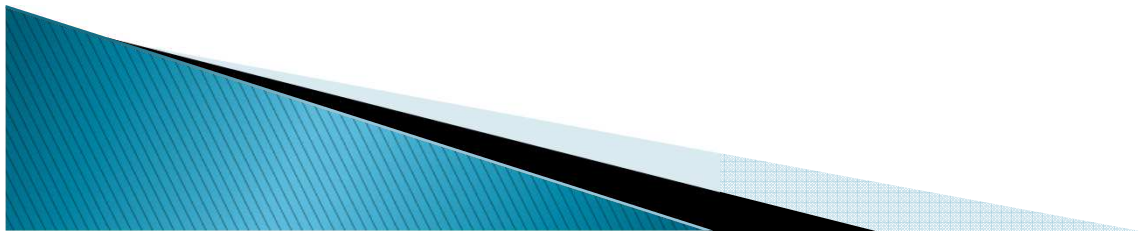
College of MPI of ZJNU

Software Quality Assurance and Testing

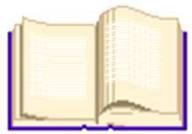
Lecturer: Zhiguo Ding
Phone: 13750985775(660775)
E-mail:dingzhiguo@zjnu.cn
Office: Room 313, Building 21

The Course Overview — Contents

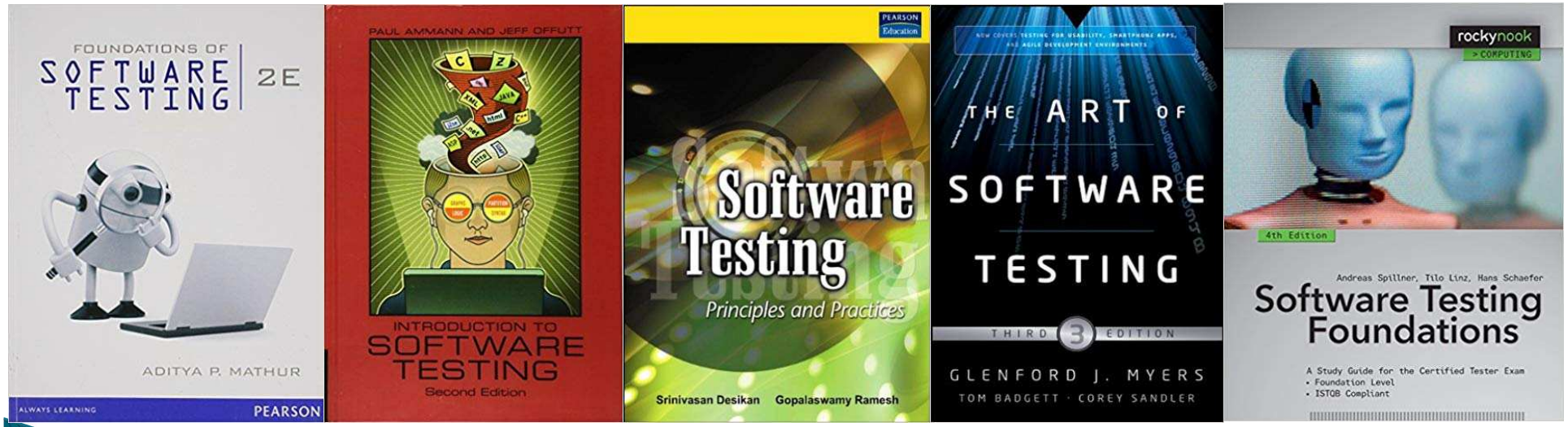
- Chapter 1: Software Testing Introduction
- Chapter 2: Black-Box Testing
- Chapter 3: White-Box Testing
- Chapter 4: Unit Testing
- Chapter 5: Integration Testing
- Chapter 6: Performance Testing
- Chapter 7: System Testing
- Chapter 8: Testing OO Software, Web Applications and Web Services
- Chapter 9: Software Testing Automation and Software Testing Management



The Course Overview — Selected Further Reading



1. **Aditya Mathur, Foundations of Software Testing (2nd Edition), Addison-Wesley Professional; 2014.**
2. **Paul Ammann, Jeff Offutt, Introduction to Software Testing(2nd Edition), Cambridge University Press; 2016.**



The Course Overview — Selected Further Reading

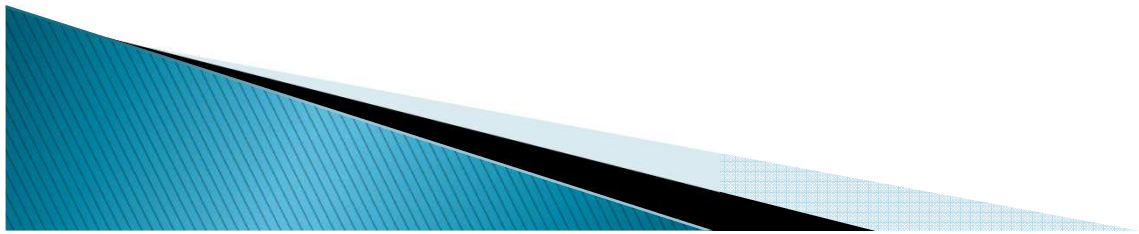
The Course Overview — Online Materials

- http://zjnu.fy.chaoxing.com/portal/courseNetwork/list?pageNum=1&keyword=%E8%BD%AF%E4%BB%B6%E8%B4%A8%E9%87%8F%E4%BF%9D%E8%AF%81%E4%B8%8E%E6%B5%8B%E8%AF%95

序号	封面	课程名称	所属院系	课程负责人	点击量	创建时间
88855400		软件质量保证与测试	数理与信息工程学院	丁智国	9955	2016-06-02

The Course Overview — Assessmen

- Attendance： 5~10%
- Labs and Reports： 30~40%
- Topic Report 20~30%
- Final Exam： 30~40%



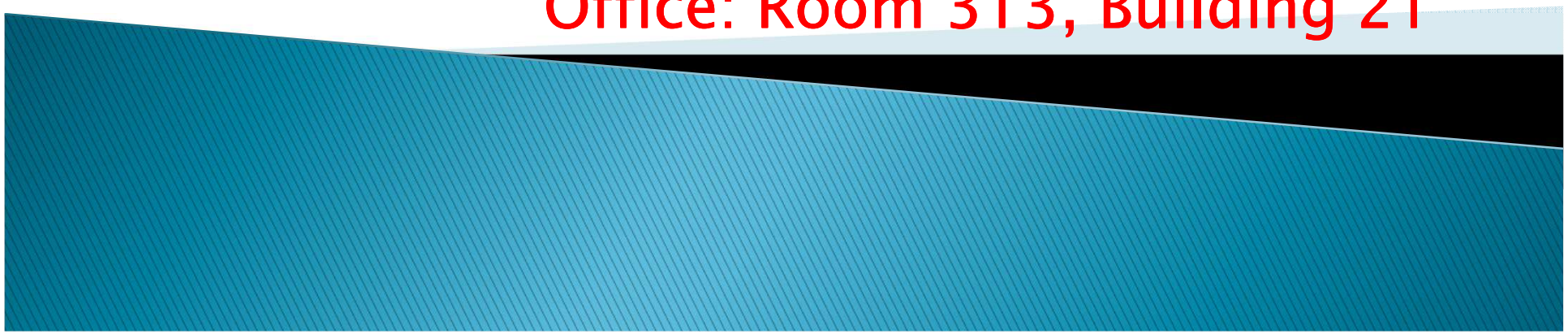
Hope you to enjoy yourself and have a good time in my class!

College of MPI of ZJNU

Software Quality Assurance and Testing

Chapter 1 Software Testing Introduction

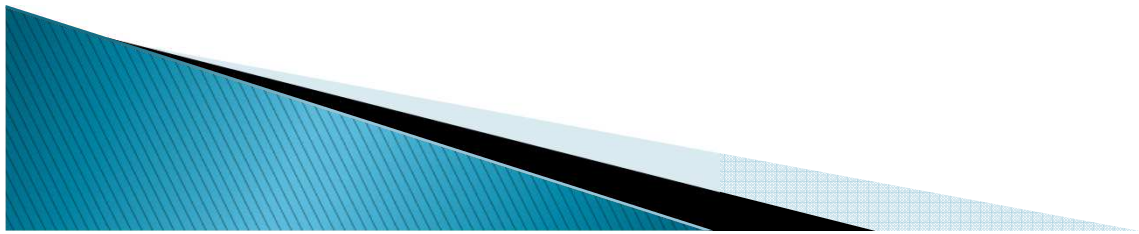
Lecturer: Zhiguo Ding
Phone: 13750985775(660775)
E-mail:dingzhiguo@zjnu.cn
Office: Room 313, Building 21



Contents

Chapter 1

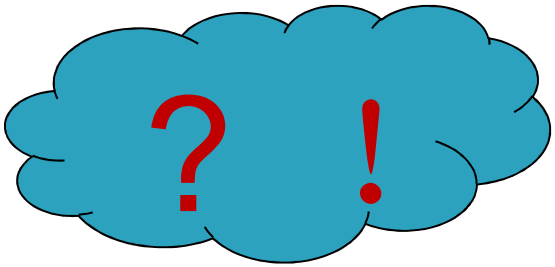
Software Testing Introduction

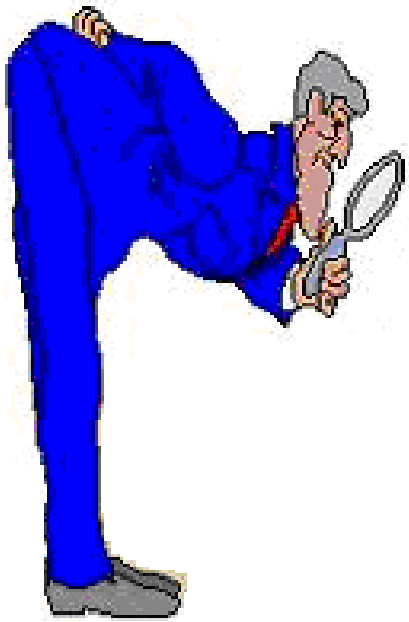


GAME: SPOT THE DIFFERENCE

Expected **Developed**

DIFFERENCE





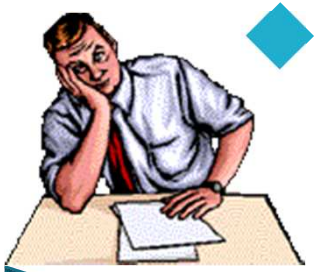
- Quantity
- Order
- Object
- ….

1.1 Something to know about Software Testing

- Software is everywhere

1.1 Something to know about Software Testing

- Qualified rate of products
  - Qualified rate of airplane manufacturing industry
    - “Aerobus747-400” are made up of 1000,000 parts
    - Qualified rate of every part: 99.9999%
    - Do you know the qualified rate ?



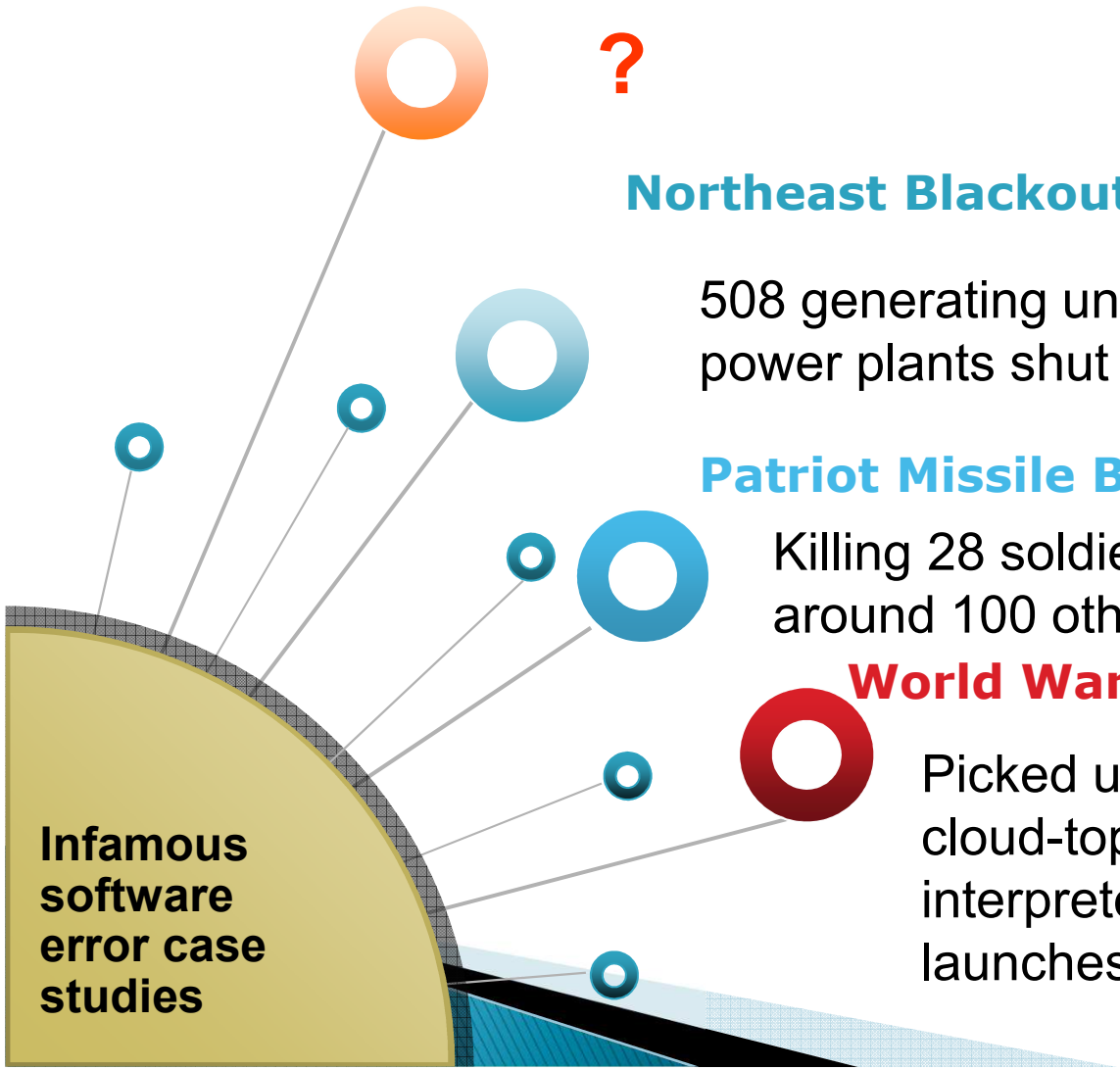
$$(99.9999\%)^{1000000} = 36.79\%$$

1.1 Something to know about Software Testing

- Qualified rate of products
  - Qualified rate of software
    - Coding line: 99%
    - Write 10,000 lines code
    - Do you know the qualified rate?

