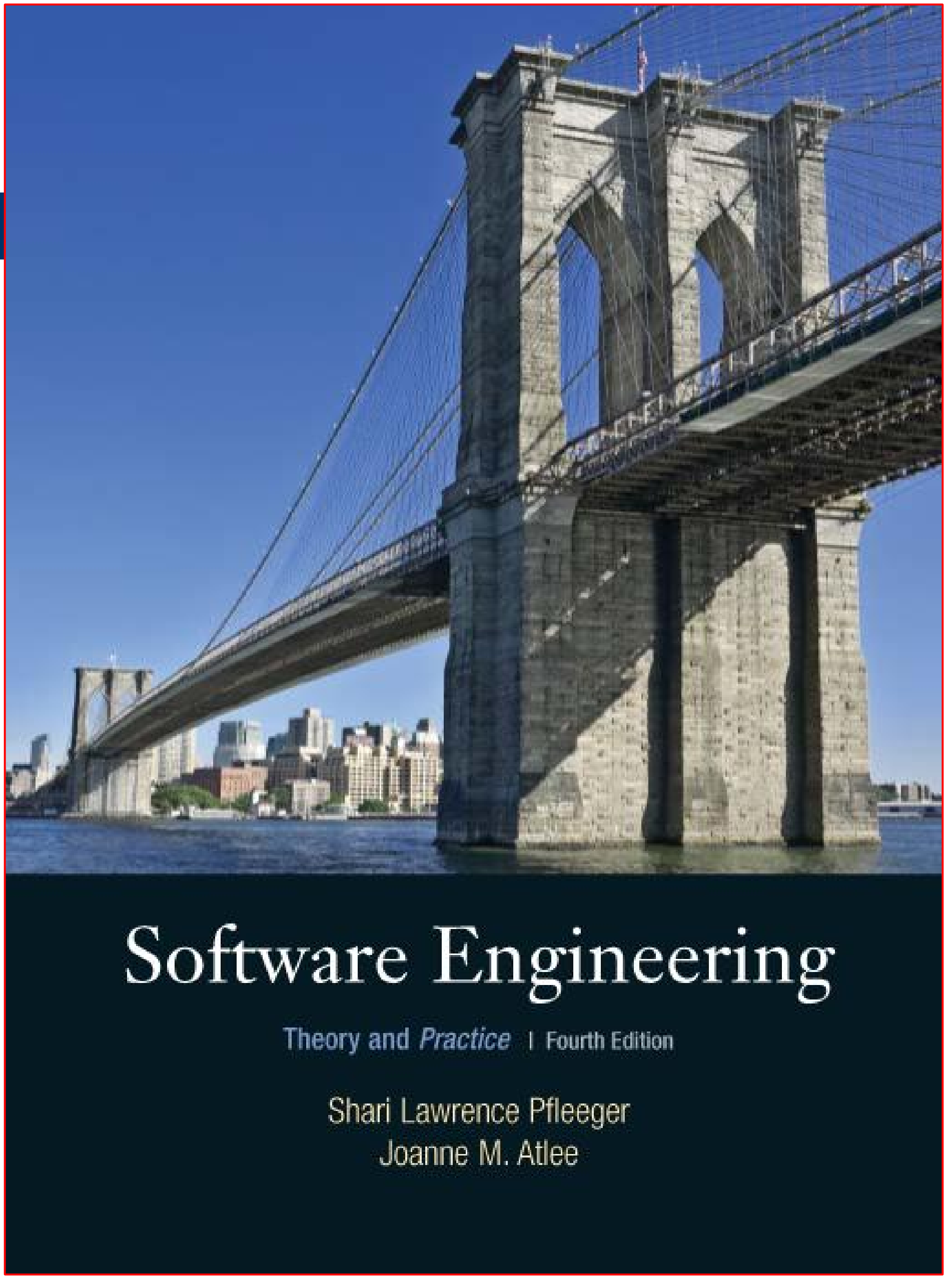
Chapter 8

Testing the Programs

Shari L. Pfleeger
Joann M. Atlee

4th Edition



理想情况下，一般认为程序员能够搞定一切，但理想很什么，现实很那个！

微 软 的 经 验 教 训

- 在微软的起步初期，微软的许多软件都出现了很多的Bug
  - 例如：在1981年，与IBM PC绑定的BASIC软件，用户使用“.1”除以10时就会出错，引起了大量用户的投诉。
  - 微软的高层领导觉得有必要引入更好的测试和质量控制方法，但是遭到很多开发人员和项目经理的反对，因为他们认为开发人员自己能测试产品，无需加入太多的人力。
  - 1984年，微软请Anderson咨询公司对其在苹果机上的电子表格软件进行测试，但是外部的测试没有能力进行的很全面，结果漏测的一个Bug，让微软为2万多个用户免费提供更新版本，损失达XXXX万美元。
- 在这以后，微软得出了一个结论：不能依赖开发人员测试，也不能依赖外部的测试，必须建立一个独立的测试部门。

Chapter 8 Testing the Programs

Note: A: several testing approaches $\longrightarrow$ delivering a quality system

B: focus on(in testing stage): finding faults

C: this chapter: unit testing and integrated testing

8.1 Software Faults and Failures

1. Introduction

① faults:

A: definition: the problem caused by errors

B: The cause of fault appearing:

X: the software itself (处理大量的系统状态, 处理复杂的公式、活动、数据及算法等)

Y: causes from customer and designer(uncertain、missing、impossible requirement, faults in design，etc.)

Z: other factor (项目的场景及规模因素, 众多参与者等)

Chapter 8 Testing the Programs

② failure:

A: definition: $\underset{\text{does}}{\text{the software}} \; \overset{\text{correspondent}}{\underset{\text{with}}{\cancel{\longleftrightarrow}}} \; \underset{\text{describe}}{\text{the requirement}}$

（软件的动作与需求描述的不相符，称之为失败(或失效)）

B: example: (P401-s1)

C: cause of failure: (P402-5 dots)

- Wrong requirement: not what the customer wants
- Missing requirement：缺失若干事件处理逻辑。
- Requirement impossible to implement：软件需求无法实现。
- Faulty design：曲解需求、设计本身问题等。
- Faulty code：代码在某些条件下有运行隐患。
- Improperly implemented design （实现不当）

Chapter 8 Testing the Programs

③ several concepts and problems (about testing)

A: purpose of testing:discover faults (not demonstrate correctness

B: fault identification: a process- P402)

C: fault correction: making chan o the system so the faults ar ved

这两者的出发点有根本性的差别:前者是假设系统有问题,而后者是假设系统没有问题

Chapter 8 Testing the Programs

D: discuss about fault

X: software faults is caused by ***human factors***

Y: “bug”----be refused to use or call by professionals

E: purpose of this chapter (P403)

----discussing techniques for ***minimize*** the occurrence of faults especially in the stage of coding .

