



SOFTWARE TESTING

苏临之

sulinzhi029@nwu.edu.cn

计算机类考研初试安排

	第一天（周六）					第二天（周日）			
上午 8:30~11:30	思想政治101 (100)					数学一301 / 数学二302 (150)			
	思修	马原	近现代史	毛中特	时政	高等数学 (微积分)	线性代数	概率统计	
下午 14:00~17:00	英语一201 / 英语二204 (100)					计算机学科专业基础综合408 (150)			
						数据结构	计算机组成 原理	操作系统	计算机 网络



考研中的软件测试

- 有些院校有软件工程专业自主命题权，第二天下午的专业课考试就可能包含“软件工程”这门课，而软件测试是软件工程中的重要组成部分。
- 有些时候，考研复试可能会涉及软件测试的相关问题，尤其是软件工程专业。而在考研面试时，软件测试在相关专业往往会成为一个重点去考察。这时往往考察一些基本术语和基本概念，以及某些测试工具（如JUnit）的简单使用情况。



Teaching Aims

- 目标1：掌握基软件测试的基本概念、基本理论、常用测试方法与测试技术。（34）
- 目标2：能够使用常见的测试自动化工具以提高测试的效率和准确性。（20）
- 目标3：能够针对实际问题选择合适的测试策略，综合应用测试理论、测试方法与技术并运用测试工具进行软件部件测试的设计与实施。（36）
- 目标4：能够管理软件测试的各个阶段。（5）
- 目标5：能够对测试结束后的相关结果做出分析评估，从而改进测试流程，完善后续测试过程。（5）



Error, Failure, Fault and Defect

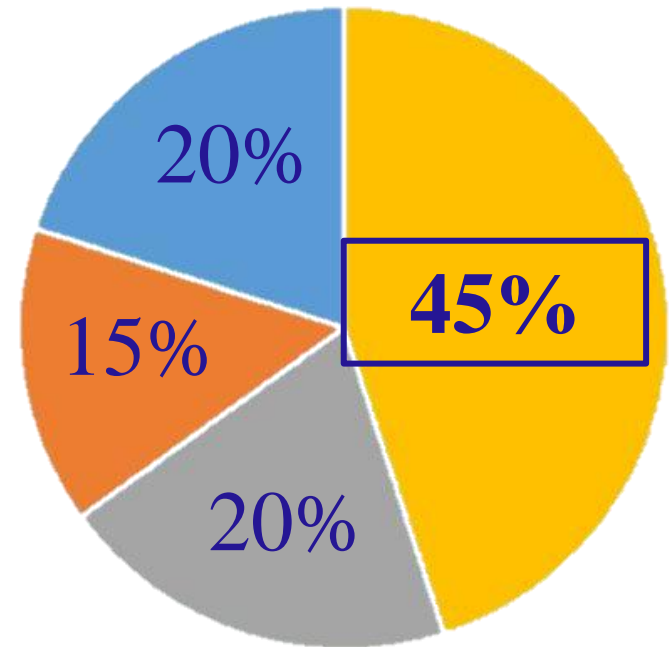
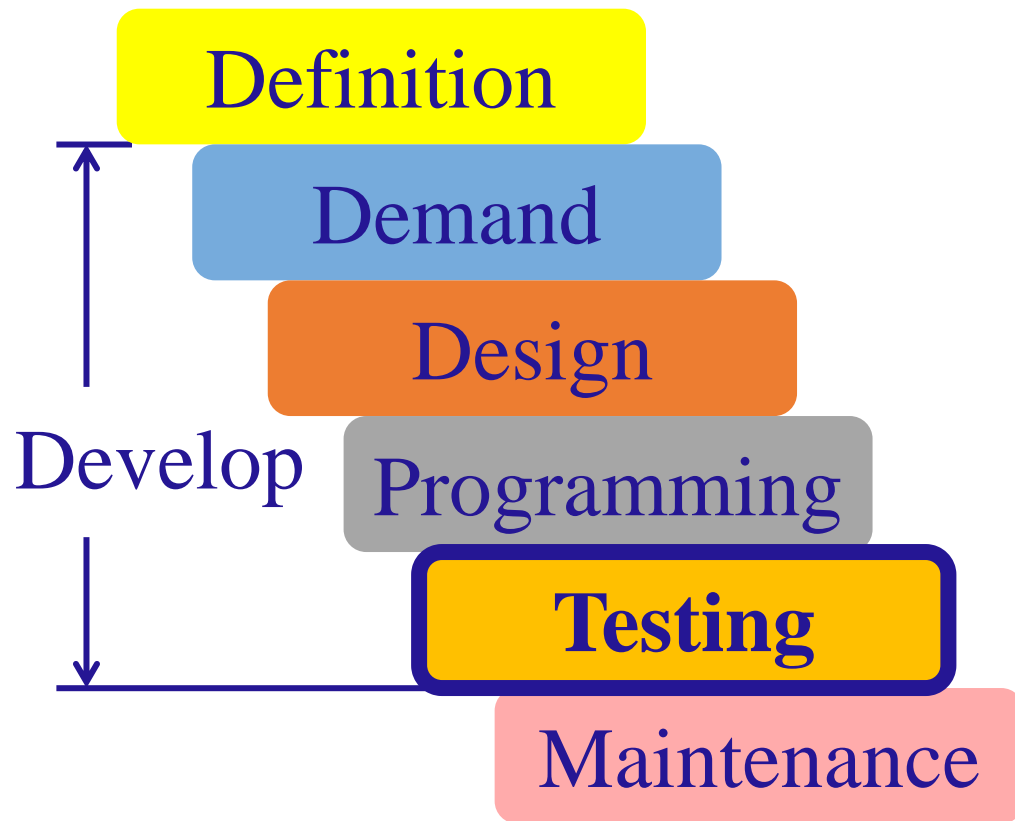
- Error: The software can not operate as fully anticipated.
- Failure: The software engenders wrong results caused by errors.
- Fault: The existing physical problems that may or may not result in failures.
- Defect: The software does not work as required. A software defect, or bug as often referred to, may or may not result in an error.



