片更换

2.3 不可能完全测试程序

- 不能够完全测试软件的原因：

1. 输入量太大
2. 输出结果太多
3. 软件实现的路径太多
4. 软件说明书没有客观标准

不可能完全测试程序举例

- 计算器程序

所有的输入：

1+1，1+2…

1+999999999999

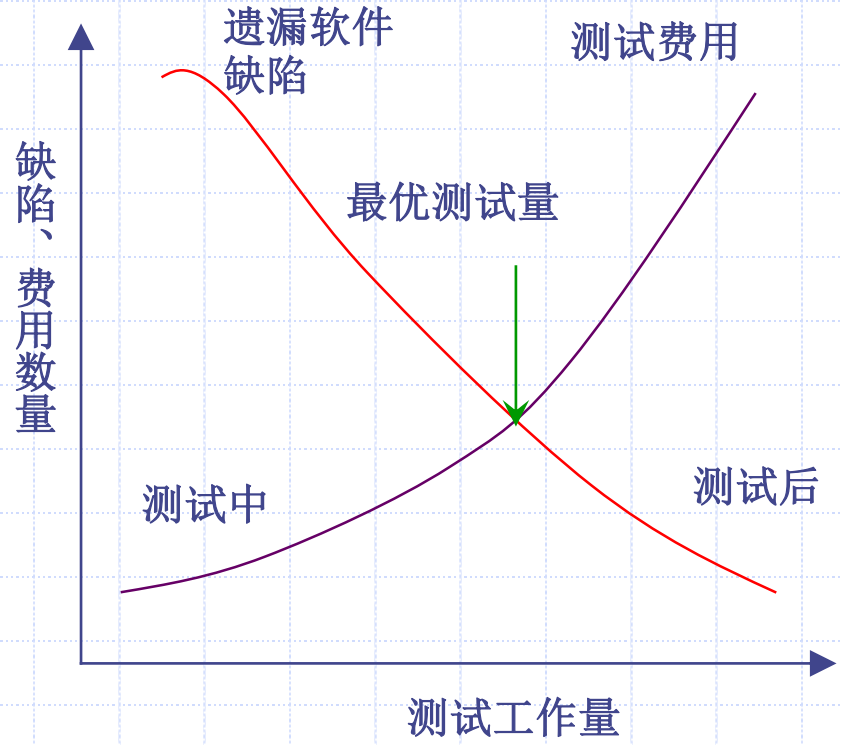
2+1，2+2

…

- 各种错误的输入

2.4 软件测试是有风险的行为

- 由于不能进行完全测试，某些隐藏的缺陷就是风险；
- 软件测试员需要学习把无穷的测试减少到可以控制的范围，以及针对风险采用什么样的明智决策，去粗存精。



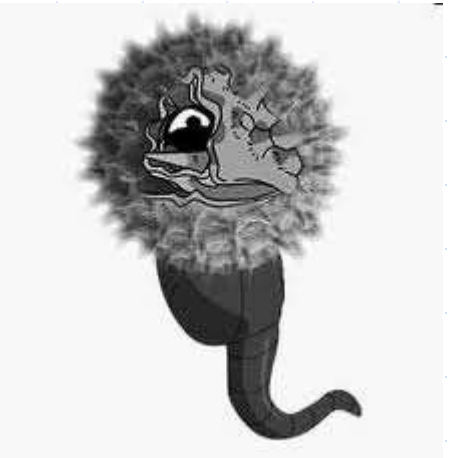
每一个软件项目都有一个最优测试量

2.5 找到的缺陷越多，说明软件的缺陷越多

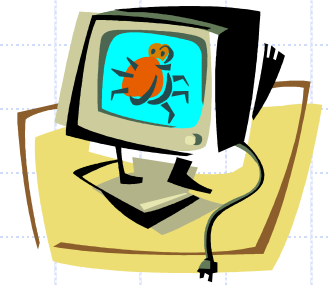
- 如果在软件测试中发现很多缺陷，那么说明这个软件的缺陷也越多
- 其中的原因是：
  - 程序员倦怠
  - 程序员容易犯同样的错误，比如，数据越界、内存泄露等
  - 某些软件缺陷是其它缺陷的征兆

2.6 杀虫剂现象

- 采用某种农药消灭害虫，后来发现害虫对这种农药具有了免疫力，这就是**杀虫剂现象**
- 对于软件，如果一直使用同样的测试方法，只能查找到同样问题，而其它软件缺陷可能被忽略
- 为了克服这种现象，需要尝试不同的测试方法，以便找出更多的软件缺陷



超级病毒



2.7 并非所有的软件缺陷都能修复

- 在软件测试中，并不是所有的软件缺陷都能被修复，这是因为：
  - 没有足够的时间
  - 不能算是真正的软件缺陷
  - 修复风险太大，修复一个错误可能引入其他软件缺陷
  - 有些错误不值得去修复，这归结为商业风险决策

2.8 如何应对产品需求的变化

- 由于IT业的发展迅速，导致一些软件在完成之前，产品的需求已经发生变化，比如竞争对手推出了新的功能，此时，我们应该如何应对？
- 软件测试员必须要想到产品需求的可能变化，因此需要灵活制定测试计划和执行测试技术（自动化测试）

2.9 软件测试员在小组中不受欢迎

- 软件测试员的任务是找出软件的缺陷，公布发现的问题，相当于监督员
- 为了克服监督员不受欢迎的局面，测试员可以采取以下措施：
  - 尽可能早地找出软件缺陷，这样容易改进
  - 控制情绪，不要把自己找到软件缺陷的喜悦在相应的程序员面前过度表达
  - 不要总报告坏消息，需要及时沟通

2.10 软件测试是一项讲究条理的技术

- 软件测试属于技术性工作，如果没有技术，只能进行黑盒的功能测试
- 性能测试、白盒测试、一些自动化测试工具等，都需要有较高的技术基础
- 有了良好技术，可以和开发人员讨论系统架构设计，验证系统是否具有可测性、发现单点失效、性能瓶颈等设计问题
- 有了良好技术，可以开发所需要的测试工具、自动化测试框架和自动化测试脚本

其它软件测试的原则

- ✓所有测试都应追溯到用户需求
- ✓应在测试工作开始前设计测试计划
- ✓测试应从小规模到大规模
- ✓避免不能重现或匆忙的测试
- ✓深入细致地审查测试结果
- ✓不要为了测试方便而修改程序
- ✓好的测试用例具有较高的发现过去未被发现过的错误的概率，而不应只表明程序运行正常

软件测试误区

- **误区一：**如果发布出去的软件有质量问题，都是软件测试人员的错
- **误区二：**软件测试技术要求不高，至少比编程容易多了
- **误区三：**有时间就多测试一些，来不及就少测试一些
- **误区四：**软件测试是测试人员的事，与开发人员无关
- **误区五：**根据软件开发瀑布模型，软件测试是开发后期的一个阶段

内容概要

1. 软件测试的模型
2. 软件测试的原则
3. 软件测试的术语
4. 软件测试基本过程
5. 完整的软件测试系统

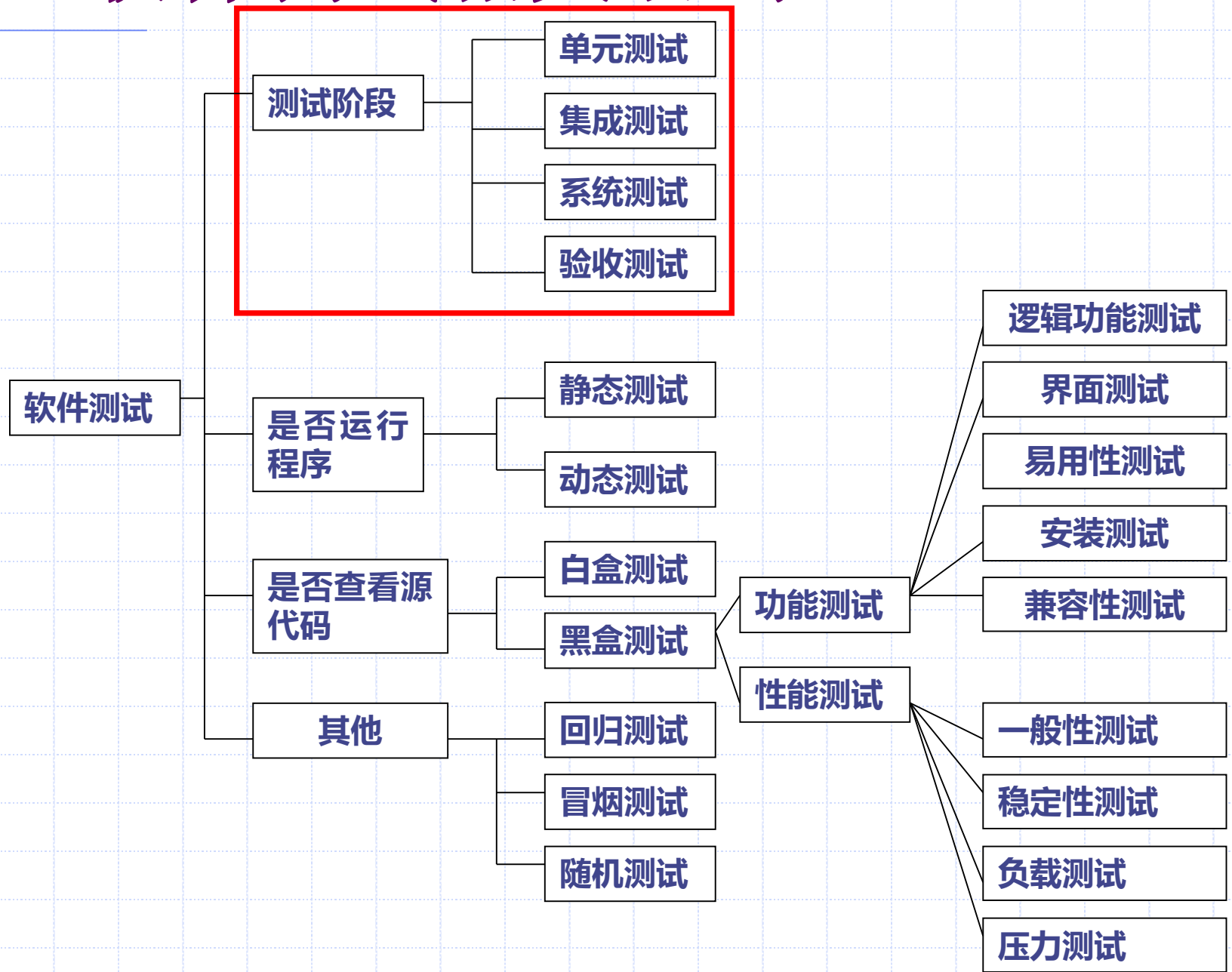
Verification & Validation

- Verification(验证) is the process confirming that software meets its specification.
  - Are we building the product right?
  - 软件是否满足需求说明
- Validation(确认) is the process confirming that it meets the user's requirements.
  - Are we building the right product?
  - 软件是否满足用户的需求