(本章研究将程序代码本身故障的出现减到最少的技术)

----本章只涉及降低代码缺陷的测试技术，不涉及“专门针对需求的测试（专门的课题）、无缺陷需求获取技术，以及无缺陷设计技术”等规范。

Chapter 8 Testing the Programs

2. Types of faults

① reason for classifying faults (P403-s2) （深海潜伏、缺陷分类、创造特定条件、饱和检测等）

② types of faults

A: algorithmic fault（算法缺陷）

X: definition:算法的某些处理步骤或逻辑有问题，以至于软件部件对给定的输入数据无法产生正确的输出 (see text P403-s3)

Y: the ways to checking faults:

m: desk checking (static)

n : give classified input $\longrightarrow$ help to identify faults

Z: typical algorithmic faults(P403 6 dots)

Chapter 8 Testing the Programs

- Branching too soon
- Branching too late
- Testing for the wrong condition √
- Forgetting to initialize variable or set loop invariants √
- Forgetting to test for a particular condition √
- Comparing variables of inappropriate data types √
- Syntax faults

Chapter 8 Testing the Programs

B: computation and precision faults (计算和精度缺陷)

----computation error , or may result in less-than -acceptable precision （算法或公式在编程实现时出现错误或最终结果达不到精度要求）

cause: see text P403-rs3(举例:不同精度变量的混合运算, 浮点数据的不当使用，意料之外的数据截断，实现时操作次序不当，数据的对象化包装不当等等，都会导致精度的下降或失真。)

C: Documentation Faults（文档缺陷）

----the documentation does not match what the program actually does

(many of us tend to believe the document)

(that will be result in faults proliferation)

Chapter 8 Testing the Programs

D: Stress or overload Faults (in functionality)(过载缺陷)

---- when running program, data structures are filled past their specified capacity （so system can’t perform function in this moment）(DS----Queues, buffers, tables, arrays or list)

E: Boundary or Capacity Faults(in performance)(能力缺陷)

---- the system’s performance becomes unacceptable as a system activity reaches its specified limit (P404-s2)

Chapter 8 Testing the Programs

同能力缺陷的区别：这里指正常条件下性能需求不能满足，组装后测试不充分

F: timing faults（时序性缺陷）

----when coordination of seve

carefully defined time sequen broken (in real time system) (hard to ide and correct)

G: performance faults(性能缺陷)(example:response time)

H: recovery faults（恢复性缺陷）(一旦将缺陷分类，程序员很难考虑健壮性等全面功能性能特征。)

---- system can’t recover from running failure

I: hardware and system software faults（硬件和系统软件缺陷）

---- when the supplied hardware or software do not actually work according to the documented operating conditions and procedures (fail to work according to hardware or software introduction)

Chapter 8 Testing the Programs

J: Standards and Procedure Faults (代码的标准和规程缺陷)

---- the code does not follow the organizational standards and procedures

③ orthogonal defect classification

A: note:

X: faults exists in any ... g

Y: classification is hel ... ber of faults

若一个错误可以属于不止一个类, 则失去了度量的意义

B: definition: one fault belongs to one category

C: two types of defect X: fault of omission （忘记赋值）

Y: fault of commission （赋错值）

Chapter 8 Testing the Programs

D: example:

orthogonal fault classification of IBM ---- table8.1

fault classification of HP---- siderbar8.1 , fig8.1

three descriptors: (siderbar8.1 , fig8.1)

X: origin（起源）

Y: type（类型）

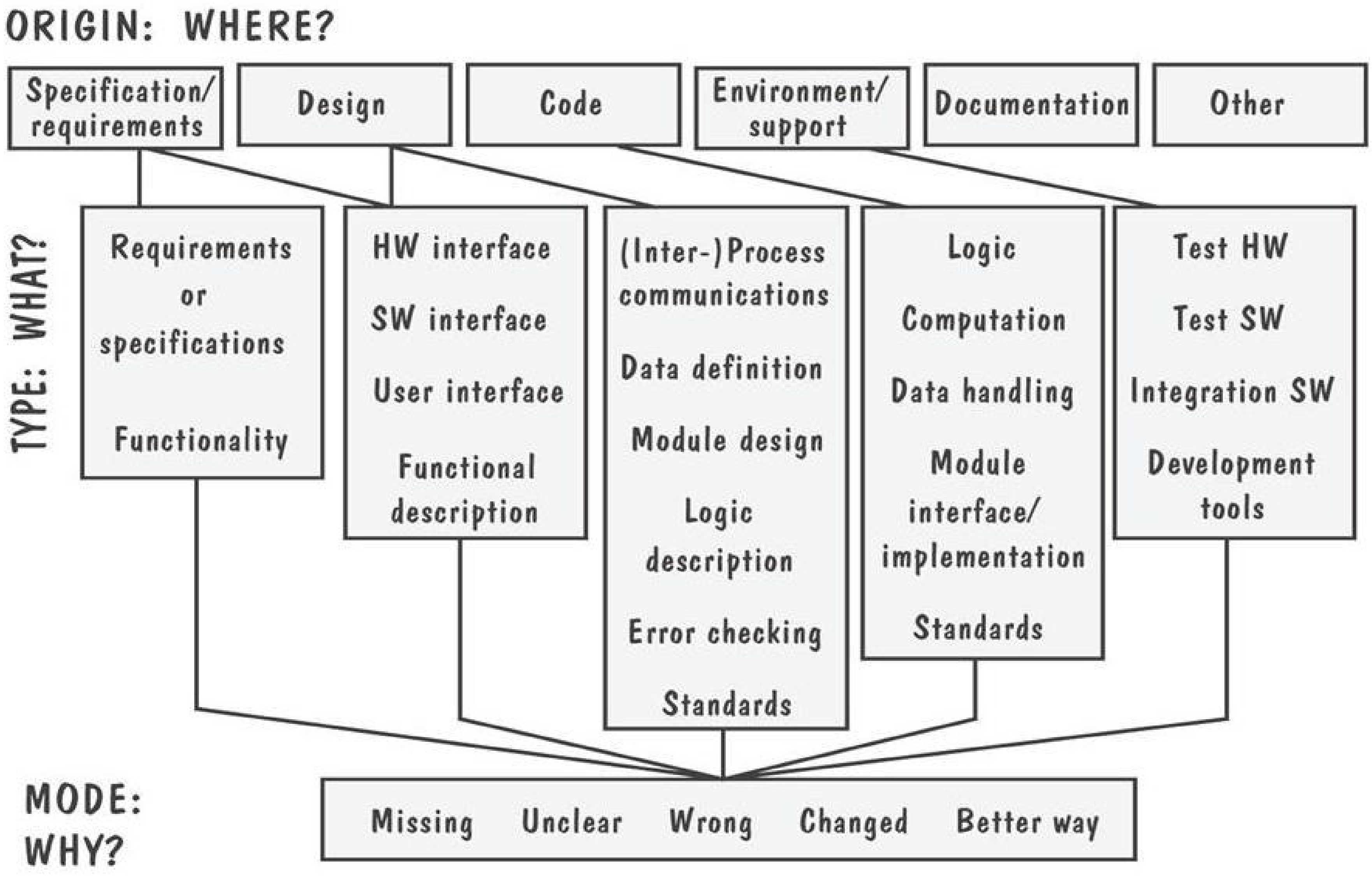
Z: mode （模式）

fig8.2: percentage of the faults for one Hewlett-Packard division

Chapter 8 Testing the Programs

Fault Type	Meaning
Function	Fault that affects capability, end-user interface, product interface with hardware architecture, or global data structure
Interface	Fault in interacting with other component or drivers via calls, macros, control, blocks or parameter lists
Checking	Fault in program logic that fails to validate data and values properly before they are used
Assignment	Fault in data structure or code block initialization
Timing/serialization	Fault in timing of shared and real-time resources
Build/package/merge	Fault that occurs because of problems in repositories management changes, or version control
Documentation	Fault that affects publications and maintenance notes
Algorithm	Fault involving efficiency or correctness of algorithm or data structure but not design

Chapter 8 Testing the Programs



Chapter 8 Testing the Programs

8.2 Testing Issues（有关测试的若干问题）

About Tests: different types、subsystems、purposes

(测试贯穿全过程，各个阶段考虑的测试问题不同，有的基于阶段划分，有的基于类型划分)

1. Test organization（测试的组织）

① several stages（几个阶段）

A: unit test: verifies the component functions (according to the program design)

B: integration test: verifies the system components work together (by system design and program design)

C: function test: check function by <SRS>

D: performance test: check performance by <SRS>

Chapter 8 Testing the Programs

E: acceptance test: check the customer's requirement definition

F: installation test: check the system in actual environment

② relationship among the testing steps

A: relationship: Fig8.3

B: this chapter: focus on unit and integration testing

C: chapter 9: system test = function test + performance test + acceptance test + installation test

