SOFTWARE TESTING

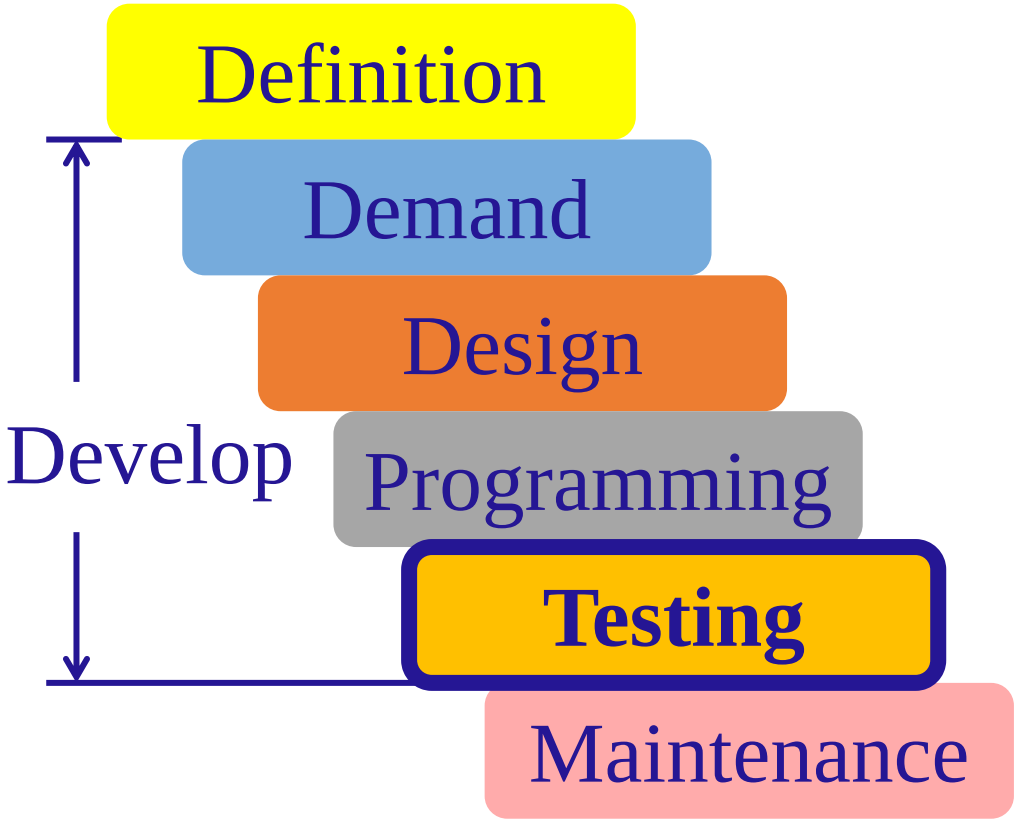
苏临之

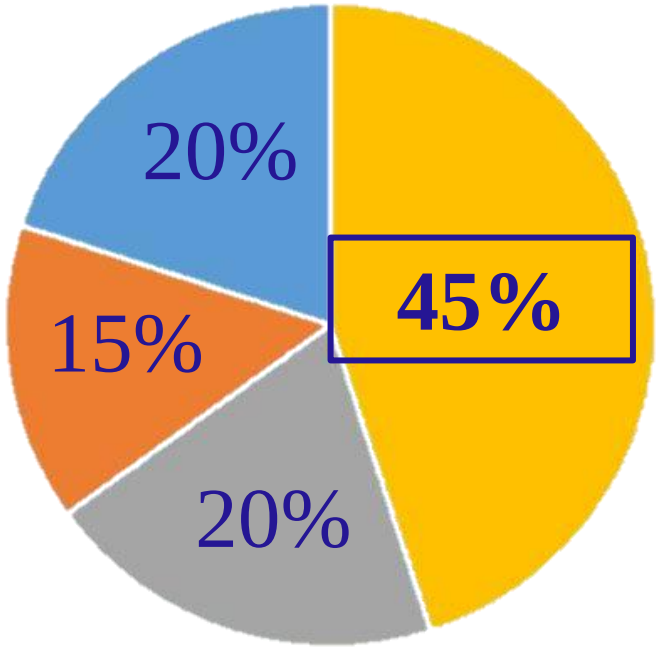
sulinzhi029@nwu.edu.cn

Background for Software Testing

- It's easy to take software for granted and not really appreciate how much it has infiltrated our daily lives.
- Software is everywhere.
- Software is written by people, so it's not perfect.