测试用例

- 测试用例（Test Case）是为某个特殊目标而编制的一组测试输入、执行条件以及预期结果，以便测试某个程序路径或核实是否满足某个特定需求。
  - 测试数据（输入）
  - 执行环境（条件）
  - 预期结果（评估）

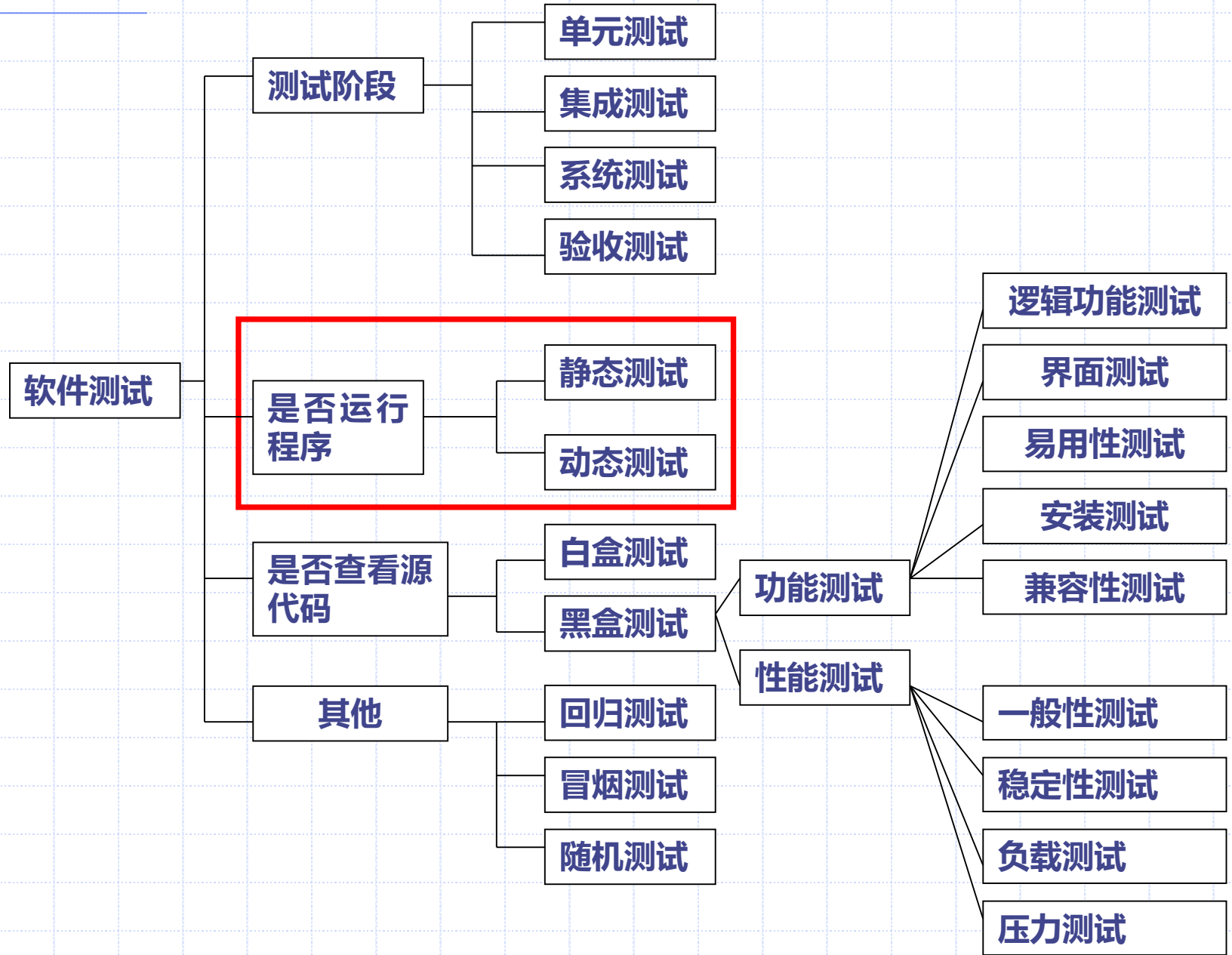
软件测试相关名词


软件测试
测试阶段
单元测试
集成测试
系统测试
验收测试
是否运行程序
静态测试
动态测试
是否查看源代码
白盒测试
黑盒测试
功能测试
逻辑功能测试
界面测试
易用性测试
安装测试
兼容性测试
性能测试
一般性测试
稳定性测试
负载测试
压力测试
其他
回归测试
冒烟测试
随机测试

测试阶段

- 单元测试(unit testing)：对每个模块或类的考察、测试；
- 集成测试(integration testing)：组装测试，联调，考察模块间的接口和联系；
- 系统测试(system testing)：把计算机和其他部件联结起来，考察软件是否满足系统总的功能要求和性能要求；
- 验收测试(acceptance testing)：用户为主的测试。

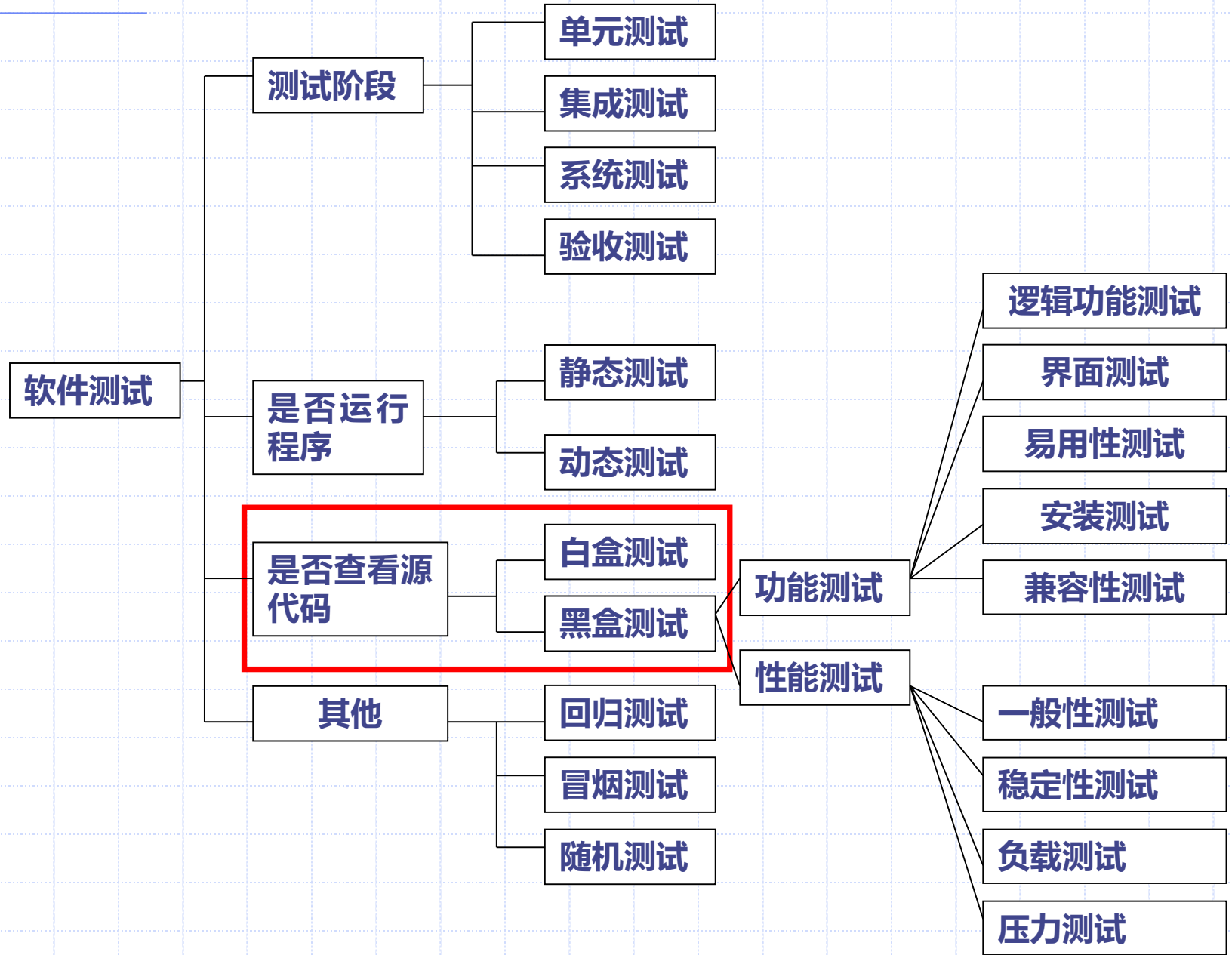
软件测试相关名词


软件测试
测试阶段
单元测试
集成测试
系统测试
验收测试
是否运行程序
静态测试
动态测试
是否查看源代码
白盒测试
黑盒测试
其他
回归测试
冒烟测试
随机测试
功能测试
逻辑功能测试
界面测试
易用性测试
安装测试
兼容性测试
性能测试
一般性测试
稳定性测试
负载测试
压力测试

