

# 第一讲：软件测试概论

郁 莲

THE SCHOOL OF SOFTWARE AND MICROELECTRONICS

PEKING UNIVERSITY

NO.24 JINYUAN RD, BEIJING 102600

# 提 纲

1 软件失效的巨大代价 (Spectacular Software Failures)

2 什么是软件测试 (What Do We mean by Testing) ?

3 软件测试关键概念 (Software Testing Terminology)

4 测试人员职业发展与素质

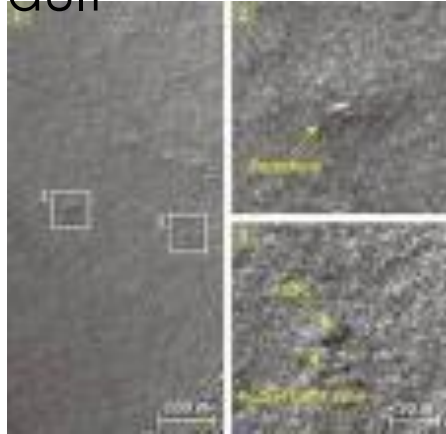
5 总结

## 高昂软件失效代价 (COSTLY SOFTWARE FAILURES)

- NIST report, “The Economic Impacts of Inadequate Infrastructure for Software Testing” (2002)
  - Inadequate software testing costs the US alone between \$22 and \$59 billion annually
  - Better approaches could cut this amount in half
- Huge losses due to web application failures
  - Financial services : \$6.5 million per hour (just in USA!)
  - Credit card sales applications : \$2.4 million per hour (in USA)
- In Dec 2006, *amazon.com*’s BOGO offer turned into a double discount
- 2007 : Symantec says that most **security vulnerabilities** are due to faulty software

# 惊人的软件失效代价 (SPECTACULAR SOFTWARE FAILURES)

- NASA's Mars lander: September 1999, crashed due to a units integration fault



Mars Polar Lander crash site?

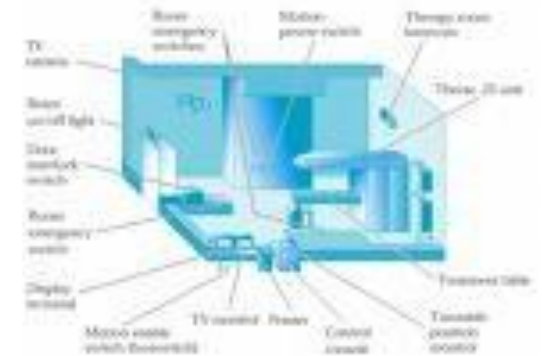
Ariane 5: exception-handling bug : forced self destruct on maiden flight (64-bit to 16-bit conversion: about 370 million \$ lost)



- Toyota brakes : Dozens dead, thousands of crashes
- Major failures: Ariane 5 explosion, Mars Polar Lander, Intel's Pentium FDIV bug
- Poor testing of safety-critical software can cost *liv*
  - THERAC-25 radiation machine: 3 dead

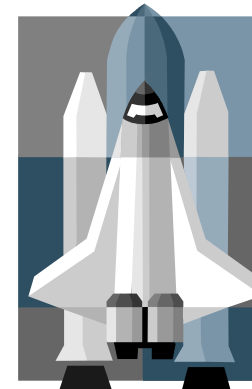
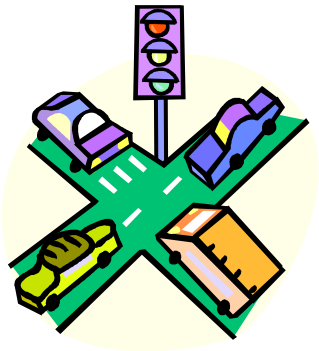
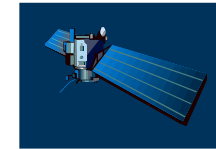


THERAC-25 design



**We need our software to be reliable**  
**Testing is how we assess reliability**

# SOFTWARE IS A SKIN THAT SURROUNDS OUR CIVILIZATION



Quote due to Dr. Mark Harman

# 提 纲

1

软件失效的巨大代价 (Spectacular Software Failures)

2

什么是软件测试 (What Do We mean by Testing) ?

3

软件测试关键概念 (Software Testing Terminology)

4

测试人员职业发展与素质

5

总结

# 什么是软件测试？

- Myers (1979)定义：
  - 测试是**执行程序**的过程，其目的是**发现错误**。
- IEEE标准610.12 (IEEE, 1990)给出了两个更为规范、约束的测试定义：
  - (1) 在特定的条件下运行系统或构件，观察或记录结果，对系统的某个方面做出评价；
  - (2) 分析某个软件项以发现现存的和要求的条件之差别（即错误）并评价此软件项的特性。

动态测试  
Dynamic  
testing

静态测试  
Static  
testing

## 注意!

- 有资料表明，60%以上的软件错误并不是程序错误，而是软件需求和软件设计错误。
- 错误的理解用户需求会导致开发完美优良的，但却是不正确的产品。
- 因此，做好软件需求和软件设计阶段的质量保证工作也是非常重要的。

## 注意!

- 英文里“testing”和“test”在翻译成中文时，虽然都译为“测试”，但是有区别的。
  - “testing”是指一个过程，
  - 而“test”一般是指执行测试的一次活动。
- 我们在看到“测试”时，要根据上下文来判断是哪种意思。
- 本课程名称中的“测试”是“testing”。