Software Life Cycle





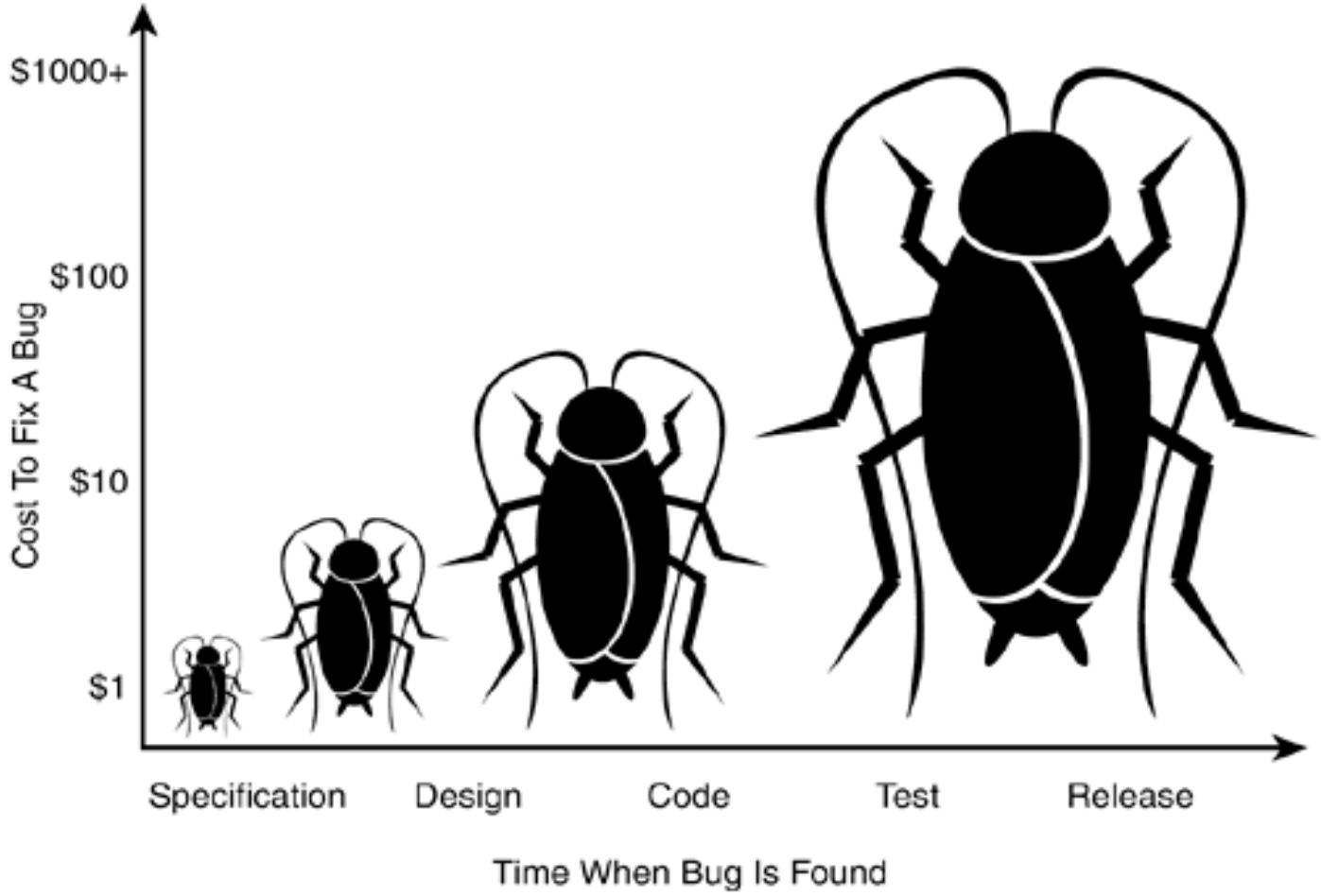
Error, Failure, Fault and Defect

- 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.

