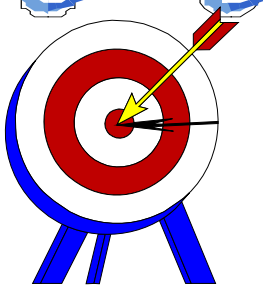
Object-Oriented Systems Development:
Using the Unified Modeling Language

Chapter 13:
Software Quality Assurance



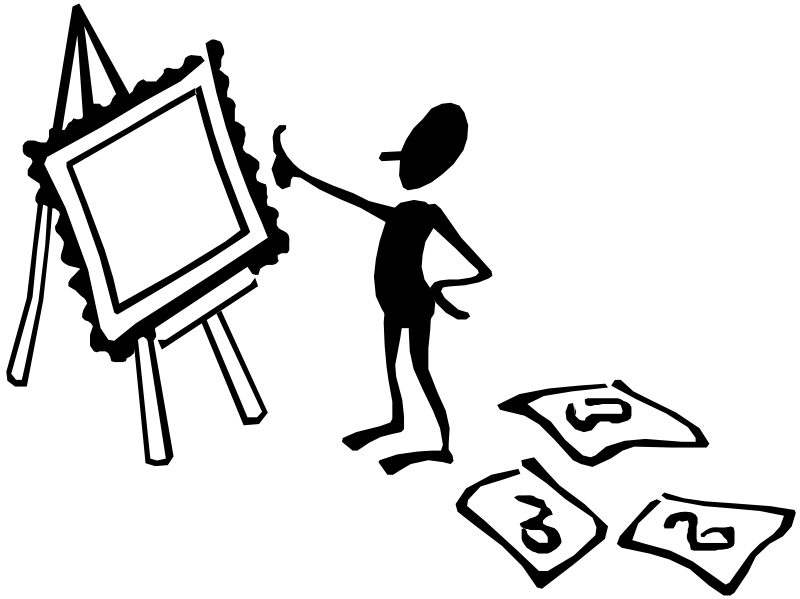
Goals

- **Bugs and Debugging**
- **Testing strategies.**
- **The impact of an object orientation on testing.**
- **How to develop test cases.**
- **How to develop test plans.**

Introduction

Two issues in software quality are:

- ***Validation*** **or user satisfaction**
- ***Verification*** **or quality assurance.**

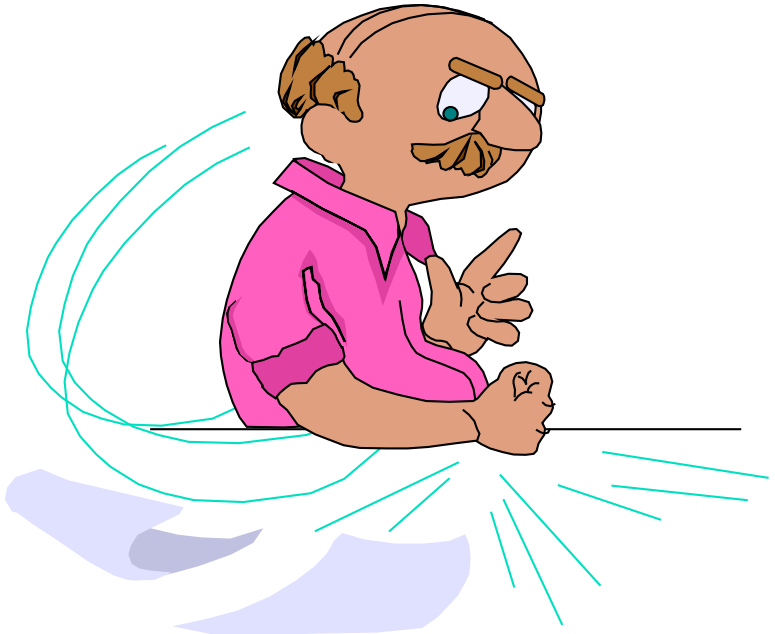


Introduction (Con't)

- **Elimination of the syntactical bug is the process of debugging.**
- **Detection and elimination of the logical bug is the process of testing.**

Introduction (Con't)

- **Error Types:**
  - **Language errors or syntax errors**
  - **Run-time errors**
  - **Logic errors**