# 软件测试的目的



Myers是这样来描述软件测试的目的：

- “测试是程序的执行过程，目的在于发现错误；
- 一个好的测试用例具有很高的可能性发现至今尚未发现的错误的测试用例；
- 一个成功的测试是指发现了至今尚未发现的错误的测试”。

Bill Hetzel[HET93]提出了：

- 测试目的不仅仅是为了发现软件缺陷与错误，而且也是对软件质量进行度量和评估，以提高软件的质量。

## 软件测试的目的（续）



### 修复 bugs

测试的目的是要以最少的人力、物力和时间找出软件中的各种错误与缺陷，通过修正各种错误与缺陷提高软件质量，避免软件发布后由于潜在的软件缺陷和错误造成的隐患所带来的经济风险。

### 满足需求

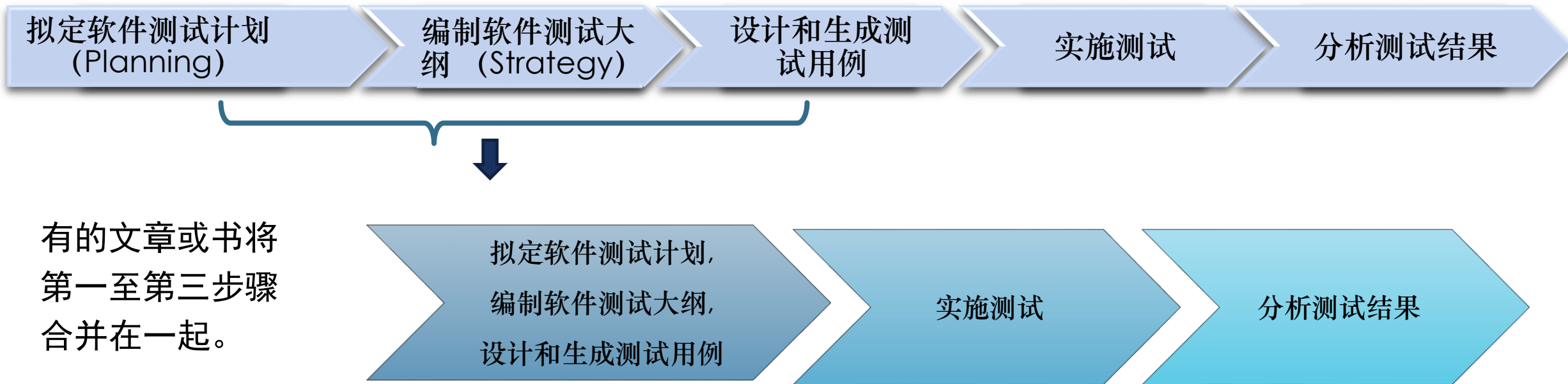
同时，测试是以评价一个程序或者系统属性为目标的活动，测试是对软件质量的度量与评价，以验证软件的质量满足用户的需求的程度，为用户选择与接受软件提供有力的依据。当然，通过最终的验收测试，也可以证明软件满足了用户的需求，树立人们是用软件的信心。

### 过程改进

此外，通过分析错误产生的原因还可以帮助发现当前开发工作所采用的软件过程的缺陷，以便进行软件过程改进。同时通过对软件结果的分析整理，为风险评估提供信息，还可以修正软件开发规则，并为软件可靠性分析提供依据。

# 软件测试过程

软件测试是一个复杂的过程，通常包括以下基本的测试活动。



# 软件测试过程（续）

## 拟定软件测试计划

1

- 确定主要的目标，测试范围，
- 系统功能和非功能需求，测试环境，
- 测试自动控制，
- 测试结果分析计划，问题解决方案与报告计划，
- 测试重用计划，
- 系统恢复计划，
- 活动时间表，
- 测试结束标准。

## 编制软件测试大纲：

2

- 软件测试大纲是软件测试的依据。
- 它明确详尽地规定了在测试中针对系统的每一项功能或特性所必须完成的基本测试项和测试完成的标准。
- 无论是自动测试还是手动测试,都必须满足测试大纲的要求。

# 软件测试过程（续）



## 设计和生成测试用例：

3

- 一般而言,测试用例是指为实施一次测试而向被测系统提供的输入数据、操作或各种环境设置, 以及被测系统的期望输出。
- 测试用例控制着软件测试的执行过程,它是对测试大纲中每个测试项的进一步实例化。

FOCUS

(续)

- 测试用例的代表性：能够代表各种合理和不合理的、合法的和非法的、边界和越界的,以及极限的输入数据、操作和环境设置等;
- 测试结果的可判定性：即测试执行结果的正确性是可判定的或可评估的;
- 测试结果的可再现性：即对同样的测试用例,系统的执行结果应当是相同的。

## 软件测试过程（续）

### 实施测试：

4

- 软件测试的实施阶段是由一系列的测试周期(Test Cycle)组成的。
- 在每个测试周期中,软件测试工程师将依据预先编制好的测试大纲和准备好的测试用例,通过执行被测软件,对其进行的测试。
- 即向被测软件输入数据或激发事件,以观察其输出结果。

### 分析测试结果：

5

- 执行软件测试过程中,收集通过与未通过的测试用例。后者将触发纠错过程。测试与纠错通常是反复交替进行的。
- 当使用专业测试人员时,测试与纠错甚至是平行进行的,从而压缩总的开发时间。
- 测试结果分析可生成软件问题报告供有关人员参考或作进一步分析。