Cost of Defects



Software Testing

- 软件测试技术是对软件产品进行验证和确认的活动过程，其目的是尽快尽早发现软件产品中存在的诸问题，包括错误、缺陷以及用户预先定义需求的不一致性等。
- IEEE在1983年对软件测试定义如下：使用人工或者自动手段来运行或测定整个系统的过程，其目的在于检验它是否满足规定的需求或是弄清预期结果与实际结果之间的差别。

Aim of Software Testing

- Software testing is actually a process or a series of processes, identifying that the computer has achieved the functions it should accomplish and has not achieved the ones it did not mention. So the aim of software testing is **to find the defects**.
- One can never think that the aim can be defined as the certification of software validity.

What Does a Software Tester Do?

- The goal of a software tester is to find bugs.
- The goal of a software tester is to find bugs and find them as early as possible.
- The goal of a software tester is to find bugs, find them as early as possible, and make sure they get fixed.

What Makes a Good Software Tester?

- They are explorers.
- They are troubleshooters.
- They are relentless.
- They are creative.
- They are pursuing perfection.
- They are tactful and diplomatic.
- They are persuasive.
- In addition to these traits, having some education in software programming is a big plus.

Tendency

- Getting Simpler
  - Various languages, systems and hardware
  - Computers used in nearly every realm
  - Rapid development of the hardware
- Getting Easier
  - Standing on the shoulders of giants, i.e. we can get much test experience from the pioneers or from a systematic training.

Software Development Process

- In a software development process, some basic elements are indispensable:

1. Customer requirements (用户需求)
2. Specifications (规格说明)
3. Schedules (开发计划详述)
4. Software design documents (软件设计文档)
5. Test documents (测试文档)

Customer Requirements

- Software is written to fulfill some need that a person or a group of people (the customer) has.
- To properly fulfill that need, the product development team must find out what the customer wants.
  - Some teams simply guess, but most collect detailed information in the form of surveys, feedback from previous versions of the software, competitive product information, magazine reviews, focus groups, and numerous other methods, some formal, some not.
  - All this information is then studied, condensed, and interpreted to decide exactly what features the software product should have.

Specifications

- The result of the customer requirements studies is really just raw data. It doesn’t describe the proposed product, and it just confirms whether it should (or shouldn't) be created and what features the customers want.
- The specifications take all this information plus any unstated but mandatory requirements and define what the product will be, what it will do, and how it will look.
- The format of specifications varies greatly. Some companies use a very rigorous process with many checks and balances. The result is an extremely detailed and thorough specification, indicating that it can not change except under very extreme conditions.

Schedules

- A key part of a software product is its schedule.
- As a project grows in size and complexity, with many pieces and many people contributing to the product, it becomes necessary to have some mechanism to track its progress.
- The goals of scheduling are to know which work has been completed, how much work is still left to do, and when it will all be finished.

Software Design Documents

- A list of a few common software design documents
  - Architecture: a document that describes the overall design of the software, including descriptions of all the major pieces and how they interact with each other.
  - Data flow diagram: a formalized diagram that shows how data moves through a program.
  - State transition diagram: another formalized diagram that breaks the software into basic states, or conditions, and shows the means for moving from one state to the next.
  - Flowchart: the means for depicting a program's logic.
  - Commented code.