Identifying Bugs and Debugging

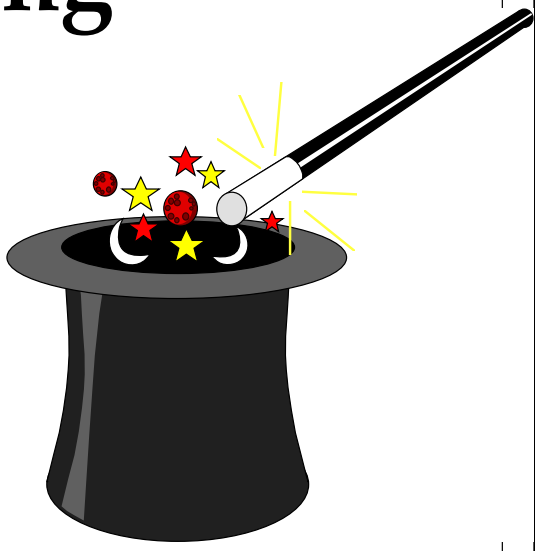
- **The first step in debugging is recognizing that a bug exists.**
- **Sometimes it's obvious, the first time you run the application, it shows itself.**
- **Other bugs might not surface until a method receives a certain value, or until you take a closer look at the output.**

Identifying Bugs and Debugging (Con't)

There are no magic tricks for debugging.

However, these steps might help:

- **Selecting appropriate testing strategies.**
- **Developing test cases and sound test plan.**
- **Debugging tools.**



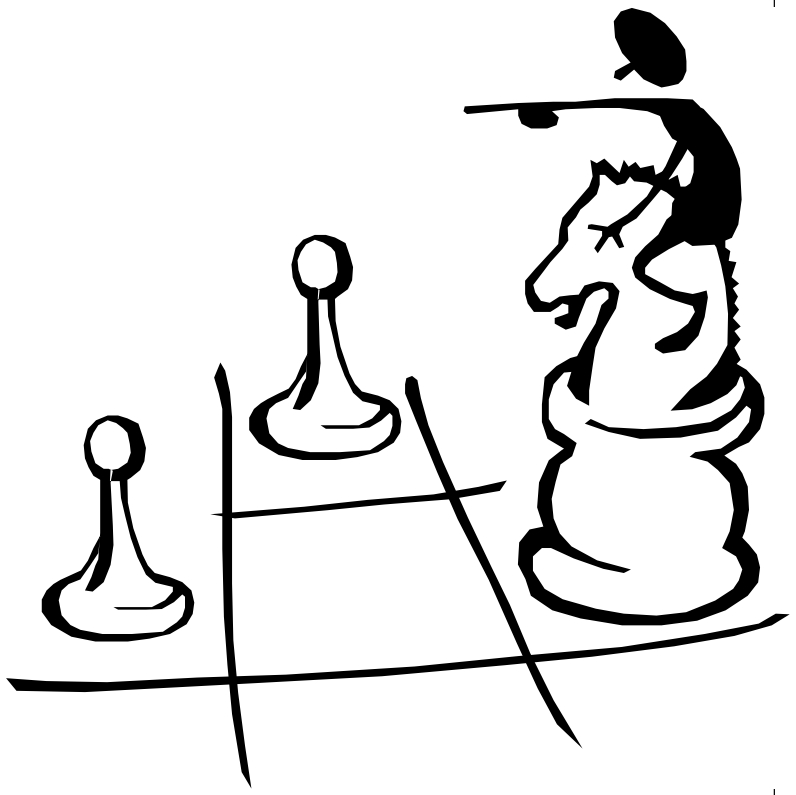
Debugging Tools

- **Debugging tools are a way of looking inside the program to help us determine what happens and why.**
- **It basically gives us a snapshot of the current state of the program.**

Testing Strategies

There are four types of testing strategies These are:

- **Black Box Testing**
- **White Box Testing**
- **Top-down Testing**
- **Bottom-up Testing**