# TEST PLAN (TAKE IEEE AS AN EXAMPLE)

## IEEE 829-1998

- a) Test plan identifier;
- b) Introduction;
- c) Approach;
- d) Test items;
- e) Features to be tested;
- f) Features not to be tested;
- g) Item pass/fail criteria;
- h) Suspension criteria and resumption requirements;

## IEEE 829-1998 (cont'd)

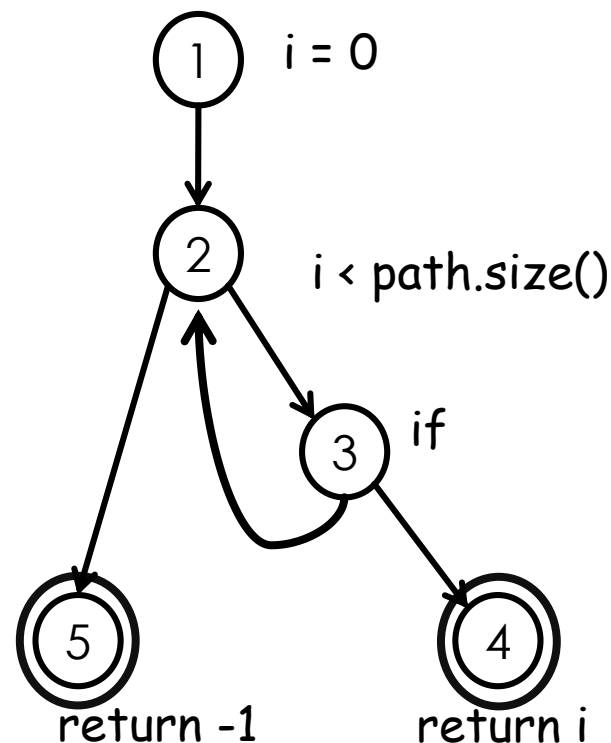
- i) Test deliverables;
- j) Testing tasks;
- k) Environmental needs;
- l) Responsibilities;
- m) Staffing and training needs;
- n) Schedule;
- o) Risks and contingencies;
- p) Approvals.

# 一个简单程序测试示例 (A SMALL ILLUSTRATIVE EXAMPLE)

## Software Artifact : Java Method

```
* Return index of node n at the  
* first position it appears,  
* -1 if it is not present  
*/  
public int indexOf (Node n)  
{  
    for (int i=0; i < path.size(); i++)  
        if (path.get(i).equals(n))  
            return i;  
    return -1;  
}
```

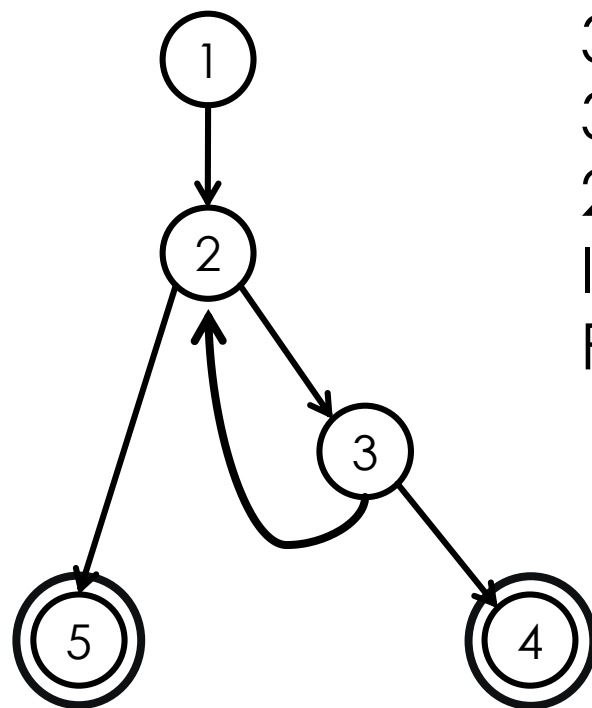
## Control Flow Graph





# 程序测试示例 (续)

Graph  
Abstract version



Edges

1 2

2 3

3 2

3 4

2 5

Initial Node: 1

Final Nodes: 4, 5

Test Paths

[1, 2, 5]

[1, 2, 3, 2, 5]

[1, 2, 3, 2, 3, 4]

6 requirements for  
Edge-Pair  
Coverage

1. [1, 2, 3]

2. [1, 2, 5]

3. [2, 3, 4]