Test Documents

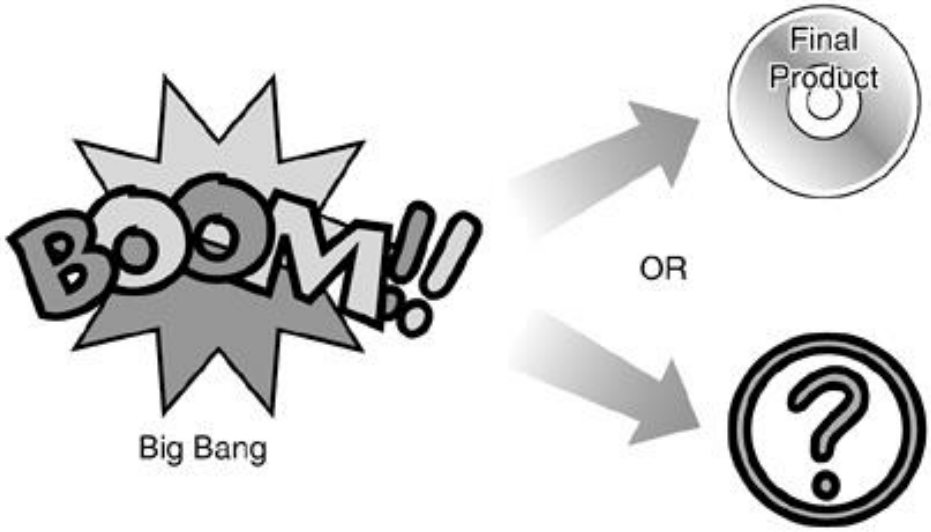
- Test documentation is integral to what makes up a software product. Software testers must plan and document their work.
  - Test plan: use to describe the overall method to be used to verify that the software meets the product specification and the customer's needs.
  - Test cases list: the specific items that will be tested and describe the detailed steps that will then verify the software.
  - Defect reports: summarization of the problems found by using the test cases.
  - Test tools and automation.
  - ……

Software Development Lifecycle

- There are many different methods that can be used for developing software, four frequently used models listed here.
  - Big-bang (大爆炸模型)
  - Code-and-Fix (边写边改模型)
  - Waterfall (瀑布模型)
  - Spiral (螺旋模型)
- 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.

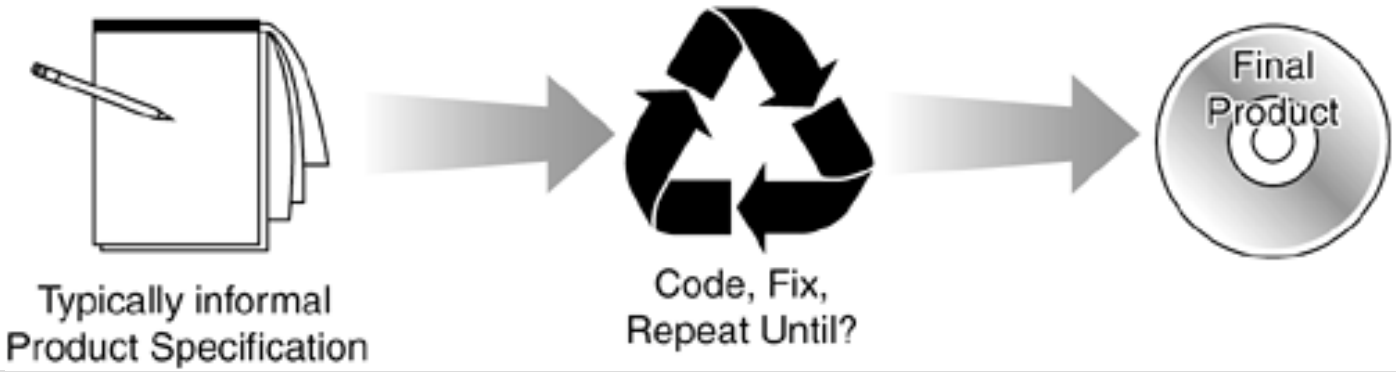
Big-Bang Model

- The big-bang model for software development involves huge amount of matter (people and money) which is put together. A lot of energy is expended often violently and outcomes the perfect software product…or it doesn't.
- The big-bang model is by far the simplest method of software development.
- There is no (or little) testing process in the model, so try to stay away from testing in this model