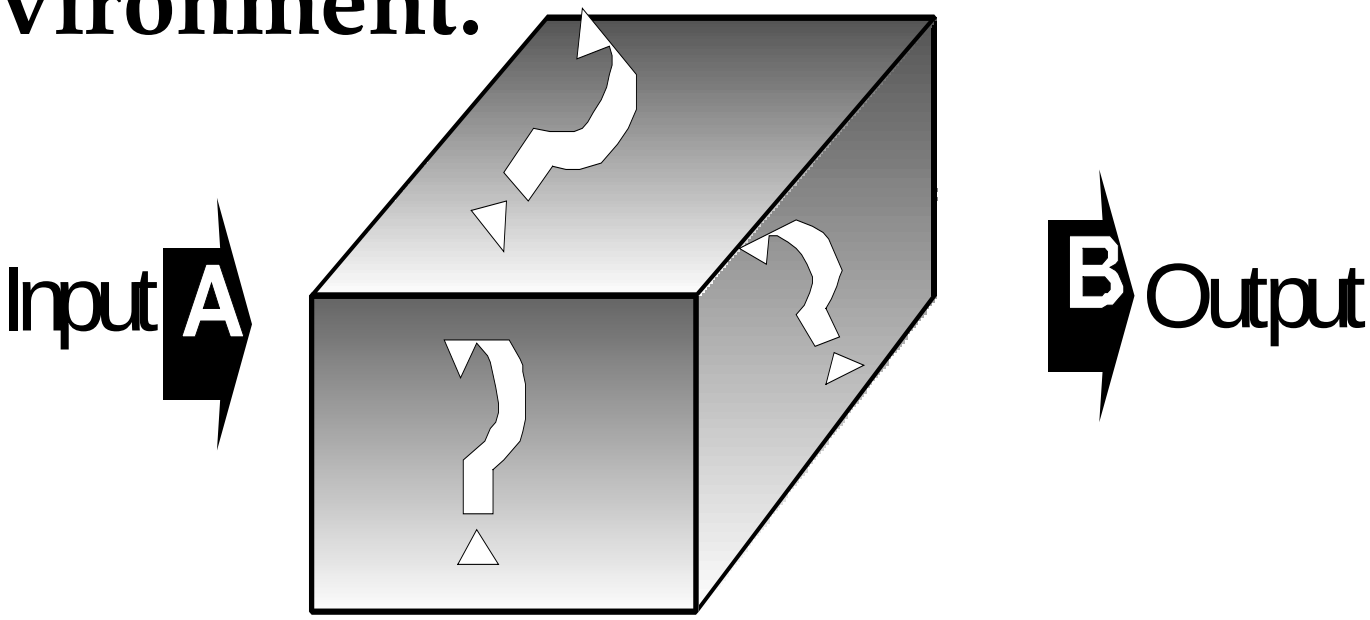


Black Box Testing

- **In a black box, the test item is treated as "black" whose logic is unknown.**
- **All that's known is what goes in and what comes out, the input and output**

Black Box Testing(Con't)