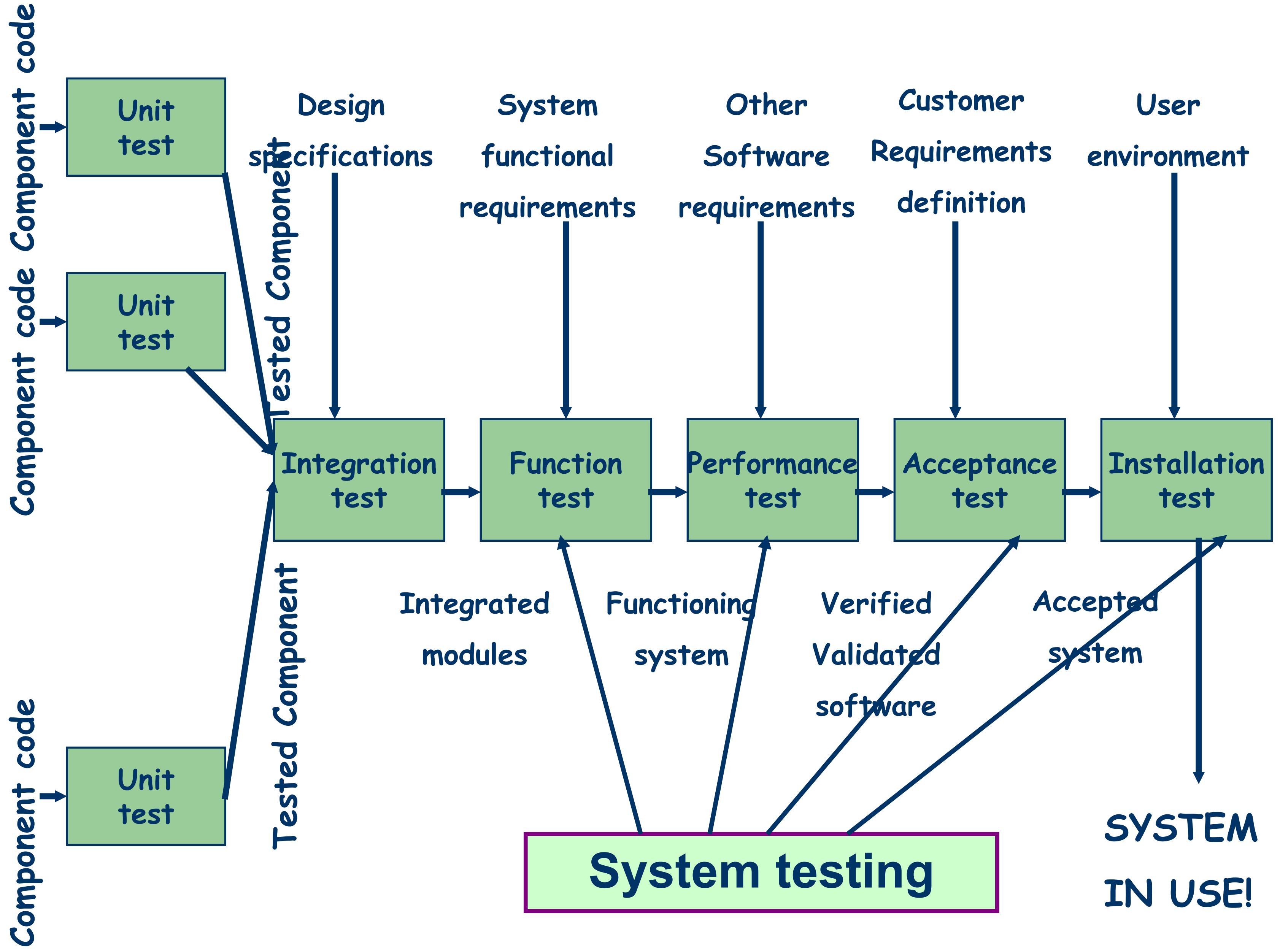


Fig 8.3 Testing steps.

Chapter 8 Testing the Programs

2. Attitudes toward testing（测试的态度）

① new programmer

----not accustomed to viewing testing as a discovery process (they only want to show the correctness, design skill and personal ability)

② customer:

----interested in being sure that the system works properly under all conditions (sometime it does not correspond the reality)

③ right attitude

---- just like egoless programming (P409-s3)
(A: view components as a part of the large system, not as property of those who wrote them
B: 不能将被测试的程序仅仅看做是否满足了解决方案，而同时应该考虑问题本身，即：有权怀疑一切！)

Chapter 8 Testing the Programs

3. Who performs the tests？（测试的人员）

① focus on: independent test team （Why？)

② reasons（of setting independent test team）

A: avoid conflict (between personal responsibility and the need to discover faults)

B: there are many choices to introduce faults (specify requirement and solution, realize algorithm, write document , etc.)

C: independent test team can participate in reviews and other test activities, work concurrently with coders

Chapter 8 Testing the Programs

4. Views of the test objects（测试的观点/方法）

① <u>black box（黑盒）</u>：

A: the <u>contents/structures are unknown</u>, only <u>test the functionality</u> of the testing object. That is , the testing feed input to the black box and note what output is produced (测试人员在完全不了解程序内部的逻辑结构和内部特性的情况下，只依据程序的需求规格及设计说明，检查程序的功能是否符合它的功能说明。)

(the component function is the basis of testing)

(备注：1、测试时应该考虑让被测模块完成一切应做的事情, 拒绝一切不应做的事情。2、黑盒测试的参考文档是系统需求、主要文档是系统设计和程序设计阶段文档。若是可重用部件，则是类似系统)

Chapter 8 Testing the Programs

B: advantage: is free of the constraints imposed by the internal structure and logic of the the test object, only **use representative test cases to finish the test .**

C: disadvantage: not always possible to run a complete test cases in this manner (理论上如此，见下页)

D: example: $ax^2+bx+c=0$ (3 inputs, 2 outputs)

$$\left\{\begin{array}{l} a: +, -, 0 \\ b: +, -, 0 \\ c: +, -, 0 \end{array}\right. \Rightarrow 3^3=27 \text{ (digital combination)}$$

others: round-off error（precision）

incompatible data type (‘a’,’m’)

黑盒测试

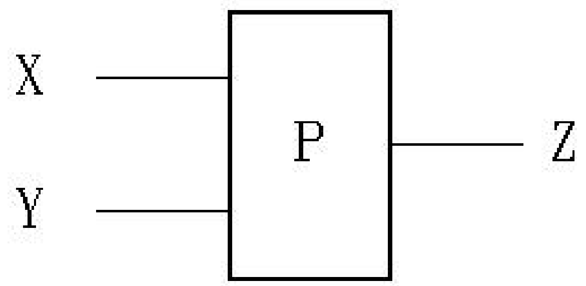
假设一个程序P有输入量X和Y及输出量Z。在字长为32位的计算机上运行。若X、Y取整数，按黑盒方法进行穷举测试：

可能采用的测试数据组：

$$2^{32} \times 2^{32} = 2^{64}$$

如果测试一 组数据需要1毫秒，一年工作365× 24小时，完成所有测试需5亿年。

因此，我们只能在大量可能的数据中，选取其中一部分有代表性的数据作为测试用例。



Chapter 8 Testing the Programs

E: another example: “federal income tax” (P410)

----not suitable in using “black box method”

② <u>white box（白盒）</u>:

A: definition: use the <u>structure</u> of the test object to test in different ways （以测试对象的内部结构为基本依据，手工或自动的展开各种测试。）

B: advantage: detailed testing for a model

C: disadvantage: may be impractical （全路径不可能）

D: example1: white box --- Fig8.4

m(100000) ∗ n(100000)=10 billions (logic paths)

$$\left\{\begin{array}{l} \text{I: } <n, =n, >n. \\ \text{J: } <m, =m, >m. \end{array}\right\}$$ 3X3=9(test cases)