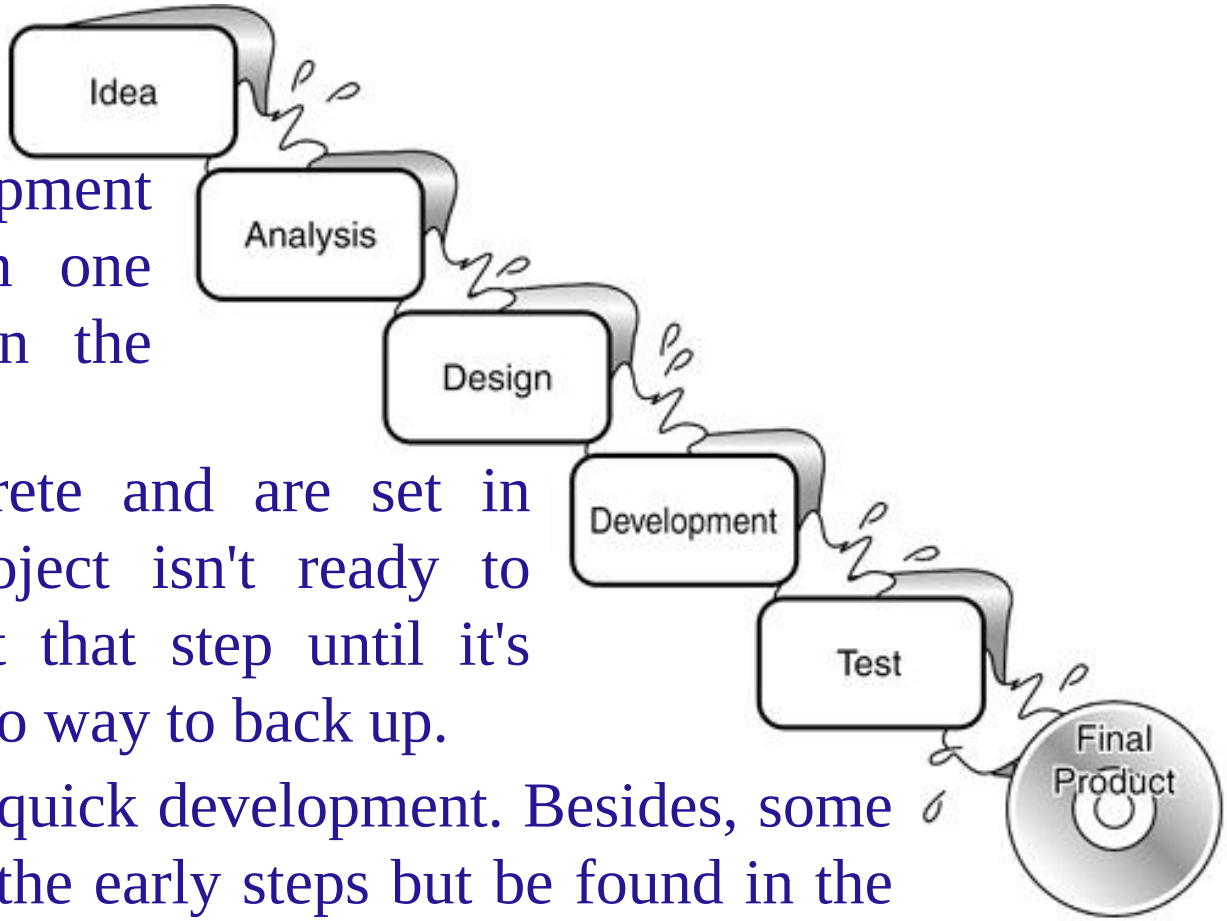


Code-and-Fix Model

- Procedurally, the code-and-fix model is a step up from the big-bang model, for it at least requires some idea of what the product requirements are.
- The code-and-fix model repeats until someone gives up. It works very well for **small projects**.
- When the new version comes out, the testing for the old version may not have finished.