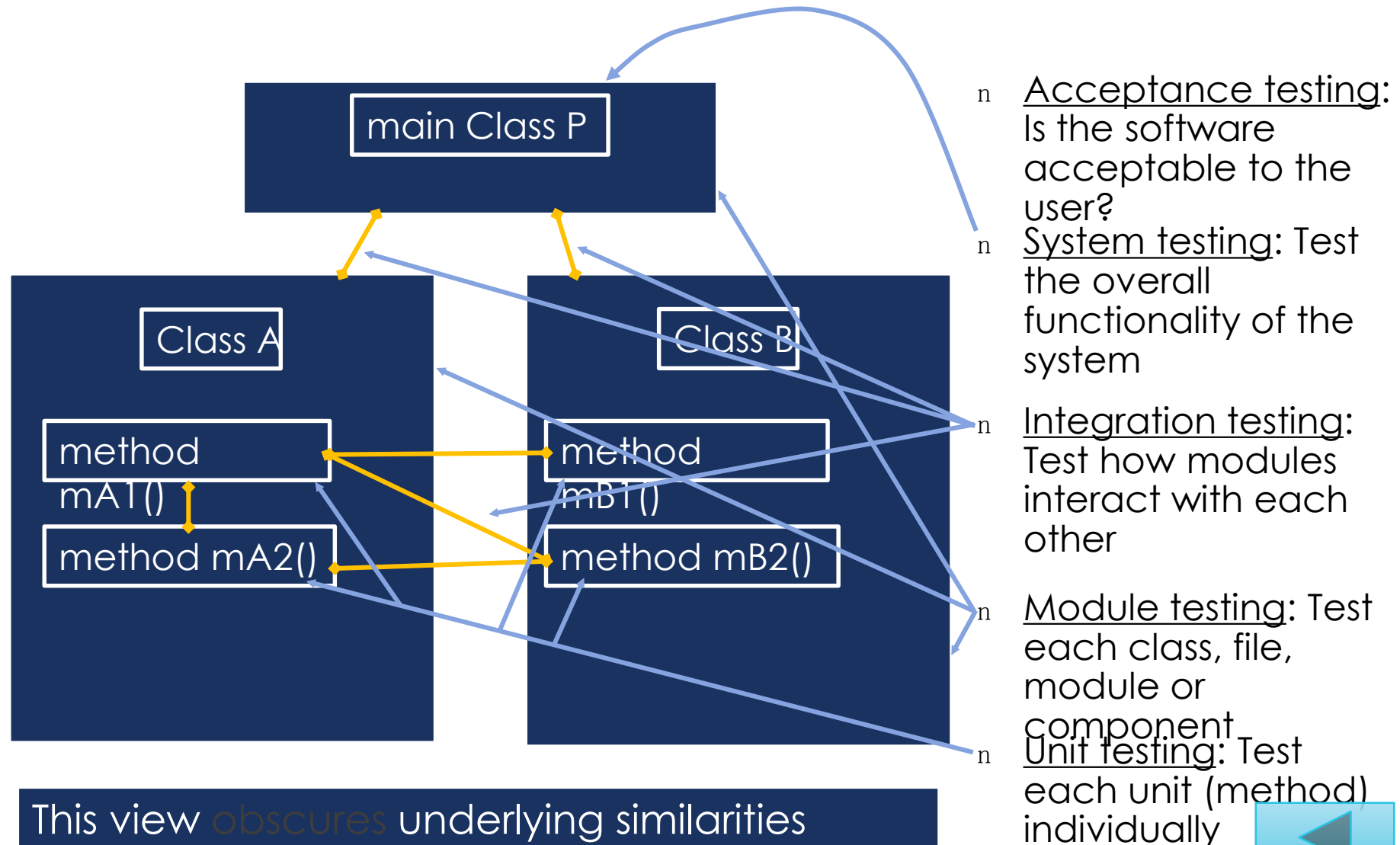
4. [2, 3, 2]

5. [3, 2, 3]

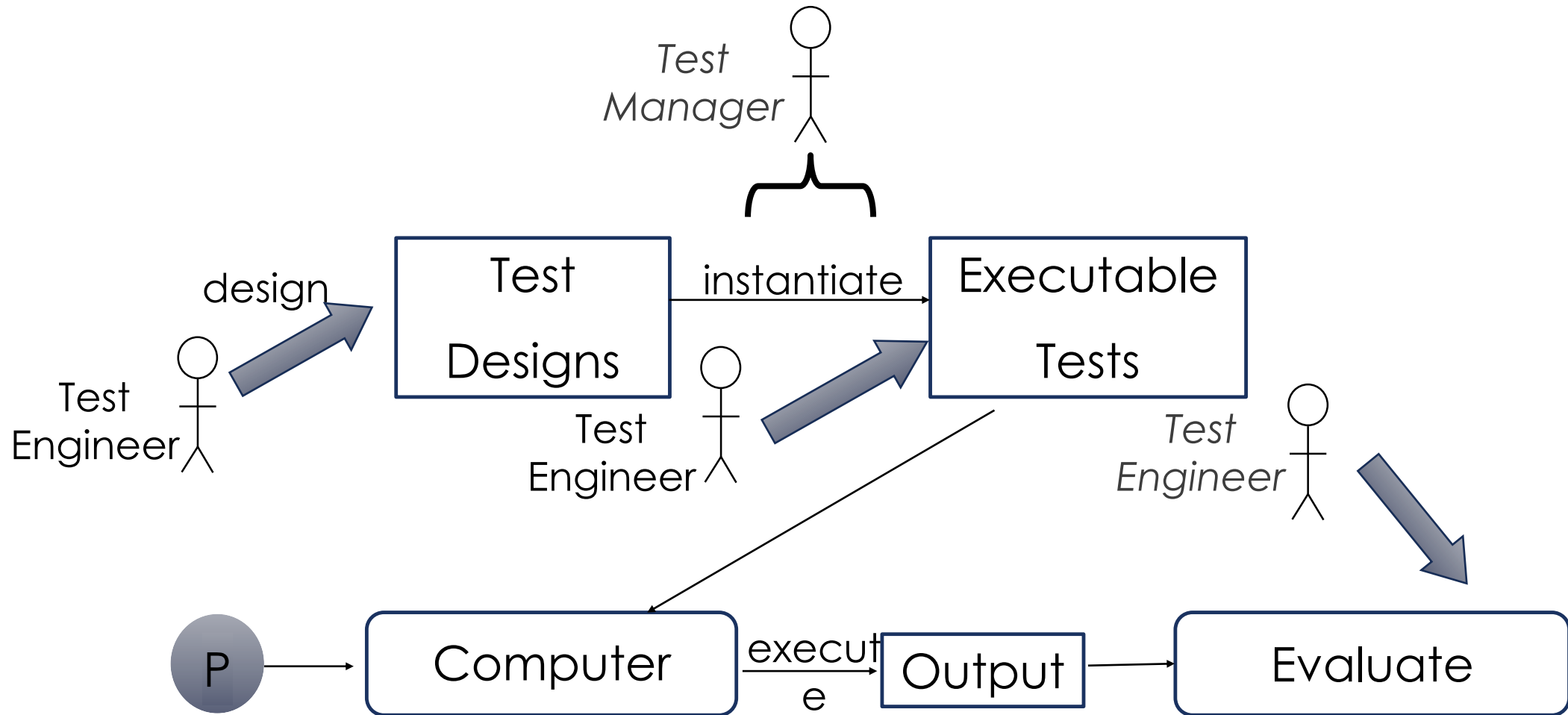
6. [3, 2, 5]

