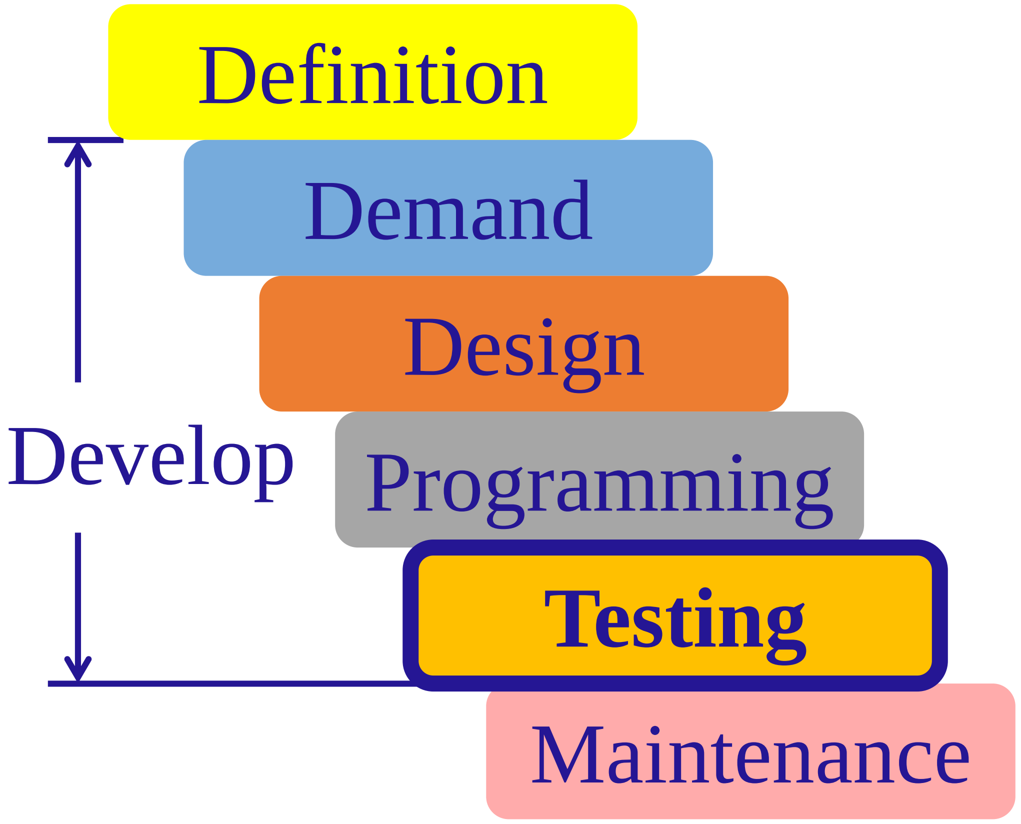
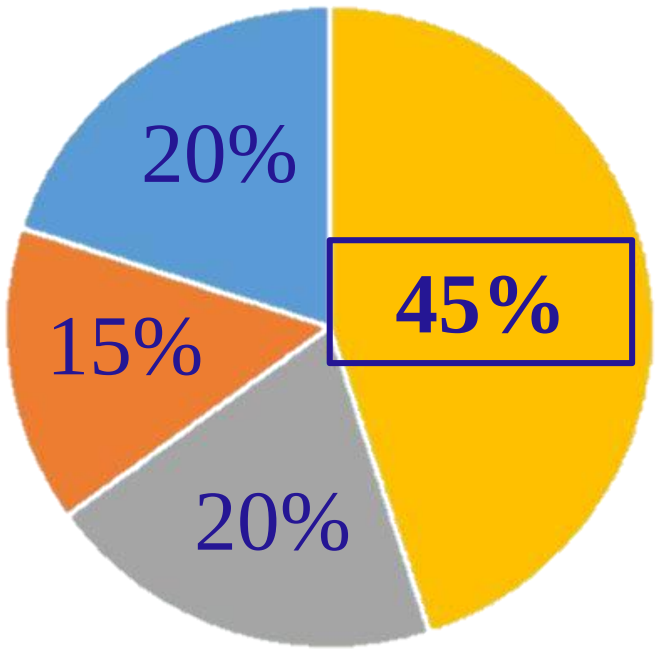
SOFTWARE TESTING