- **Black box test works very nicely in testing objects in an O-O environment.**

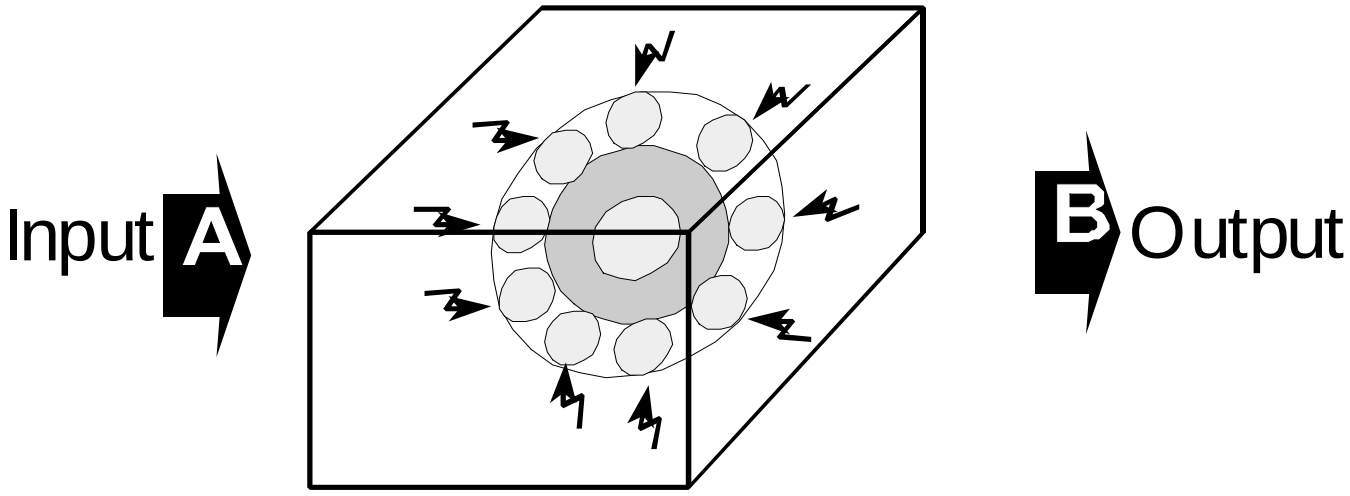


Black-Object Testing

- **Once you have created fully tested and debugged classes of objects you will put them into library for use or reuse.**

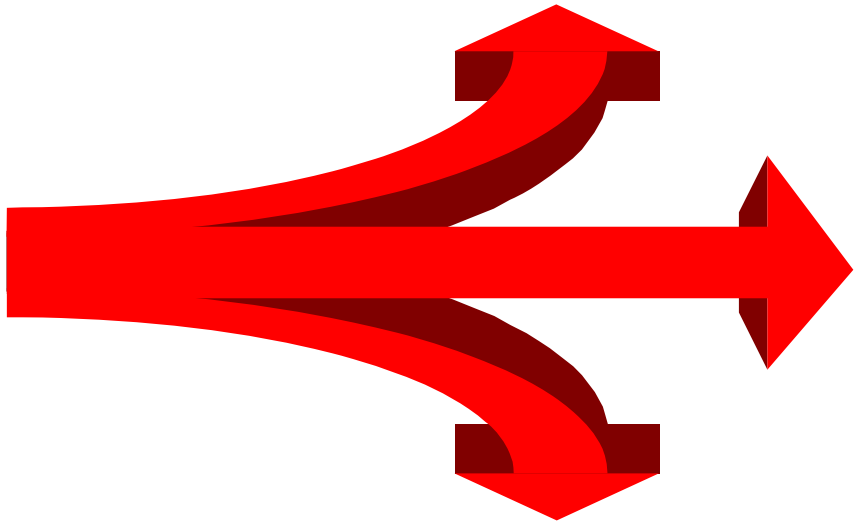
White Box Testing

- **White box testing assumes that specific logic is important, and must be tested to guarantee system's proper functioning.**



White Box Testing (Con't)

- **One form of white box testing is called *path testing*.**
- **It makes certain that each path in a program is executed at least once during testing.**



White Box Testing (Con't)

Two types of *path testing* are:

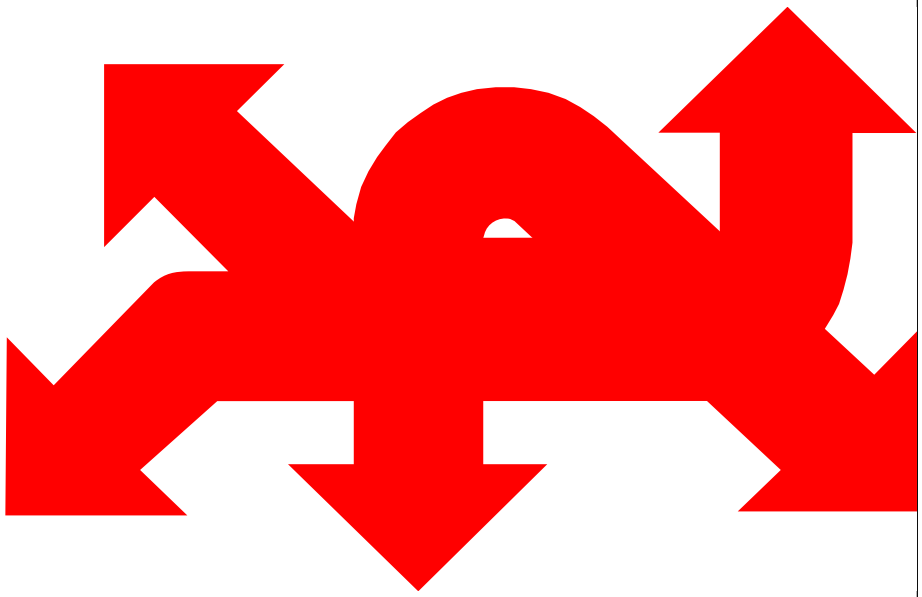
- **Statement testing coverage, and**
- **Branch testing coverage.**