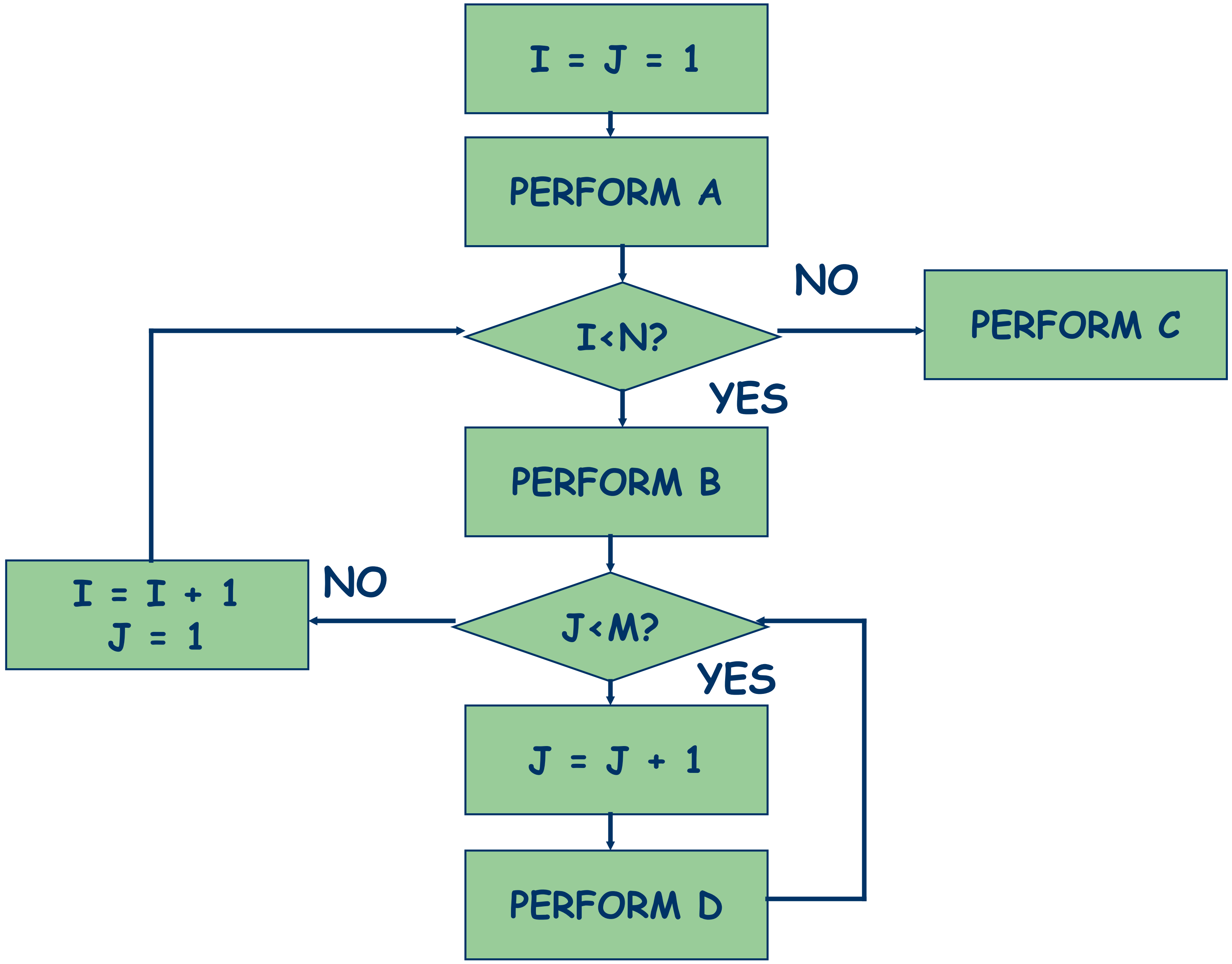


Fig 8.4 Example logic structure

Chapter 8 Testing the Programs

example2: $ax^2+bx+c=0$ （此处显示了白盒的优势）

b^2-4ac : +, -, 0 $\longrightarrow$ 3 test cases

③注: 制定软件测试策略得根据实际情况,主要是基于数据, 结构, 功能或其他规则；而决定如何进行测试时, 若条件允许，则很多时候不必把黑盒测试和白盒测试截然分开.（所有文档齐全时，不可以太矫情。）

④ several factors for choice of test case or method

A: the number of possible logical paths （可能的路径）

B: the nature/intend of the input data （数据代表性）

C: the amount of computation involved （计算量）

D: the complexity of the algorithms （算法复杂度）

Chapter 8 Testing the Programs

8.3 Unit Testing

steps: A: examine the code (静态检查代码，与需求及设计比较)

B: compile the code (compiling and debugging)

C: develop test cases and do testing

1. Examining the Code（检查代码）(面向提交)

① code review: (by an objective group of experts,

(代码复审)   review the code and its document for misunderstanding , inconsistencies, performance and other faults)

② two types of code review

A: code walkthrough（代码走查） (informal) （只标注）

Chapter 8 Testing the Programs

B: code inspection（代码检查） (formal review)

----the review team check code and document according a prepared list of concerns(关注点列表)

example—(检查需求与设计的所有内容) review data type and structure, algorithms, comments, interfaces, performance, etc.

steps: { meeting
checking and marking
meeting—discussing the results

③ success of code reviews（代码复审的成功之处）

A: success in detecting faults and be adopted by almost all organizations .

B: early finding faults ,and lower cost (in fact, review is for all processes: requirement, design, coding, etc.)

Chapter 8 Testing the Programs

④ statistics in review

----Fagan's statistic result ( )

----table 8.2 typical preparation times and meeting times

----table 8.3 Jone's result: fault discovery rates

(rate=number of faults / number of thousands of lines of code) (千行代码缺陷率)

2. Testing Program Components (测试程序模块)

① choosing test cases

A: test case: input data choosing for testing a program (以测试程序为目的而挑选的输入数据,包括对应的期望结果)

Chapter 8 Testing the Programs

- 接上页：测试用例的给出与测试报告的构成。
  ---------一般二次修改就能达到目的。
  ---------内容逐渐增长的过程，后补内容较易识别。

输入数据组合	预期结果	实际结果1	原因及修改内容	实际结果2	。。。

Chapter 8 Testing the Programs

B: 程序测试的基本步骤:

- **X**: 确定测试的目标和计划（确定各种测试类型、开始与结束时间及条件等）
- **Y**: 选择测试用例
  - ----黑盒法
    - ----等价分类法等等.
  - ----白盒法
    - ----各种覆盖方法等.
- **Z**: 执行测试计划

Chapter 8 Testing the Programs

C: the methods of giving test cases

X: black box: according to "SRS" and other Docus .

Y: white box: according to internal logic of a module

Z: example: $aX^2+bX+c=0$

black box: first method:
$$\left.\begin{array}{r} a: +, -, 0 \\ b, +, -, 0 \\ c, +, -, 0 \end{array}\right\} 3^3=27\text{(test cases)}$$

second method:
$$\left.\begin{array}{r} a>b>c \\ b>c>a \\ c>b>a \end{array}\right\} 3\text{(test cases)}$$

white box: $b^2- 4ac$: >0, <0, ==0. 3(test cases)

Chapter 8 Testing the Programs

other method: a,b,c=“F” (nonnumeric data)

W: criteria for classify the test data （黑盒等价分类法

(注：该分类原则只适用于black box法) (P422—1,2,3)

(1). 每一个可能的输入数据（测试用例）必属于某一分类。（此时的分类一般指的是输入域）

(2). 各个分类之间没有交集。

(3). 每个类的特定测试用例可以代表这个类。

（有时我们可以适当放松这个约束，即：如果一个数据元素成员属于一个分类并经运行显示了一个错误，则该类的任意元素成员运行显示同样错误的概率较高(接近100%)。）

（弱一般等价类，软件失效是基于单缺陷假设。）

Chapter 8 Testing the Programs

有时称为灰盒测试，既关注黑盒的输出合理性，又关注白盒的细致性

U: drawbacks of black box and white box

black box: uncertainty (in finding a particular error)
（黑盒法以SRS及SAD等文档作为依据，有一定的盲目性和不确定性，不可能揭示所有的错误）

white box: pay too much effort but still difficult in giving test case(when internal logic is too complex)
（该法以模块内部逻辑为依据，当内部逻辑过于复杂时，则不能给出好的或合适的测试用例）