苏临之

sulinzhi029@nwu.edu.cn

Software Life Cycle





Error, Failure, Fault and Defect

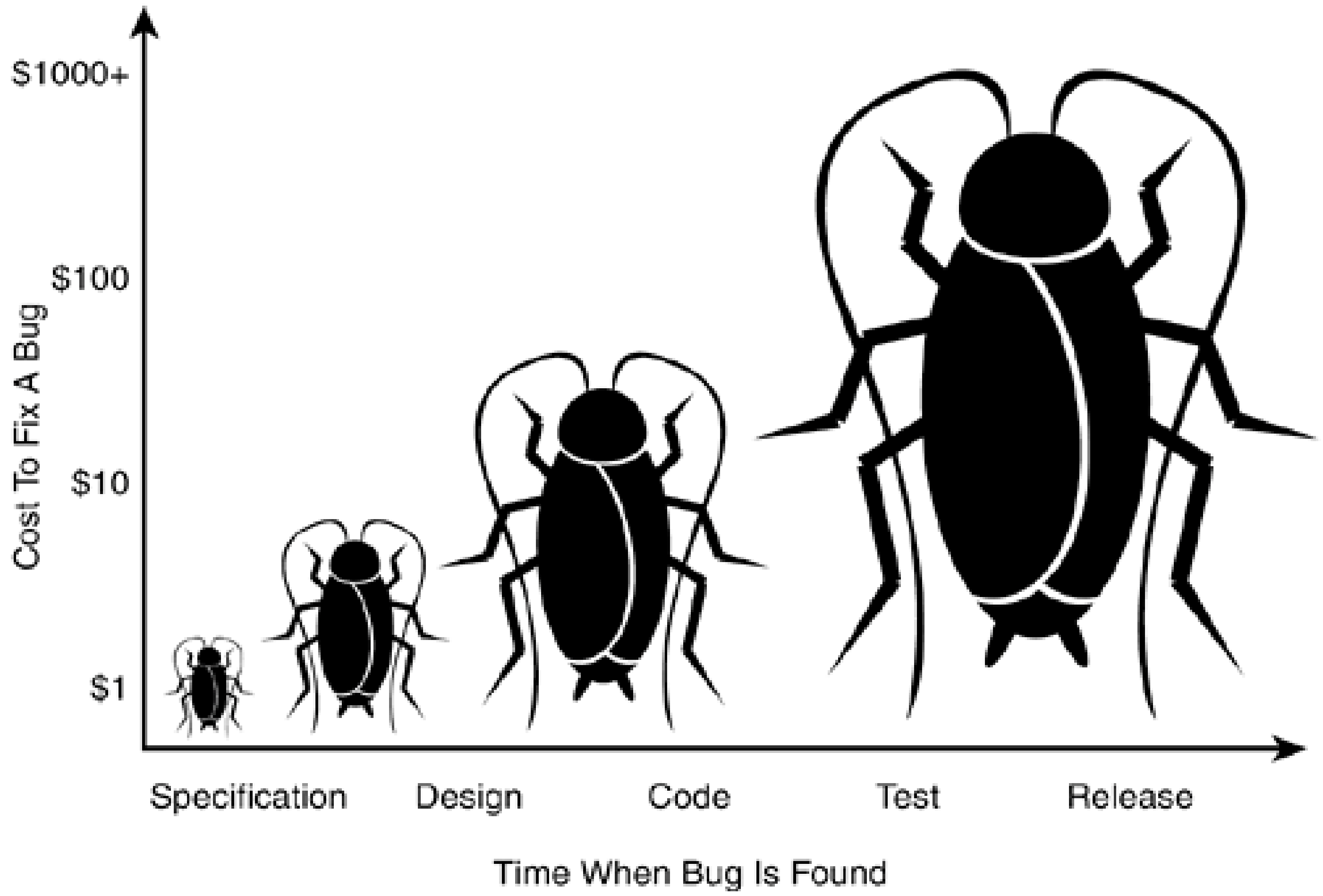
- Error: The software can not operate as fully anticipated.
- Failure: The software engenders wrong results caused by errors.
- Fault: The existing physical problems that may or may not result in failures.
- Defect: The software does not work as required. A software defect, or bug as often referred to,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