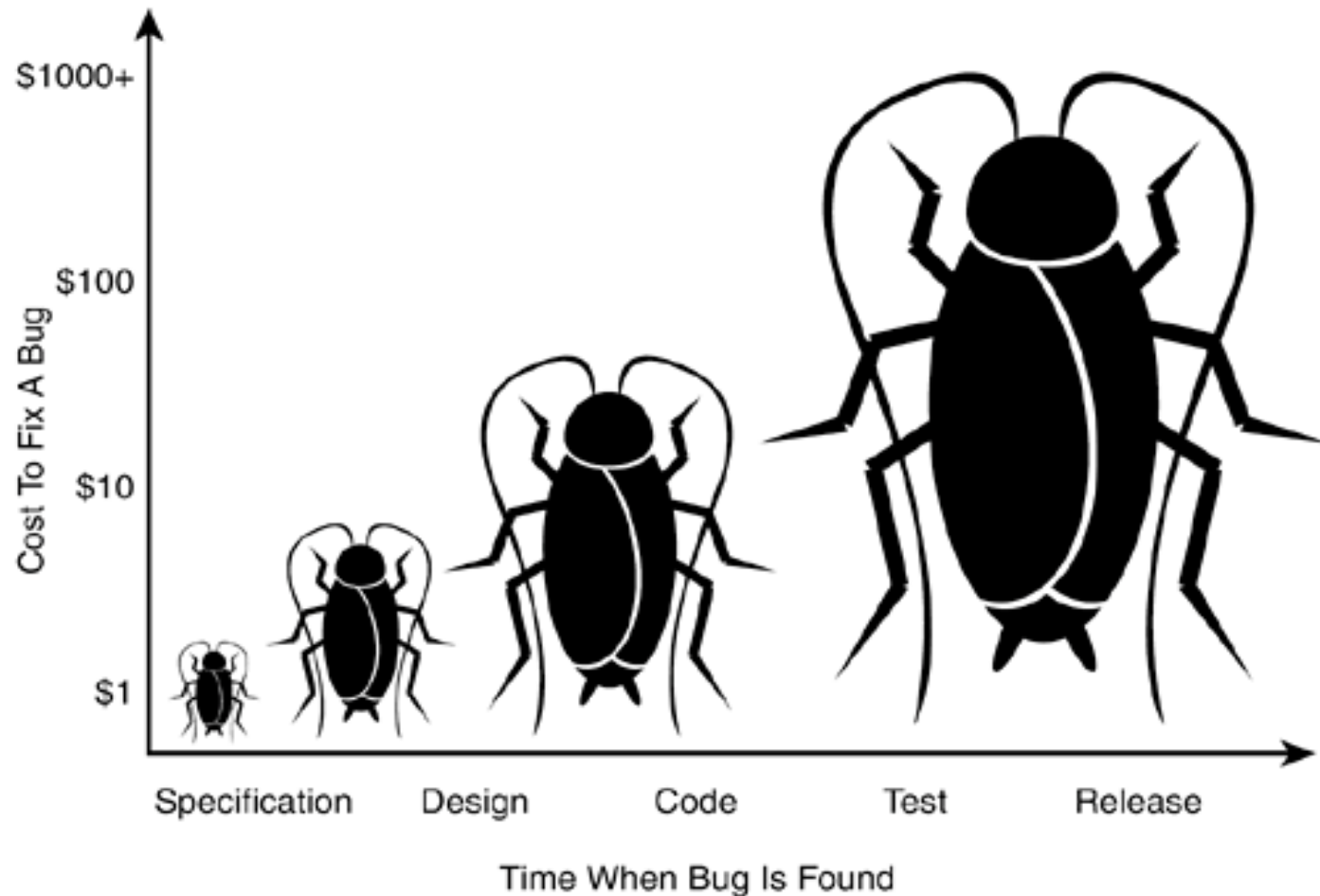
Identification of a Software Defect

- ❖ A software defect occurs when one or more of the following five rules is true:
 1. The software doesn't do something that the product specification says it should do.
 2. The software does something that the product specification says it shouldn't do.
 3. The software does something that the product specification doesn't mention.
 4. The software doesn't do something that the product specification doesn't mention but should.
 5. The software is difficult to understand, hard to use, slow, or in the tester's eyes will be viewed not good by the end user.

Software Life Cycle



Cost of Defects





Software Development Process

- ❖ In a software development process, some basic elements are indispensable:
 1. Customer requirements
 2. Specifications (Spec)
 3. Schedules
 4. Software design documents
 5. Test documents



Software Development Lifecycle

- ❖ There are many different methods that can be used for developing software, four frequently used models listed here.
 - Big-Bang
 - Code-and-Fix
 - Waterfall
 - Spiral
- ❖ No model is necessarily the best for a particular project. Each model has its advantages and disadvantages.
- ❖ As a tester, one will likely encounter them all and will need to tailor (adjust) his test approach to fit the model being used for his current project.



Some Testing Axioms

- Not all the found bugs will be fixed. Generally four reasons:
 - There's not enough time.
 - It's really not a bug.
 - It's too risky to fix.
 - It's just not worth it.
- The more bugs one have found, the more bugs there are.
- The more one tests software, the more immune it becomes to his tests.



Test Cases

- ❖ Test cases are the specific inputs that one will try and the procedures that you'll follow when you test the software.
- ❖ 测试用例是为某个特殊目的而编制的一组输入数据，用于测试输出、执行条件以及预期后果，以便测试某个顺序途径或核实能否满足某个特定需求。



Test-to-Pass & Test-to-Fail

❖ Two fundamental approaches to testing software

■ Test-to-Pass

- Designing and running test cases to assure only what the software can minimally work is called test-to-pass.

■ Test-to-Fail

- Designing and running test cases with the sole purpose of breaking the software is called test-to-fail or error-forcing.



Four Basic Techniques

- Static Black-Box Testing
- Dynamic Black-Box Testing
- Static White-Box Testing
- Dynamic White-Box Testing



Static Black-Box Testing

- ❖ High-Level Review of the Specification
 - Pretend to be the customer
 - Research existing standards and guidelines
 - Review and test similar software
- ❖ Low-Level Review of the Specification
 - Specification attributes
 - Specification terminology

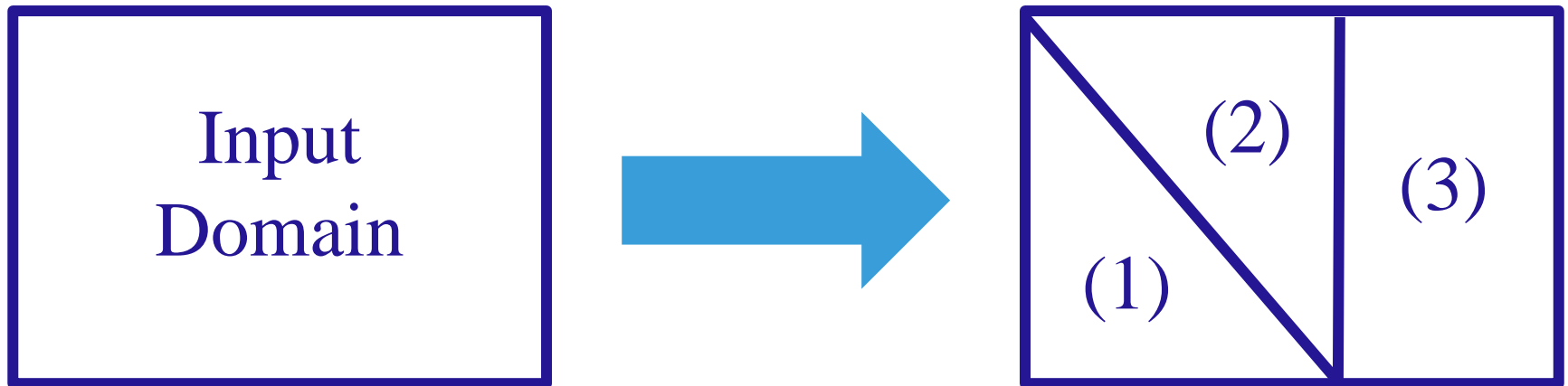


Dynamic Black-Box Testing

- Equivalence Partitioning
- Boundary Value Analysis
- Decision Table
- Cause-Effect Diagram
- Error Guessing

Equivalence Class

- An equivalence class is a subset that consists of test cases, testing the same thing or revealing the same bug.
- If we view the input domain as the universe, a certain subset is an equivalence class.