静态测试和动态测试

- **静态测试：**不执行程序，仅分析、检查和审阅系统的相关文档，如需求文档、系统设计图、程序代码等。
- **动态测试：**运行并使用软件，通过测试用例，比较实际输出与预期输出，以发现存在的还未被揭示的错误。

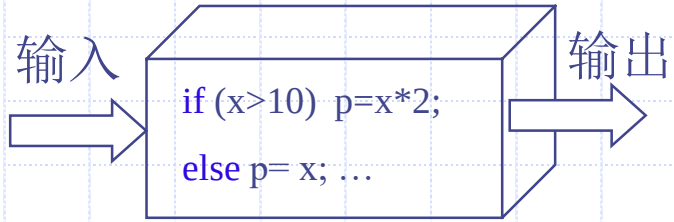
软件测试相关名词


软件测试
测试阶段
单元测试
集成测试
系统测试
验收测试
是否运行程序
静态测试
动态测试
是否查看源代码
白盒测试
黑盒测试
功能测试
逻辑功能测试
界面测试
易用性测试
安装测试
兼容性测试
性能测试
一般性测试
稳定性测试
负载测试
压力测试
其他
回归测试
冒烟测试
随机测试

黑盒测试和白盒测试

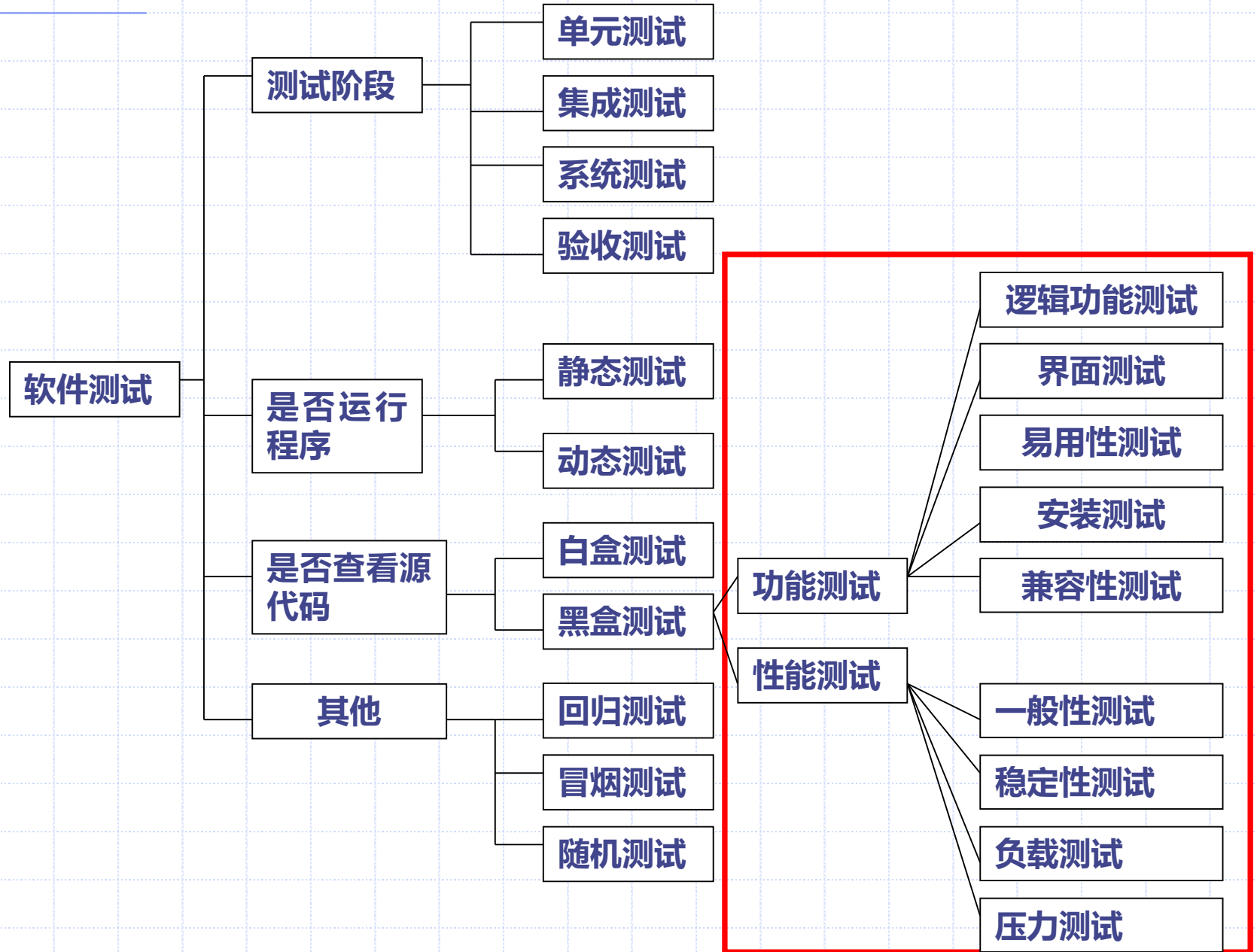
- **黑盒测试**——是**基于需求说明书**的软件测试，在这种测试下，不需要了解软件的内部结构，以及软件代码的具体实现
- **白盒测试**——了解软件的**内部结构**以及实现的具体代码，从而可以根据程序的内部情况来设计更具有针对性的测试案例

黑盒测试



白盒测试

软件测试相关名词


软件测试
测试阶段
单元测试
集成测试
系统测试
验收测试
是否运行程序
静态测试
动态测试
是否查看源代码
白盒测试
黑盒测试
其他
回归测试
冒烟测试
随机测试
功能测试
逻辑功能测试
界面测试
易用性测试
安装测试
兼容性测试
性能测试
一般性测试
稳定性测试
负载测试
压力测试

功能测试

- 是黑盒测试的一方面，检查软件功能是否符合用户的需求。
- 功能测试是黑盒测试中最重要的测试，比性能测试更加重要。因为软件的性能比较差用户还可以忍耐，但如果软件实现的根本不是用户所需要的功能……
- 功能测试分为：逻辑功能测试、界面测试、易用性测试（残疾人）、安装测试、兼容性测试。

性能测试

- 是软件测试的高端领域，性能测试工程师的待遇和白盒测试工程师不相上下。通常招聘时所说的高级软件测试工程师一般就指性能测试工程师或白盒测试工程师。性能测试一般要用到自动化测试工具。
- 软件性能包括很多方面，主要有时间性能和空间性能两种。主要有一般性测试、稳定性测试、负载测试、压力测试。

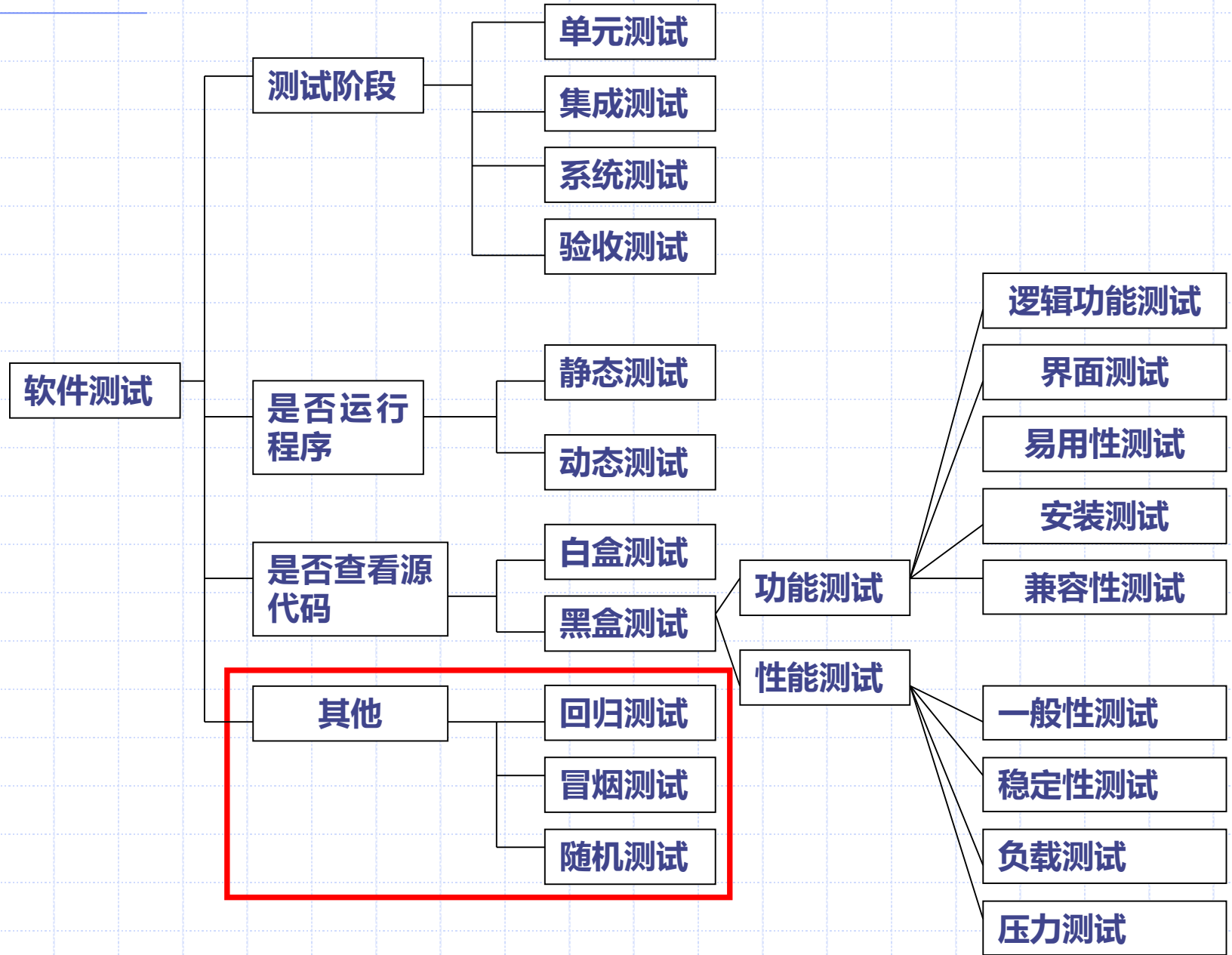
性能测试——负载测试

- 通过测试系统在**资源超负荷**情况下的表现，以发现设计上的错误或验证系统的负载能力。
- 负载测试中，将使测试对象承担不同的工作量，以评测和评估测试对象在不同工作量条件下的性能行为，以及持续正常运行的能力。
- 负载测试的目标是确定并确保系统在超出最大预期工作量的情况下仍能正常运行。