Find values ...

# 程序测试不同层次 (TESTING AT DIFFERENT LEVELS)



# 测试工程师任务 (TEST ENGINEER ACTIVITIES)



# 提 纲

1

Spectacular Software Failures

2

What Do We Do When We Test ?

3

Software Testing Terminology

4

测试人员职业发展与素质

5

总结

## 确认与验证 VALIDATION & VERIFICATION (*IEEE*)

- Validation : The process of evaluating software at the end of software development to ensure compliance with intended usage
- Verification : The process of determining whether the products of a given phase of the software development process fulfill the requirements established during the previous phase

# TESTING & DEBUGGING

- Testing : Finding inputs that cause the software to fail
- Debugging : The process of finding a fault given a failure



## IEEE Std 610.12 [IEEE90] 测试用例的定义如下：

测试用例是（A）一组输入即运行前提条件，和为某特定的目标而生成的预期结果，

- 例如：测试某一特定的程序路径或验证程序是否符合某特定需求。

测试用例是（B）一个文档，详细说明输入、期望输出，和为一测试项所准备一组的执行条件。

## 白盒测试

需要了解产品的内部工作，关注程序的结构和内部逻辑，  
根据程序的结构和内部逻辑设计用例。

### 常用的白盒测试技术有：

基本路径测试 (Basis path testing): 为测试控制流路径设计测试用例；

条件测试 (Condition testing): 为测试每个条件的结果而设计测试用例；

数据流测试 (Data flow testing): 为测试数据元素定义和引用关系而设计的测试用例。



# 黑盒测试

需要了解功能性的规格说明，关注对功能的需求，  
为测试系统的功能设计测试用例。

## 常用的黑盒测试技术有：

边界值分析(Boundary value analysis)：根据变量的边界值设计测试用例；

因果测试(casual-effect analysis)：根据触发—响应和输入—输出的关系设计测试用例；

等价划分(equivalence partitioning)：将输入、输出域划分成不相交的区域，根据这种划分设计测试用例。

# 软件测试类型



从四种不同的角度讲述测试类型：

开发阶段	测试技术	测试实施状态	测试实施主体
<ul style="list-style-type: none"><li>• 单元测试</li><li>• 集成测试</li><li>• 确认测试</li><li>• 系统测试</li><li>• 验收测试</li></ul>	<ul style="list-style-type: none"><li>• 白盒测试</li><li>• 黑盒测试</li><li>• 灰盒测试</li></ul>	<ul style="list-style-type: none"><li>• 静态测试</li><li>• 动态测试</li></ul>	<ul style="list-style-type: none"><li>• 开发方测试</li><li>• 用户测试</li><li>• 第三方测试</li></ul>



# 按照执行状态划分

## 静态测试：

- 指不运行程序，而通过人工或利用工具对程序和文档进行分析与检查。静态测试技术又称为静态分析技术，是对软件中的需求说明书、设计说明书、程序源代码等进行非运行的检查。
- 静态测试包括：走查，审查，符号执行等

## 动态测试：

- 指通过人工或利用工具运行程序进行检查，分析程序的执行状态和程序的外部表现。
- 单元测试、集成测试、确认测试，系统测试、验收测试、白盒测试、黑盒测试及灰盒测试等是指的动态测试。



# 软件质量衡量



## 时间方法

- 在正确的时间用正确的方法把个工作做正确 (Doing the right things right at the right time.)

## 应用标准

- 符合一些应用标准的要求，比如不同国家的用户不同的操作习惯和要求，项目工程中的可维护性、可测试性等要求。

## 符合规约

- 质量本身就是软件达到了最开始所设定的要求，而优美或精巧的表现技巧并不代表软件的高质量  
(Quality is defined as conformance to requirements, not as "goodness" or "elegance".)