Equivalence Partitioning

- The input domain is divided into several equivalence classes, and we select only a few (maybe only one) representative test cases from each class, reducing the inputs and thus avoiding redundant full testing. This is called the equivalence partitioning.
- Two basic steps:
 - Establishing the equivalence class table
 - Designing the corresponding test cases



Basic Principles

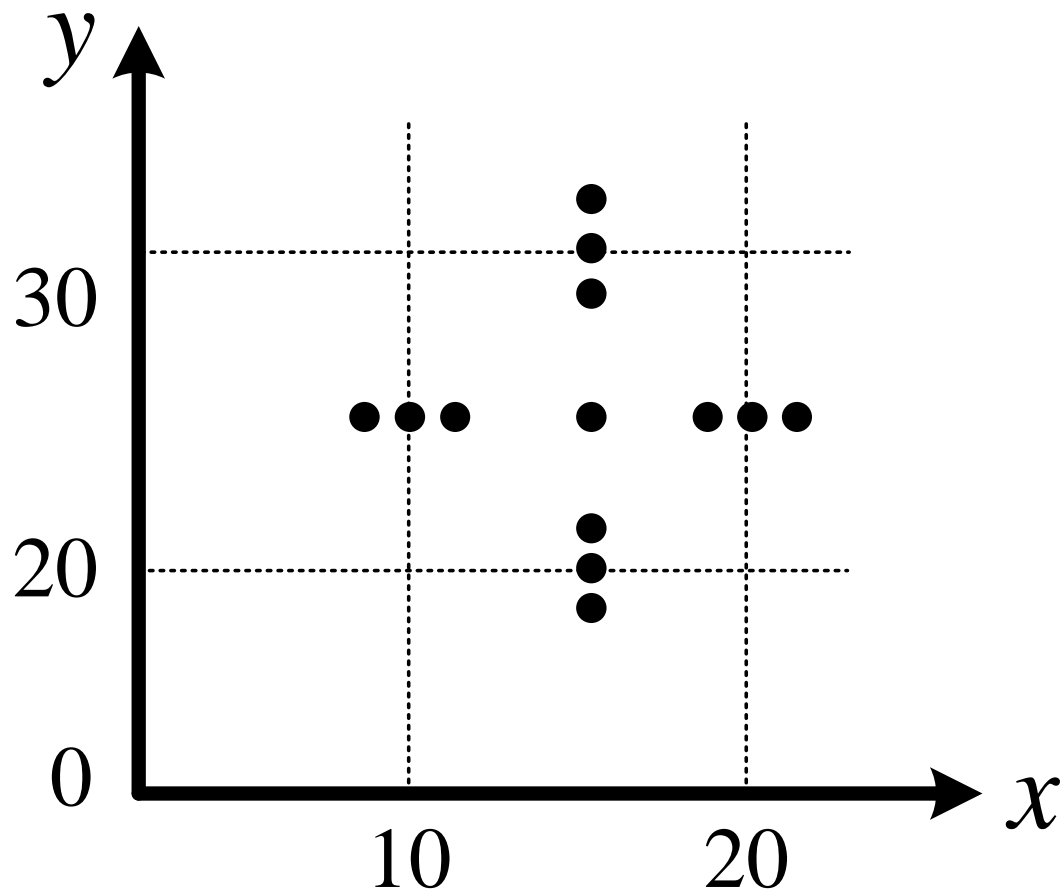
- Value Range: one VEC & two IECs
- Value Number: one VEC & two IECs
- Multiple Branching: one VEC for each branch value & one IEC
- "Must-Be or Should-Be": one VEC & one IEC
- If the elements are thought to be treated unequally for any reason, or if an EC is too general, then we should divide it into some smaller ECs.



Additional Principles

- If the condition contains the OR relationship, whether valid or invalid, we usually establish several dependent ECs; provided the relationship is AND, a further consideration is needed.
- "*n* numeric characters" indicates both "numerical digit equals *n*" and "must be numeric characters". By using the NOT operation, three IECs should be established.
- Multiple branches can also be expressed as "must be A, B or C".
- A special requirement for the numerical digit can be translated into the numerical value range.
- Sometimes the De Morgan's laws will facilitate the analysis.

Graphic Analysis





Decision Table

- A decision table is usually employed to describe the cases where the input variables are logical concerned or the inputs and outputs are limited by causalities.
- A decision table consists of two parts. One is the condition part (consisting of condition stubs) and the other is the action part (consisting off action stubs).



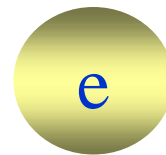
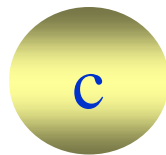
Procedure Summary

- 首先确定各个条件桩和动作桩；
- 对于可以预见的任意项加以合并，并对不可能并存的条件桩组合加以删除；
- 根据条件桩的组合和题目要求，画出对应输出了哪些动作桩。
- 检查判定表，根据最后的结果再次合并无关项；
- 根据判定表的条件桩组合给出测试用例，以检查结果是否和动作桩组合相一致。



Cause-Effect Diagram

- The cause-effect diagram (CED), or cause-and-effect diagram in some literature, is used to describe the test issue where multiple inputs are involved. At the same time, the CED is also able to point out the imperfection and ambiguousness in the specification.
- Each single input is a cause, and each single output is an effect. They are shown as follows.

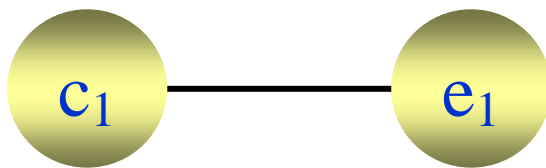




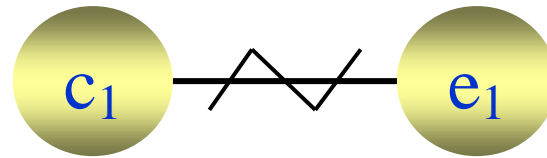
Basic Relationships and Constraints

- There are four basic relationships between the causes and the effects: Equal, Not, Or & And.
- There are five basic constraints among the causes or among the effects. For input causes, the four constraints are: Exclusive (E), Inclusive (I), Only one (O) & Require (R). For output effects, there is only one constraint: Mask (M).