C: combining black-and white-box testing to generate test data（黑盒与白盒法相结合产生测试用例）

X: method： black box + white box + other method

Y: example: ----“input positive value”
(black box + other method)   (8.3.3—7 dots)

Chapter 8 Testing the Programs

② test thoroughness（测试的彻底性）(about white box)

A: three methods:

X: statement (coverage) testing（语句(覆盖)测试）

----给出的测试用例能使模块中每一语句至少执行一遍

Y: branch testing（分支测试）

Z: path testing（路径测试）

B: example—Fig8.7

statement: $\begin{cases} x>k \\ result>0 \end{cases}$ 1-2-3-4-5-6-7 (1 test case)

branch: 1-2-3-4-5-6-7 and 1-2-4-5-6-1 (2 test cases)

path: 1-2-3-4-5-6-7 ＋ 1-2-3-4-5-6-1 ＋ 1-2-4-5-6-7

＋ 1-2-4-5-6-1   4 paths---(4 test cases)

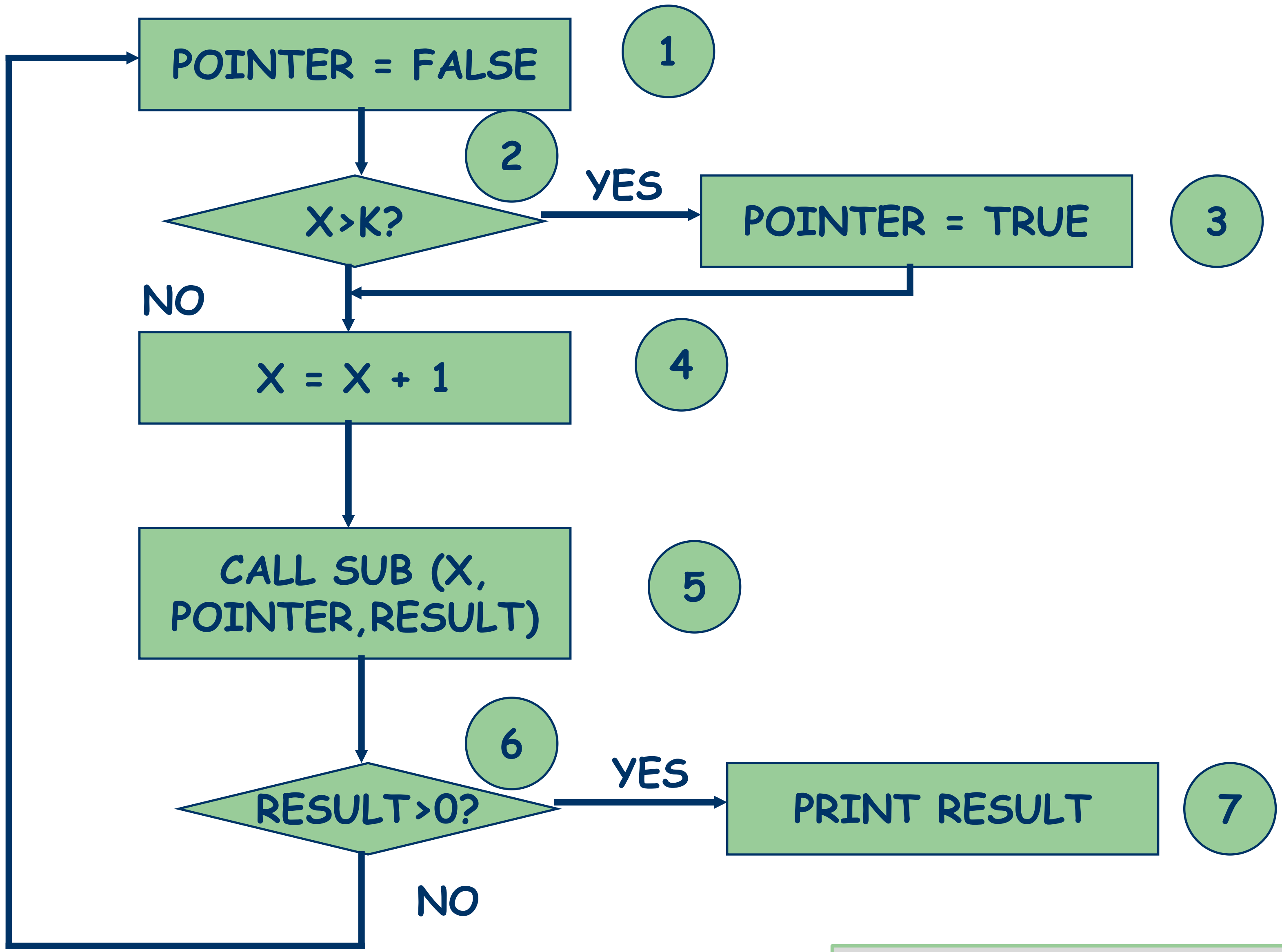


Fig 8.7 Logic flow.

结论：一般来说，语句测试使用的测试用例最少，分支测试较多，路径测试最多。

Chapter 8 Testing the Programs

③ **supplemental material (*)**

- **A: black box**
  - **X: equivalence partitioning**
  - **Y: boundary value analysis**
  - **Z: error guessing**
  - **U: cause-effect diagram**
  - **example:**
- **B: white box**
  - **X: 4 logical coverage methods :**
    statement, branch, condition,
    condition combination

Chapter 8 Testing the Programs

Y: 3 path coverage methods

node:

edge:

path:

Z: the difference between logical coverage and path coverage

C: example

Chapter 8 Testing the Programs

Sidebar 8.4
Fault Discovery Efficiency at Contel IPC

- 17.3% during inspections of the system design
- 19.1% during component design inspection
- **15.1% during code inspection**
- **29.4% during integration testing**
- 16.6% during system and regression testing
- 0.1% after the system was placed in the field

Chapter 8 Testing the Programs

8.4 Integration Testing（集成测试）

A:goal----**working system**(can perform basic functions)

B: focus on: importance of the way/strategy/plan of combining and testing the components

知道此时组长有多紧张吗？

w: when a failure occurs, we should have some idea to guess what the reason is by the way of units integration.