Statement Testing Coverage

- **Every statement in the program must be executed at least once.**
- **Beizer states, *"Testing less than this for new software is unconscionable and should be criminalized."***

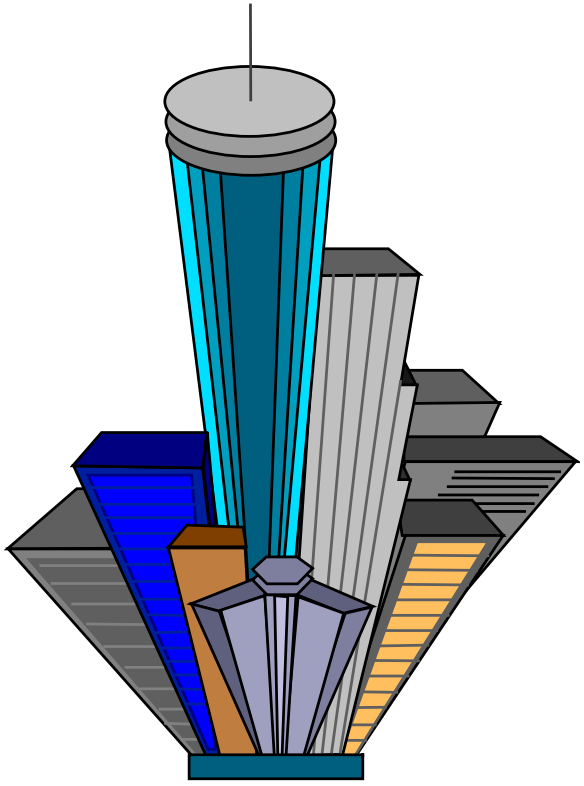
Branch Testing Coverage

- **Every branch alternative must be executed at least once under some test.**



Top-down Testing

- **It assumes that the main logic of the application needs more testing than supporting logic.**

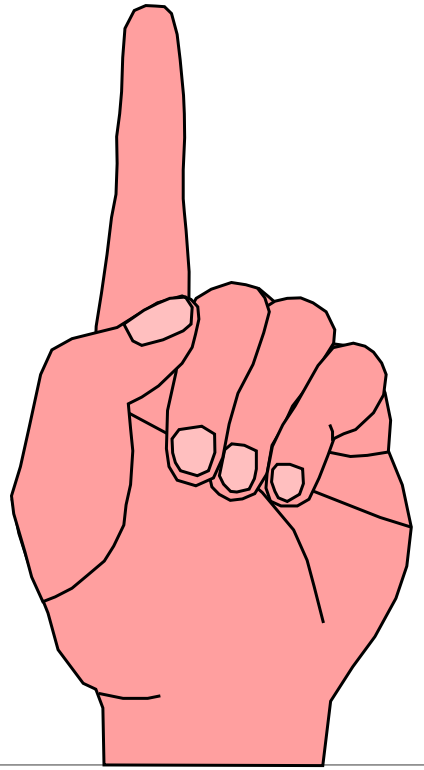


Bottom-up Approach

- **It takes an opposite approach.**
- **It assumes that individual programs and modules are fully developed as stand alone processes.**
- **These modules are tested individually, then combined for integration testing.**

Bottom-up Approach (Con't)

- **Bottom-up testing is more appropriate for O-O systems.**
- **You test each object then combine them and test the whole system by utilizing the top-down approach.**