$$(99\%)^{10000} = 2.25 \times 10^{-44}$$

1.1 Something to know about Software Testing



What is the next ?

?

Northeast Blackout (2003)

508 generating units and 256 power plants shut down

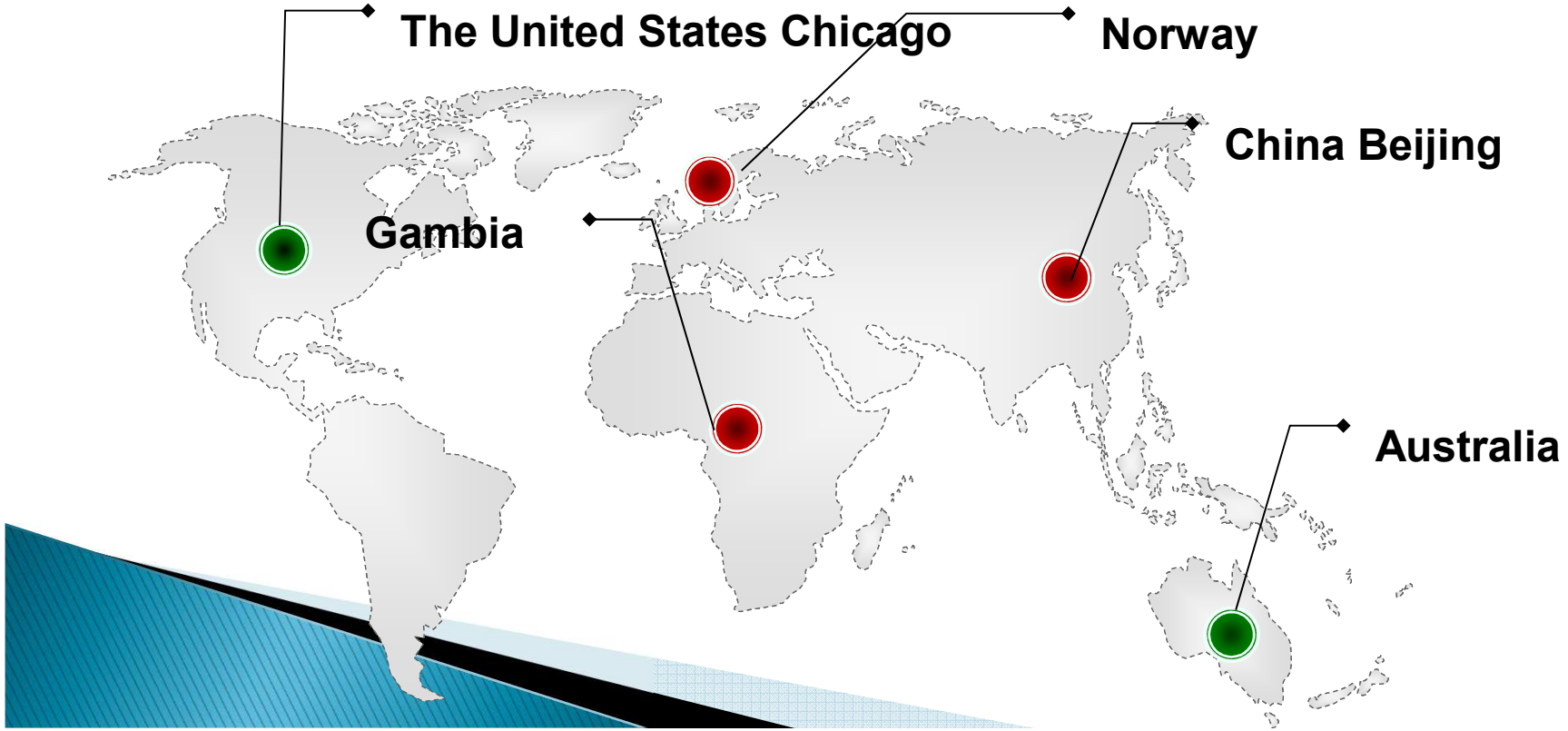
Patriot Missile Bug (1991)

Killing 28 soldiers and injuring around 100 other people

World War III… Almost (1983)

Picked up sunlight reflections off cloud-tops and mistakenly interpreted them as missile launches in the United States.

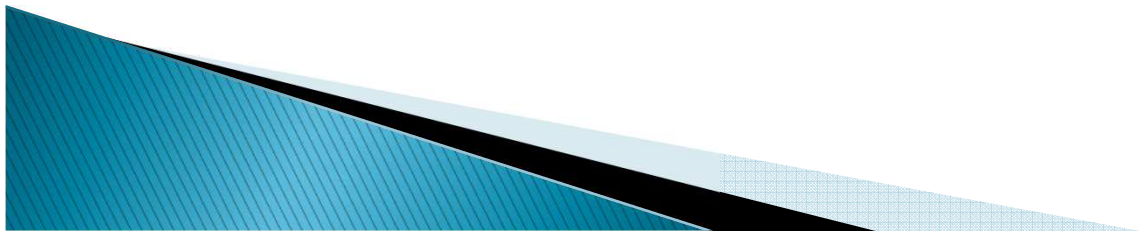
1.1 Something to know about Software Testing



1.1 Something to know about Software Testing

Year Ambiguities

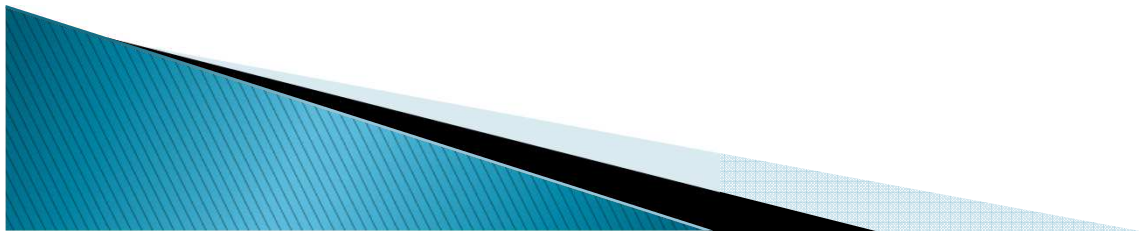
- In 1992, Mary Bandar received an invitation to attend a kindergarten in Winona, Minnesota, along with others born in '88.
- Mary was 104 years old at the time.



1.1 Something to know about Software Testing

Year Ambiguities (Cont'd)

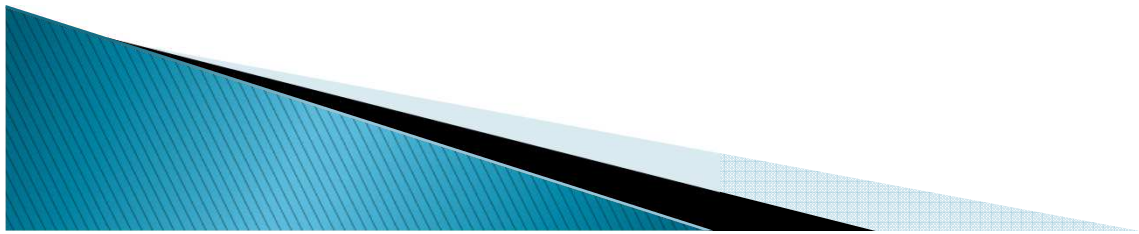
- Mr. Blodgett's auto insurance rate tripled when he turned 101.
- He was the computer program's first driver over 100, and his age was interpreted as 1.
- This is a double blunder because the program's definition of a teenager is someone under 20!



1.1 Something to know about Software Testing

Dates, Times, and Integers

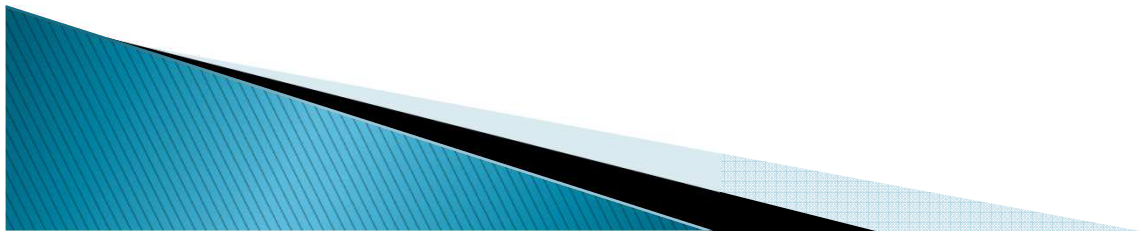
- The number 32,768 =2^{15} has caused all sorts of grief from the overflowing of 16-bit words.
- A Washington D.C. hospital computer system collapsed on September 19, 1989, 2^{15} days after January 1, 1900, forcing a lengthy period of manual operation.



1.1 Something to know about Software Testing

Bank Generosity

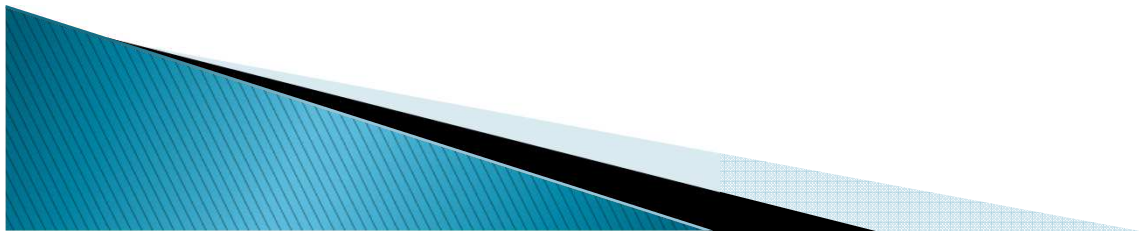
- A Norwegian bank ATM consistently dispersed 10 times the amount required.
- Many people joyously joined the queues as the word spread.



1.1 Something to know about Software Testing

Therac-25 Radiation “Therapy”

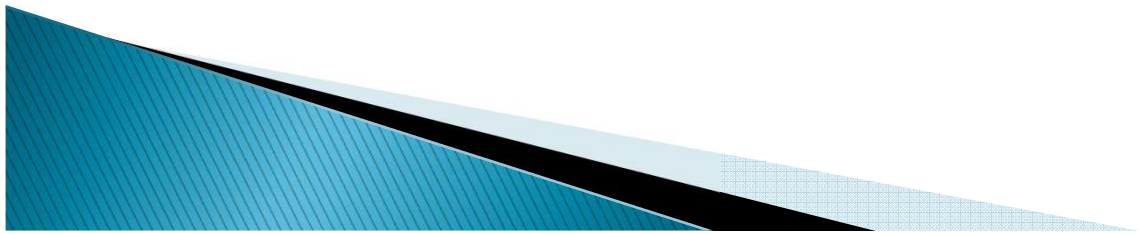
- In Texas, 1986, a man received between 16,500-25,000 rads in less than 1 sec, over an area of about 1 cm.
- He lost his left arm, and died of complications 5 months later.
- In Texas, 1986, a man received at least 4,000 rads in the right temporal lobe of his brain.
- The patient eventually died as a result of the overdose.