X: the strategy affects the integration time and the order of coding

Y: influence to the choice of test cases

Z: influence to the cost and thoroughness of the testing

Chapter 8 Testing the Programs

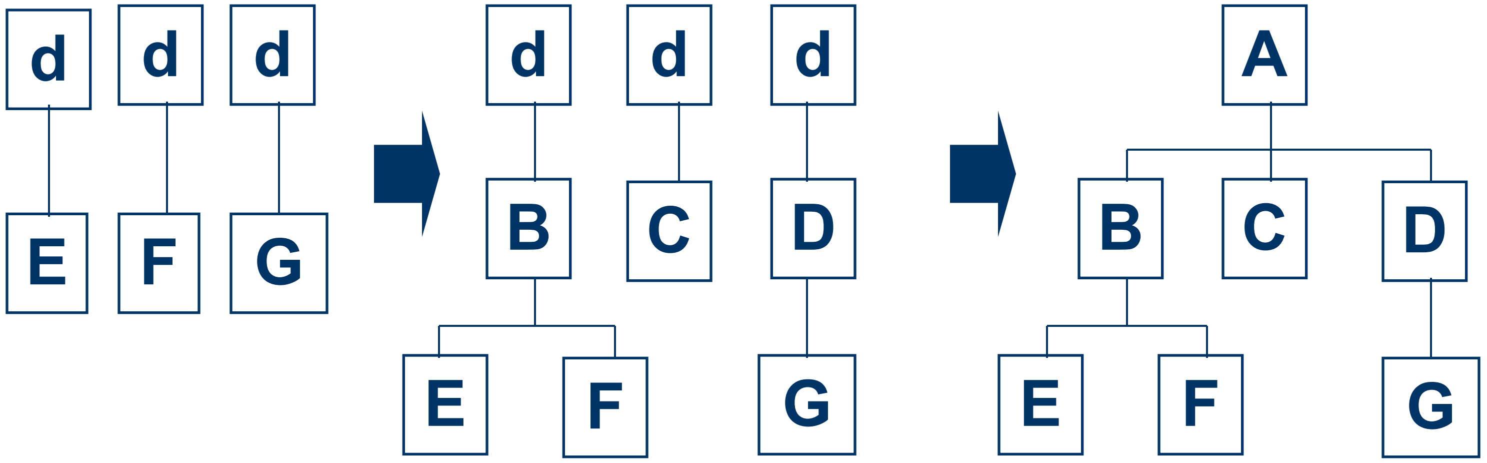
1. Bottom-Up Integration（由底向上集成测试）

① meaning: (P426)从模块结构图的最低层开始，由下而上按调用关系逐步添加新模块，组成子系统并分别测试，直到全部模块组装完毕。

② example:

A: component driver (驱动模块: 代替上级模块传递测试用例的程序)

B: fig-8.8, fig-8.9 bottom-up testing procedure:



Chapter 8 Testing the Programs

③ advantage/feature:

A: easy to generate a test case (每次底层都是真实模块)

B: be suitable for OO approach （每次加入的是经过测试的对象，也符合消息的传递方式。）

C: be suitable when many of the low level components are general-purpose utility routines that are invoked often by others.

④ drawback: the major faults in high level units can't correct as soon as possible

⑤ 注：图8.8部件层次结构的含义: 指模块之间一个对象调用另一个对象的方法，或一个部件传递参数给另一个部件，或部件间传递消息等。

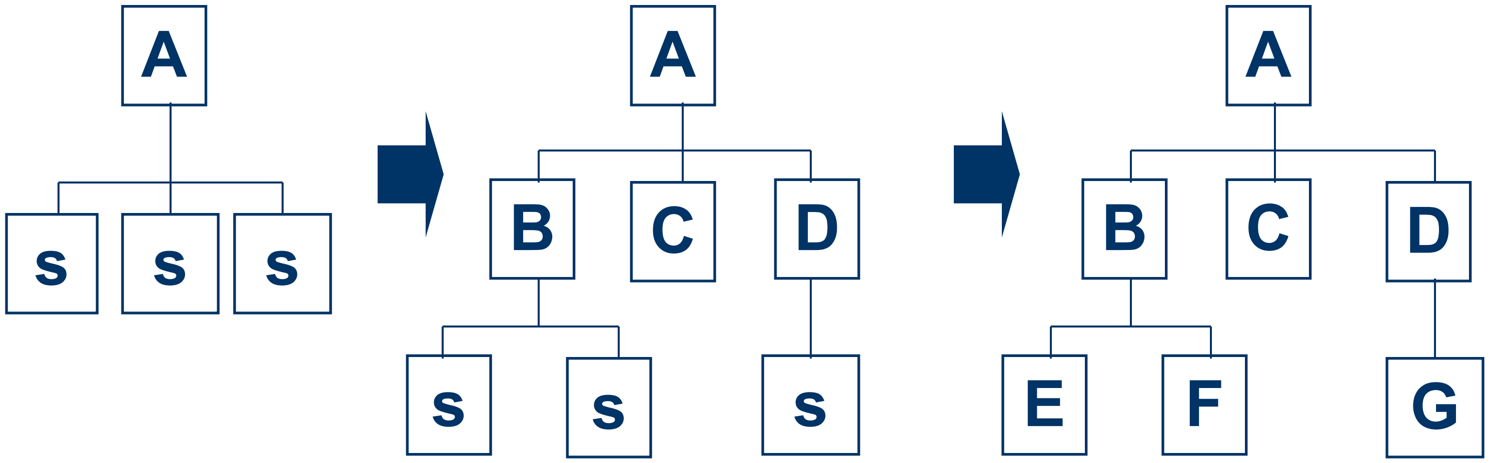
2. Top-Down Integration（自顶向下集成测试）

① meaning: (P428)从顶层控制组件开始，首先对它本身进行测试，然后将被测组件调用的下级组件组合起来，对这个更大的子系统进行测试，反复采用这种组装方法，直到包含了所有组件为止。

② example:

A: stub（桩模块：代替下级模块的仿真程序）

B: fig-8.8, fig-8.10 top-down testing procedure:



Chapter 8 Testing the Programs

③ advantage: the upper models have more testing choice , so the major question will be found in early stage

④ drawback:

A: generating test cases can be difficult (P429)

B: a very large number of stubs may be required (modified top-down testing—Fig8.11)

3. Big-bang Integration（莽撞测试）

① meaning:unit testing$\longrightarrow$integrate all units in one time

② drawback:

A: stubs+drivers (when unit test a module)

Chapter 8 Testing the Programs

B: its difficult to find the cause of any failure

C: interface faults cannot be distinguished easily

4. Sandwich Integration（混合方式测试）

① meaning: A: three layers:

X: target layer（middle layer）

Y: upper layer

Z: lower layer

B: strategy

X: upper layer— top-down testing

Y: lower layer --- bottom-up testing

Z: target layer --- testing(stub+driver)

W: integration of all three layers。

Chapter 8 Testing the Programs

C: example: Fig8.13

② feature: A: full testing for top level

B: easy to generate test case for lower level

5. Comparison of Integration Strategies

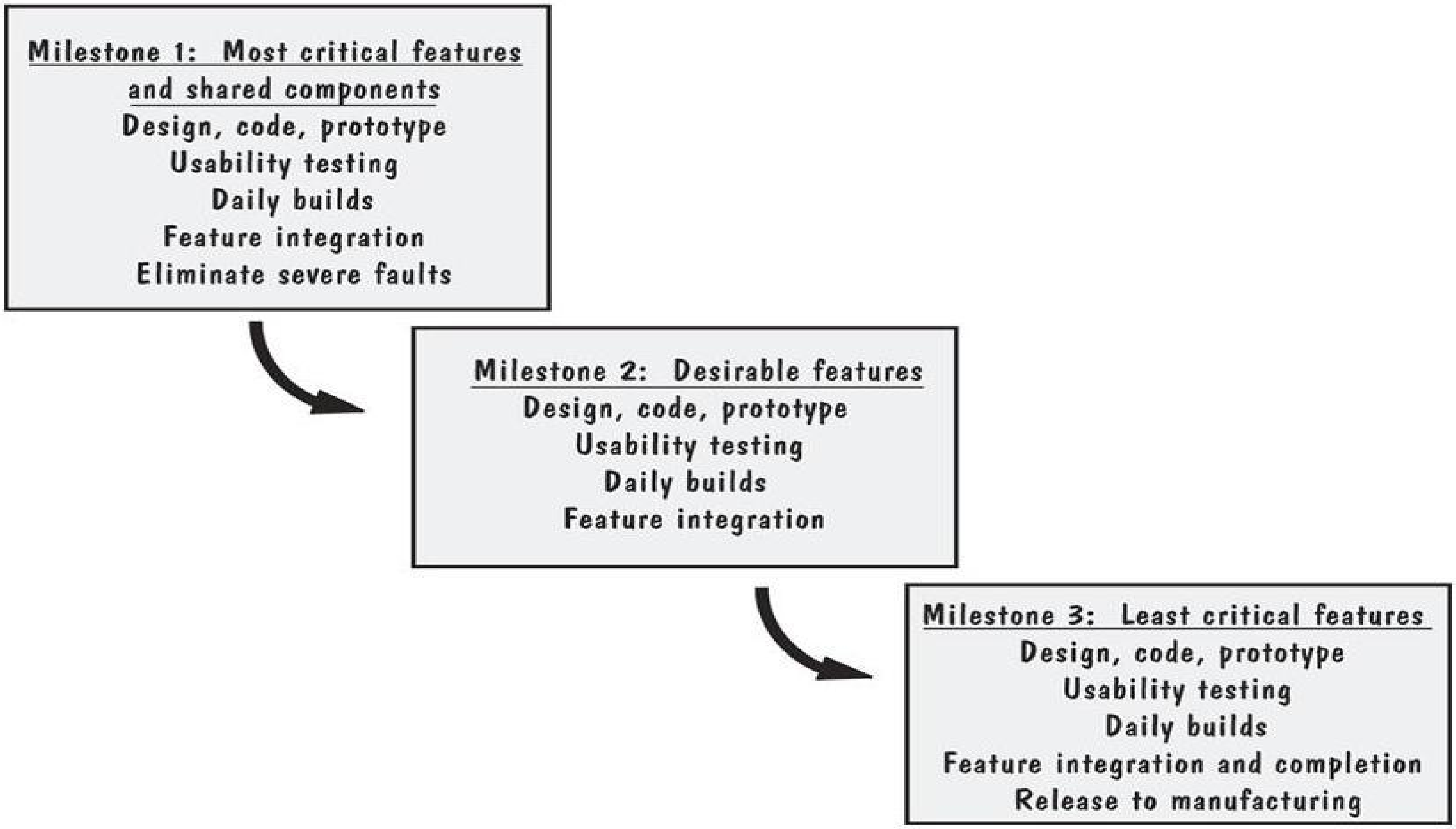
① focus on: choosing an integration strategy depends not only on system characteristics, but also on customer expectations . (P430) (so, test strategy should be discussed with customer in the early stage of a project)