性能测试——压力测试

- 压力测试的基本思路：在计算机数量较少或系统资源匮乏的条件下运行测试。
- 通常要进行压力测试的资源包括内存、CPU 可用性、磁盘空间和网络带宽等。一般用并发来做压力测试。

软件测试相关名词


软件测试
测试阶段
单元测试
集成测试
系统测试
验收测试
是否运行程序
静态测试
动态测试
是否查看源代码
白盒测试
黑盒测试
功能测试
逻辑功能测试
界面测试
易用性测试
安装测试
兼容性测试
性能测试
一般性测试
稳定性测试
负载测试
压力测试
其他
回归测试
冒烟测试
随机测试

回归测试

- 对软件的新版本进行测试，以便确认新版本的修改没有引入其它错误。
- 可能重复执行上一个版本测试时的用例，也可能需要设计新的测试用例。
- 回归测试可以在任何测试阶段进行，既有黑盒回归也有白盒回归。
- 回归测试时如何选择测试用例是目前的一个研究热点。

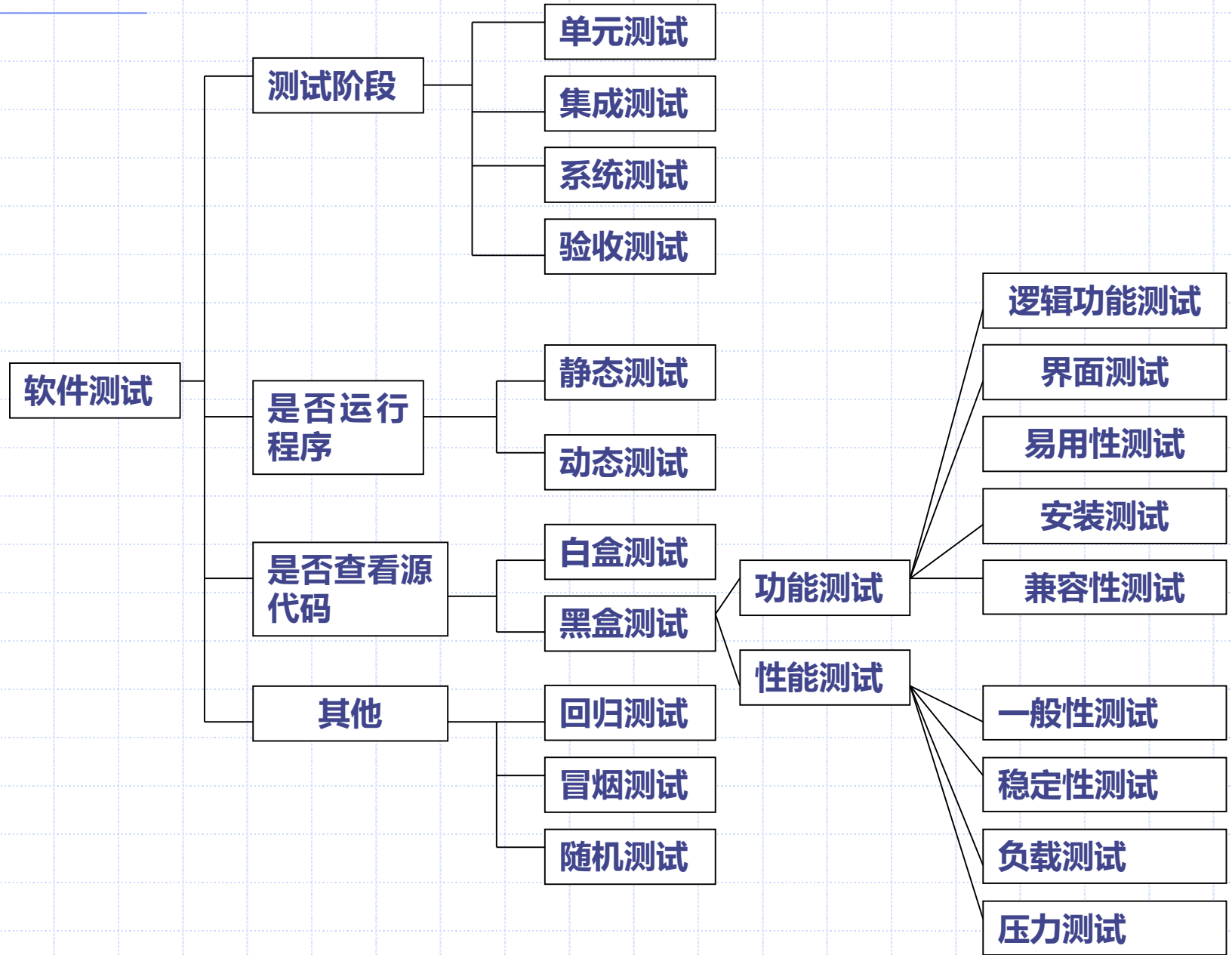
冒烟测试

- 来源：电路板质量检查，通电看是否冒烟。
- 软件冒烟测试的目的：在对一个新版本进行系统大规模的测试之前，先验证一下软件的基本功能是否实现，是否具备可测试性；
- 在软件测试中，冒烟测试主要指测试小组对主要功能进行先期测试，如果连主要功能都没有实现，直接打回去进行开发。

随机测试

- 也叫猴子测试，测试中所有的输入数据都是随机生成的，其目的是模拟用户的正式操作，并发现一些边缘性的错误。
- 优点是能够发现一些意想不到的错误
- 随机测试的缺点与不足
  - **测试不系统，无法统计代码覆盖率和需求覆盖率，很难再现、回归测试。**

软件测试相关名词


软件测试
测试阶段
单元测试
集成测试
系统测试
验收测试
是否运行程序
静态测试
动态测试
是否查看源代码
白盒测试
黑盒测试
功能测试
逻辑功能测试
界面测试
易用性测试
安装测试
兼容性测试
性能测试
一般性测试
稳定性测试
负载测试
压力测试
其他
回归测试
冒烟测试
随机测试

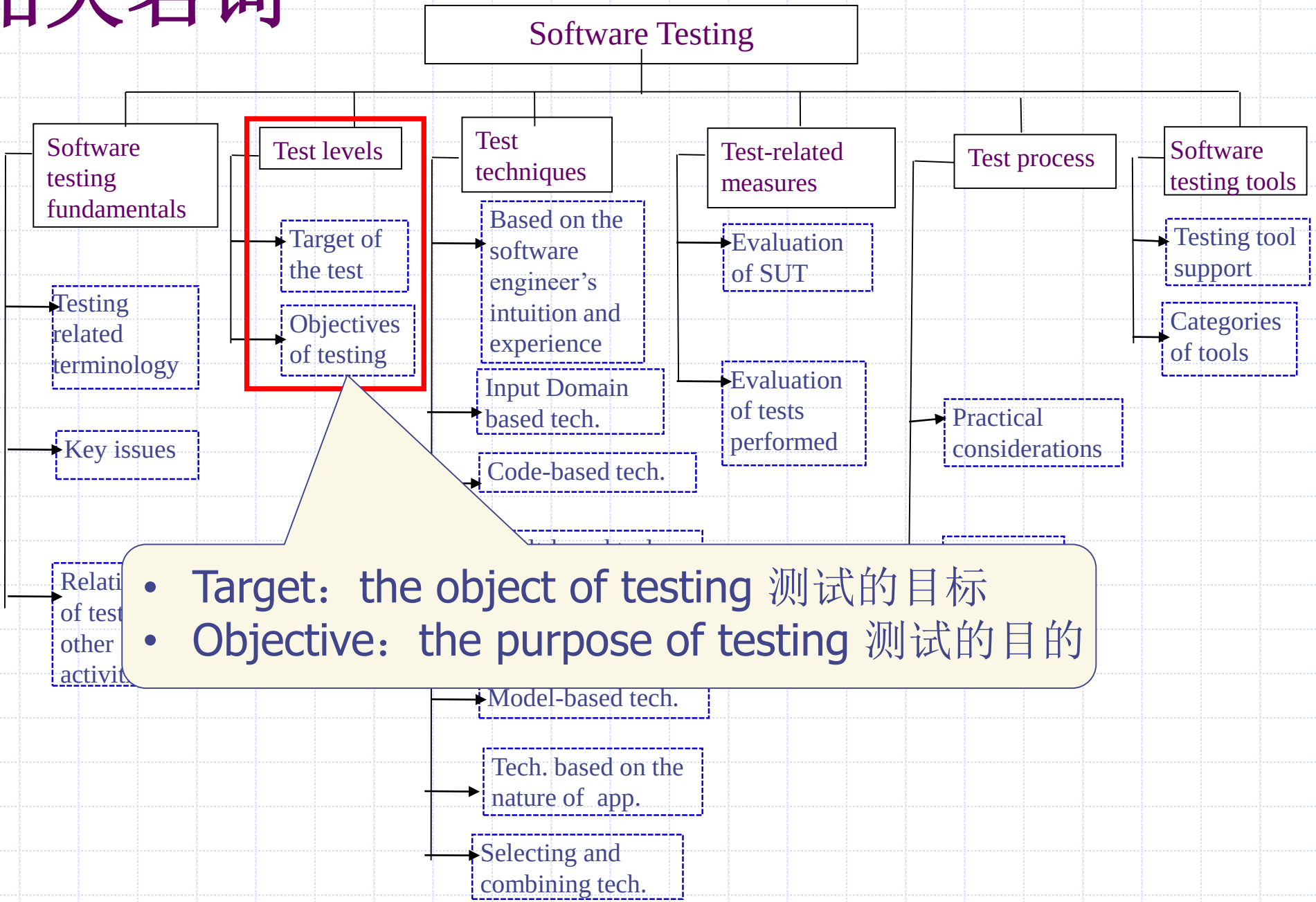
软件测试相关名词

组合使用不同的测试方法：

	白盒测试方法	黑盒测试方法
静态测试方法	**静态白盒测试**：对源程序代码进行语法检查、扫描、评审等	**静态黑盒测试**对需求文档、需求规格说明书的审查活动，一些非技术性文档测试等
动态测试方法	**动态白盒测试**用于单元测试：运行代码，对结果进行检查、验证和调试等	**动态黑盒测试**：运行程序，输入运行数据，对软件进行功能测试，从用户角度验证软件的各项功能

软件测试相关名词

Software Testing Knowledge Area in SWEBOK3.0



Test levels——The target of the test 测试的目标

- **unit testing:** verifies the functioning in isolation of software elements that are separately testable. A unit could be an individual subprogram or a larger component made of highly cohesive units.
- **integration testing:** the process of verifying the interactions among software components.
- **system testing:** testing the behavior of an entire system. System testing is usually considered appropriate for assessing the non-functional system requirements—such as security, speed, accuracy, and reliability. External interfaces to other applications, utilities, hardware devices, or the operating environments are also usually evaluated at this level.