1.1 Something to know about Software Testing

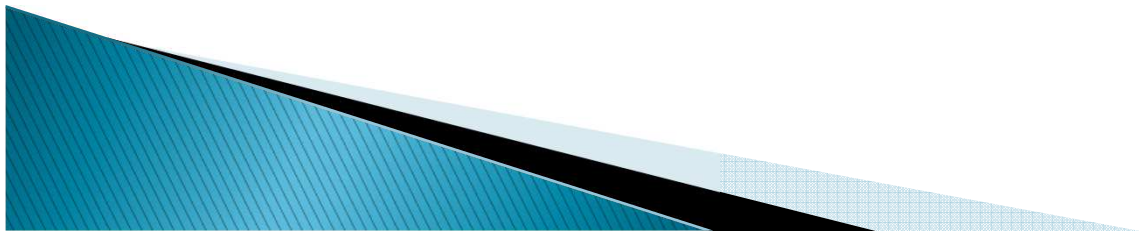
Military Aviation Problems

- An F-18 crashed because of a missing exception condition: if ... then ... without the else clause that was thought could not possibly arise.
- In simulation, an F-16 program bug caused the virtual plane to flip over whenever it crossed the equator, as a result of a missing minus sign to indicate south latitude.



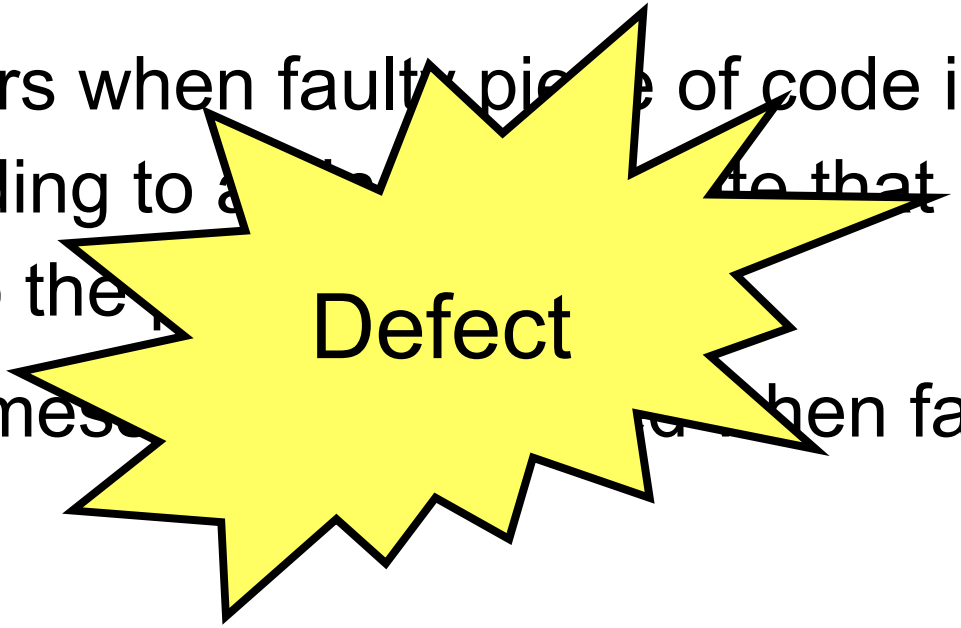
1.1 Something to know about Software Testing

- What is Software defect?
- The History of Software Testing



1.1 Something to know about Software Testing

- Error : occurs in the process of writing a program.
- Fault : is the manifestation of one or more errors.
- Failure : occurs when faulty pie e of code is executed leading to a to that propagates to the
- Incident : no mes hen failure occurs.



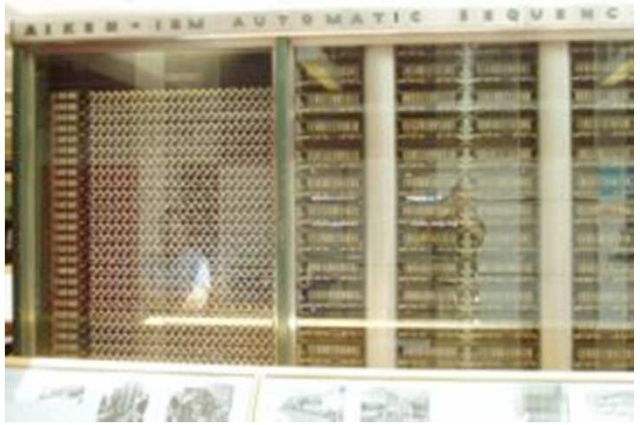
1.1 Something to know about Software Testing

- Defect
- Fault
- Problem
- Error
- Incident
- Anomaly
- Variance

- Failure
- Inconsistency
- Product Anomaly
- Product Incidence
- Feature :-)

1.1 Something to know about Software Testing

What is a computer bug?



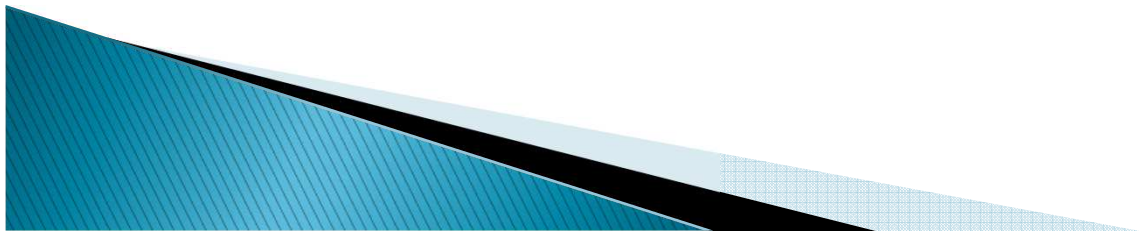
- In 1947 Harvard University was operating a room-sized computer called the Mark II.
  - mechanical relays
  - glowing vacuum tubes
  - technicians program the computer by reconfiguring it
  - Technicians had to change the occasional vacuum tube.
- A moth flew into the computer and was zapped by the high voltage when it landed on a relay.
- Hence, the first computer bug!
  - I am not making this up :-)

1.1 Something to know about Software Testing

- Software defect definition from IEEE 1983 of IEEE Standard 729
- From inside of product
- From outside of product

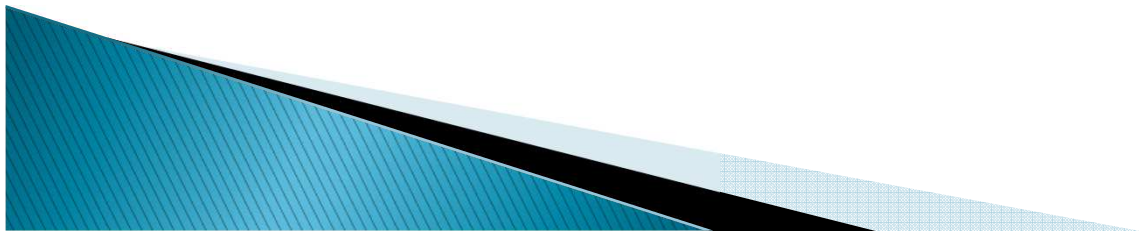
1.1 Something to know about Software Testing

- Software defect definition
  - Out of accord with user expectancy
  - Software function can be executed incorrectly
  - All kinds of software problems
    - E.g. Inconsistency , user interface fault
  - Defect of Software=Bug



1.1 Something to know about Software Testing

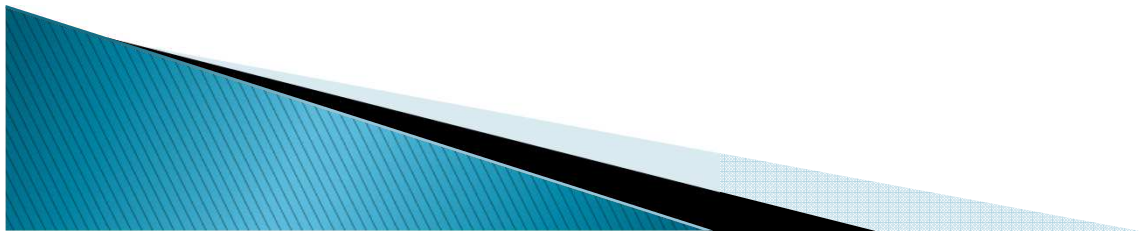
- Defect example：
  - Shortcoming: running slowly.
  - Inconsistency: Ctrl+S can’t save all applications.
  - User interface design defect: an button should show 5 words on it ,but only 3 words could be seen.



1.1 Something to know about Software Testing

Adverse Effects of Faulty Software

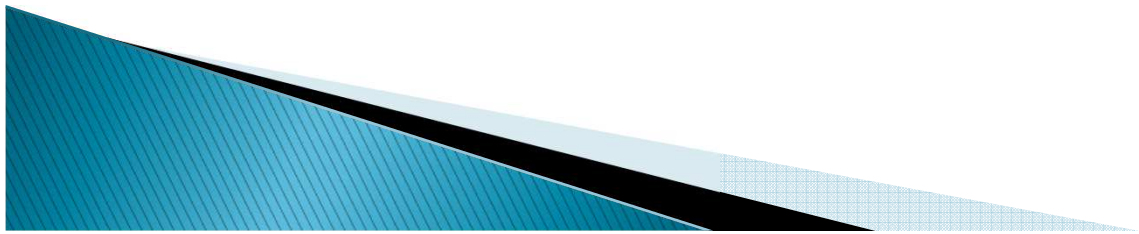
- Communications: Loss or corruption of communication media, non delivery of data.
- Space Applications: Lost lives, launch delays.
- Defense and Warfare: Misidentification of friend or foe.



1.1 Something to know about Software Testing

Adverse Effects of Faulty Software (Cont'd)

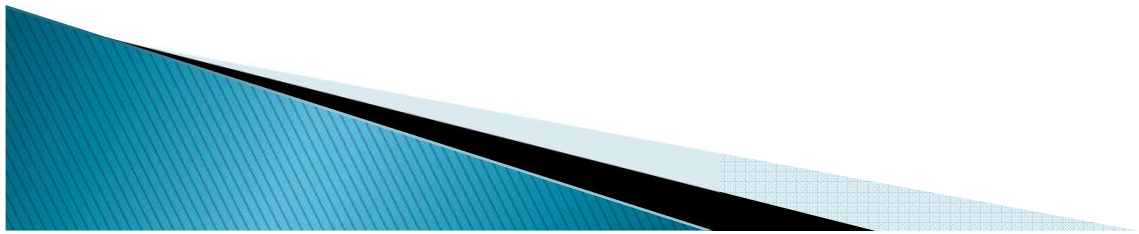
- Transportation: Deaths, delays, sudden acceleration, inability to brake.
- Safety-critical Applications: Death, injuries.
- Electric Power: Death, injuries, power outages, long-term health hazards (radiation).



1.1 Something to know about Software Testing

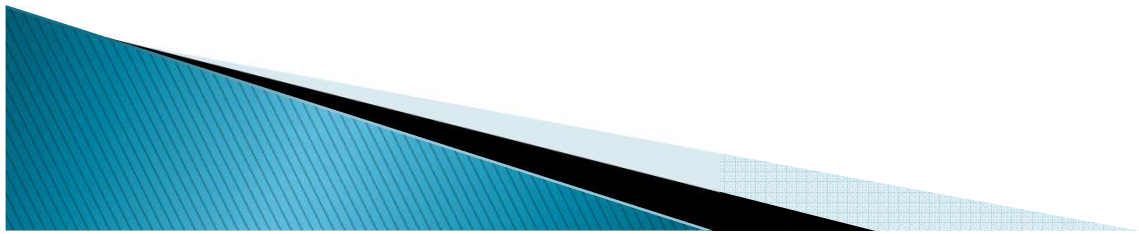
Adverse Effects of Faulty Software (Cont'd)

- Money Management: Fraud, violation of privacy, shutdown of stock exchanges and banks, negative interest rates.
- Control of Elections: Wrong results (intentional or non-intentional).
- Control of Jails: Technology-aided escape attempts and successes, accidental release of inmates, failures in software controlled locks.
- Law Enforcement: False arrests and imprisonments.