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.

Software Development Process

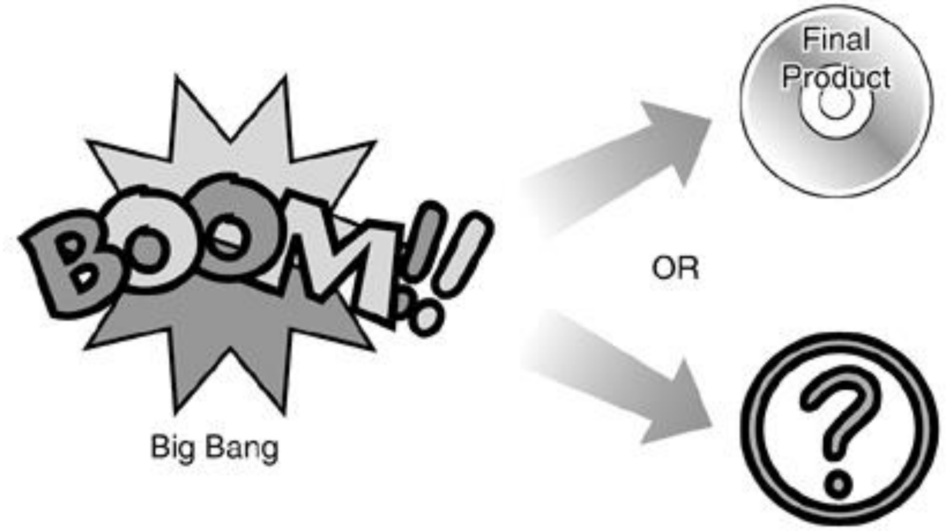
- In a software development process, some basic elements are indispensable:
  1. Customer requirements (用户需求)
  2. Specifications (规格说明)
  3. Schedules (开发计划详述)
  4. Software design documents (软件设计文档)
  5. Test documents (测试文档)

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)
  - Code-and-Fix (边写边改)
  - Waterfall (瀑布)
  - Spiral (螺旋)
- 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.