② table8.7—comparison of integration strategies

sidebar8.5 + Fig8.15 ---- Microsoft’s strategy： driven by market pressures。

Sidebar 8.5 Builds at Microsoft

- The feature teams synchronize their work by building the product and finding and fixing faults on a daily basis



Chapter 8 Testing the Programs

6. Integration Test practical Strategies (集成测试中的具体措施)

① 深度优先策略：首先集成一个主控路径下的所有模块（无论是由底向上还是自顶向下），然后选择其他路径进行集成（路径的次序选择有任意性），直到全部模块集成完毕。

② 广度优先策略：将软件结构中每一层次所有直接隶属于上层的模块集成起来进行测试（无论是由底向上还是自顶向下），然后进行另一个层次的测试，直到全部模块集成完毕。

③按照由底向上还是自顶向下的分类，采用相应的驱动模块或桩模块。

Chapter 8 Testing the Programs

8.5 Testing Object-Oriented Systems (OO测试)

Focus on: OO testing has special characteristics

----should take several additional steps

1. Testing the Code

① several problem about OO code testing (P433--3 dots)

----OO code 存在于对象的方法中,而完成软件任务需要界面类、控制类、实体类等的交互，因而OO测试不仅仅局限于对方法的测试，而是比较复杂和广泛。

----注意的几个问题 (提问)（暗含需求方面的测试）

---- 意外的输入能否产生意外的结果。(路径应该唯一)

---- 意料之内的输入能否产生意料之外的结果。(结果唯一)

---- 在选择有用的用例时，有没有考虑不周的情形？

有的团队首先根据需求来测试，也算合理

Chapter 8 Testing the Programs

② several aspects about the faults of objects (P433-434 5 dots)

note: 关于对象和类的检查不会都在设计阶段的静态检查过程中完成, 有很多时候是在测试阶段才发现其不合理性(特别是OO), 然后又需要调整设计 .

----发现不对称或不合理的关联或继承。（需要调整设计）

----发现一个类中存在关系不密切的属性和动作。(合理性)

----一个类担任两个以上角色。（目标明确性问题）

----操作目标不明确。(例如: 某个类没有合适的方法等等)

----发现两个同名的或同目的的关联关系等等。

Chapter 8 Testing the Programs

③ **message : making special testing**

----传统方法可很好的应用于功能测试, 但没有考虑测试类所需的对象状态及其交互协作关系, 而这涉及到消息 .

----OO方法开发的系统, 消息是软件任务完成的载体和线索, 需要对消息进行较全面的测试, 同时也完成了对对象的状态测试 .(例如：响应消息的准确性需要测试)

2. Difference between OO and traditional Testing

① **focus on: retest all methods when we add new subclass in a way of inheritance**

(换句话说: 封装时被充分测试的程序在实现继承组合时可能没有被充分测试)（添加子类时）

Chapter 8 Testing the Programs

② difference between OO and traditional testing

A: traditional testing

system changed $\longrightarrow$ original test case+new test case

(ordinary testing)(回归测试)

B: OO testing: retest the overriding subclass, and may use different test cases（导出类具有特定功能）

C: OO testing: unit testing ---- easy （对象小粒度，构件内的复杂度可能转移到接口上。）

integration testing ---- extensive(涉及接口)

(Fig8.16---- show the easier and harder parts)

D: difference (with traditional testing)