1.1 Something to know about Software Testing

- The source of defects
  - Requirements definition
  - Design
  - Implementation
  - Support systems
  - Inadequate testing of software
  - Evolution



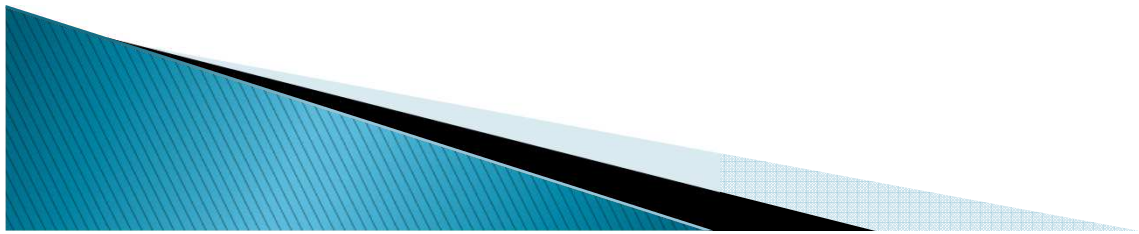
1.1 Something to know about Software Testing

- When does defect occur?
  - The software does not do something that the specification says it should do.
  - The software does something that the specification says it should not do.
  - The software does something that the specification does not mention.
  - The software does not do something that the product specification does not mention but should.
  - The software is difficult to understand, hard to use, slow …

1.1 Something to know about Software Testing

Sources of defects

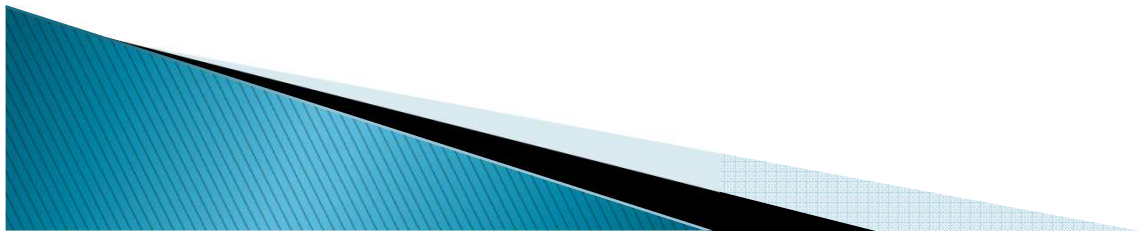
- Requirements Definition: Erroneous, incomplete, inconsistent requirements.
- Design: Fundamental design flaws in the software.
- Implementation: Mistakes in chip fabrication, wiring, programming faults, malicious code.
- Support Systems: Poor programming languages, faulty compilers and debuggers, misleading development tools.



1.1 Something to know about Software Testing

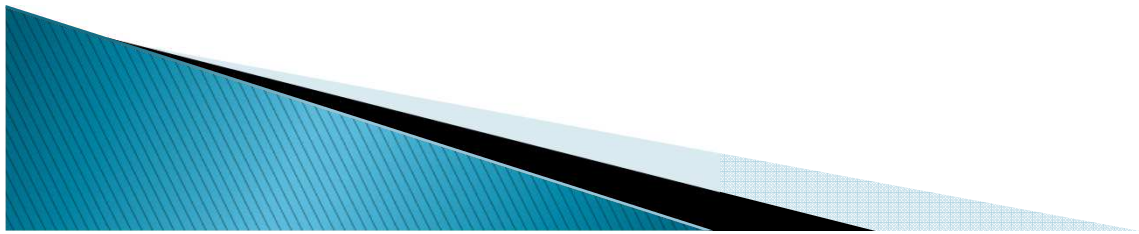
Sources of defects (Cont'd)

- Inadequate Testing of Software: Incomplete testing, poor verification, mistakes in debugging.
- Evolution: Sloppy redevelopment or maintenance, introduction of new flaws in attempts to fix old flaws, incremental escalation to inordinate complexity.

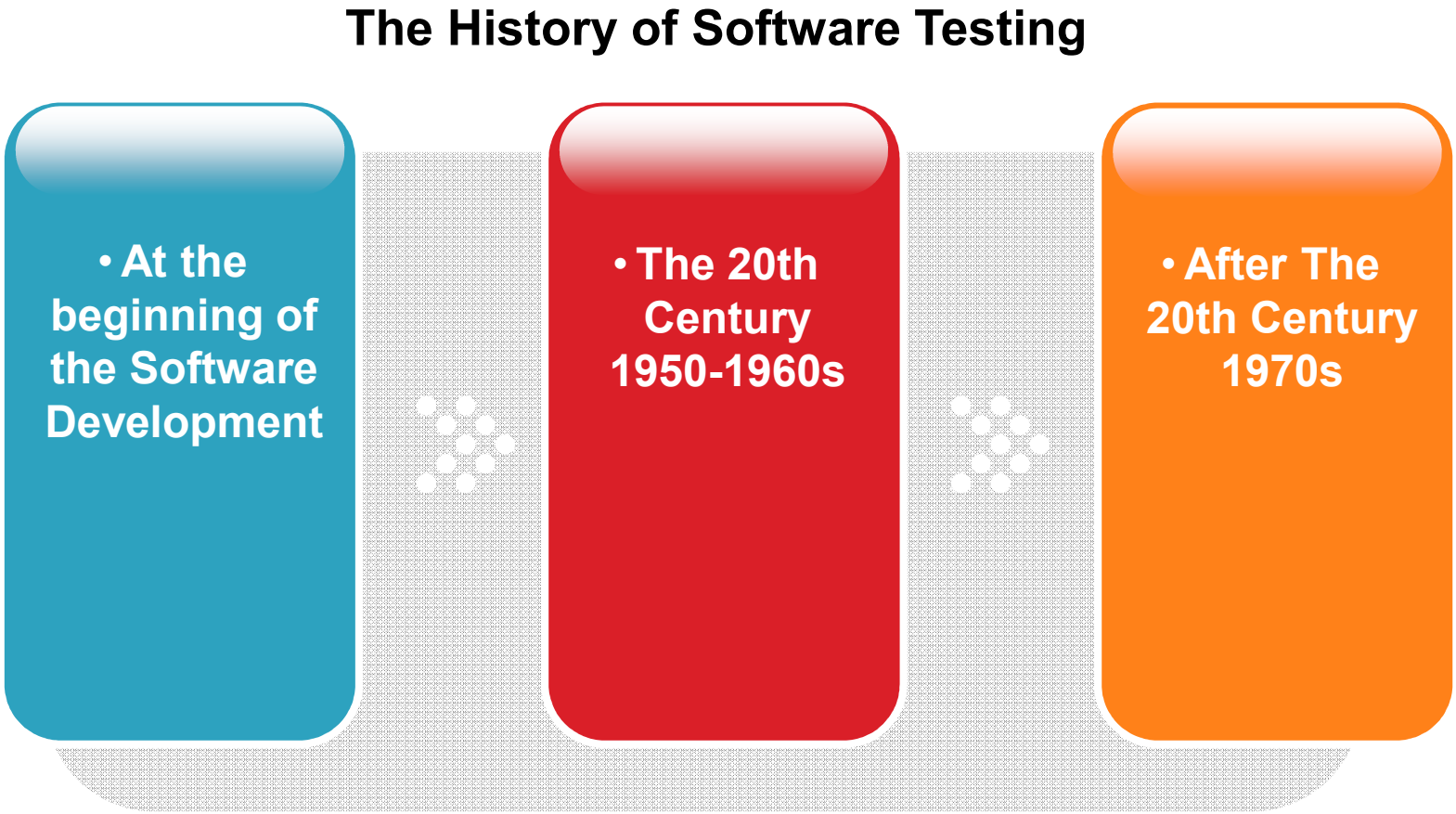


1.1 Something to know about Software Testing

- Correct program
  - No syntax error
  - No obvious errors during running
  - No improper statements
  - Valid input - correct output
  - Invalid input - correct output
  - Any possible input - correct output



1.1 Something to know about Software Testing

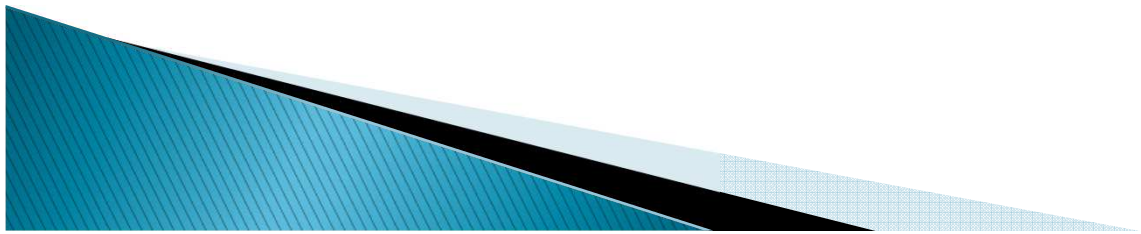


1.2 Why Software Testing is Needed

- Why do we test ?
  - We want our programs to be reliable.

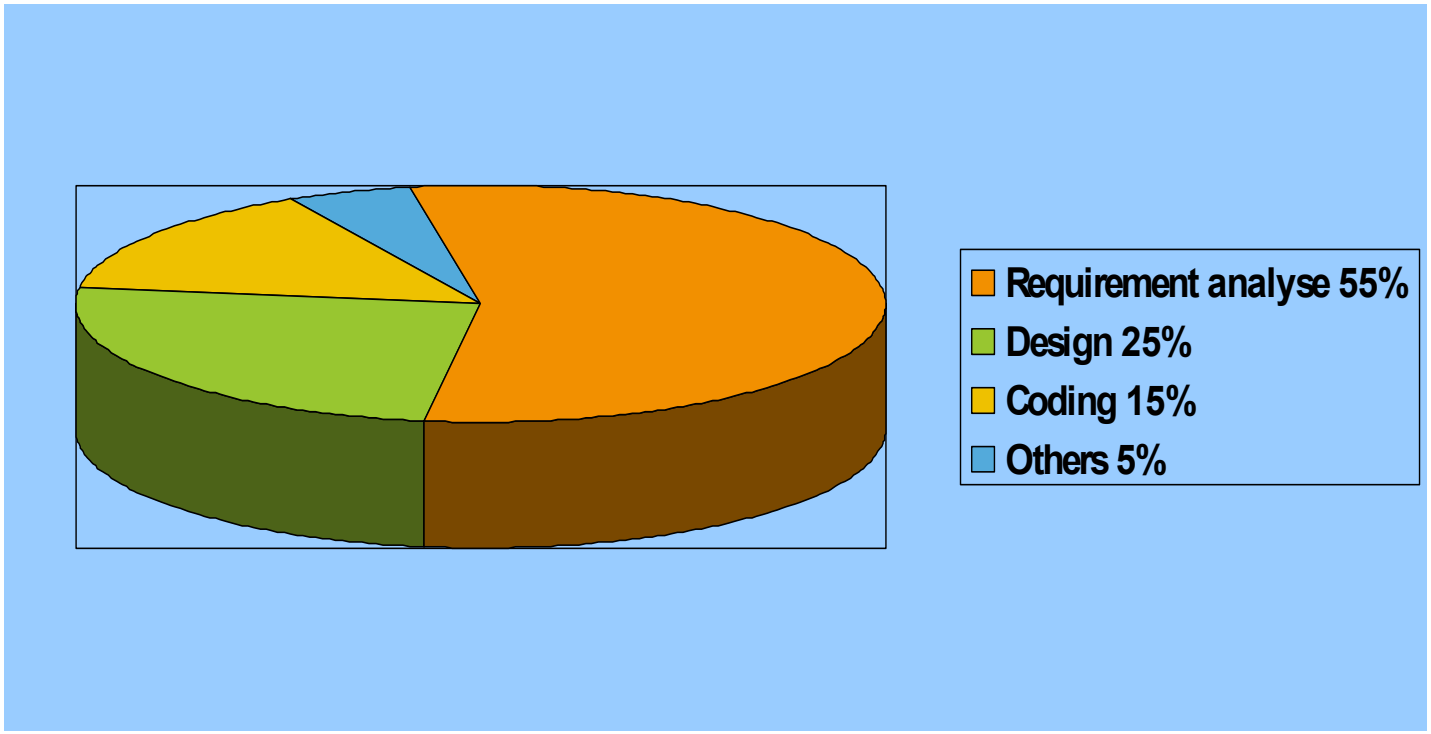
1.2 Why Software Testing is Needed

- Why do we test ?
  - Most security vulnerabilities are due to faulty software.
  - World-wide monetary loss due to poor software is staggering.
  - Stronger testing could solve most of these problems.



1.2 Why Software Testing is Needed

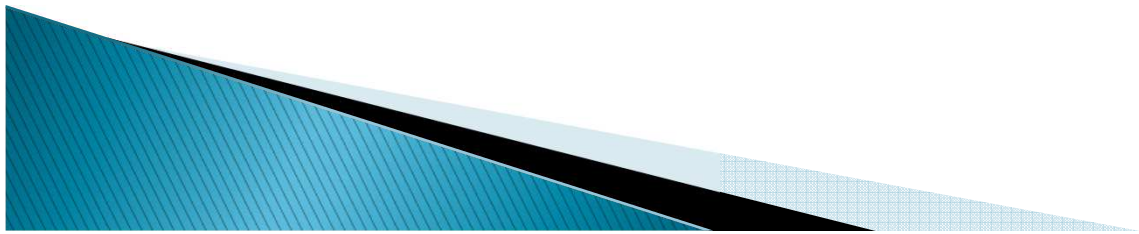
- Defects can be imported from any phrase during software development, and they will be amplified.