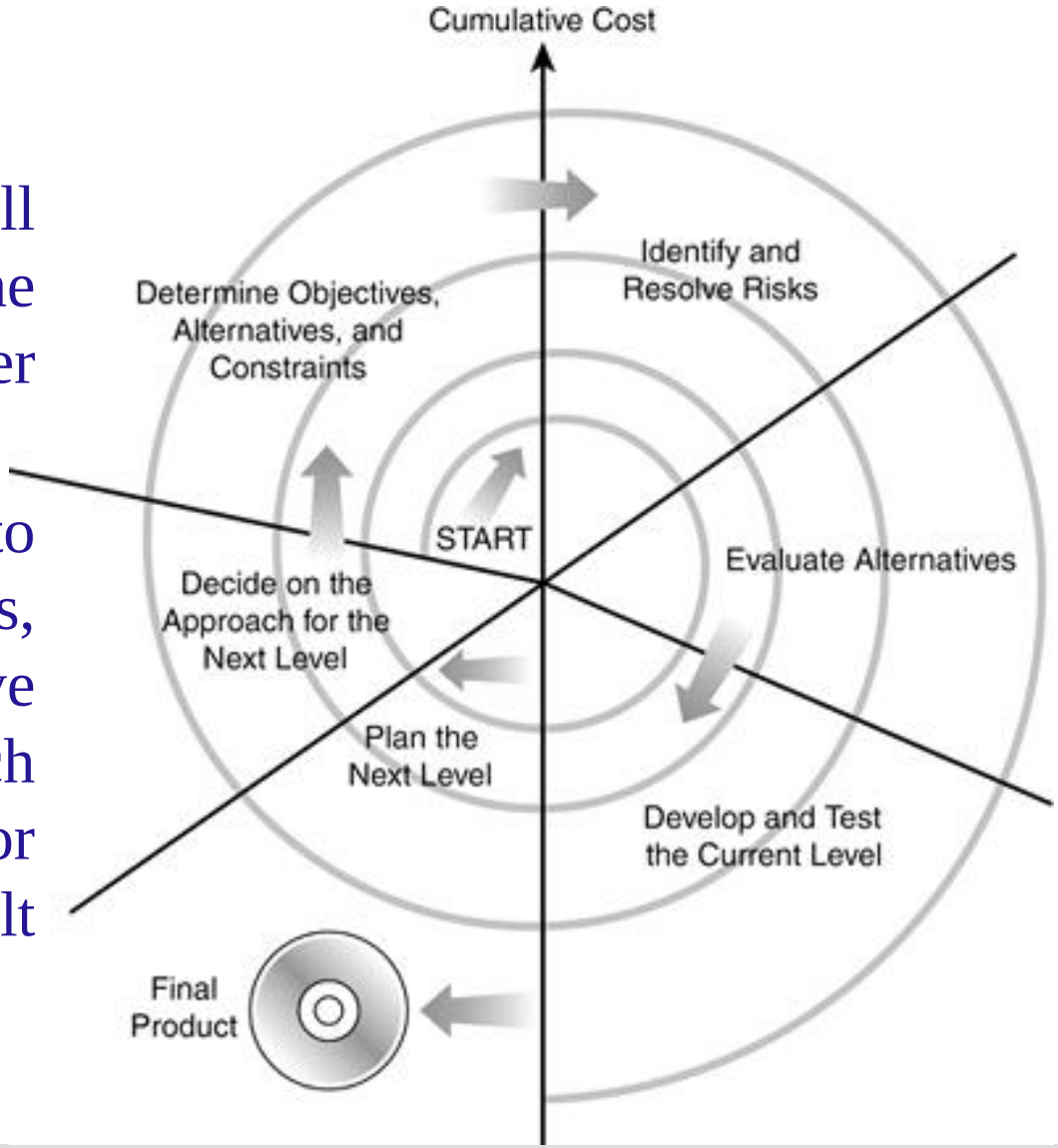
Waterfall Model



- The software development process flows from one step to the next in the waterfall model.
- The steps are discrete and are set in advance. If the project isn't ready to progress, it stays at that step until it's ready. And there is no way to back up.
- It is not suitable for quick development. Besides, some bugs may appear in the early steps but be found in the final step, which is against the testing criteria.

Spiral Model

- The spiral model starts small and gradually expands as the project becomes better defined and gains stability.
- It will take a long time to develop the model. Besides, although the customers have chance to participate each step, it is still difficult for them to convince the result is controllable.



Summary

- As can be seen, there's no definitive approach.
  - The four models presented here are just examples. There are many others and lots of variations of these.
  - Each company, each project, and each team will choose what works for them.
  - Sometimes they may choose right, but sometimes they may choose wrong.
- A software tester will work the best he can in the development model he is in, applying the testing skills (that will be learned later) to create the best software possible.

Testing Axioms

- Here are some testing axioms that are worth repeating.
  - It's impossible to test a program completely.
  - Software testing is a risk-based exercise, i.e. one should both test everything and reduce the huge domain of the tests, otherwise he would be likely to miss some bugs.
  - Testing can show that bugs exist, but it can't show that bugs don't exist.
  - **The more bugs one have found, the more bugs there are.**
  - **The more one tests software, the more immune it becomes to his tests, i.e. the pesticide paradox** (杀虫剂悖论).