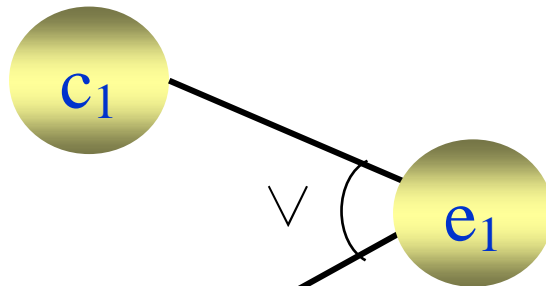
Four Basic Relationships



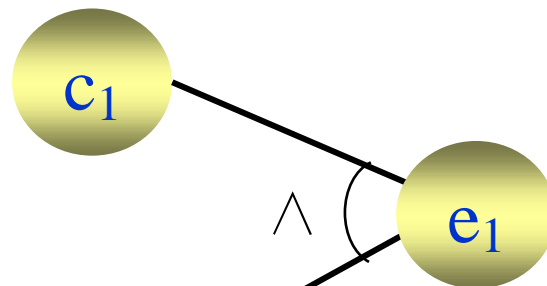
Equal



Not

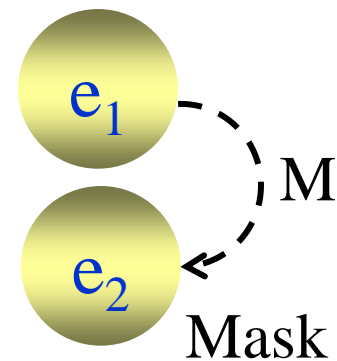
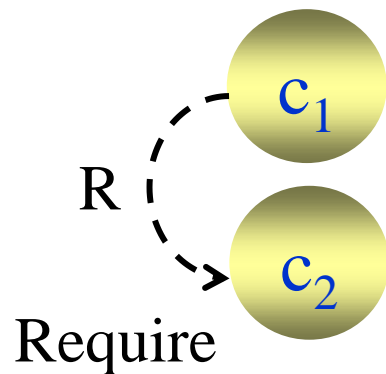
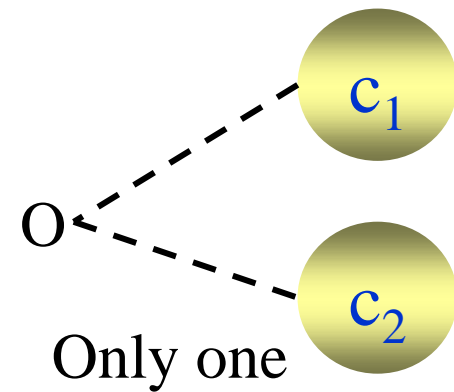
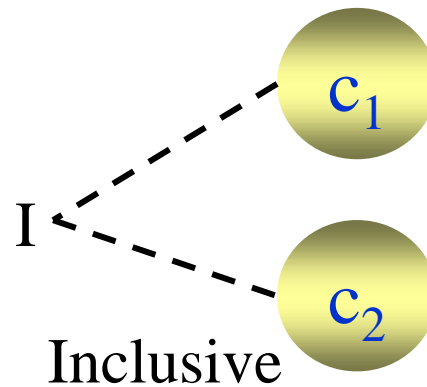
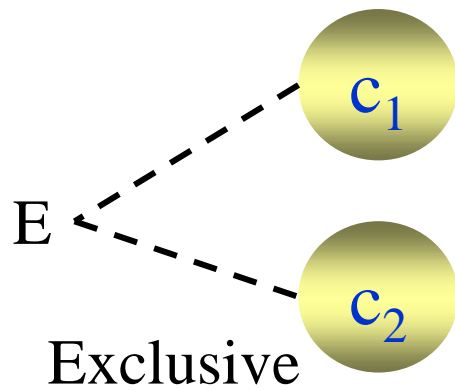


Or



And

Five Basic Constraints





Procedure

- 1、列出作输入条件的原因和作输出条件的结果；
- 2、通过语义分析原因之间的约束，标于图上；
- 3、分析每一个结果由哪些原因组合构成的；
- 4、通过因果分析构建出恒等关系；
- 5、构建中间变量，使一次传播中仅一种基本关系；
- 6、完成因果图绘制，并分析结果间的约束；
- 7、转换为判定表；
- 8、划掉不满足诸约束的组合；
- 9、根据判定表给出相应的测试用例。



Procedure

- 1、因果列举；
- 2、原因约束；
- 3、因果分析；
- 4、恒等构建；
- 5、中间构建；
- 6、结果约束；
- 7、转判定表；
- 8、弃违约项；
- 9、测试用例。



Static White-Box Testing

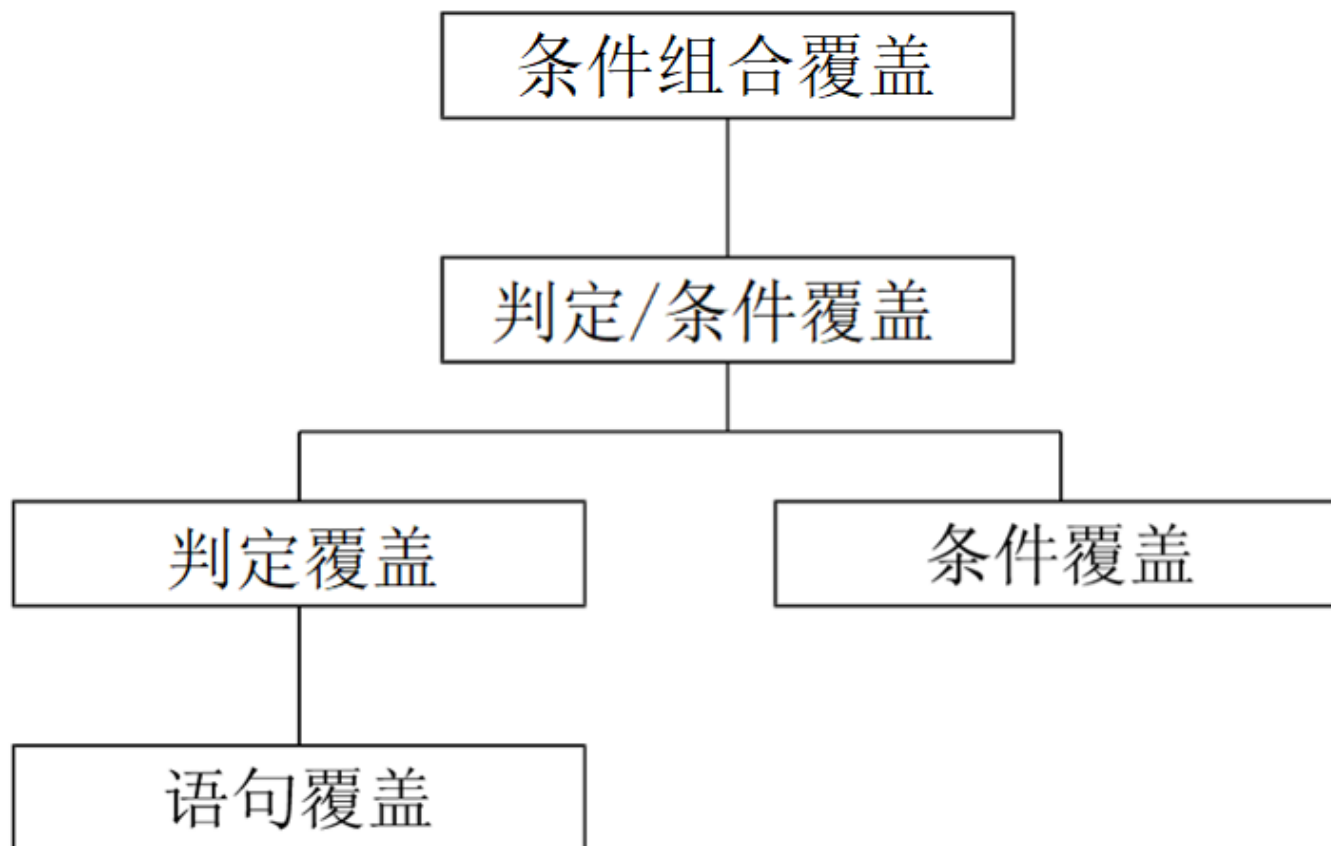
- The list generally contains the follow eight issues.
 1. Data Reference Errors
 2. Data Declaration Errors
 3. Computation Errors
 4. Comparison Errors
 5. Control Flow Errors
 6. Subroutine Parameter Errors
 7. Input / Output Errors
 8. Other Checks



Techniques in DBWT

- Statement Coverage
- Decision Coverage
- Condition Coverage
- Decision / Condition Coverage
- Multiple Condition Coverage
- Path Coverage

Techniques in DBWT



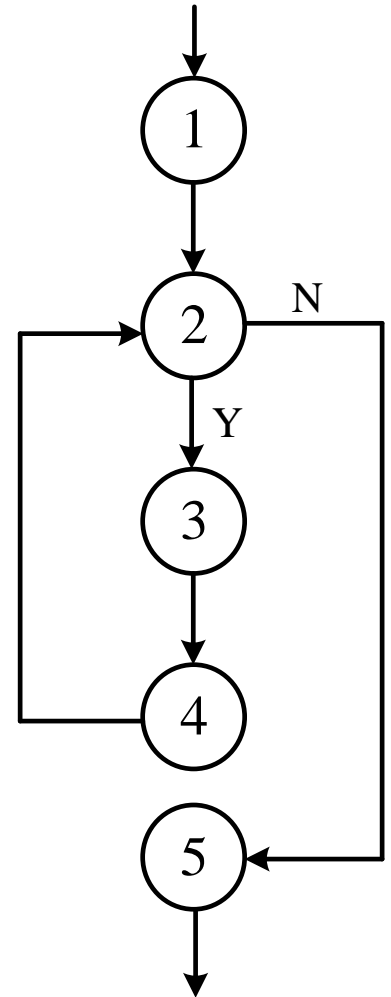


Program Control Flow Graph

- A program, no matter how complicated it is, is made of three basic structure elements: sequence, branch and loop. A program control flow graph be used to describe it.
- In the flow graph, two elements are involved: **node** and **edge**.