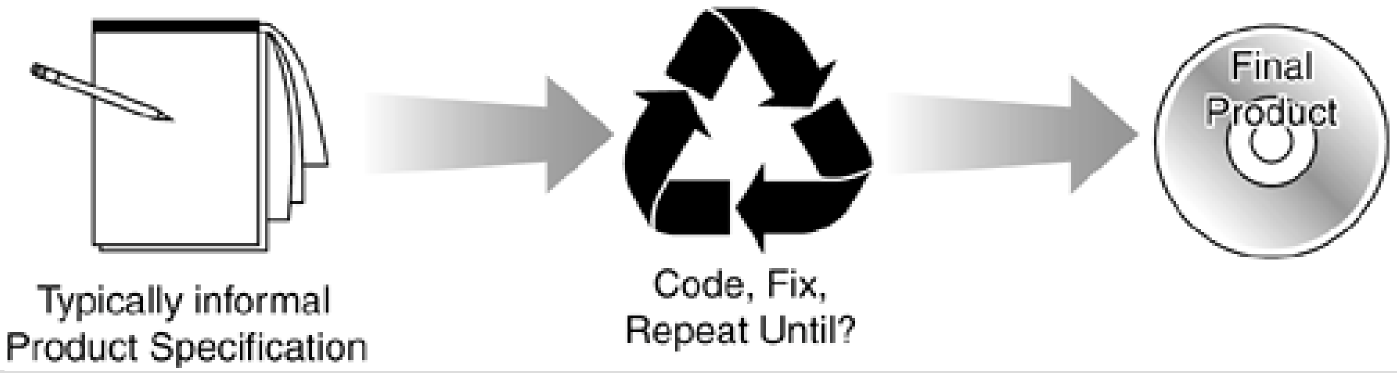
Big-Bang Model

- 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

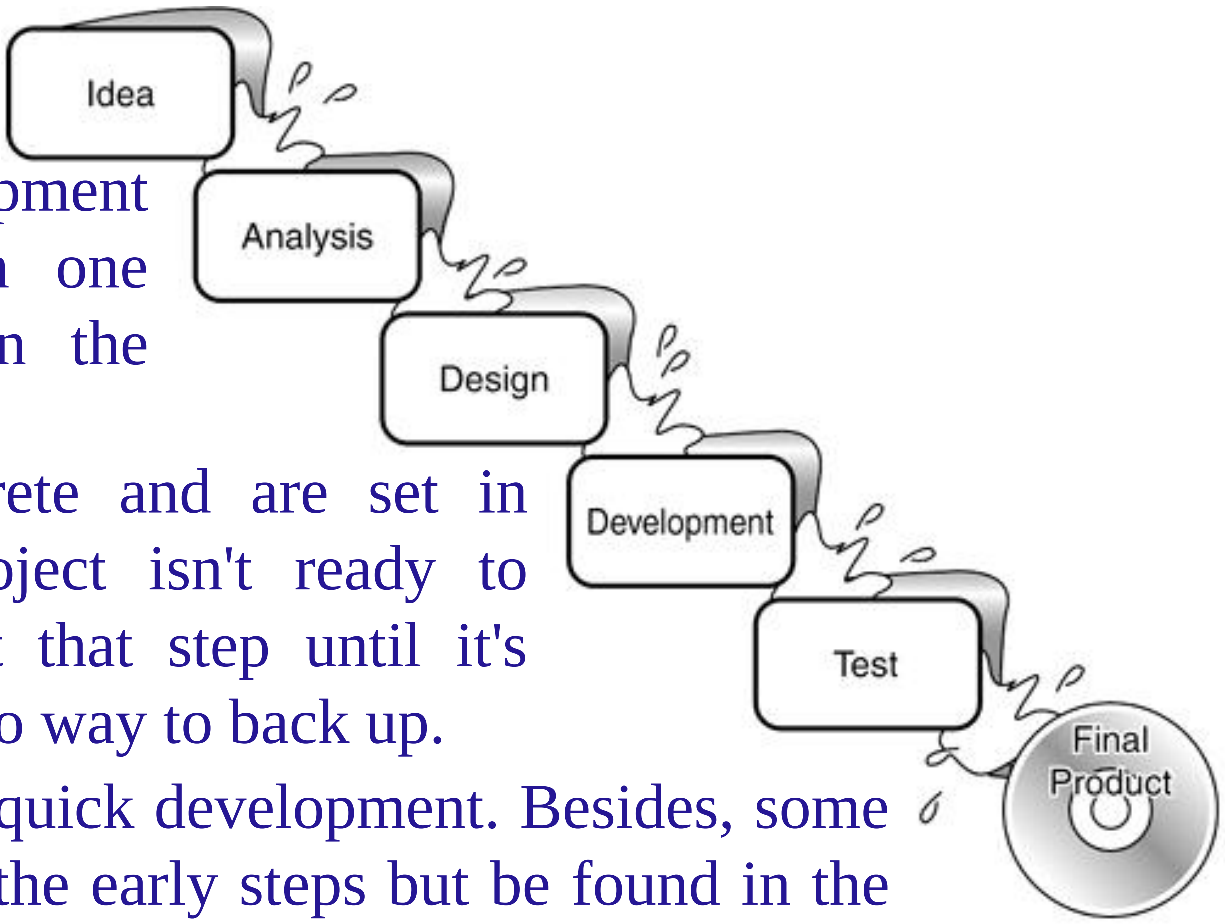


Code-and-Fix Model

- 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.