## 客户的需要

- 作为软件测试这个行业，最重要的一件事就是从客户的需求出发，从客户的角度去看产品，客户会怎么去使用这个产品，使用过程中会遇到什么样的问题。

# 软件测试原理



所有的测试都应追溯到用户需求

测试计划的制定应先于测试的执行

帕累托法则适用于软件测试

软件测试应从“小规模”开始，然后扩展到“大规模”

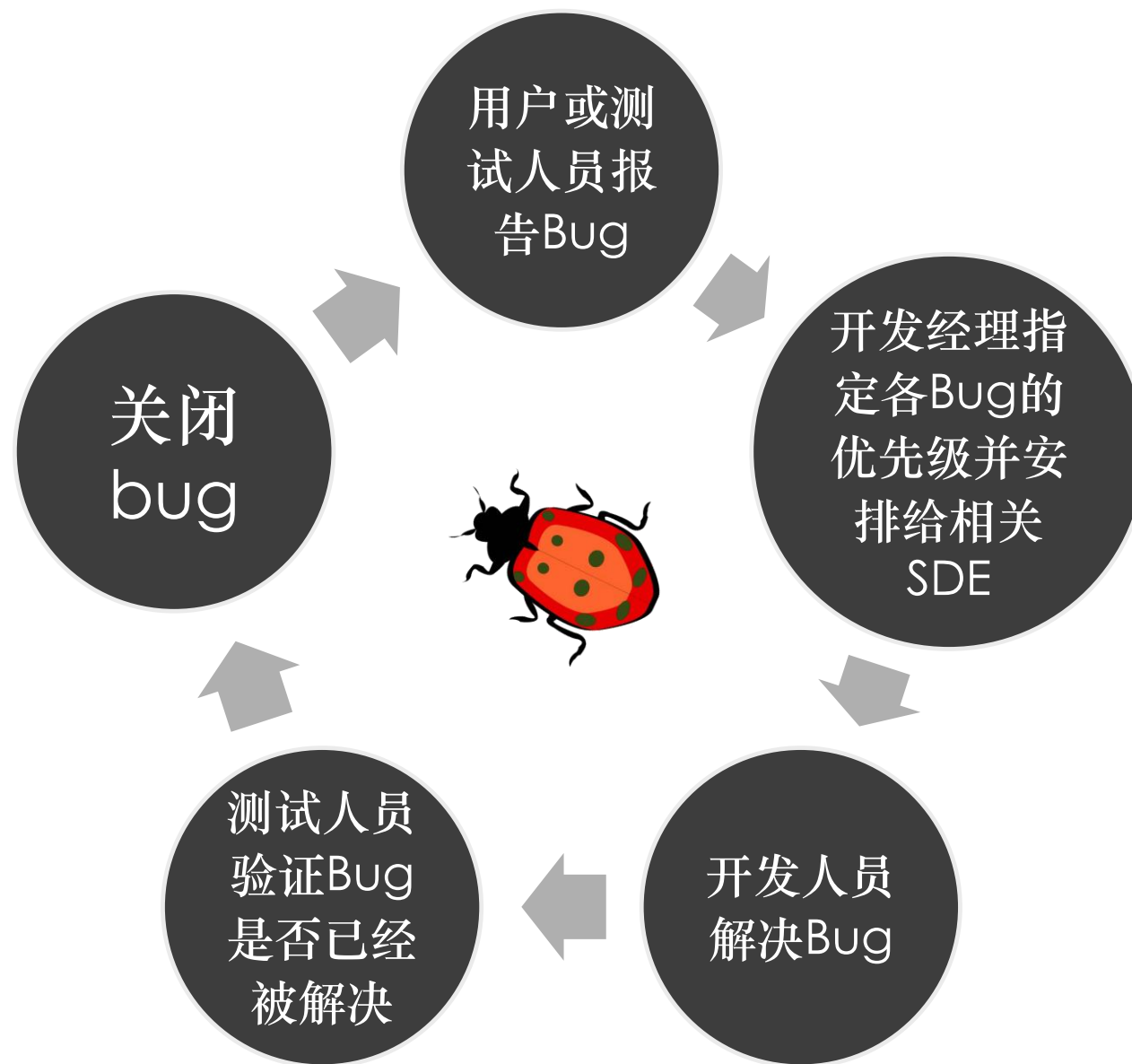
完全测试是不可能的

要是测试更为有效，测试应由独立的第三方进行

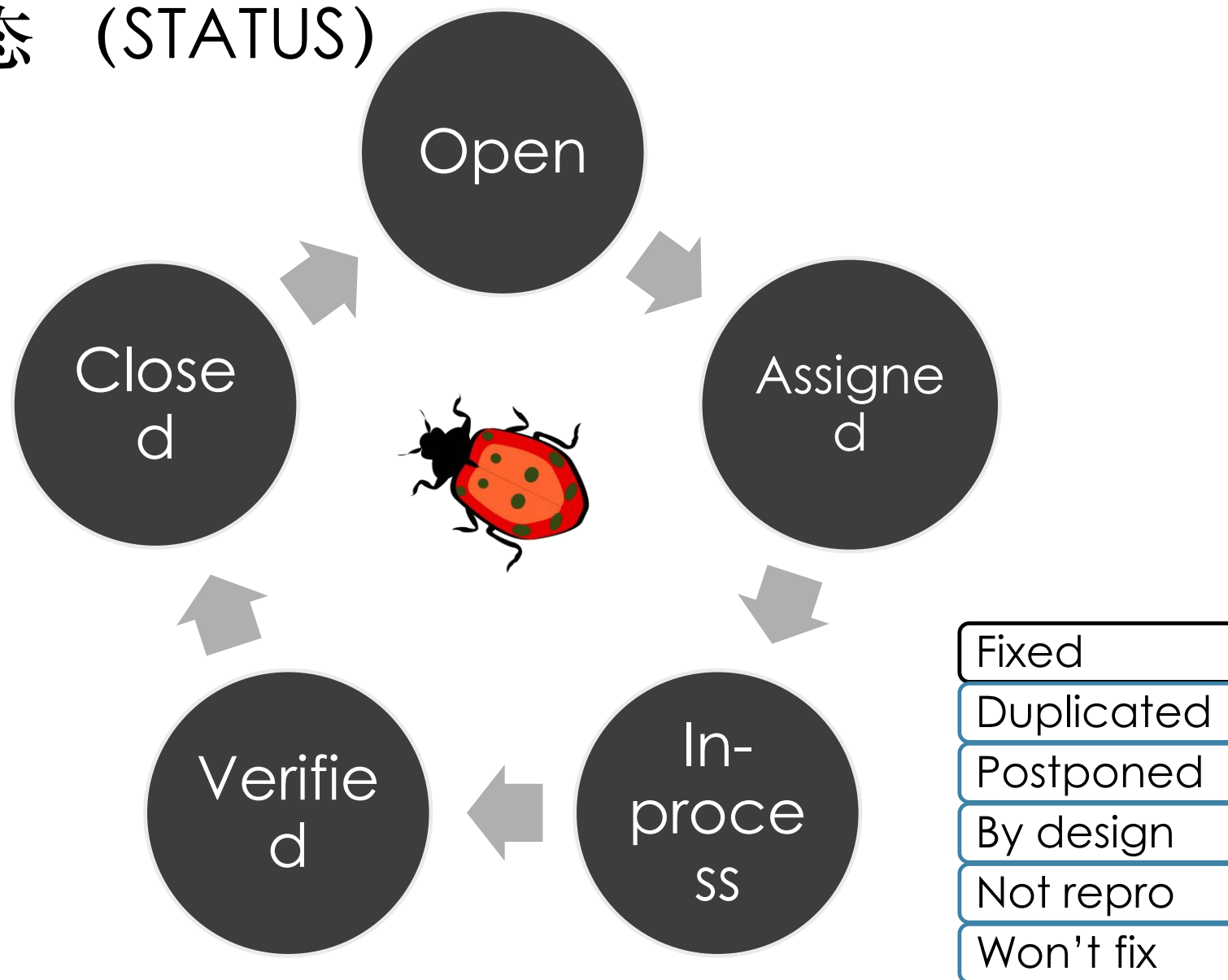
# 软件测试成熟度 (BEISER'S TESTING LEVELS BASED ON TEST PROCESS MATURITY)

- Level 0 : There's no difference between testing and debugging
- Level 1 : The purpose of testing is to show correctness
- Level 2 : The purpose of testing is to show that the software doesn't work
- Level 3 : The purpose of testing is not to prove anything specific, but to reduce the risk of using the software
- Level 4 : Testing is a mental discipline that helps all IT professionals develop higher quality software