Test levels—— Objectives of testing测试的目的

- **Acceptance/qualification testing:** determines whether a system satisfies its acceptance criteria, usually by checking desired system behaviors against the customer's requirements.
- **Installation testing:** verified the installation in the target environment. Installation testing can be viewed as system testing conducted in the operational environment of hardware configurations and other operational constraints.
- **Alpha and Beta testing:** Before software is released, it is sometimes given to a small, selected group of potential users for trial use (alpha testing) and/or to a larger set of representative users (beta testing).

Test levels—— Objectives of testing测试的目的

- **Reliability Achievement and Evaluation:** Testing improves reliability by identifying and correcting faults.
- **Regression testing:** the selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements.
- **Performance testing:** verifies that the software meets the specified performance requirements and assesses评估 performance characteristics.
- **Security testing:** verifies that the software is protected from external attacks.

  注意与safety testing区别

Test levels—— Objectives of testing测试的目的

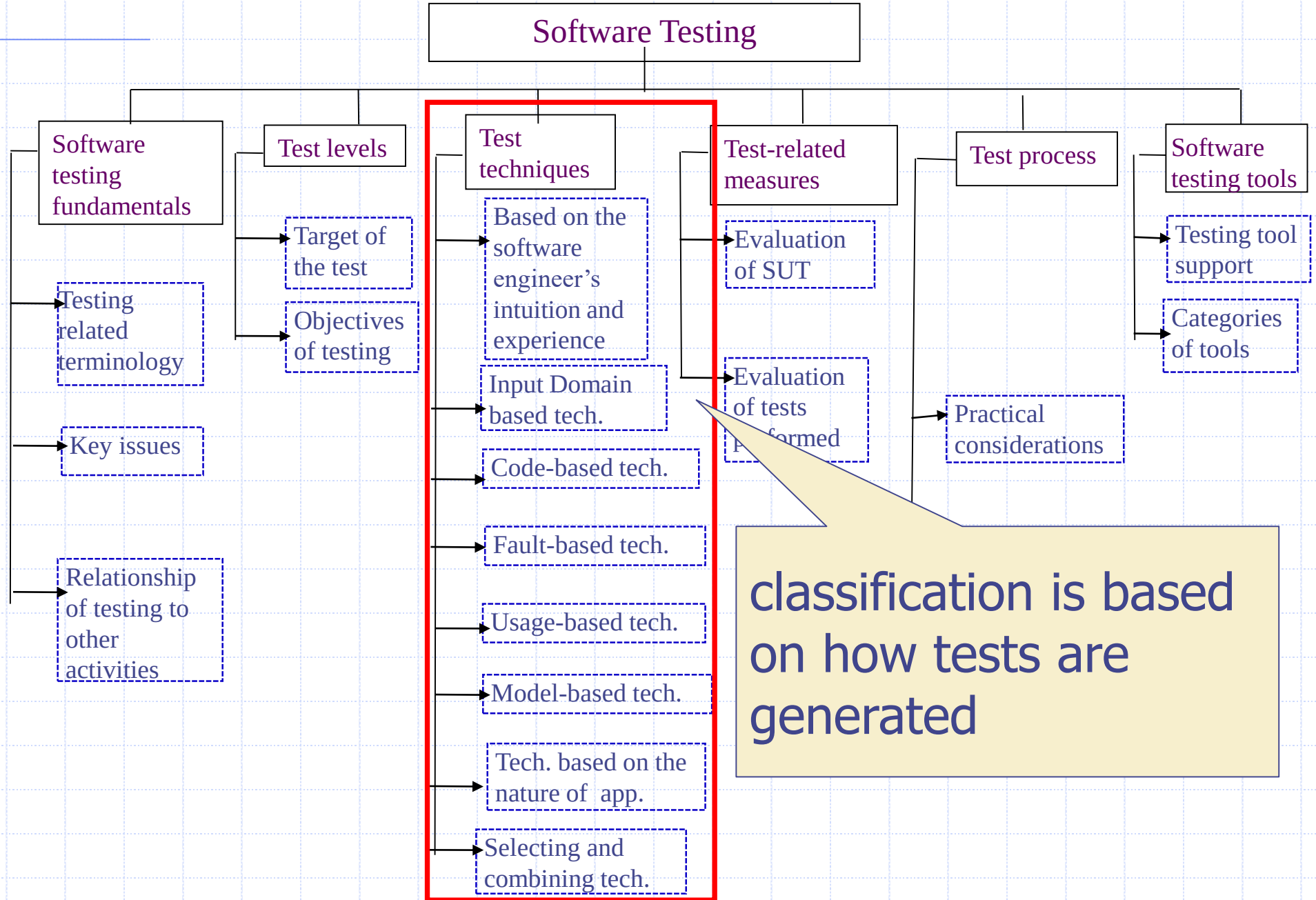
- **Stress testing:** exercises software at the maximum design load, as well as beyond it, with the goal of determining the behavioral limits, and to test defense mechanisms in critical systems.
- **Back-to-Back testing:** testing in which two or more variants of a program are executed with the same inputs, the outputs are compared, and errors are analyzed in case of discrepancies不一致.
- **Recovery testing:** verifies that the software restart capabilities after a system crash or other "disaster."
- **Interface testing:** verifies whether the components interface correctly to provide the correct exchange of data and control information.
  - A specific objective of interface testing is to simulate the use of APIs by end-user applications.

Test levels—— Objectives of testing测试的目的

- **Configuration testing:** In cases where software is built to serve different users, configuration testing verifies the software under different specified configurations.
- **Usability and Human Computer Interaction testing:** to evaluate how easy it is for end users to learn and to use the software.

  for example:
  - testing the software functions that supports user tasks
  - testing documentation that aids users
  - testing the ability of the system to recover from user errors

Software Testing Knowledge Area in SWEBOK3.0



Test Techniques

- **Based on the Software Engineer's Intuition and Experience基于直觉和经验的测试**
  - **Ad hoc: 自由测试**，根据测试人员的检验、直觉进行测试。在用常规方法难以确定测试用例时（测试易用性、用户体验等），可以使用。
  - **Exploratory Testing：探索式测试** tests are not defined in advance in an established test plan, but are dynamically designed, executed, and modified. 在测试的过程中，根据具体情况、经验、直觉等，动态确定下一步测试的目标。

Test Techniques

- **Input Domain-Based Techniques基于输入域的测试**
  - **Equivalence Partitioning: 等价类划分**
  - **Pairwise Testing：成对测试**
  - **Boundary-Value Analysis：边界值分析**
  - **Random Testing：随机测试**
  - **注：上面技术都属于黑盒测试方法**

Test Techniques

- **Code-Based Techniques基于代码的测试**
  - **Control Flow-Based Criteria: 基于控制流的覆盖准则**
  - **Data Flow-Based Criteria：基于数据流的覆盖准则**
  - **Reference Models for Code-Based Testing：将程序的控制流或数据流画成有向图，使用图的理论进行测试设计**
  - **注：上面技术都属于白盒测试方法**

Test Techniques

- **Fault-Based Techniques基于缺陷的测试：** devise test cases specifically aimed at revealing categories of likely or predefined faults (*fault model*).
  - **Error Guessing: 根据之前的测试或者经验，猜测最有可能出错的情况，进行测试。**
    - **比如：猜测程序中容易出现的问题，空指针、内存泄漏、session失效等**
  - **Mutation Testing：变异测试，在程序中植入错误（变异），通过测试找到它们。主要在白盒测试中使用**

Test Techniques

- **Usage-Based Techniques基于用户使用的测试**
  - **Operational Profile:** 模拟软件的实际使用环境进行测试，目的是推测软件在未来实际使用中的可靠性。
  - **User Observation Heuristics:** 测试系统的易用性。侧重于用户界面测试，可以假设自己是用户，可以通过调查问卷、访谈等形式。