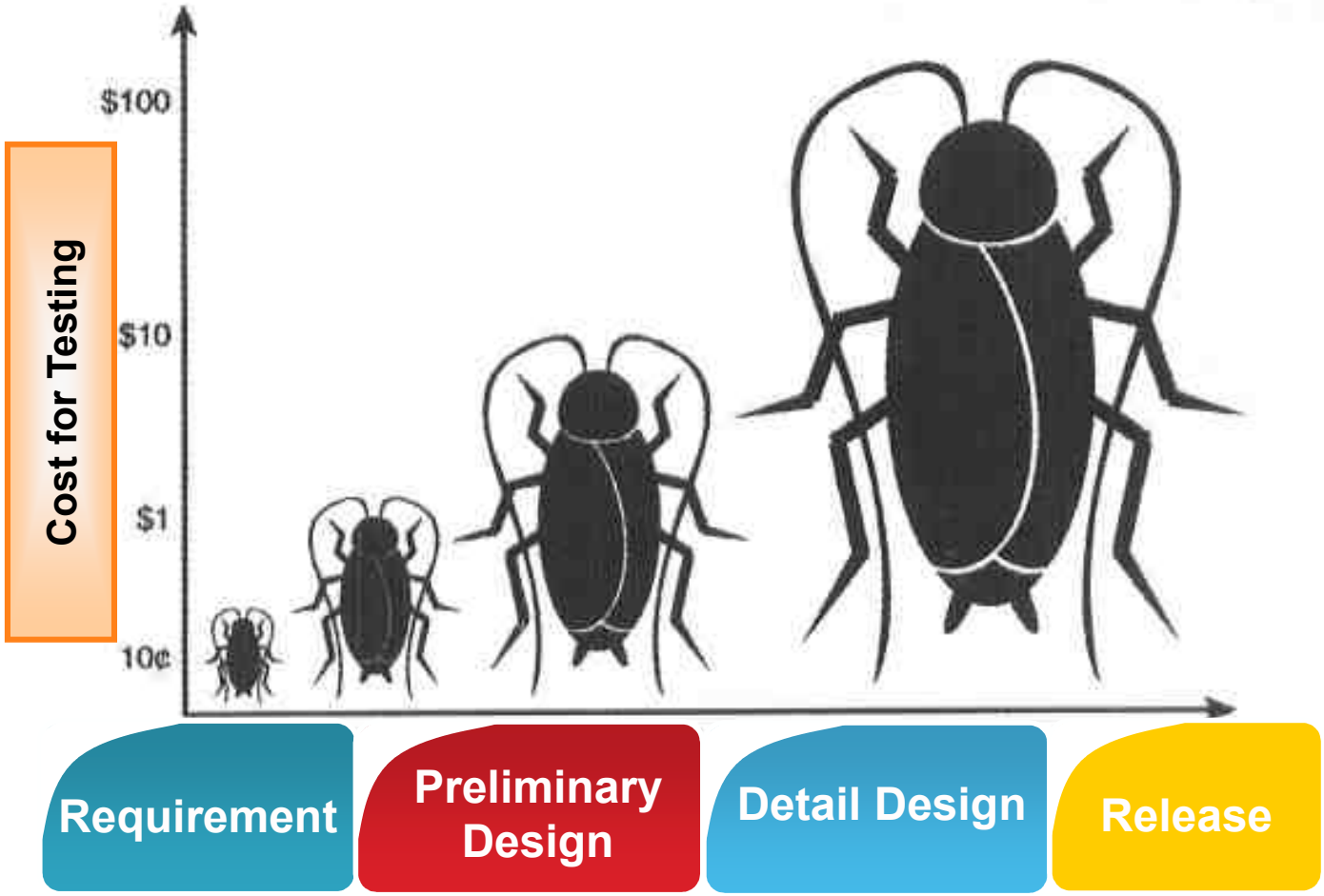
1.2 Why Software Testing is Needed

Problems with Testing

- Since it is impossible to find every fault in a software system, some bugs will be found by customers after the product is released.
- Q: If all software is released to customers with faults, why should we spend so much time, effort, and money on testing?



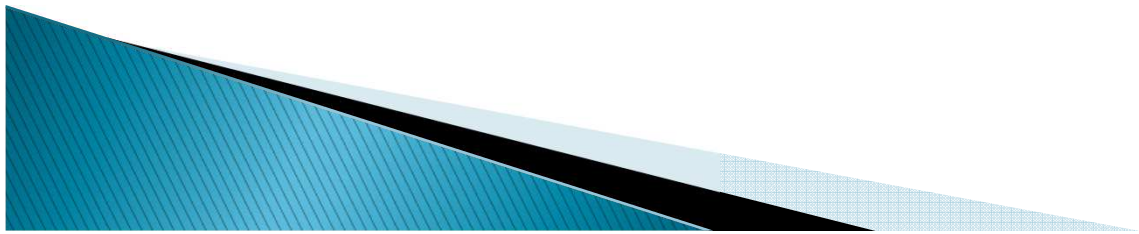
1.2 Why Software Testing is Needed



1.2 Why Software Testing is Needed

Goal of a software testing

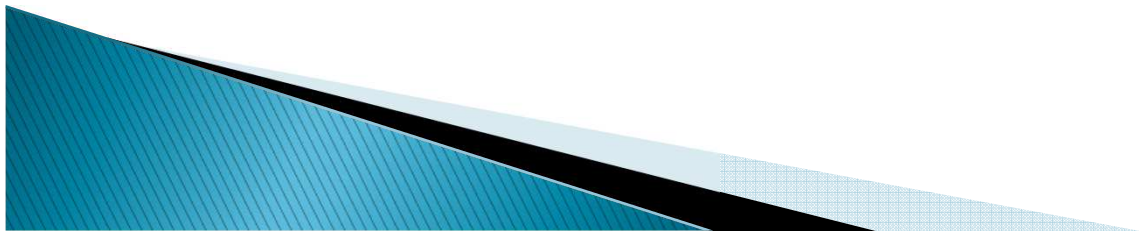
- … to *find* bugs
- … as *early* in the software development processes as possible
- … and make sure they get *fixed*.



1.2 Why Software Testing is Needed

Bug Free Software ?

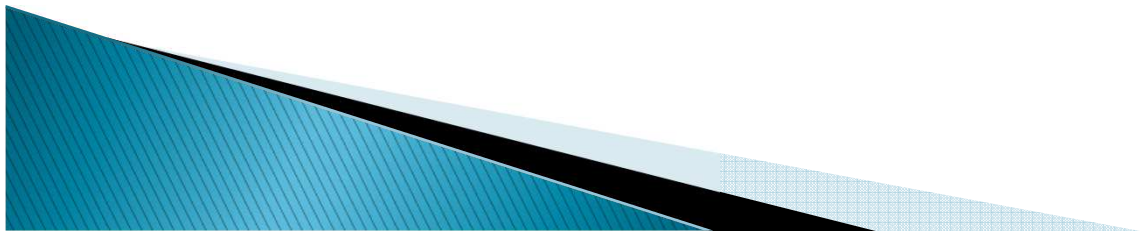
- Why can't software engineers develop software that just works?
  - As software gets more features and supports more platforms it becomes increasingly difficult to make it create bug-free.
  - Infinite test is impossible



1.2 Why Software Testing is Needed

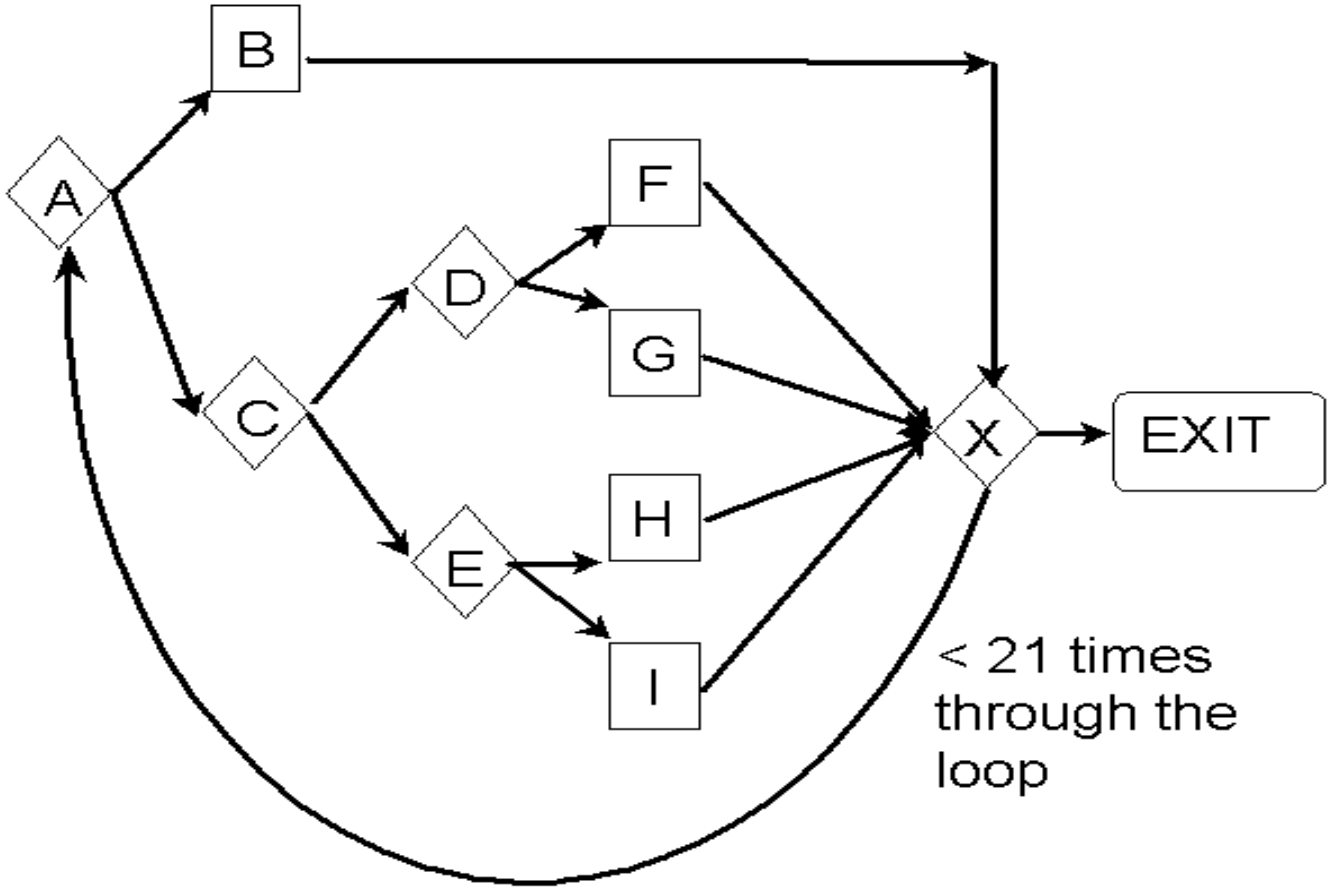
Why Can't Every Bug be Found?

- Too many possible paths.
- Too many possible inputs.
- Too many possible user environments.



1.2 Why Software Testing is Needed

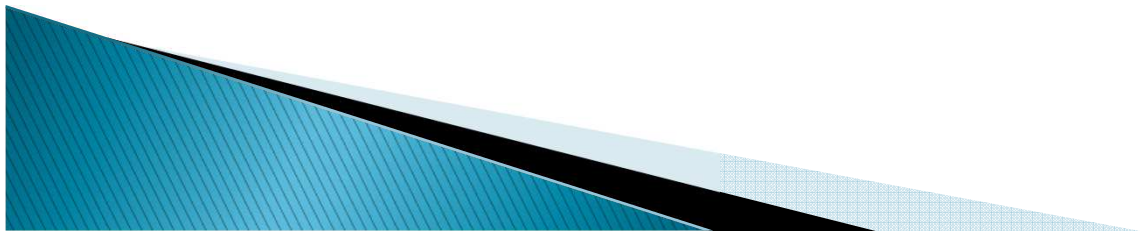
Too Many Possible Paths



1.2 Why Software Testing is Needed

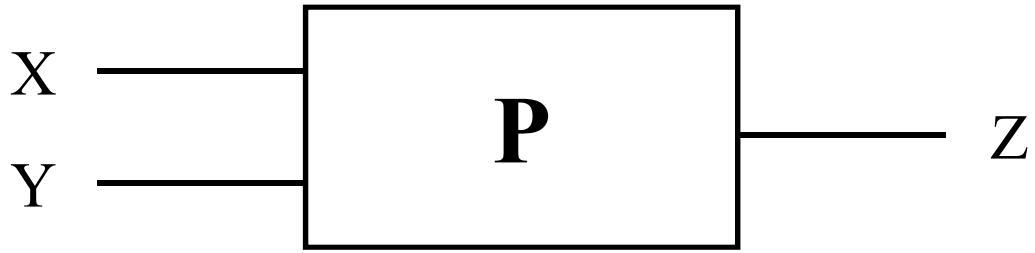
Too Many Possible Paths

- There are 5 paths from A to X without passing through the loop.
- There are 5^{20} paths from A to X after passing through the loop 20 times.
- There are $5 + 5^2 + 5^3 + \ldots + 5^{20}$ = 100 trillion possible paths in this program.
- If you could test a path per second it would take more than 3 million years!