## Bug 生命周期管理



## BUG 的五种不同状态 (STATUS)



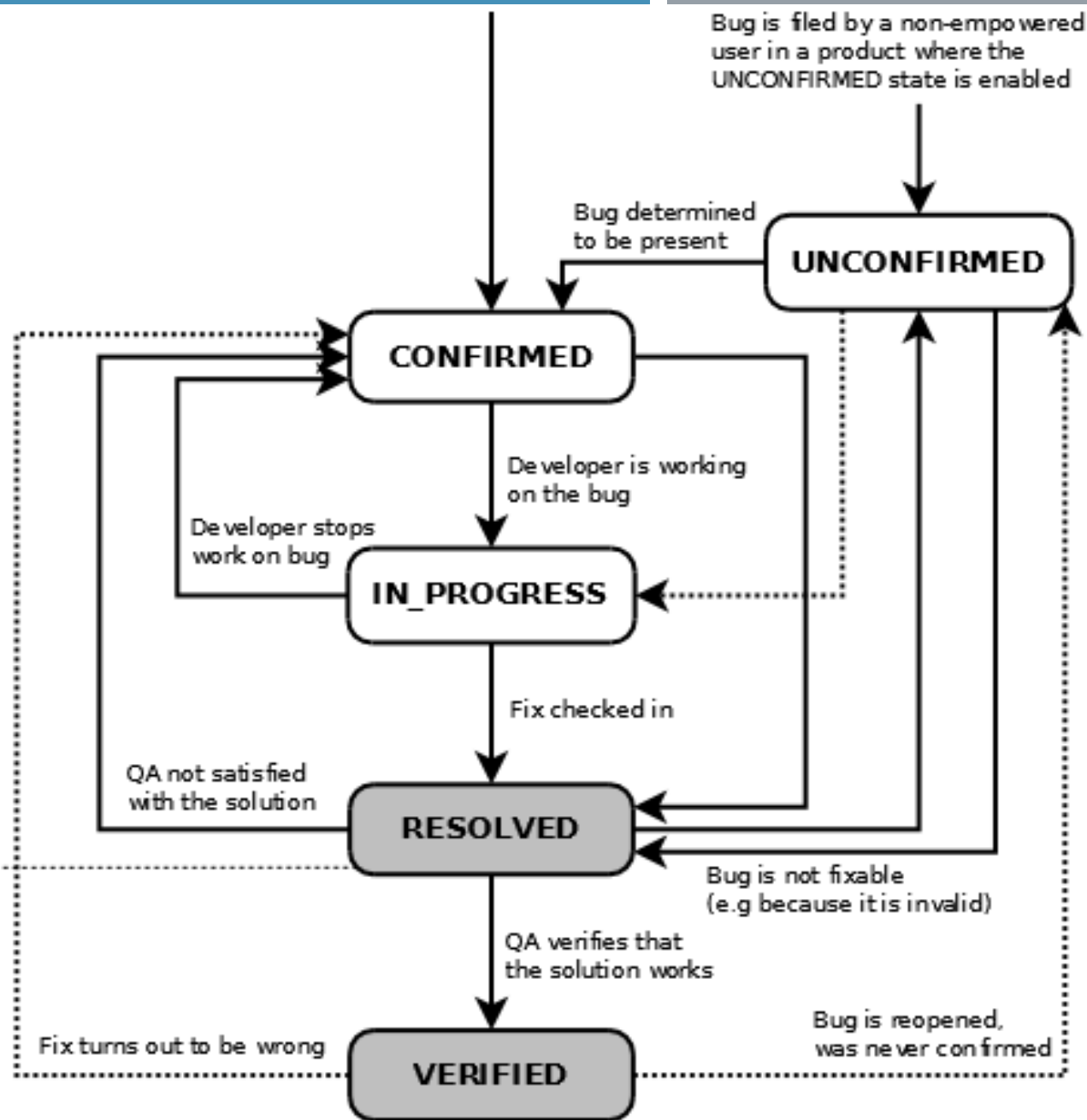


# 缺陷跟踪系统 Bugzilla



Bugzilla

Possible resolutions:  
FIXED  
DUPLICATE  
WONTFIX  
WORKSFORME  
INVALID





## BUG REPORTS

Bug subject/title

Software version

Priority and Severity

Report steps of test

Result

Expected result

Other information

# 提 纲

1

Spectacular Software Failures

2

What Do We Do When We Test ?

3

Software Testing Terminology

4

测试人员职业发展与素质

5

总结



# 优秀测试人员应该具备的态度

## 具有独立性

- 首先需要独立于开发者。为什么呢？一般来说，开发者有意识或无意识地倾向于偏袒他们的错误。

## 了解顾客

- 其次必须能从顾客的角度考虑问题。为什么呢？最终是用户来使用产品，他们要从中获益，所以一个优秀的测试人员应该为顾客着想。

## 测试所期望的功能性

- 这是测试的基本目的之一。一个优秀的测试人员应该测试每个所期望的功能性，以确保该产品就是顾客所要的。

## 测试所不期望的功能性

- 有时候叫做破坏（Break-it）测试（或Dirty testing）。在这个过程中，测试者有意地要造成编码失败，帮助发现一些特殊情况中可能出现的编码错误。

# 提 纲

1

Spectacular Software Failures

2

What Do We Do When We Test ?

3

Software Testing Terminology

4

测试人员职业发展与素质

5

总结

# 总结

测试是要发现语义的或逻辑的错误，而不是要发现语法的或符号的错误


软件测试是为了确认软件做了所期望的事情（Do the right thing），另一方面是确认软件以正确的方式来做了这个事件（Do it right）。

软件测试不仅是在测试软件产品的本身，而且还包括软件开发的过程。

如果一个软件产品开发完成之后发现了很多问题，这说明此软件开发过程很可能是有缺陷的。因此软件测试是保证整个软件开发过程是高质量的。

# 课后思考题

- 什么是软件测试？
- 软件测试过程或步骤？
- 什么是测试计划？为什么要做测试计划？
- 软件测试原理是什么？
- 如何进行缺陷管理？
- 优秀测试人员应该具备怎样的态度？
- 我为什么要学习软件测试？



# 附录



## LEVEL 0 THINKING

- Testing is the same as debugging
- Does not distinguish between incorrect behavior and mistakes in the program
- Does not help develop software that is reliable or safe

This is what we teach undergraduate CS  
majors

## LEVEL 1 THINKING

- Purpose is to show correctness
- Correctness is impossible to achieve
- What do we know if no failures?
  - Good software or bad tests?
- Test engineers have no:
  - Strict goal
  - Real stopping rule
  - Formal test technique
  - Test managers are **powerless**

This is what hardware engineers often

expect

## LEVEL 2 THINKING

- Purpose is to show failures
- Looking for failures is a negative activity
- Puts testers and developers into an adversarial relationship
- What if there are no failures?

This describes most software companies.  
How can we move to a team approach ??

## LEVEL 3 THINKING

- Testing can only show the presence of failures
- Whenever we use software, we incur some risk
- Risk may be small and consequences unimportant
- Risk may be great and the consequences catastrophic
- Testers and developers work together to reduce risk

This describes a few “enlightened” software  
companies

## LEVEL 4 THINKING

A mental discipline that increases quality

- Testing is only one way to increase quality
- Test engineers can become technical leaders of the project
- Primary responsibility to measure and improve software quality
- Their expertise should help the developers

This is the way “traditional” engineering  
works