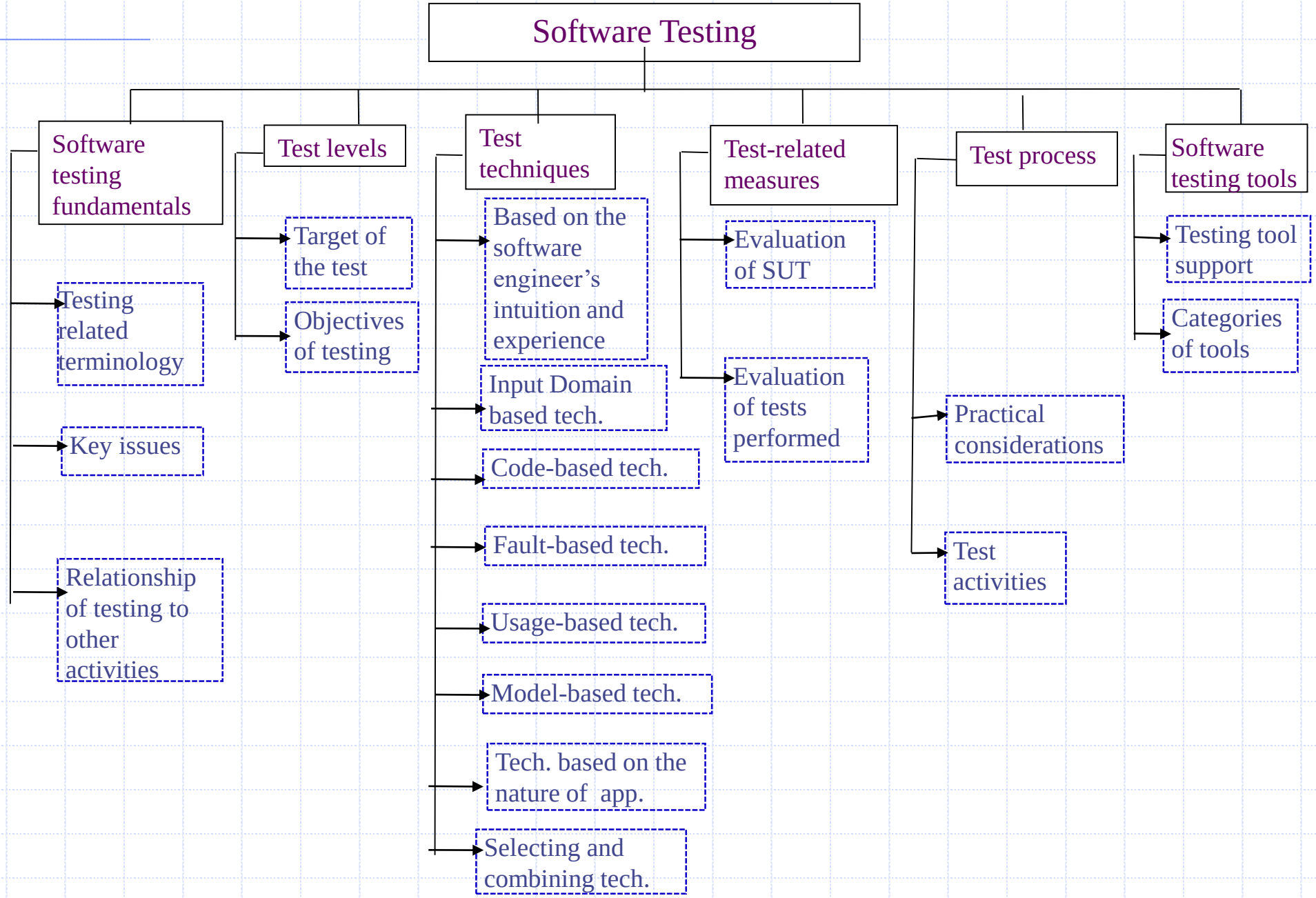
Test Techniques

- **Model-Based Testing Techniques基于模型的测试：** 用模型表示被测系统，validate requirements, check their consistency, and generate test cases focused on the behavioral aspects of the software.
  - **Decision Tables:** 决策表，各种输入输出的组合。
  - **Finite-State Machines:** 有限状态机，测试系统各种状态及其转换。
  - **Formal Specifications:** 用形式语言描述被测系统，并自动生成测试用例。
  - **Workflow Models:** 测试系统的一些典型工作流程（或叫场景）。主要用于业务流程测试中。
  - 注：上面技术都属于黑盒测试方法

Test Techniques

- **Techniques Based on the Nature of the Application：** 根据被测系统的特征进行不同的测试。典型系统类型：
  - object-oriented software
  - component-based software
  - web-based software
  - concurrent programs
  - protocol-based software
  - real-time systems
  - safety-critical systems
  - service-oriented software
  - open-source software
  - embedded software
  - 可以综合运用前面的各种测试方法进行测试

Software Testing Knowledge Area in SWEBOK3.0



正确认识各种软件测试技术

- 不论各种测试技术从原理上看多么好，都要通过客观的评估方式来评估其优劣。

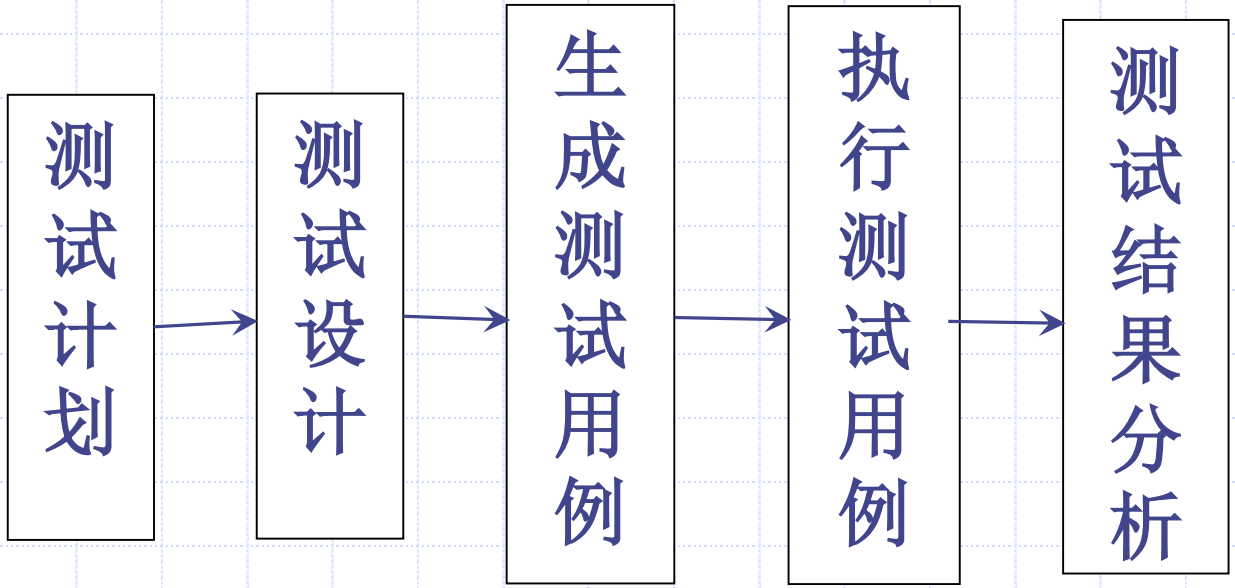
梅特林克的实验：聪明不一定总是好的

内容概要

1. 软件测试的模型
2. 软件测试的原则
3. 软件测试的术语
4. 软件测试基本过程
5. 完整的软件测试系统

4.1 最简单的测试过程

- 很多人认为，测试软件就是运行软件，看软件是否出错。实际上，运行软件只是软件测试的一部分。
- 测试过程一般包括测试计划、测试设计、生成测试用例、执行测试用例、测试结果分析几个阶段。



4.1.1 一个软件测试的例子

- 对“**Hello world**”程序进行测试，程序代码如下：

```
void main (int argc, char* argv[] )
{
    printf( “Hello world”);
    return 0;
}
```

测试过程

（1）测试计划：黑盒测试

（2）测试设计：运行程序，查看运行结果

（3）建立测试用例：使用表格的方式来构建简单的测试用例

测试Hello world程序的测试用例

用例编号	说明	操作过程	输入值	期望的结果
1	测试程序功能	运行软件	无	在控制台上打印出”Hello world”

测试过程

（4）执行测试用例

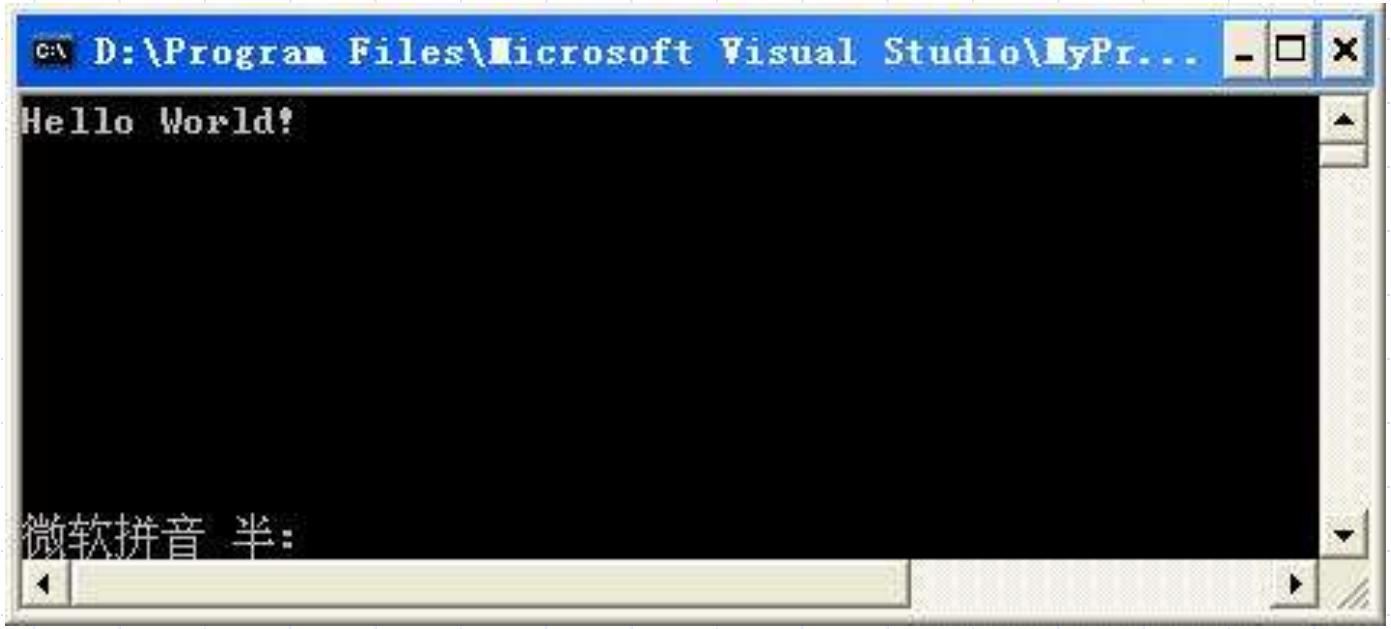
将Hello world 程序编译、连接，得到可执行程序Hello.exe，然后运行它，因为测试不要求输入值，因此运行软件即是执行测试