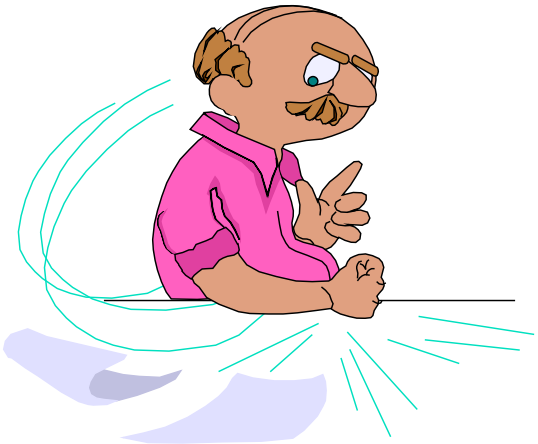


Impact of Object Orientation on Testing

- **The impact of an object orientation on testing is summarized by the following:**
  - **Some types of error could become less plausible (not worth testing for).**
  - **Some types of error could become more plausible (worth testing for now).**
  - **Some new types of error might appear.**

Impact of Inheritance in Testing

- **Inheritance does make testing a system more difficult.**
- **If you do not follow the OOD guidelines, you will end up with objects that are extremely hard to debug and maintain.**



Reusability of Tests

- **The simpler is a test, the more likely it is to be reusable in subclasses.**
- **However, simple tests tend to find only the faults you specifically target; complex tests are better at both finding those faults and stumbling across others by sheer luck.**

Test Cases

- **All methods of your system must be checked by at least one test.**
- **Construct some test input cases, then describe how the output will look like.**
- **Compare the outcomes with the expected output.**

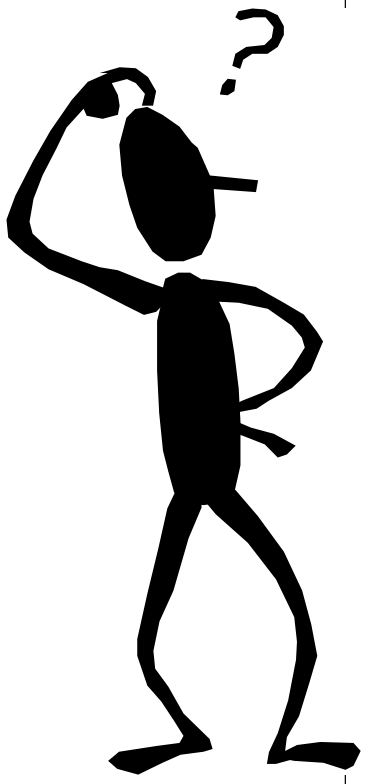
Objective of Testing

- **Myers describes testing as follows :**
- **Testing is the process of executing a program with the intent of finding errors.**
- **A good test case is the one that has a high probability of detecting an as-yet undiscovered error.**
- **A successful test case is the one that detects an as-yet undiscovered error.**

Guidelines For Developing Test Cases

- **Test case is a set of *What-if* questions**
- **The general format is:**

  If it receives certain input, it produces certain output.



Guidelines For Developing Test Cases (Con't)

- **Describe which feature or service (external or internal) your test attempts to cover.**
- **If the test case is based on a use case, it is a good idea to refer to the use case name.**
- **Remember that the use cases are the source of test cases.**

Guidelines For Developing Test Cases (Con't)

- **Specify what you are testing and which particular feature (methods).**
- **Specify what you are going to do to test the feature and what you expect to happen.**

Guidelines For Developing Test Cases (Con't)

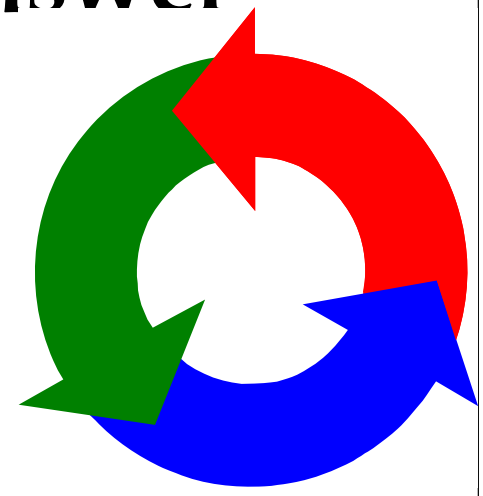
- **Test normal use of the function.**
- **Test abnormal but reasonable use of function.**
- **Test abnormal and unreasonable use of function.**
- **Test the boundary conditions.**

Guidelines For Developing Test Cases (Con't)