Node and Edge

- Each circle in the graph is a node, and if there are several statements in a sequence structure, they are actually viewed as one node.
- An edge shows the sequence of nodes.
- The edge which enter the program is called the entry edge, and the one which leave the program is called the exit edge. The other edges are the active edge.





Cyclomatic Complexity

- Three methods for calculating the cyclomatic complexity:

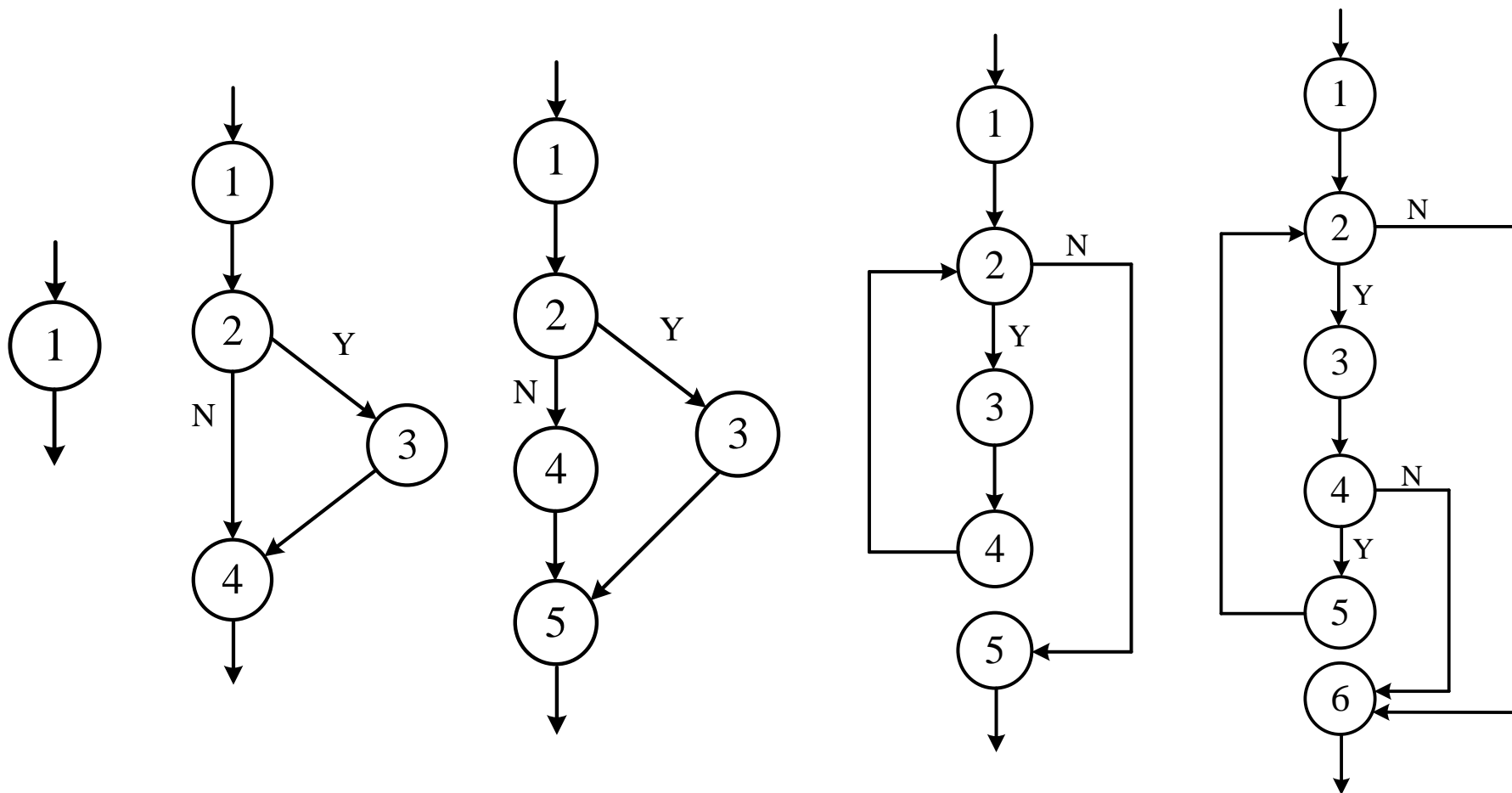
- Two Formulae

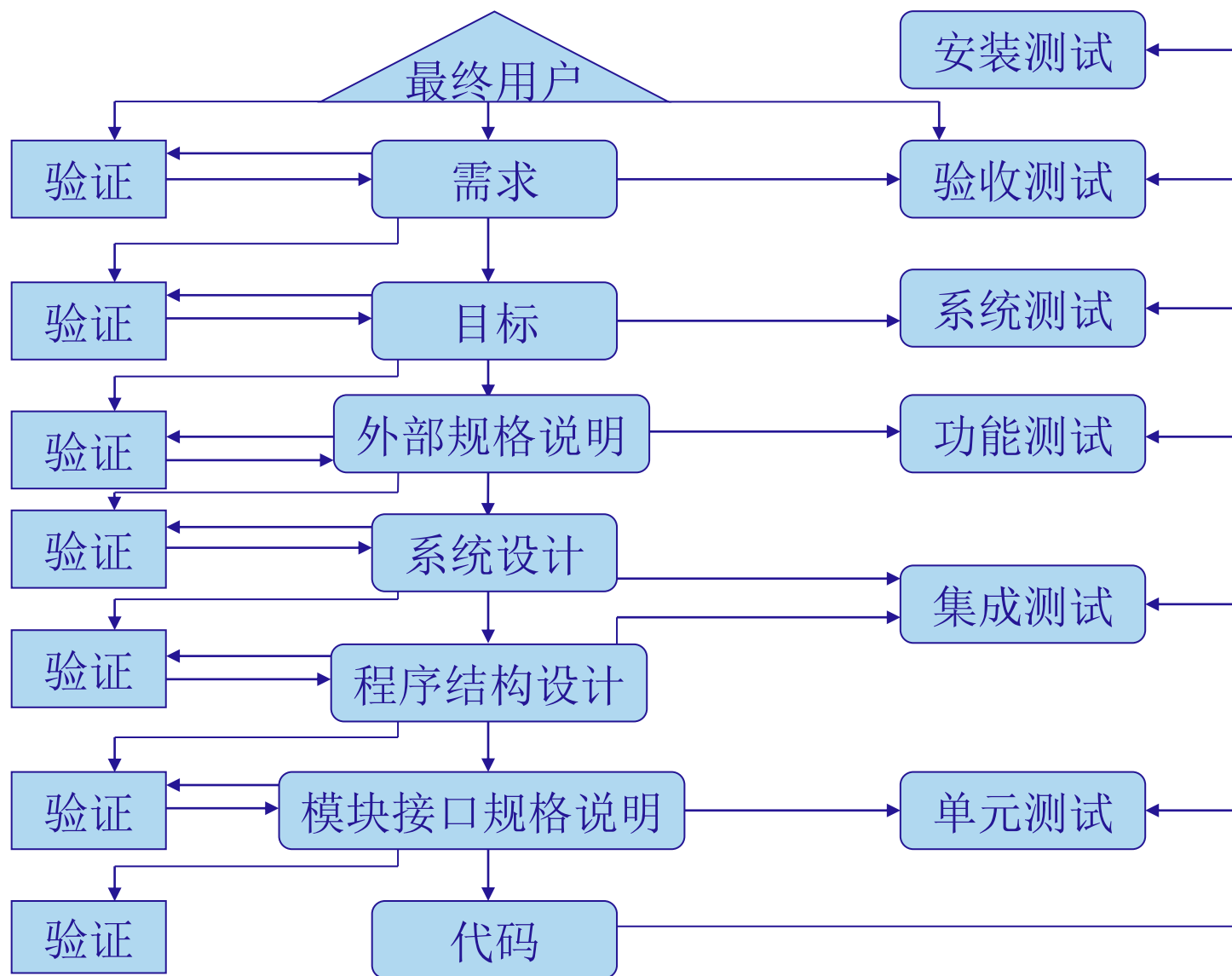
$$V = E - N + 2P \qquad V = E_0 - N$$

- Number of Plane Region

- Number of Binary Predicate Nodes plus 1

Structures and Basis Path Set