>Hello ↵

测试过程

（5）记录运行结果并分析

控制台上打印出**Hello world**字样



结果分析：测试的实际结果与期望的结果一致，程序的打印功能是正确的。

4.1.2 另一个软件测试的例子

对**Hello World**程序进行改进

```
void static main (int argc, char* argv[] )
{  switch ( *argv[1] ) {
   case      '1':              // 输入1打印Hello world
        printf( "Hello world!");   break;
   case      '2':              // 输入2打印Hello guy
        printf( "Hello guy!");     break;
   default:              // 输入其它参数打印Hello
        printf( "Hello!");  break;
} }
```

测试过程

（1）测试计划

黑盒测试

（2） 测试设计

运行程序，检查不同输入参数（合理输入和不合理输入）情况下的运行结果

测试过程

（3）建立测试用例

选择不同的输入参数来进行测试，新的测试用例：

编号	说明	操作过程	输入值	期望的结果
1	测试程序功能	运行软件	1	控制台显示”Hello world!”
2	测试程序功能	运行软件	2	控制台显示”Hello guy!”
3	测试程序功能	运行软件	3	控制台显示”Hello！”
4	测试程序健壮性	运行软件	无	控制台显示”Hello！”

测试过程

（4）执行测试用例

在控制台运行该程序，分别给程序带不同的输入参数，可执行程序的名字是**Hello.exe**

>Hello 1 ↵

>Hello 2 ↵

>Hello 3 ↵

>Hello ↵

测试过程

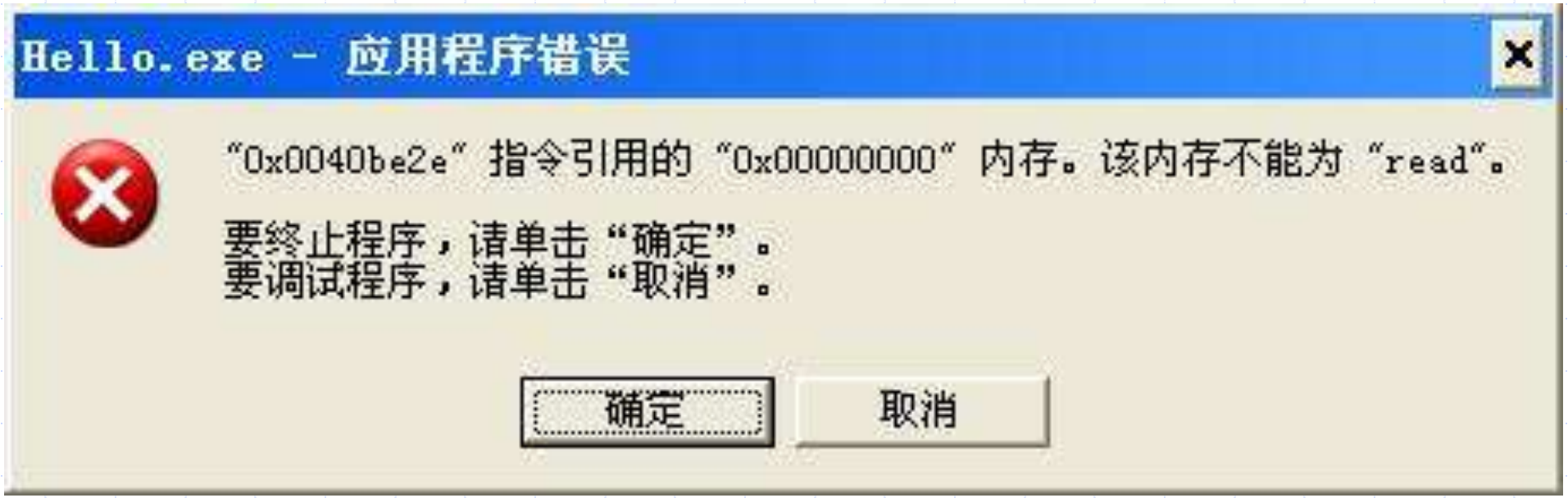
（5）记录运行结果并分析

编号	输入	期望结果	实际结果	通过/失败
1	1	打印”Hello world!”	打印”Hello world!”	通过
2	2	打印”Hello guy!”	打印”Hello guy!”	通过
3	3	打印”Hello!”	打印”Hello!”	通过
4	无	打印”Hello!”	弹出错误对话框	失败

测试过程

（5）记录运行结果并分析

执行不带参数的**Hello**程序后，弹出错误对话框，程序崩溃



测试过程

（5）记录运行结果并分析：分析被测程序

```
void static main (int argc, char* argv[] )
{  switch ( *argv[1] ) {
    case       '1':              // 输入1打印Hello world
         printf( "Hello world!");   break;
    case       '2':              // 输入2打印Hello guy
         printf( "Hello guy!");     break;
    default:              // 输入其它参数打印Hello
         printf( "Hello!");  break;
} }
```

测试过程

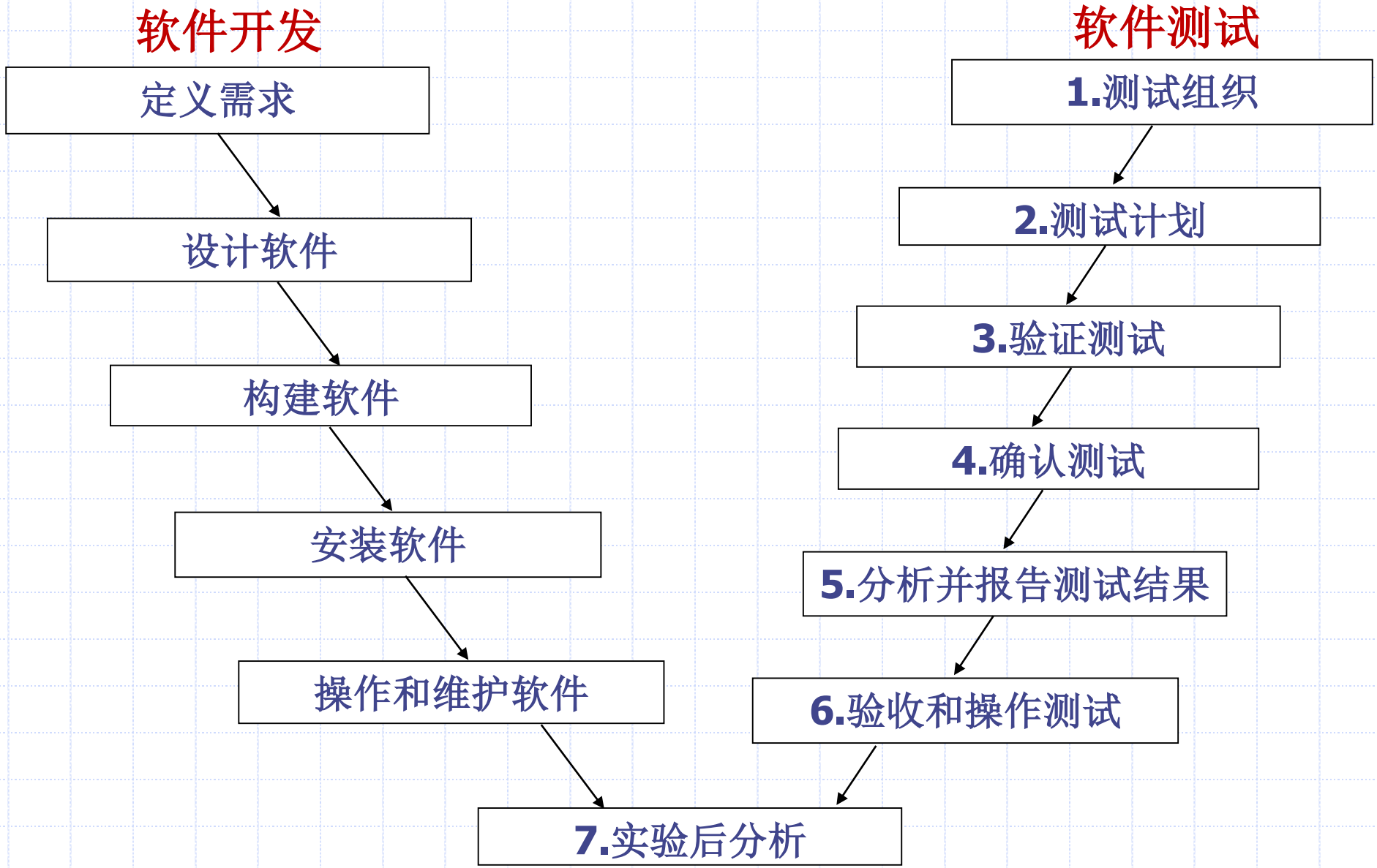
（5）记录运行结果并分析

- 程序中使用了*argv[1]，不输入任何参数时，argv[1]为空（NULL），程序访问空指针当然会出现问题；
- 如果在程序中对参数个数进行检查，就不会出现上面的问题
- 真是想不到，这么简单的一个程序也会出错
- 是的，即使我们认为再简单的程序也可能包含错误，软件测试可以找到这些错误

4.2 Perry的7步软件测试流程

- 美国质量保证协会**QAI**的创始人之一**William.E.Perry**创建，基于**1000**多家公司的实际测试经验，这些公司都是**QAI**会员。
- 该测试过程是迄今为止最完善、最实用、最具有实际指导意义的测试模型。

Perry软件测试流程图示



Perry软件测试流程说明

- Perry的7步软件测试流程是与软件开发过程相对应的，相融合的，是一个顺序向下的一个过程

- 开发和测试不应当是完全独立的过程，将开发和测试整合为一个过程，对于开发高质量的软件而言至关重要

4.2.1 组织测试

- 对测试进行组织属于测试前的准备活动之一，两个目的：
  - 确定测试的内容

    没有测试计划那么详细，只是确定测试的范围。比如只进行功能测试，需采用静态和/或动态测试等；
  - 建立测试团队

    确定测试团队中每个成员的任务。

4.2.2 测试计划

- 测试计划也属于测试准备活动之一，是测试工作的详细指南，包含测试风险分析、策略、方法、资源、进度等重要内容；
- 测试计划将确定测试的具体目标，测试工作都应该是按计划进行的，没有测试计划的测试很难到达测试目的。