1.2 Why Software Testing is Needed

- Suppose a program *P* has inputs(*X* and *Y*) and output (*Z*). It run at a computer with 32 bits. if *X* and *Y* are integer, how many time is needed?(1 group data/ms)

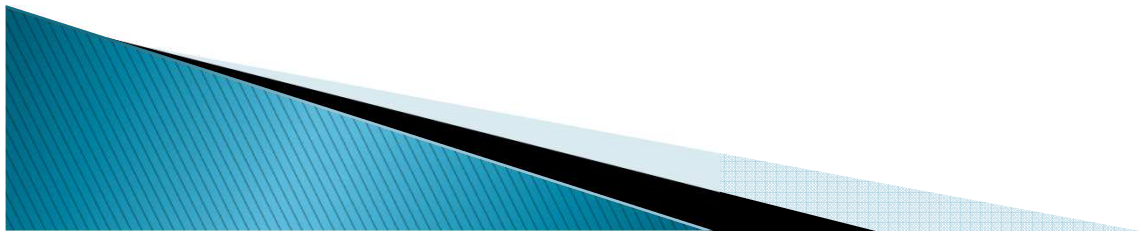


$2^{32} \times 2^{32} / (365 \times 24 \times 60 \times 60 \times 1000) = 5$ Million Years

1.2 Why Software Testing is Needed

Too Many Possible Inputs

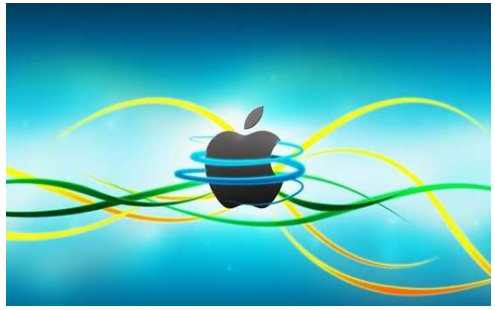
- Programs take input in a variety of ways: mouse, keyboard, and other devices.
- Must test Valid and Invalid inputs.
- Most importantly, there are an infinite amount of sequences of inputs to be tested.

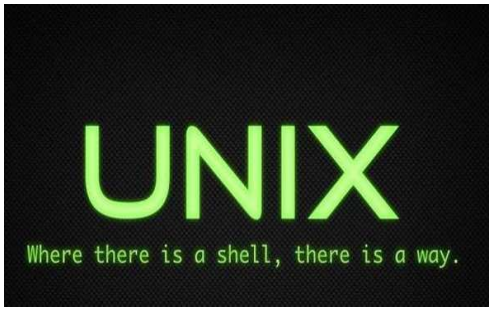


1.2 Why Software Testing is Needed

Too Many Possible User Environments

- Difficult to replicate the user's combination of hardware, peripherals, OS, and applications.
- Impossible to replicate a thousand-node network to test networking software.

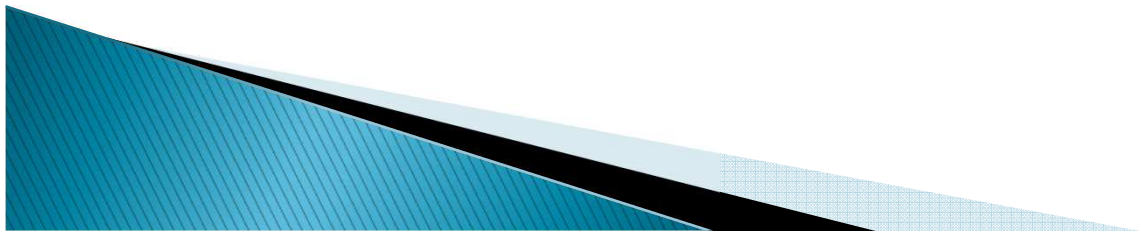




1.2 Why Software Testing is Needed

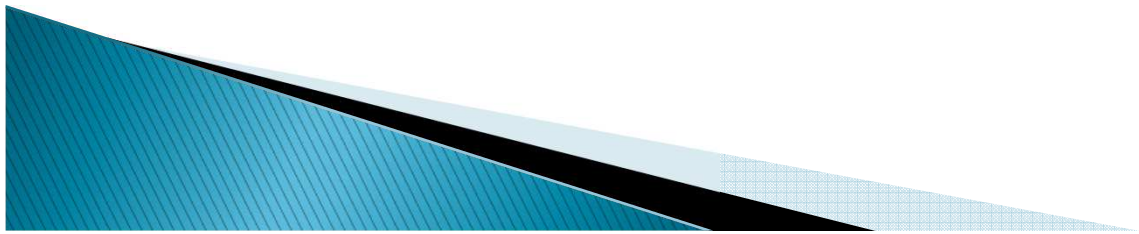
Reasons that Bugs Escape Testing

- User executed untested code.
- User executed statements in a different order than was tested.
- User entered an untested combination of inputs.
- User's operating environment was not tested.



1.3 Software testing definition

- Definition
- Verification &Validation
- Test & debug
- Purpose of software testing
- Types of testing



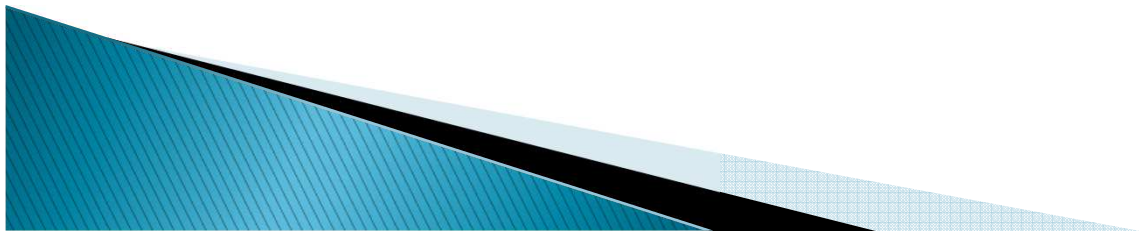
1.3 Software testing definition

- Definition from IEEE(1983)

https://en.wikipedia.org/wiki/Software_testing

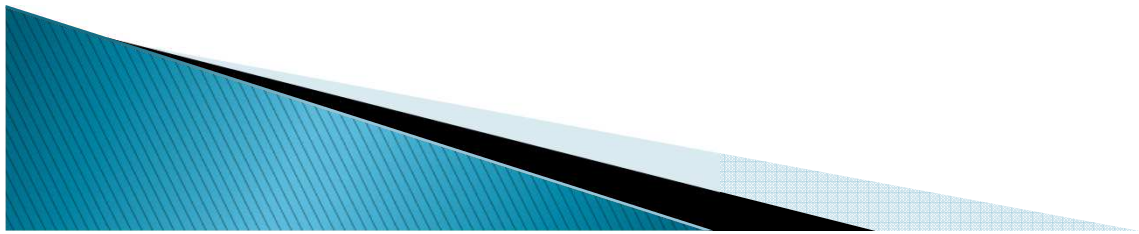
https://en.wikipedia.org/wiki/Software_test_documentation

https://en.wikipedia.org/wiki/Software_quality_assurance



1.3 Software testing definition

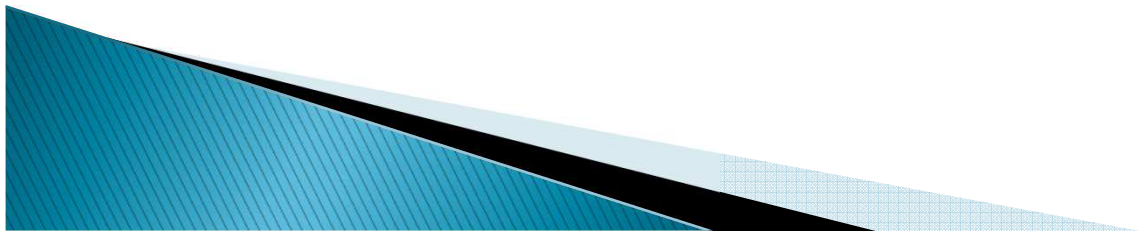
- Software testing is the essential step which is planned and systematic.
- It is an empirical investigation conducted to provide stakeholders with information about the quality of the product or service under test.
- We can know whether the users expectations are realized.
- Software testing is the key step of software quality assurance.



1.3 Software testing definition

Software testing is the process of executing a software system to determine whether it matches its specification and executes in its intended environment.

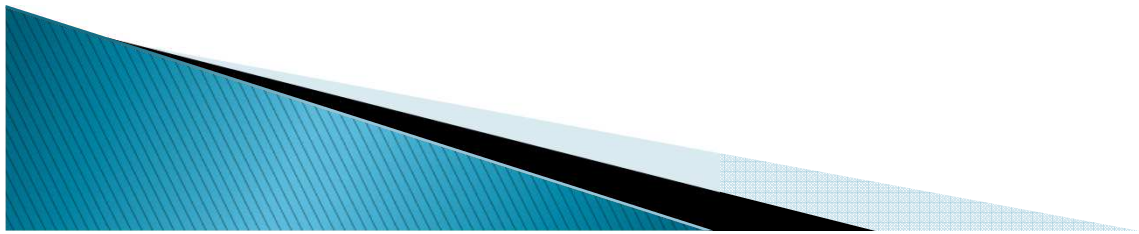
"Testing is the process of executing a program with the intention of finding errors."
– **Myers**



1.3 Software testing definition

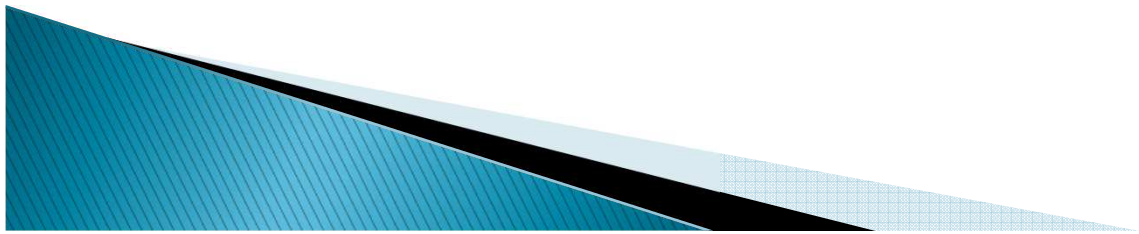
“Program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence” [Dijkstra, 1972]

“Testing can show the presence of bugs but never their absence.” - Dijkstra



1.3 Software testing definition

- Purpose of software testing ,according to the view of G.J.Myers.
  - Find errors of software
  - Decrease the risk of software doesn't work
  - First time & Never



1.3 Software testing definition

Myers's Idea about software testing

- Testing

  The execution(?) of a program to find its faults

- Verification

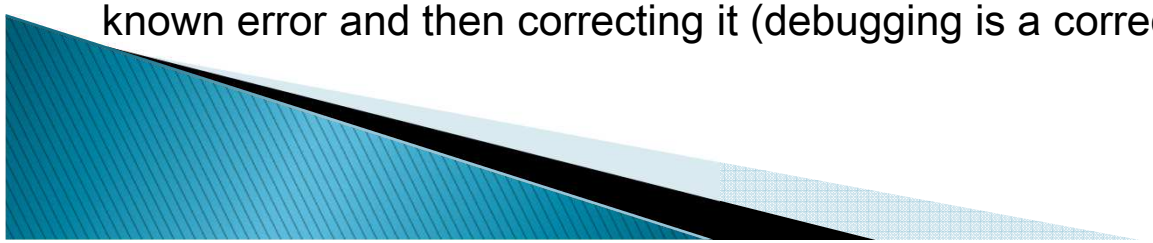
  The process of proving the programs correctness.(An attempt to find errors by executing a program in a test or simulated environment (it is now preferable to view verification as the process of proving the program's correctness))

- Validation

  The process of finding errors by executing the program in a real environment

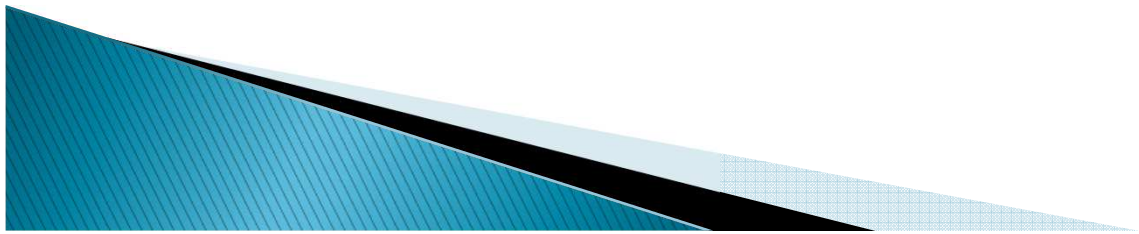
- Debugging

  Diagnosing the error and correct it (Diagnosing the precise nature of a known error and then correcting it (debugging is a correction and not a testing activity)



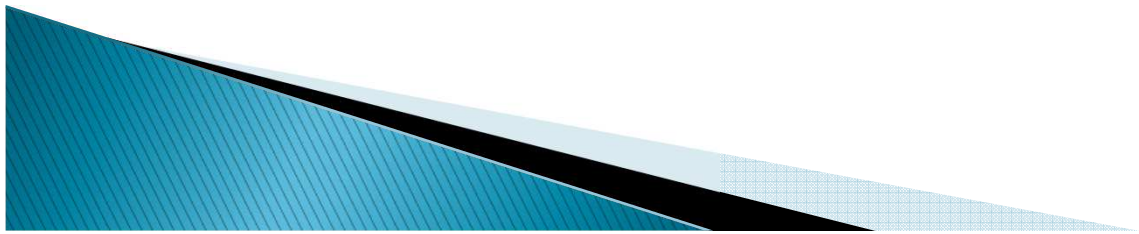
1.3 Software testing definition

- Attention：
  - The essential function of Software testing is Verification and Validation.
    - **Verification**: The software should conform to its specification (Are we building the product right?)
    - **Validation**: The software should do what the user really requires (Are we building the right product?)