Testing Axioms

- **Not all the found bugs will be fixed**. Generally four reasons: **(i) There's not enough time; (ii) it's really not a bug; (iii) it's too risky to fix; (iv) it's just not worth it.**
- Software testers may not be popular in a project team. Here are a couple of tips to keep the peace with your fellow teammates:
  - Find bugs early.
  - Temper your enthusiasm.
  - Avoid always reporting bad news

Testing Axioms

- Product specifications are never final.
- Software testing is a disciplined technical profession.
- Software testing can now be a career choice, a job that requires training and discipline, and allows for advancement.

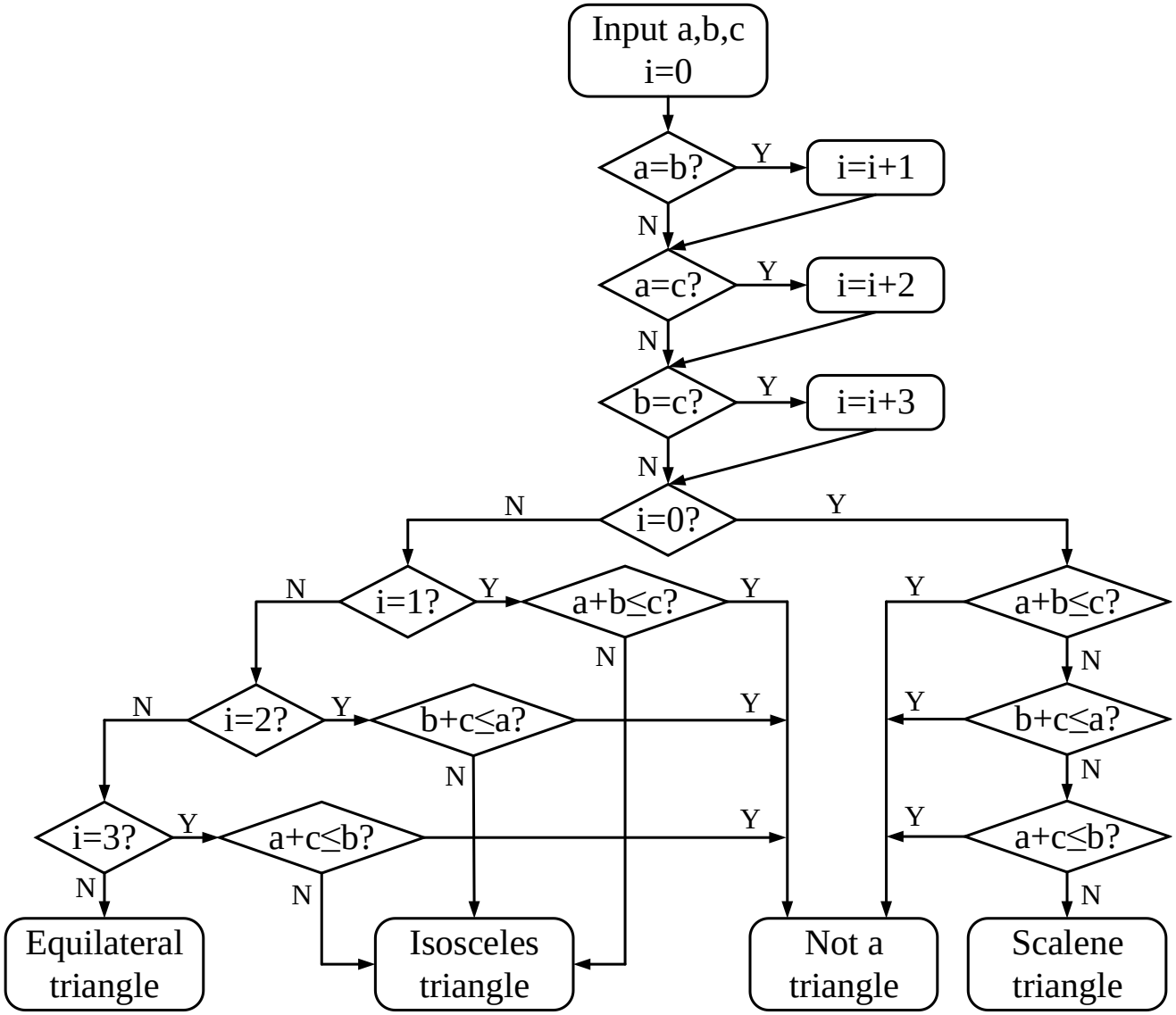
Test Cases

- 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。
- Once one knows the inputs and outputs of the software he is about to test, the next step is to start defining the test cases.
- Selecting test cases is the single most important task that software testers do.