- **Test objects' interactions and the messages sent among them. If you have developed sequence diagrams, they can assist you in this process.**
- **When the revisions have been made, document the cases so they become the starting basis for the follow-up test.**

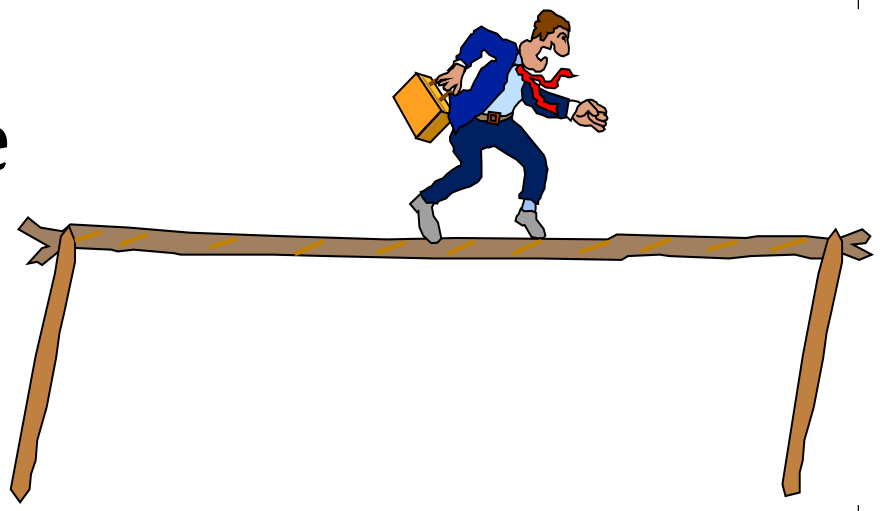
Guidelines For Developing Test Cases (Con't)

- **Attempting to reach agreement on answers generally will raise other what-if questions.**
- **Add these to the list and answer them, repeat the process until the list is stabilized, then you need not add any more questions.**



Test Plan

- **Testing is a balance of art, science, and luck.**
- **Test plan offers a road map for testing activity.**
- **Test plan should state test objective and how to meet them.**



Test Plan (Con't)

- **The plan need not be very large.**
- **Devoting too much time to the plans can be counter productive.**

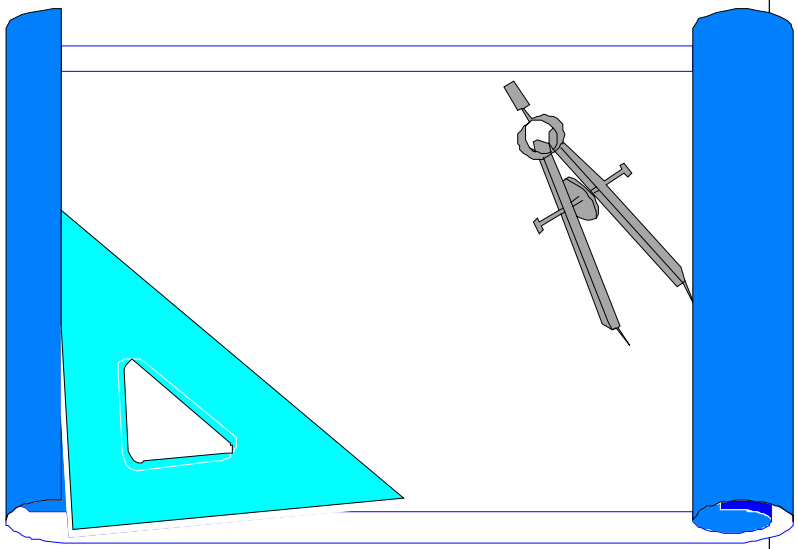
Test Plan (Con't)

The followings are the steps needed to create a test plan:

I. Objectives of Test.

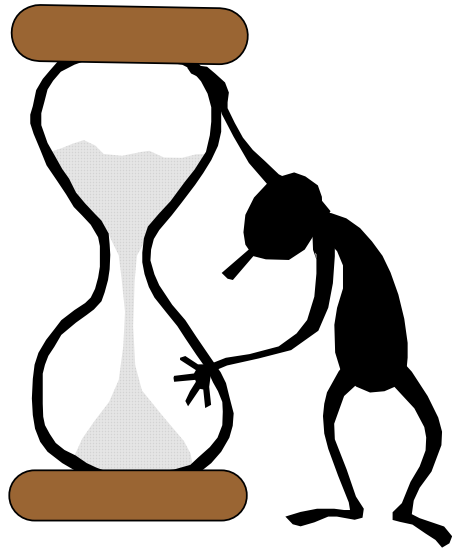
2. Develop Test Case.