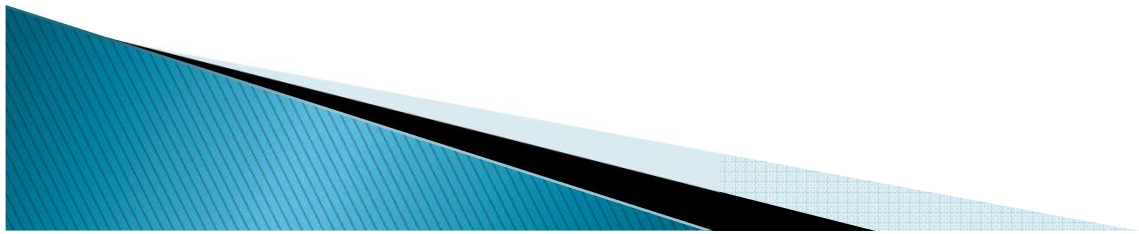
1.3 Software testing definition

- Attention：
  - Test & debug
    - Automated test vs. manual operate
    - Don’t know details are OK vs. must know details
    - Correctness proof and how to do with failure vs. correctness proof only
    - Checking vs. reasoning
    - Plan , under control vs. out of control



1.3 Software testing definition

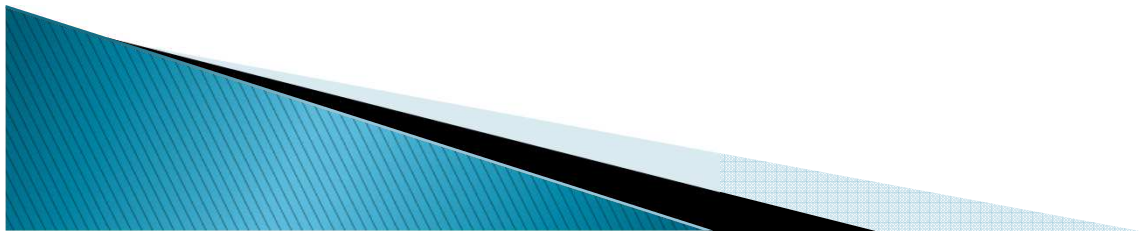
- Types of testing
  - C1: Source of test generation
  - C2: Life cycle phase in which testing takes place
  - C3: Goal of a specific testing activity
  - C4: Characteristics of the artifact under test
  - C5: Test process models



1.3 Software testing definition

- C1:Source of test generation

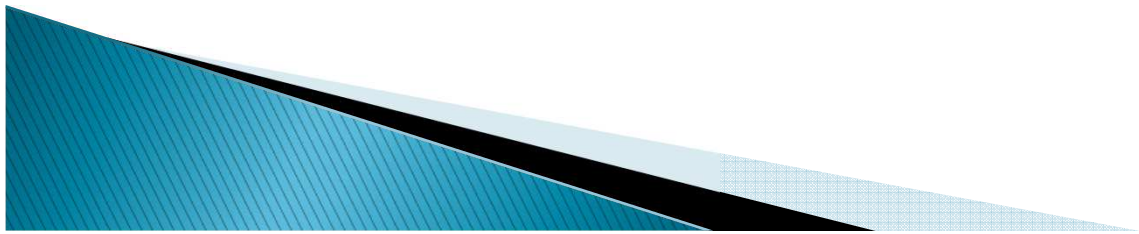
Artifact	Technique
Requirements (informal)	Black-box
Code	White-box
Requirements and code	Black-box and white-box
Formal model: graphical or mathematical specification	Model-based specification
Component's interface	Interface testing



1.3 Software testing definition

- C2:Life cycle phase

Phase	Technique
Coding	Unit testing
Integration	Integration testing
System integration	System testing
Maintenance	Regression testing
Post system, pre-release	Alpha-testing, Beta-testing (Acceptance testing)



1.3 Software testing definition

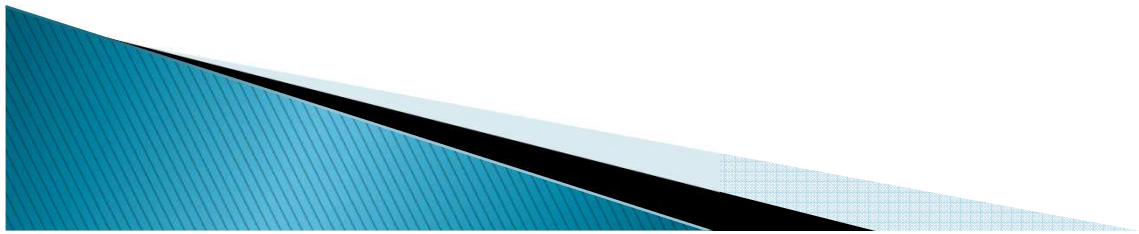
- C3:Goal-directed testing

Goal	Technique
Advertised features	Functional
Security	Security
Invalid inputs	Robustness
Vulnerabilities	Vulnerability
Errors in GUI	GUI
Operational correctness	Operational
Reliability assessment	Reliability
Resistance to penetration	Penetration
System performance	Performance
Customer acceptability	Acceptance
Business compatibility	Compatibility
Peripherals compatibility	Configuration
Foreign language compatibility	Foreign language

1.3 Software testing definition

- C4:Artifact under test

Characteristic	Technique
Application component	Component testing
Batch processing	Equivalence partitioning, finite-state model-based testing, …
Client and server	Client-server testing
Compiler	Compiler testing
Design	Design testing
Code	Code testing



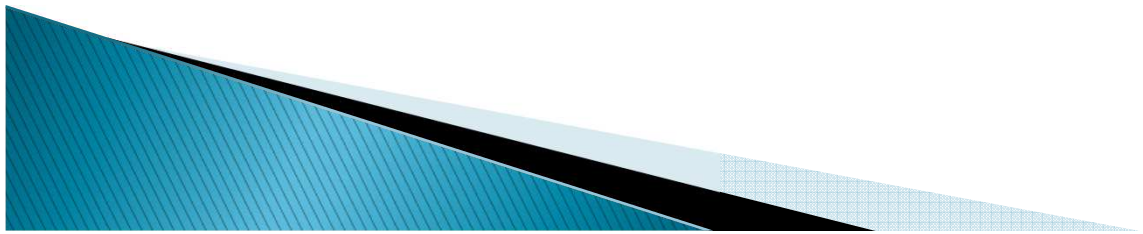
1.3 Software testing definition

- C4:Artifact under test

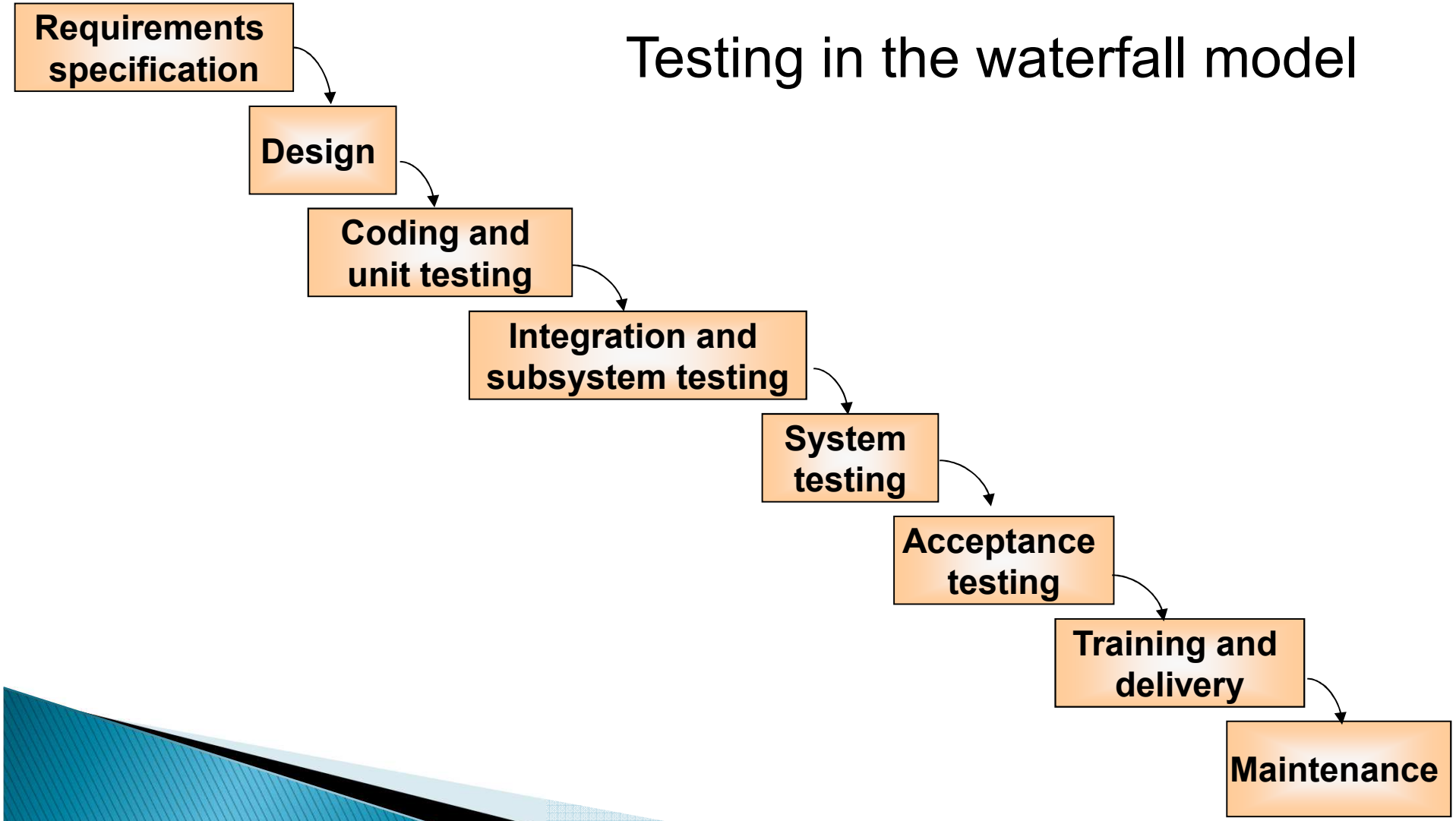
Characteristic	Technique
Database system	Transaction-flow testing
OO software	OO-testing
Operating system	OS testing
Real-time software	Real-time testing
Requirements	Requirement testing
Software	Software testing
Web service	Web-service testing

1.3 Software testing definition

- C5:Test process models
  - Testing in waterfall model
  - Testing in V-model
  - Spiral testing
  - Agile testing
  - Test-driven development (TDD)
    - Requirements specified as tests