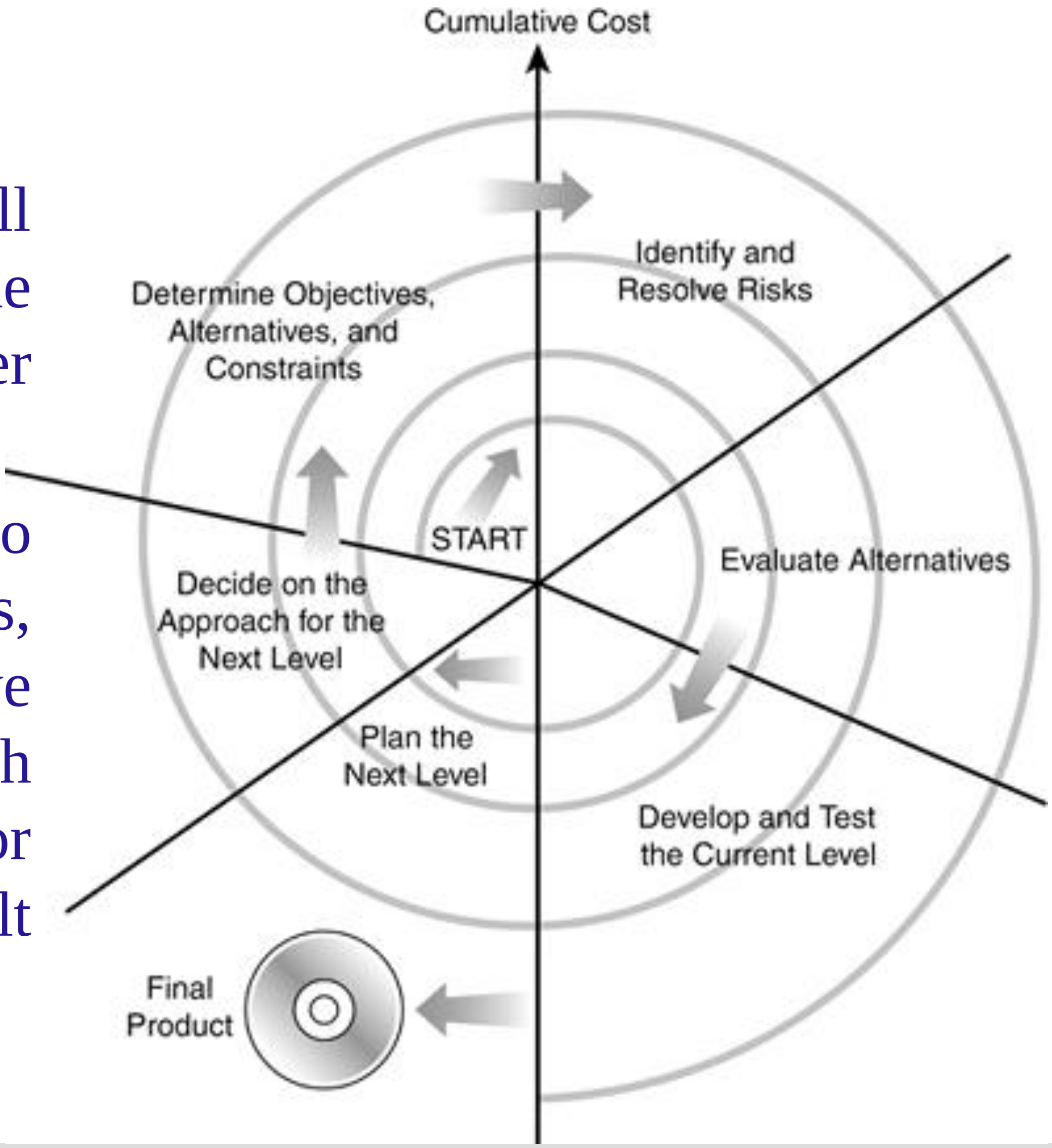
Waterfall Model



- 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.



Cumulative Cost
Identify and Resolve Risks
Determine Objectives, Alternatives, and Constraints
START
Evaluate Alternatives
Decide on the Approach for the Next Level
Plan the Next Level
Develop and Test the Current Level
Final Product

Some Important 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, i.e. the pesticide paradox (杀虫剂效应).

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**

How to Select the Test Cases

- Since our goal is to find defects, 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?

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
  - Dynamic Testing
- According to the aspects of testing
  - Black-Box Testing
  - White-Box Testing

Static Testing

- In static testing, the software program itself is not run in a computer but examined or checked manually. The aspects to be checked involve the code grammar, structure, flowchart, specification and so on. So the proficiency of programming and management is needed.

```
int i, a[10];
for(i=0;i<=10;i++);
    a[i]=i;
```

```
int i, a[10];
for(i=0;i<=9;i++)
    a[i]=i;
```

Static Testing

- Static is not as accurate as dynamic testing. However, it is a successful testing and is able to improve the effectiveness and reliability.
  - We are able to find the bugs in the early stage by using ST, and thus the cost to fix the bugs will be low.
  - Psychologically, the programmer will err less if the bugs are found in the early ST stage.
  - It will be easier to locate and find some similar bugs, reducing the cost of follow-up testing.
  - 30%~70% of the logic and code errors can be found in ST.