First Attempt of a Test Example

- **Description of the example**
  - 运行在8位处理机字长的程序从一个输入对话框中读取三个正整数值。这三个整数值代表了三角形三边的长度。程序显示提示信息，指出该三角形究竟是不等边三角形、等腰三角形还是等边三角形。
- **请设计测试用例，对上述程序进行测试**

Flowchart


Input a,b,c
i=0
a=b?
Y
i=i+1
N
a=c?
Y
i=i+2
N
b=c?
Y
i=i+3
N
i=0?
N
Y
i=1?
N
Y
a+b≤c?
Y
N
i=2?
N
Y
b+c≤a?
Y
N
i=3?
N
Y
a+c≤b?
Y
N
Equilateral triangle
Isosceles triangle
Y
a+b≤c?
N
Y
b+c≤a?
N
Y
a+c≤b?
N
Not a triangle
Scalene triangle

Interface of the Software

```
Please input the three edges (separated with a space):
```

Interface of the Software

```
Please input the three edges (separated with a space):
3 5 6
```

Interface of the Software

```
Please input the three edges (separated with a space):
3 5 6
Scalene triangle.
```

How to Select the Test Cases

- Since our goal is to find bugs, then what test cases should we use?

How to Select the Test Cases

- Since our goal is to find bugs, then what test cases should we use?
- Consider the cases below according to the example.
  1. Illegal input: what if they are not three integers?
  2. Overflow input: what if there is an overflow in the addition calculation?
  3. Verification of the input: if the legal values are input, will the software show the right result as we have anticipated?

Selection of the Test Cases

- What are the corresponding outputs according to the inputs?
  1. {x y 1}、{-1 2 1.2}、{1 2}…
  2. {70 60 50}、{10, 120, 115}…
  3. {3 4 5}、{1, 2, 3}…

Difficulties

- Software testing is difficult because:
  - Bugs may appear in any development step.
  - There do exist some approaches or rules for selecting test cases, but we may not find the cases only by following them. Occasionally, we have to test the aspects that are not mentioned in the specification, which further aggravates the difficulties.

Test-to-Pass & Test-to-Fail

- Two fundamental approaches to testing software
  - Test-to-pass (通过性测试)
    - Designing and running test cases to assure only what the software can minimally work is called test-to-pass.
  - Test-to-fail (失效性测试)
    - Designing and running test cases with the sole purpose of breaking the software is called test-to-fail or error-forcing.

Ways to Test a Software

- According to the way of testing
  - Static testing: testing something that's not running examining and reviewing it in a computer
  - Dynamic testing: testing something by running and using the software in a computer
- According to the aspects of testing
  - Black-box testing: functional or behavioral testing where one only knows what the software is supposed to do and he can't look in the box to see how it operates
  - White-box testing: structural testing where one has access to the program's code and can examine it for clues to help him with his testing he can see inside the box.

Ways to Test a Software

- Hence, we have four basic software testing techniques:
  - Static Black-Box Testing
  - Dynamic Black-Box Testing
  - Static White-Box Testing
  - Dynamic White-Box Testing
- Relatively, it is easier to undertake the static testing than the dynamic testing. Thus, great emphasis will be put on dynamic black-box testing and dynamic white-box testing, in which some useful techniques will be learned.

Dynamic Black-Box Testing

- Several techniques for dynamic black-box will be learned then:
  - Equivalence Partitioning
  - Boundary Value Analysis
  - Decision Table
  - Cause-Effect Diagram
  - Error Guessing

Dynamic White-Box Testing

- Several techniques for dynamic white-box will be learned then:
  - Statement Coverage, Decision Coverage, Condition Coverage, Decision / Condition Coverage, Multiple Condition Coverage
  - Path Coverage

THANK YOU!