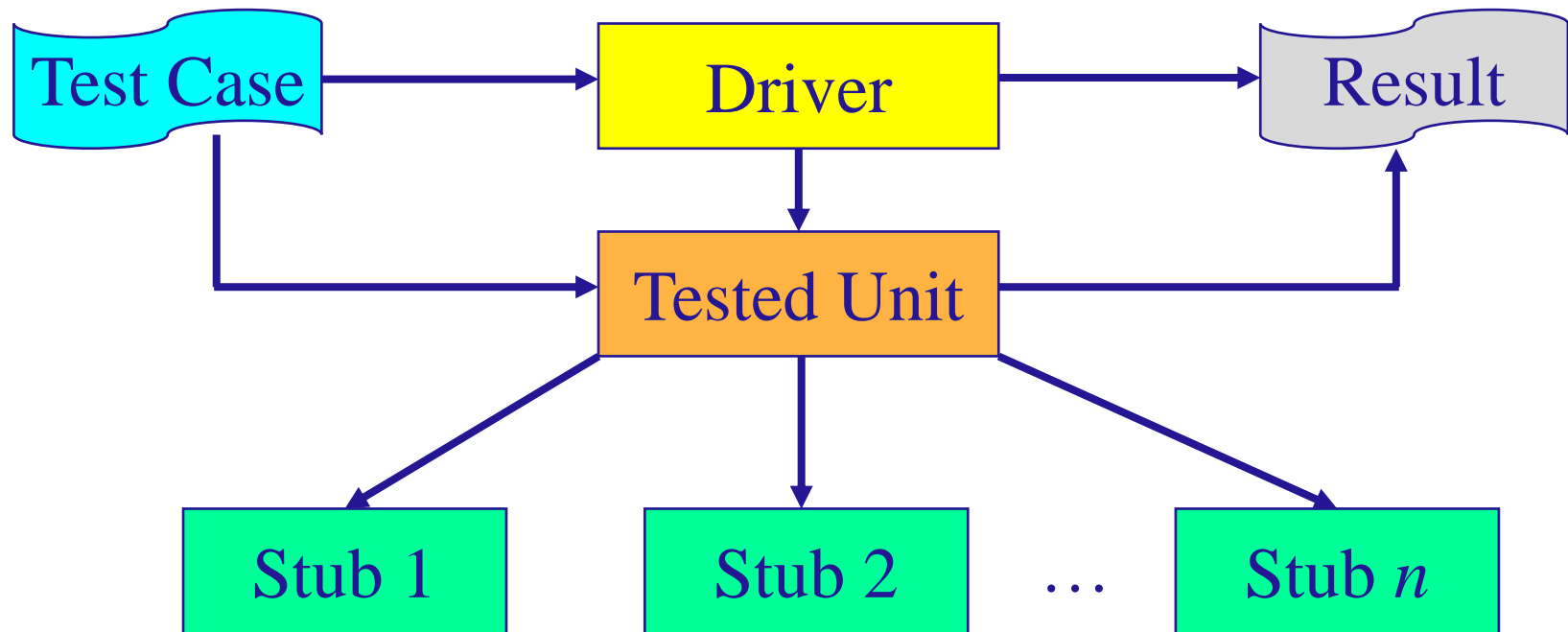




Unit Testing

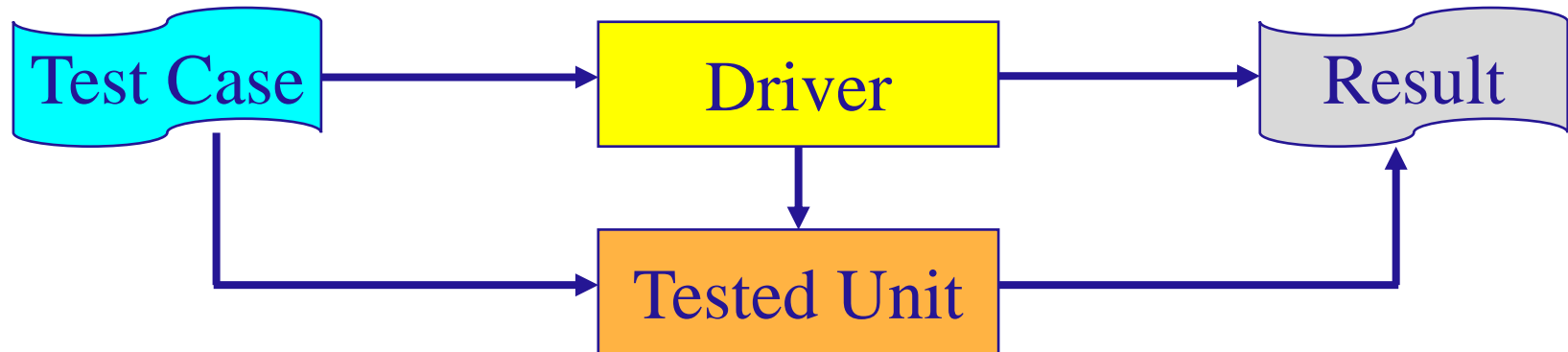
- Unit testing is also called the module testing, which alleviates the difficulty in the testing process. One can also test several units in parallel.
- The unit testing is generally a white-box testing, which means that it is essential to know the basic code in the module, including the input and output as well as its function.

Basic Model



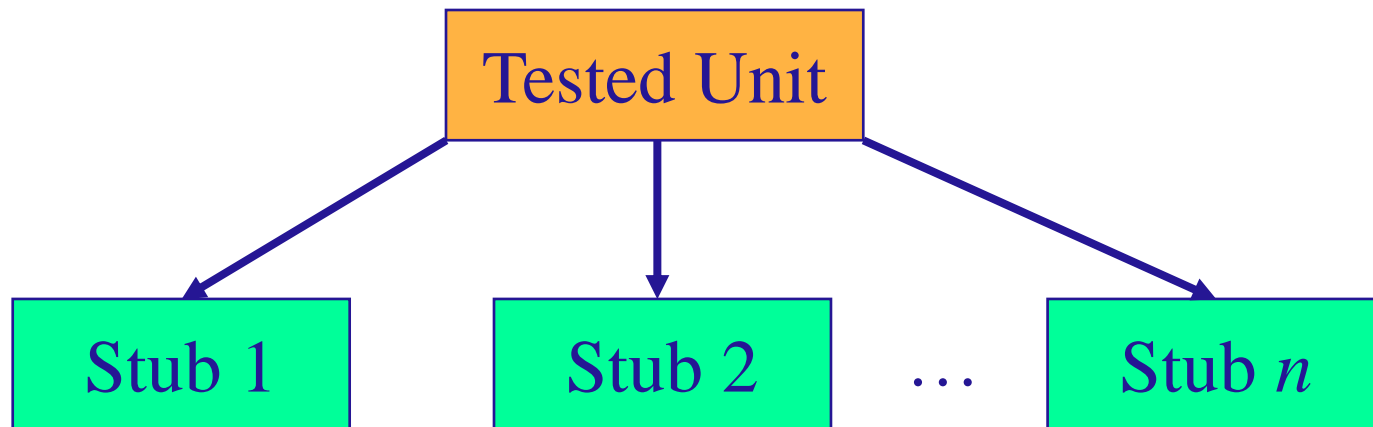
Driver Module

- The driver module acts as the higher part of the unit. It sends data to the unit and then shows the corresponding result after operation.