Dynamic Testing

- In dynamic testing, the program is run in a computer to show the results, checking the difference between the run results and the anticipated ones. Three elements are indispensable: test cases, operating and the analysis of results.

```
a=[];
for i=1:1000000
    a=[a,i];
end
```

```
N=1000000;
a=zeros(1,N);
for i=1:1000000
    a(i)=i;
end
```

Dynamic Testing

- In dynamic testing, the program is run in a computer to show the results, checking the difference between the run results and the anticipated ones. Three elements are indispensable: test cases, operating and the analysis of results.

```
a=[];
for i=1:1000000
    a=[a,i];
end
The elapsed time is 2393.7s.
```

```
N=1000000;
a=zeros(1,N);
for i=1:1000000
    a(i)=i;
end
The elapsed time is 0.0101s.
```

Black-Box Testing

- In the black-box testing, the tester only knows what the software is supposed to do, whereas he can't look in the box to see how it operates.
- Black-box testing is sometimes referred to as functional testing or behavioral testing.
- Actually, functional testing is after unit testing and integration testing, where the detailed testing has ended. So all one needs to do is test its overall behavior, and hence the BBT is employed.

White-Box Testing

- In white-box testing, the software tester has access to the program's code and can examine it for clues to help him with his testing.
- Based on what he sees, the tester may determine that certain numbers are more or less likely to fail and can tailor (adjust) his testing based on that information.
- In unit testing and integration testing, the WBT is usually employed. Besides, the testing tool "JUnit" which is based on Java programming is also WBT.