4.2.3 验证测试（软件评审）

- 验证测试是高层测试，对软件需求说明、设计等进行验证，目的是确保构建正确的系统。
- 验证测试采用的技术通常是手动的，比如评审、走查、审查、需求跟踪、静态分析等等，这些静态技术的运用可以找到系统的高层缺陷。

4.2.4 确认测试

- 确认测试是利用各种测试技术和方法确定所构建系统的运行正确性
- 确认测试是在系统上运行测试用例
- 确认测试包含3方面的任务：
  - 建立测试用例（数据/脚本）
  - 执行测试
  - 记录测试结果

4.2.5 分析并报告测试结果

- 分析和报告测试结果是测试的输出，目的是将测试中获得的信息告知相关人员。包括软件在哪些方面存在问题，也包括测试员的一些建设性意见。
- 通常测试人员要编写两种测试报告：中间报告和最终的测试报告。
- 测试报告应该阐述缺陷造成的利害关系，以利于尽快地纠正缺陷。

4.2.6 验收和操作测试

- 验收测试的目标是确定软件各个方面都能满足用户的需求.
- 通过内部测试的软件需要在用户的实际环境中最后确认，只有在用户的实际环境中可用的系统才是真正有效地系统。

4.2.7 实现后分析（测试评估）

- 实现后分析是面向未来的。主要回答如下问题：测试是否有效？测试人员的表现如何？如何改进测试过程？等等。
- 在每次软件测试结束后，通过评估测试效率可以最大限度地改进测试。
- 这种评估除了由测试员参加之外，还应该邀请开发人员、软件用户以及质量专业人员共同参与。

内容概要

1. 软件测试的模型
2. 软件测试的原则
3. 软件测试的术语
4. 软件测试基本过程
5. 完整的软件测试系统

完整的软件测试系统

- 测试本身包含丰富的内容，为了能够真正高效和有效地进行软件测试，首先要构建一个完整的测试系统，这个测试系统将作为测试的基础

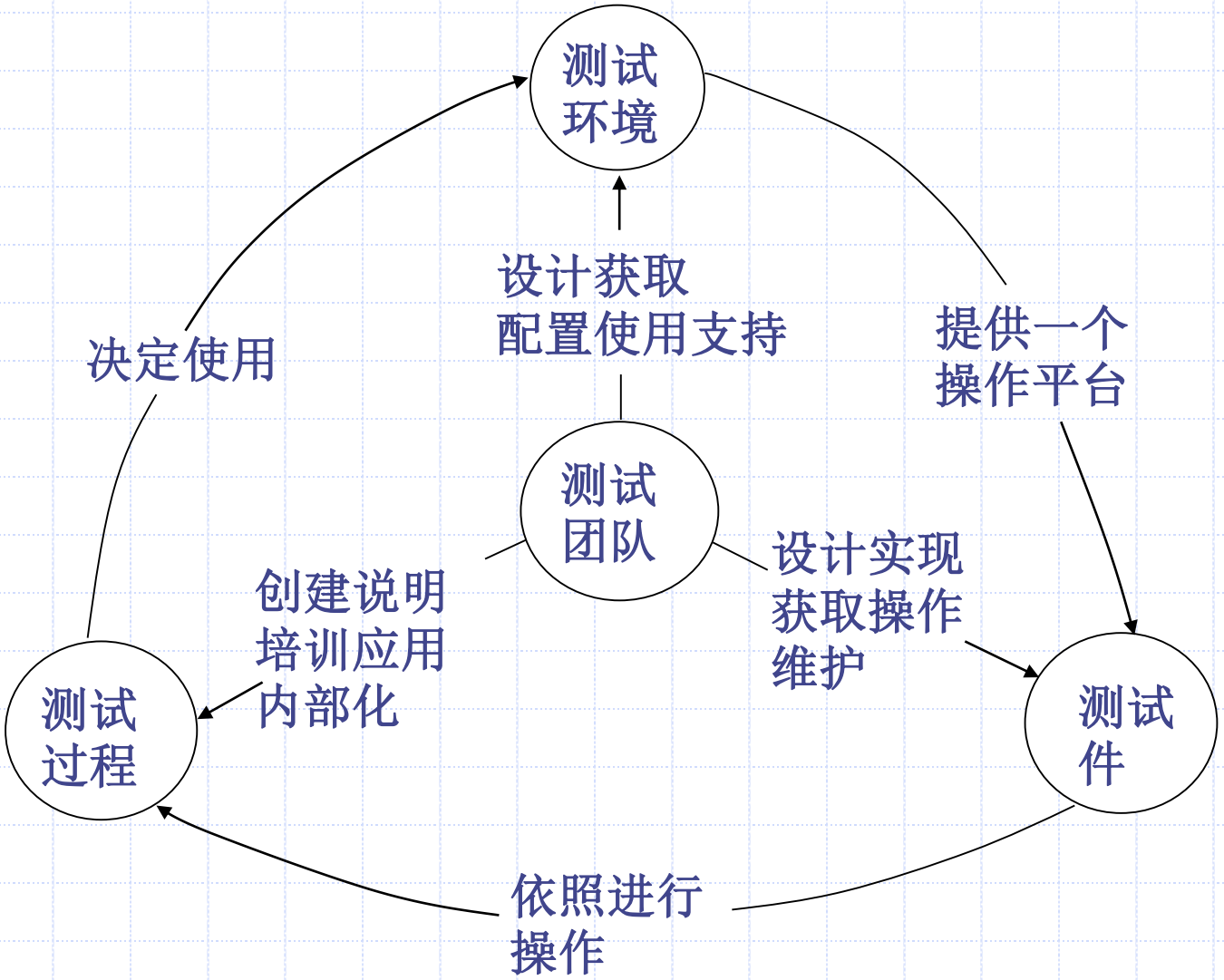
5.1 什么是软件测试系统

- 软件测试系统（Software testing system）是进行软件测试所包含的全部内容
- 包括测试人员、测试环境、测试过程、测试件、测试原则、测试管理、测试策略以及测试技术等

5.2 软件测试系统的作用

- 软件测试系统彼此联系完成整个测试任务，其输入是一项测试任务，输出是测试的结果，如测试总结报告等

5.3 Rex.Black 的软件测试系统



5.3.1 Rex.Black 测试系统说明

- 在Rex. Black的软件测试系统中：测试团队居中，处于核心位置，测试团队根据测试任务设计、创建测试环境、测试件以及测试过程；然后依赖测试环境并按照测试过程操作测试件，完成测试工作，最后得出的测试报告又加入到测试件中

5.3.2 软件测试系统的相关术语

1. 测试团队(test team)——进行软件测试工作的相关人员，包括测试经理、测试工程师、测试技术员等，他们分工协作，团结一致，利用自己的测试知识、技术以及经验共同完成测试任务

- 测试团队处于测试系统的核心位置

5.3.2 软件测试系统的相关术语

2. 测试件（testware）——测试件是软件测试中使用的所有人工制品，包括在测试过程中产生的测试计划、测试设计和执行测试以及测试报告所需要的人工制品，例如文档、脚本、输入、预期结果、安装和清理步骤、文件、数据库、环境和任何测试中使用的软件和工具

5.3.2 软件测试系统的相关术语

3. 测试环境（test environment）——执行测试需要的环境，包括硬件、仪器、模拟器等，软件工具和其他支持要素，比如测试空间
4. 测试过程（test process）——完成测试工作的流程，它由一组文档化的指导原则指导进行，比如制定测试计划、设计测试、执行测试以及测试总结等

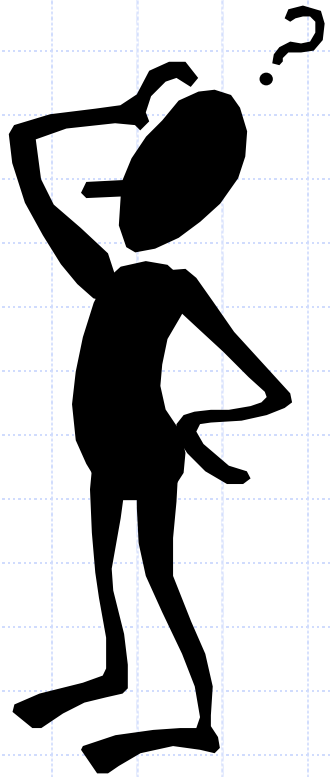
举例

- 为了测试前面的**Hello**软件，建立一个测试系统，包括：

1. 测试团队：包含一名测试员
2. 测试环境：放在实验室的一台电脑，当然还包括放电脑的桌子和测试员要坐的椅子
3. 测试件：包括测试用例，测试记录的**Excel**软件及制作的测试表格
4. 测试过程：执行的简单流程，生成测试用例，执行测试用例，记录结果及分析结果

The End

Any Question?



作业1：

按照第4部分（软件测试基本过程）介绍的简单测试流程，构造下面程序段的测试流程 。

```
void static main (int argc, char* argv[])
{
   int   x = atoi ( *argv[1] );
   int   y = atoi ( *argv[3] );
   switch ( *argv[2] ) {
   case'+':                          /// 两个数相加
        printf( "%d + %d  = %d", x, y, x+y);        break;
   case'-':                          /// 两个数相减
        printf( "%d - %d  = %d", x, y, x-y);        break;
   case'*':                          /// 两个数相乘
        printf( "%d * %d  = %d", x, y, x*y);        break;
   case'/':                          /// 两个数相除
        printf( "%d / %d  = %d", x, y, x/y);        break;
   default:                          /// 输入其它符号打印正确的使用方法
        printf( "Correct use: x +(-*/) y ");   break;
    }
}
```

作业2：简答题

1. 你认为软件测试和软件调试的区别是什么？

2. 移动（手机）应用程序测试与普通应用程序测试相比，你认为其功能测试、性能测试有哪些不同的特点？（自由发挥）