Stub Module

- The stub module acts as the lower sub-routine of the unit. By simulating the module called by the tested unit, the stub module itself does not operate but just return a static value.





Testing Methods

- Non-incremental Testing: Also called the Big-bang integration. The units are tested individually and then integrated together for the integration testing.
- Incremental Testing: The tested modules are arranged into the structure of the program in advance. In this way the module testing also includes the integration testing, which will not serve as an individual part.
 - Top-Down Testing
 - Bottom-Up Testing



Comparison

增量测试	非增量测试
工作量小：使用前面测试过的模块来取代非增量测试中所需要的驱动模块或桩模块	工作量较大：要不断设计驱动模块和桩模块
可以较早发现模块中与不匹配接口、不正确假设等编程错误	到了测试过程的最后阶段，模块之间才能“互相看到”
容易进行调试，新出现的错误往往与最近添加的模块有关	直到整个程序组装之后，模块之间接口相关的错误才会浮现，难以定位



Comparison

增量测试	非增量测试
测试可以进行地更彻底，每个模块经受了更多的检验	使用驱动模块和桩模块而非实际模块，对被测试模块的测试只影响自身
在测试上花费的时间多，设计驱动模块和桩模块所用时间少	测试时间少，但设计驱动模块和桩模块需要大量时间
并行性差	可以同时并行测试很多模块



Comparison

自顶 向下	优点	1、若主要缺陷出现于顶层则非常有利 2、预知框架结构，因此能提早发现主要控制问题
	缺点	1、必须开发桩模块，可能它们比最初表现更复杂 2、创建测试环境可能很难，甚至无法实现 3、观测测试输出比较困难
自底 向上	优点	1、若主要的缺陷发生在程序的底层将非常有利 2、提早发现程序当中的主要算法问题 3、测试环境比较容易建立 4、观测测试输出比较容易
	缺点	1、必须开发驱动模块 2、直到最后一个模块添加进去，程序才形成整体



Functional Testing

- In the functional testing, one tries to find out the difference between the actual function of the program and its specification. Upon the analysis of the specification, one will get a set of test cases.
- Except for some very small programs, the black-box testing techniques are usually applied, and the techniques, such as equivalence partitioning, boundary value analysis, decision table and cause-effect diagram, are especially applicable here.



System Testing

- System testing, whose aim is to compare the entire system with the initial goals, is a process to find out whether the system is not capable of meeting the goals. So without these goals, one can never executing the system testing.
- It is worth noting that the system testing is not to test the function of the system because it is not based on the specification. Actually the system testing can be viewed as a non-functional testing.
- It is crucial to determine who will be in charge of system testing because the testers here must know customers' attitude and the circumstance for using. Those who develop the software can not act as the system tester.



Acceptance and Installation Testing

- The acceptance testing is the process to compare the program with the initial requirement and the customers' final requirement. It is process executed by customers or the end users, so it is not the duty of those who developed the software.
- The aim of installation testing is not to find the error in the software, but to find the error in the installation process in which many options should be determined by the users and valid configurations should be involved.



Common Issues in System Testing

- Ability Testing
- Load Testing
- Stress Testing
- Performance Testing
- Memory Testing



Ability Testing

- The ability testing aims to check whether every ability item has been realized or not.
- The ability testing is not the functional testing, which focuses on whether the software can realize the corresponding functions. The ability testing focuses more on the specific quality to realize a function, such as stability or compatibility.



Load Testing & Stress Testing

- The load testing makes the system experience large amount of data, aiming to check whether the software can cope with the load as demonstrated. It will occupy much source. For example, the popular artificial neural networks need load testing when they are run in a computer.
- The stress testing aims to test whether the system can handle high stress, which means that in a short time the data or operations reach their maximum. For example, the actual power source supply will consider the maximum power transfer, so the stress testing is indispensable.



Specific Issues in System Testing

- Configuration Testing
- Compatibility Testing
- Foreign Language Testing
- Usability Testing
- Documentation Testing
- Safety Testing



Configuration Testing

- ❖ A process for various hardware to run a software.
- ❖ 配置测试是必不可少的，原因是硬件的生产厂商并没有执行严格的标准，有时候仅执行松散规范，这样导致软件使用某种硬件配置无法正常工作。
- ❖ 如何判断缺陷是由配置问题产生而不仅仅是一个普通缺陷？最可靠的方法是：在另外一台由完全不同硬件配置的机器上执行导致问题的相同操作。
- ❖ 配置测试的工作量可能是巨大的，因此可以使用等价类划分的方法来减少工作量。



Compatibility Testing

- ❖ 软件经常需要向其他程序导入和导出数据，在各种操作系统和网络浏览器上运行，与同时运行在同一硬件上的其他软件交叉操作。
- ❖ 软件兼容性测试的目标是保证软件按照用户期望的方式与其它软件进行交互。
 - **Forward** Compatibility: Being able to use the versions in the **future**.
 - **Backward** Compatibility: Being able to use the versions in the **past**.



Foreign Language Testing

- 逐字直译单词是容易的，但要想使整个操作提示意思明确、实用，就需要投入更多的时间和精力。好的翻译要是外文翻译得读起来和原文一样。
- 软件适应特定地域特征，照顾到语言、方言、地区习俗和文化的过程称为本地化或国际化。



Usability Testing

- There are seven basic element to design a good UI.
 - Following standards and guidelines (符合要求规范)
 - Intuitive (直观)
 - Consistent (一致)
 - Flexible (灵活)
 - Comfortable (舒适)
 - Correct (正确)
 - Useful (实用)



Test of Independence

- Test of independence corresponds to an independent group of people who participate the testing process. They do not develop the software.
- 优势：1) 测试人员从中立的角度看待每个缺陷；2)测试人员完全没有偏见，即不带主观人至偏颇；3)测试人员对质量没有任何假设。
- 劣势：1) 与开发团队的隔离有时会导致过时的文档或版本引用；2) 独立的测试执行通常是最后一个阶段，在此过程中任何延迟都会影响受到版本或产品的发布；3) 开发人员可能对质量不负责任，因为他们可能认为独立测试团队的测试系统中有问题；4) 独立测试一定会受到通信阻碍。



Manual & Automatic Testing

人工测试	自动测试
消耗时间并单调：由于测试用例是由人力资源执行，所以非常缓慢并乏味。	时间消耗少：快速自动化运行测试用例时明显比人力资快。
人力资源上投资巨大：由于测试用例需要人工执行，所以在人工测试上需要更多的试验员。	人力资源投资较少：测试用例由自动工具执行，所以在自动测试中需要较少的试验员。



Manual & Automatic Testing

人工测试	自动测试
可信度较低：人工测试可信度较低是可能由于人工错误导致测试运行时不够精确。	可信度更高：自动化测试每次运行时精确地执行相同的操作。
非程式化：编写复杂并可以获取隐藏的信息的测试的话，这样的程序无法编写。	程式化：试验员可以编写复杂的测试来显示隐藏信息。