Four Basic Techniques

- Hence, we have four basic software testing techniques, which will be learned one by one:
  - Static Black-Box Testing
  - Dynamic Black-Box Testing
  - Static White-Box Testing
  - Dynamic White-Box Testing

Static Black-Box Testing

- Testing the specification (spec for short)
  - The specification is a document, not an executing program, so it's considered static. Therefore, by SBBT is meant the testing of the spec.
- The form of specification
  - Textual, graphical, self-documenting computer language

High-Level Review of the Spec

- The first step is to stand back and view it from a high level (宏观审查) rather than to jump in and look for specific bugs.
- Examine the spec for large fundamental problems, oversights, and omissions.
- This research is a means to better understand what the software should do. If one has a better understanding of the whys and hows behind the spec, he will be much better at examining it in detail.

High-Level Review of the Spec

- Pretend to be the customer.
  - Do some research about who the customers will be.
  - It's important to understand the customer's expectations.
  - Assume nothing, i.e. don't skip anything that is difficult to understand, for the spec will be used to design tests.
  - Software security, which must be specified in detail, should never be overlooked.

High-Level Review of the Spec

- Research existing standards and guidelines.
  - Corporate terminology and conventions
  - Industry requirements.
  - Government standards.
  - Graphical user interface (GUI).
  - Security standards

High-Level Review of the Spec

- Review and test similar software. Some things to look for when reviewing competitive products include:
  - Scale
  - Complexity
  - Testability
  - Quality & Reliability
  - Security

Low-Level Review of the Spec

- After completing the high-level review of the product specification, one will have a better understanding of what your product is and what external influences affect its design.
- Armed with this information, he can move on to testing the specification at a lower level (微观审查).
  - Specification Attributes (属性)
  - Specification Terminology (术语)

Specification Attributes Checklist

- Complete
- Accurate
- Precise, unambiguous, and clear
- Consistent
- Relevant
- Feasible
- Code-free
- Testable

Specification Terminology Checklist

- Always, Every, All, None, Never.
- Certainly, Therefore, Clearly, Obviously, Evidently.
- Some, Sometimes, Often, Usually, Ordinarily, Customarily, Most, Mostly.
- Etc., And So Forth, And So on, Such as.
- Good, Fast, Cheap, Efficient, Small, Stable.
- Handled, Processed, Rejected, Skipped, Eliminated.
- If…Then…(but missing Else).

Dynamic Black-Box Testing

- Suppose the software is a black box, and for each input *F*, the box processes it by using its inner function *H*, and finally we can get an output *Y*.
- So if there is an error in the box (*H*), then *Y* will also be wrong. Thus, we design proper test cases (as *F*) and can check *H* by examining *Y*.

$$\begin{tikzcd} F \arrow[r] & \boxed{H} \arrow[r] & Y \end{tikzcd}$$

Dynamic Black-Box Testing

- If the entire available inputs are viewed as a universal set (universe), then the test cases comprise one subset. So to design test cases entails finding a specific subset of the input universe.
- Given fixed time and space, which subset will lead to finding the most bugs? That serves as the key point to design test cases, and several techniques have been proposed.

An Example

- **Description**
  - 某企业的报表处理系统要求用户输入处理报表的日期，日期限制在2003年1月至2008年12月，即系统只能对该段期间内的报表进行处理，如日期不在此范围内，则显示输入错误信息。输入信息要求：系统日期规定由表征年、月的6位连续数字字符组成，前4位代表年，后2位代表月。
- **Please consider why it is suitable to use dynamic black-box testing.**

Analysis

- In the example, the aim is to test the function of the system, i.e. it is not necessary to know the way it works. Actually, the system must consist of several units and somehow be combined together, but we do not care that. All we need to do is check whether the outputs to test cases are what they should be.
- Therefore, it is suitable to use dynamic black-box testing.

An Example

- **Description**
  - 某企业的报表处理系统要求用户输入处理报表的日期，日期限制在2003年1月至2008年12月，即系统只能对该段期间内的报表进行处理，如日期不在此范围内，则显示输入错误信息。输入信息要求：系统日期规定由表征年、月的6位连续数字字符组成，前4位代表年，后2位代表月。
- **Please design test cases to check the function for date input.**

Techniques

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

THANK YOU !