3. Identify Bugs and Debugging.



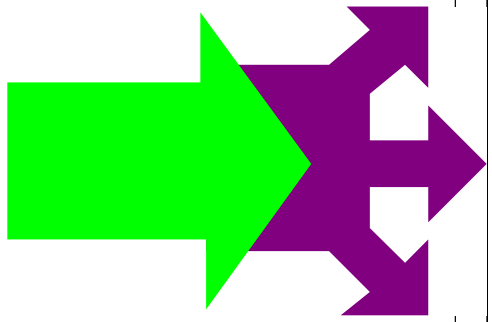
Guidelines For Developing Test Plans

- **Requirements might dictate the format of the test plan.**
- **The test plan should contain a schedule and a list of required resources.**
- **Document every type of testing that you plan to complete.**



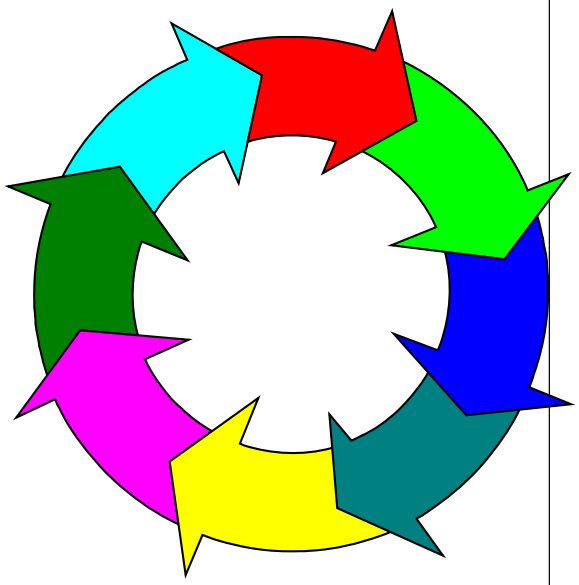
Guidelines For Developing Test Plans (Con't)

- **A configuration control system provides a way of tracking the changes to the code.**
- **Process must be in place to routinely bringing the test plan in sync with the product and/or product specification.**



Continuous Testing

- **Testing must take place on a continuous basis, and this refining cycle must continue throughout the development process until you are satisfied with the results.**



Continuous Testing (Con't)

- **The steps to successful testing:**
  - ***Understand and communicate the business case*** **for improved testing.**
  - ***Develop an internal infrastructure*** **to support continuous testing.**
  - ***Look for leaders*** **who will commit to and own the process.**
  - ***Measure*** **and document your findings in a defect recording system.**
  - ***Publicize improvements*** **as they are made and let people know what they are doing better.**

Myers Debugging Principles
Bug Locating Principles

- **Think.**
- **If you reach an impasse, sleep on it.**
- **If you reach an impasse, describe the problem to someone else.**
- **Use debugging tools**
- **Experimentation should be done as a last resort.**

Myers Debugging Principles (Con't)

- **Where there is one bug, there is likely to be another.**
- **Fix the error, not just the symptom of it.**
- **The probability of the fix being correct, drops as the size of the program increases.**

Myers Debugging Principles (Con't)

- **Beware of the possibility that an error correction creates a new error.**

Summary

- **Testing is a balance of art, science, and luck.**
- **Testing may be conducted for different reasons.**
- **Quality assurance testing looks for potential problems in a proposed design.**

Summary (Con't)

- **We must develop a test plan for locating and removing bugs.**
- **A test plan offers a road map for testing activity; it should state test objectives and how to meet them.**
- **The plan need not be very large; in fact, devoting too much time to the plan can be counterproductive.**

Summary (Con't)

- **There are no magic tricks to debugging; however, by selecting appropriate testing strategies and a sound test plan, you can locate the errors in your system and fix them by utilizing debugging tools.**