Typical E-Business Structure

- Web Browser
- User Interface Layer
- Business Logic Layer
- Data Access Layer




Debugging

- ❖ 调试指的是定位错误和修改错误的技术。某种程度上属于灰盒测试。
- ❖ 调试步骤
 - 第一步：定位错误（大概95%）。
 - 第二步：修改错误。



Five Debugging Techniques

- Brutal Force Debugging
- Inclusive Debugging
- Deductive Debugging
- Back-Tracking Debugging
- Test Debugging



α - and β -Testing

- ❖ α 测试是由一个用户在开发环境下进行的测试，也可以是公司内部的用户在模拟实际操作环境下进行的受控测试，不能由程序员或测试员完成。
- ❖ β 测试是软件的多个用户在一个或多个用户的实际使用环境下进行的测试。开发者通常不在测试现场，不能由程序员或测试员完成。所以 β 测试是在开发者无法控制的环境下进行的软件现场应用。



Difference between α - and β -Testing

- ❖ 主要区别：测试的场所、环境和时序不同：
 - α 测试可以是指把用户请到开发方的场所来测试； β 测试是指在一个或多个用户的场所进行的测试。
 - α 测试可以的环境是受开发方控制的，时间集中； β 测试的环境是不受开发方控制的，时间不集中。
 - α 测试需要先于 β 测试执行。



Defect Management

- ❖ 软件缺陷管理（Defect Management）是在软件生命周期中识别、管理、沟通任何缺陷的过程。该过程从缺陷的识别开始，直到到缺陷的解决关闭为止，确保缺陷被跟踪管理而不丢失。一般的，需要跟踪管理工具来帮助进行缺陷全流程管理。
- ❖ 每一个软件组织需要明确必须妥善处理软件中的缺陷，因为这关系到软件组织机构生存、发展的质量根本。但是并非所有的软件组织都明确应当如何有效地管理自己软件中的缺陷。



Seven Defect Statuses

状态	状态说明	操作人员
New	首次录入	测试人员
Open	已经确认是缺陷，并等待修改	开发经理/项目经理
Fixed	修改完成后待回归测试验证	开发经理/开发人员
Reopen	回归测试验证不通过，再次等待修改	测试人员
Closed	回归测试验证通过	测试人员
Rejected	讨论后认为不是缺陷或拒绝修改	开发经理/开发人员
Delayed	延迟修改	开发经理/开发人员/缺陷 评审委员会



Process of Defect Management

- 缺陷管理的流程可以概括为：测试人员提交新的错误入库，缺陷状态为“发现错误·新建”；项目经理将缺陷分配给相应的开发人员，缺陷状态为“打开”；开发人员查重现缺陷，做如下处理：如果不是缺陷，则置状态为“拒绝”；如果是缺陷则修复并置状态为“已修复”；如果不能解决的错误，要留下文字说明并保持错误为“拒绝”状态。测试人员查询状态为“已修复”的错误，验证错误是否已解决，做如下处理：如问题解决了置错误的状态为“关闭”，如果问题得不到立即解决则需要置状态为“延迟”。

Five Defect Levels

等级	说明	举例
低	可在发布后再商量是否改进	A. 某些测试的建议
中	不影响功能的正常使用,可以在时间和资源允许的情况下再解决	A. 辅助性说明描述不清楚 B. 显示格式不规范 C. 长时间操作未给用户进度提示 D. 提示窗口文字未采用行业术语 E. 可输入区域和只读区域没有明显的区分标志 F. 系统处理未优化
高	事件是重要的,但是由于解决问题需要花费一定的时间,所以可以用较长的时间解决,如果时间紧可以留下个版本解决	A. 界面文字等错误 B. 打印内容、格式错误 C. 简单的输入限制未放在前台进行控制 D. 删除操作未给出提示



Five Defect Levels

等级	说明	举例
很高	事件是重要的，并且应该在紧急的事件处理之后尽快得到解决，必须在发布前解决	A. 功能不符 B. 缺少功能，与需求不符 C. 数据流错误 D. 程序接口错误 E. 轻微的数值计算错误
紧急	事件非常重要，测试工作无法继续进行，需要马上给予关注解决。	A. 由于程序所引起的死机，非法退出，运行中断，应用程序崩溃 B. 死循环 C. 导致数据库发生死锁 D. 数据通讯错误 E. 严重的数值计算错误



Goals of Planning Testing

- ❖ 如果测试员之间不交流计划测试的对象，需要什么资源，进度如何安排，整个项目就很难成功。
- ❖ 软件测试计划是软件测试员与产品开发小组交流意图的主要方式。
- ❖ 测试计划的目标就是：规定测试活动的范围、方法、资源和进度；明确正在测试的项目、需要测试的特性、要执行的测试任务、每个任务的负责人，以及与计划相关的风险。



Quality Cost

- ❖ 一致性费用（Cost of Conformance, COC）指的是一次性计划和执行测试所相关的全部费用，用于保证软件按照预期方式去运行。
- ❖ 非一致性费用（Cost of Non-Conformance, CONC）既包含在开发中由质量缺陷产生的费用（特别是因缺陷导致的返工），又包含了产品交付后的诸多费用（例如产品召回、官司等一些巨大损失产生的费用）。



Internal & External Failures

- ❖ 内部失败（Internal Failures）指的是在发布产品之前所需要的各种缺陷。由内部失败产生的非一致性费用相对较小。
- ❖ 外部失败（External Failures）指的是在发布产品之后所发现的各种缺陷。由外部失败产生的非一致性费用相对较大，有时候甚至导致公司无法运作以至破产。
- ❖ 一致性费用和内部失败引起的非一致性费用之和往往会小于外部失败引起的非一致性费用。



CMM

- 能力成熟度模型（CMM）是一个行业标准模型，用于定义和评价软件公司开发过程的成熟度。共分为5个等级。
 - 1级：初始的。随意和混乱的过程，项目成功依赖于个人英雄主义和运气。这是软件公司存在最多的级别。
 - 2级：可重复的。项目级的思想，类似的项目成功经验可以重复使用。多数的软件公司位于这个等级。
 - 3级：定义的。组织级别的思想。具有预先考虑和文档化、标准化管理和工程活动。少部分软件公司位于此等级。
 - 4级：可管理的。可控的过程，且对过程和产品质量有详细的评估和理解。极少数软件公司位于此等级。
 - 5级：不断优化的。通过量化的反馈和新途径实现持续改进。几乎没有几家公司能达到这个标准。



Exercise

1. Software testing occupies about ____ of the software life cycle.

A. 10%

B. 20%

C. 35%

D. 45%



Exercise

1. Software testing occupies about ____ of the software life cycle.

A. 10%

B. 20%

C. 35%

D. 45%



Exercise

2. In the static black-box testing, _____ is mainly examined.

- A. the code
- B. the schedule
- C. the specification
- D. the product packaging



Exercise

2. In the static black-box testing, _____ is mainly examined.

A. the code

B. the schedule

C. the specification

D. the product packaging



Exercise

3. The bottom-up testing involves the establishment of _____ modules.

A. the driver

B. the stub

C. both the driver and the stub

D. neither the driver nor the stub



Exercise

3. The bottom-up testing involves the establishment of _____ modules.

A. the driver

B. the stub

C. both the driver and the stub

D. neither the driver nor the stub



Exercise

4. In the process of defect management, a tester found a defect and submitted to the system. Then the project manager assigned it to the development group. At this time the status of defect was "_____".

- A. New
- B. Open
- C. Fixed
- D. Delayed



Exercise

4. In the process of defect management, a tester found a defect and submitted to the system. Then the project manager assigned it to the development group. At this time the status of defect was "_____".

A. New

B. Open

C. Fixed

D. Delayed



THANK YOU!