Fig8.17---- four different aspects

两者说的是同一个问题

③ difficulty in OO testing（OO测试难处：4 dots）

Chapter 8 Testing the Programs

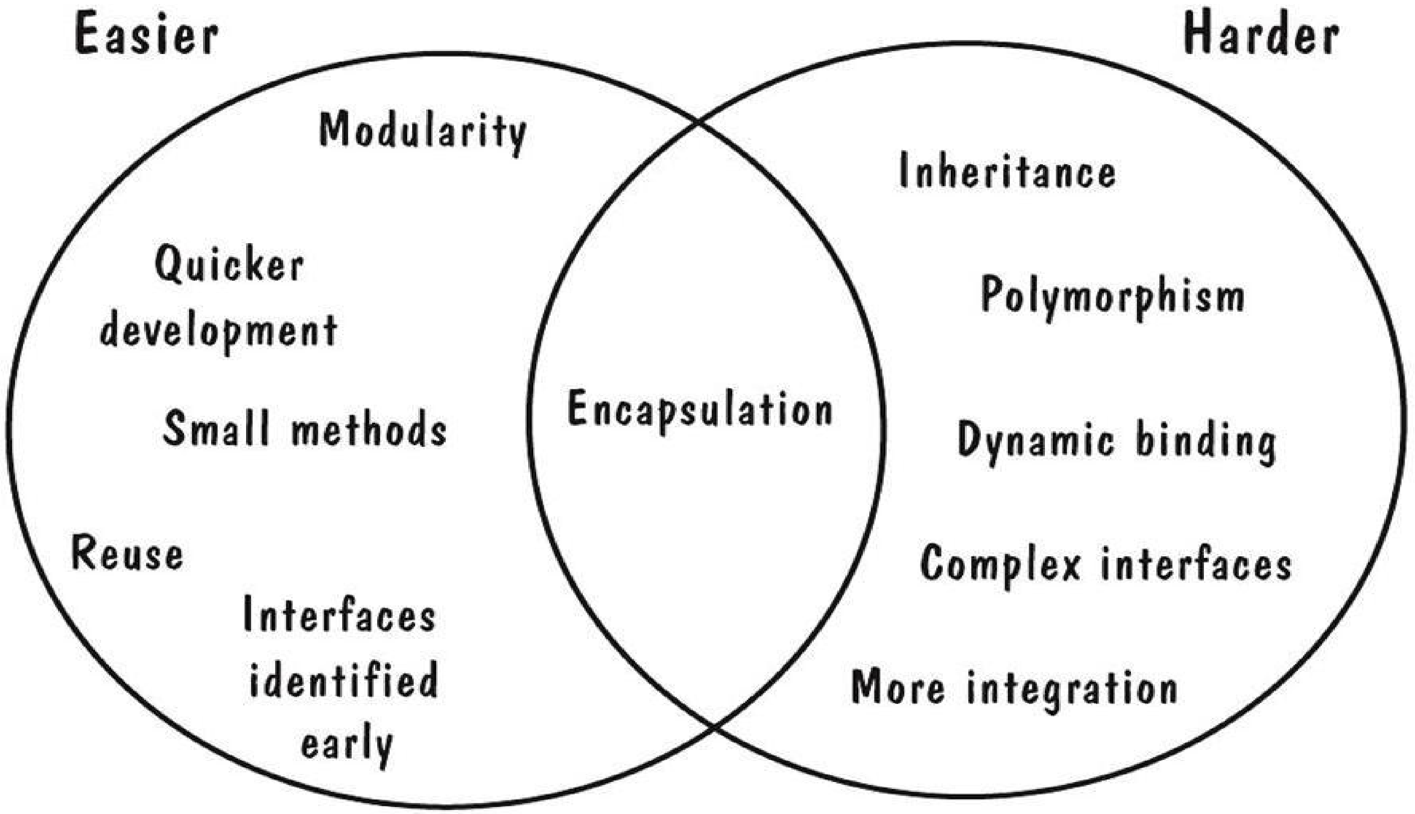
A: 需求文档的验证缺乏工具支持。（很多时候依赖人工）

B: 测试工具生成的测试用例，处理OO模型中的对象和方法时，其针对性不强。（某些OO关系是测试工具本身搞不清楚其内在逻辑关系的）

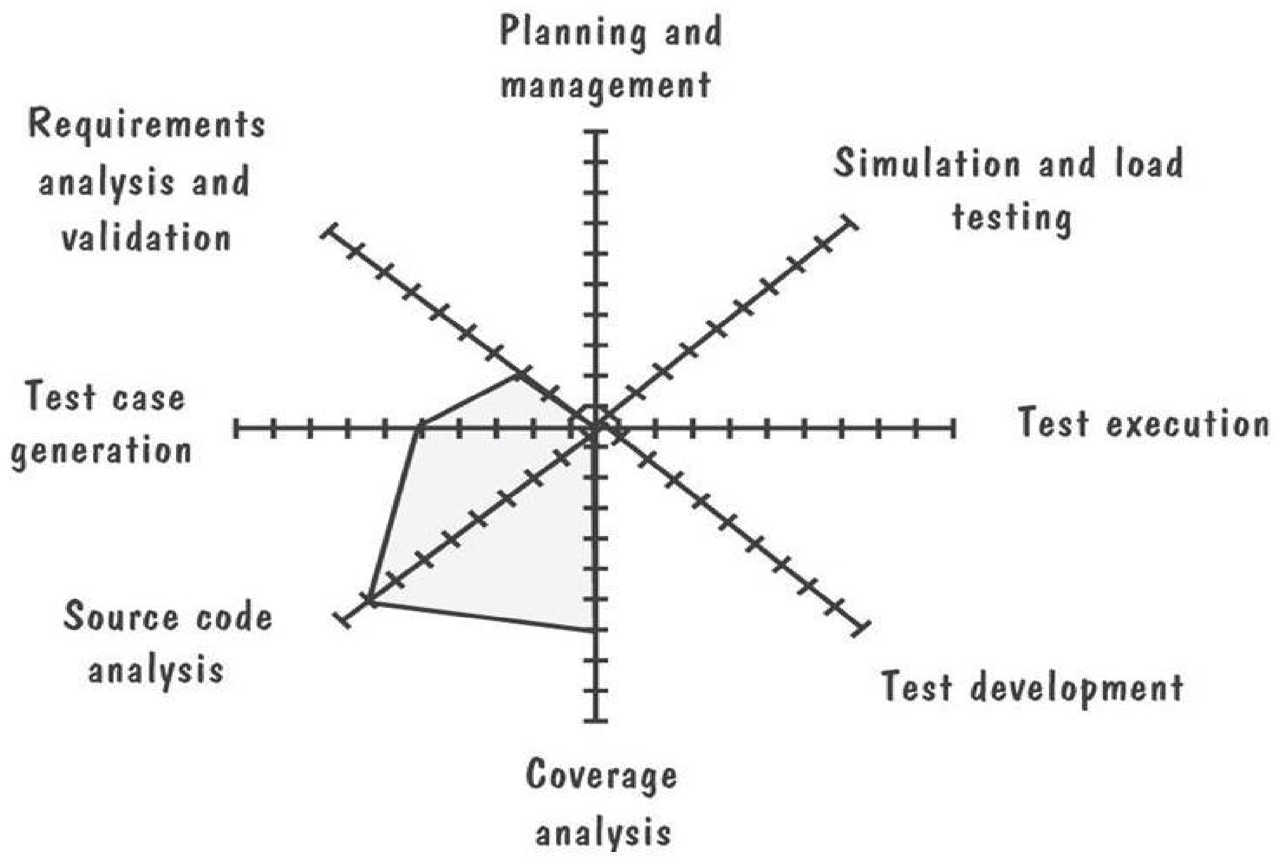
C: 传统的测试方法（如环路复杂度等）在评价OO系统的规模和复杂性时，还不是很有效，或没有太强的针对性。

D: 对象的交互是OO系统复杂性的根源，传统的测试方法和根据/依据的作用有限。

- OO unit testing is less difficult, but integration testing is more extensive (图8-16)



- The farther the gray line is out, the more the difference（图8-17）



Chapter 8 Testing the Programs

8.6 Testing Planning（测试计划）

① significance: (P436)

careful test planning $\xrightarrow[\text{us}]{\text{help}}$ design and organize the tests appropriately and thoroughly

② <u>several steps</u> about test plan (P436— "1-6")

1. 测试目标 (测试的各个总体计划与策略, 进度安排, 优先级及其他量化指标：例如集成测试后的缺陷数目/千行代码《0.5, 单元测试产生的缺陷≈0.067，其他经验数据等等)
2. 用例的分类设计 (选择分类方法、附加方法等完成分类)
3. 书写测试用例
4. 复审测试用例 (审查用例的合理性,有时还编写测试程序)
5. 运行测试用例
6. 评价测试结果

Chapter 8 Testing the Programs

1. Purpose of Plan

① purpose: realize test objectives by a test strategy

② test plan: ----describes the way in which we will show our customers that the software works correctly (the testing method include: who, why, how, when/schedule) (P437-s1)

③ demanding (制定测试计划的要求):

A: know modular hierarchy

B: choosing test objectives, define test strategy, generate test cases (to wait the testing executing)

2. Contents of the Plan

① test objective and designing test cases

A: objective and steps (address the test stages)
(量化的测试目标(象PSP,TSP中的测试统计目标一样),针对测试目标制定的具体步骤, 以及结束测试的原则等 .)

B: classifying , designing test cases, and choosing a few representative test cases .

② methods and techniques

----integration methods + review methods + all kinds of testing reports

③ detailed list of test cases

Chapter 8 Testing the Programs

3. Final Purpose of Test Plan (P438)

----having a complete picture of how and why testing will be performed

4. example about test plan

----see “test plan example.doc”

Chapter 8 Testing the Programs

- 模拟面试问答：如果让您来带领一个测试团队，您会做哪些工作？
- 对于需求文档，如何进行测试？

  （从什么方面考虑？有什么样的原则？）
- 课后作业3：练习题7.（课本）

Chapter 8 Testing the Programs

- 大学教育只是给人一种眼光，而绝不能保证你将来做什么！
- 唯一能做的是提高自身的能力和素质。

Chapter 9 Testing the System

Note A:unit and integration testing----by yourself or a small part of the development team

B:system testing----by the entire develop team

- **9.1 Principles of system testing**

Focus A: ① objective of unit and integration

------ensure the code implemented the design properly ①②③④⑤⑥ ⑦ ⑧⑨⑩