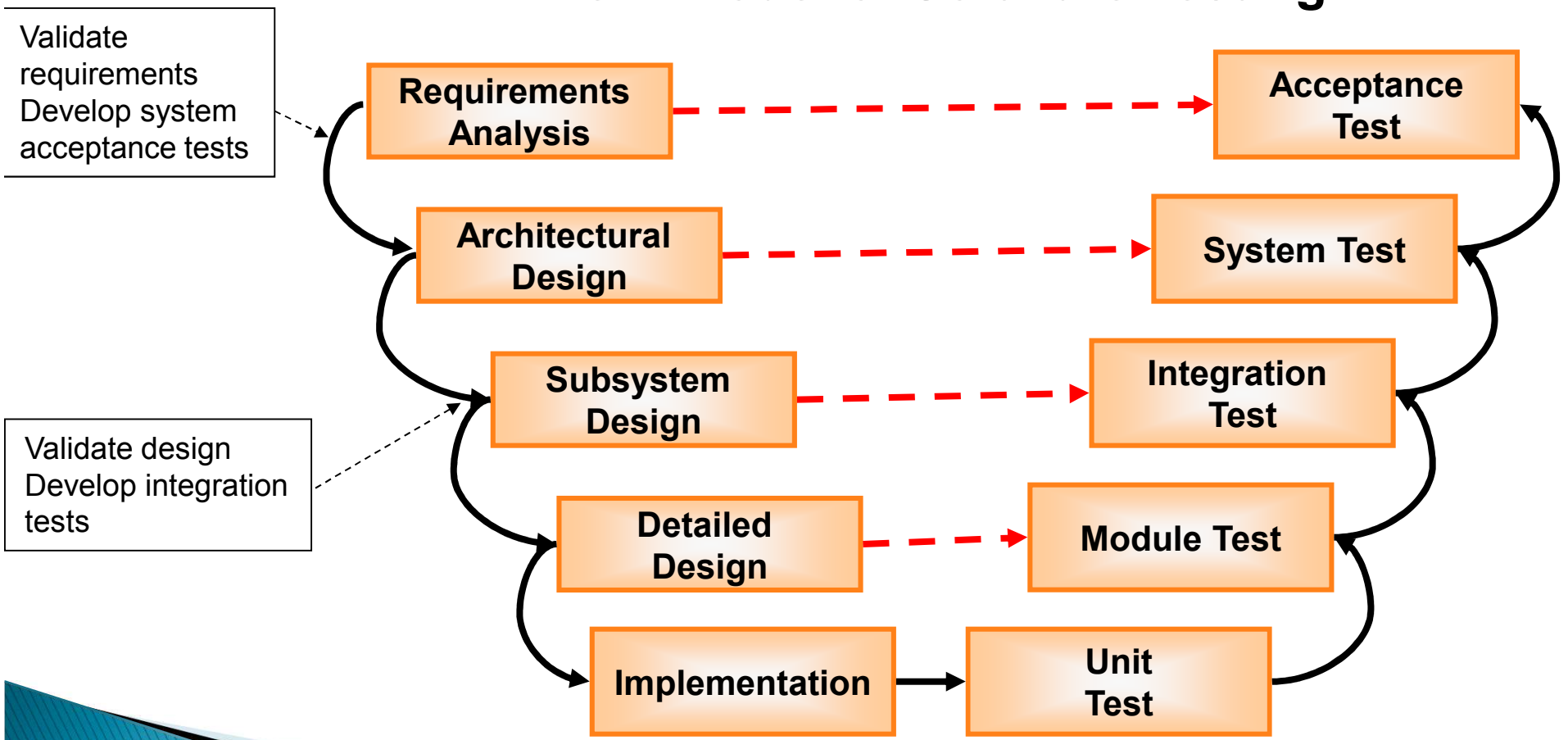
Types of Testing



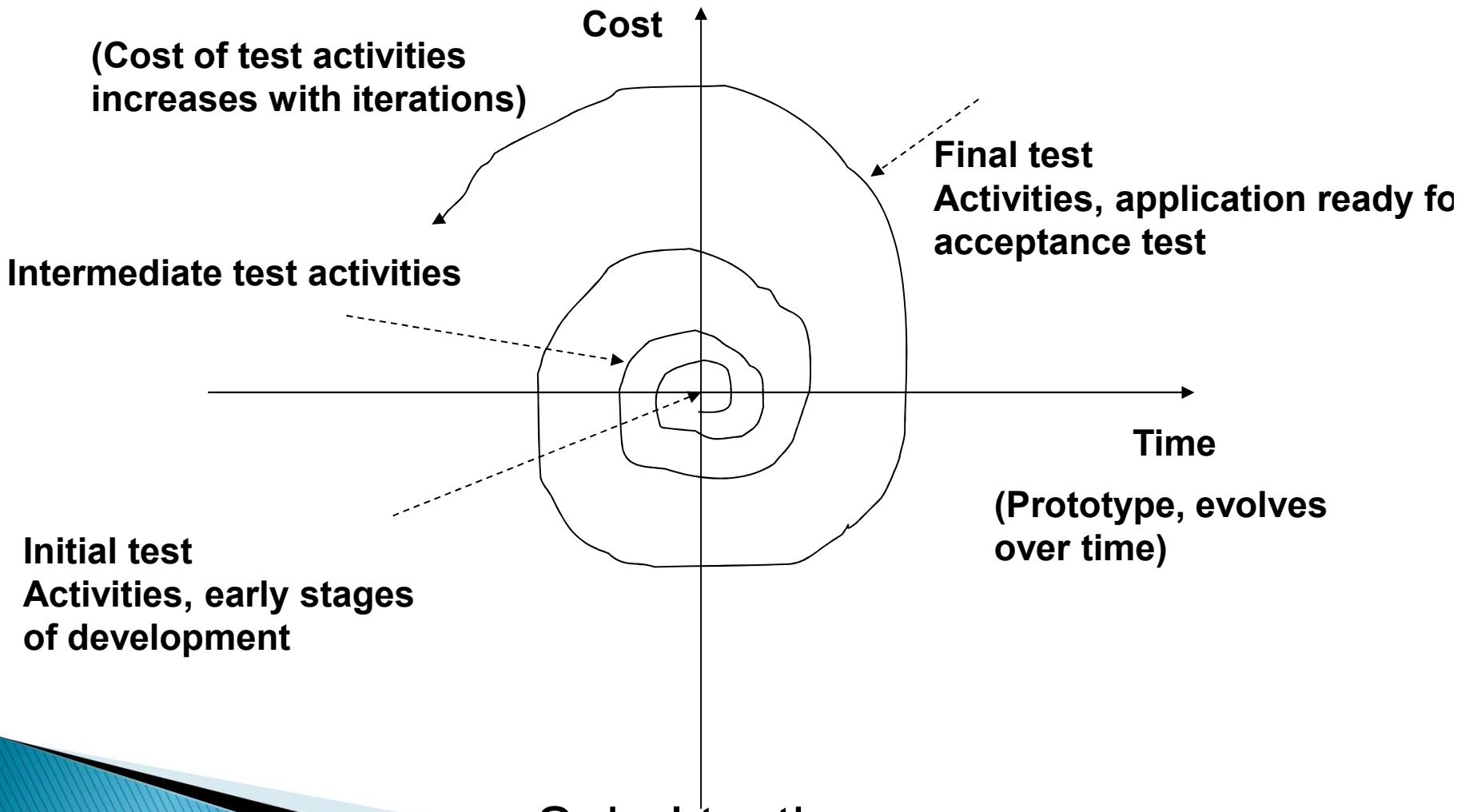
Testing in the waterfall model

Types of Testing

- **The V model of Software Testing**



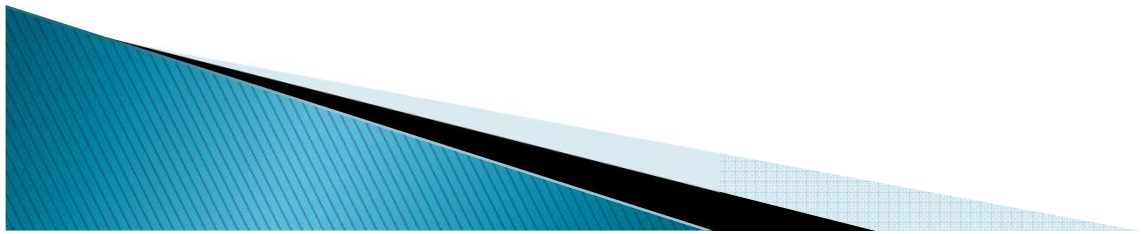
Types of Testing



Spiral testing

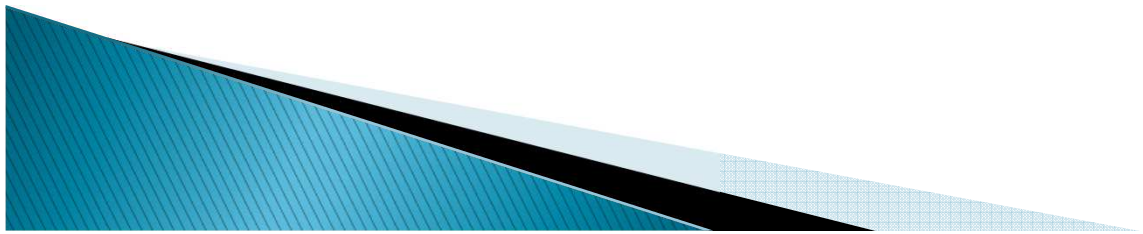
Types of Testing

- Agile testing promotes the following ideas
  - Include testing-related activities throughout a development project starting from the requirements phase
  - Work collaboratively with the customer who specifies requirements in terms of tests
  - Testers an developers must collaborate with each other rather than serve as adversaries
  - Test often and in small chunks



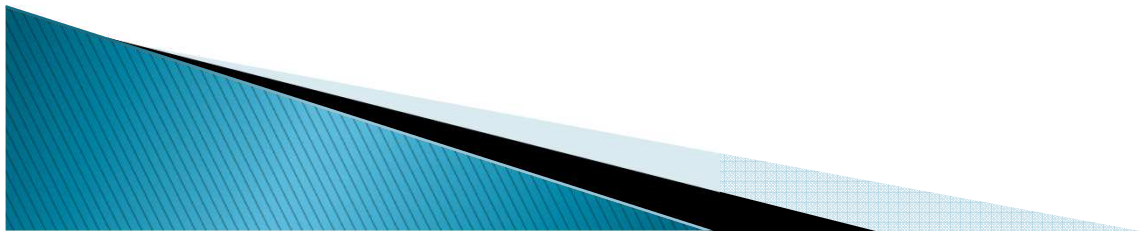
1.3 Software testing definition

- Example
  - Consider a Web service W to be tested. When executed, W converts a given value of temperature from one scale to another, for example from Fahrenheit scale to the Celsius scale.
  - Regardless of the technique used for test generation, we can refer to the testing of W as Web-services testing(C4).
  - Let's examine various types of test-generation techniques that could be used for testing W.



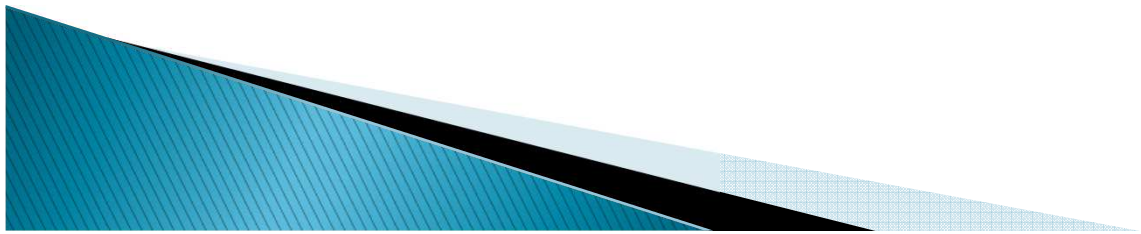
1.3 Software testing definition

- Supposing tester A tests W by supplying sample inputs and checking the outputs. No specific method is used to generate the inputs.
  - C1 : black-box testing
  - C2: unit testing
  - C3: GUI testing (If W has a GUI to interface with a user)



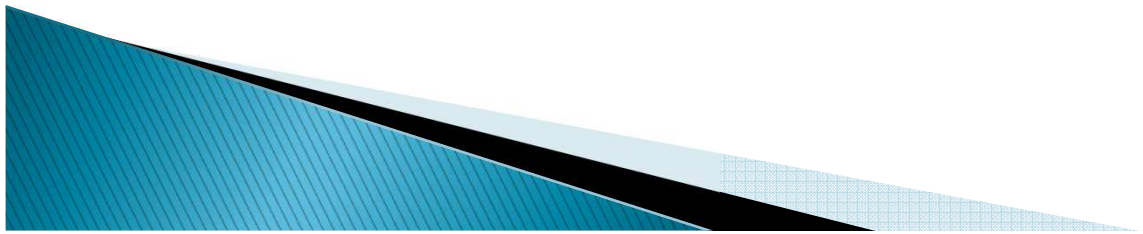
1.3 Software testing definition

- Supposing tester B writes a set of formal specifications for W using the Z notation. The tester generates, and uses, tests from the specification.
  - C1: black-box testing



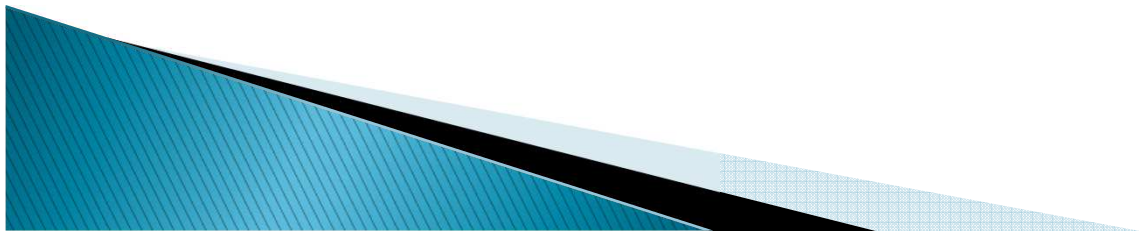
1.3 Software testing definition

- Supposing tester C generates tests using the formal specifications for W. C then tests W and evaluates the code coverage using one of the several code-coverage criteria. C finds that the code coverage is not 100%
  - C1: black-box testing and white-box testing



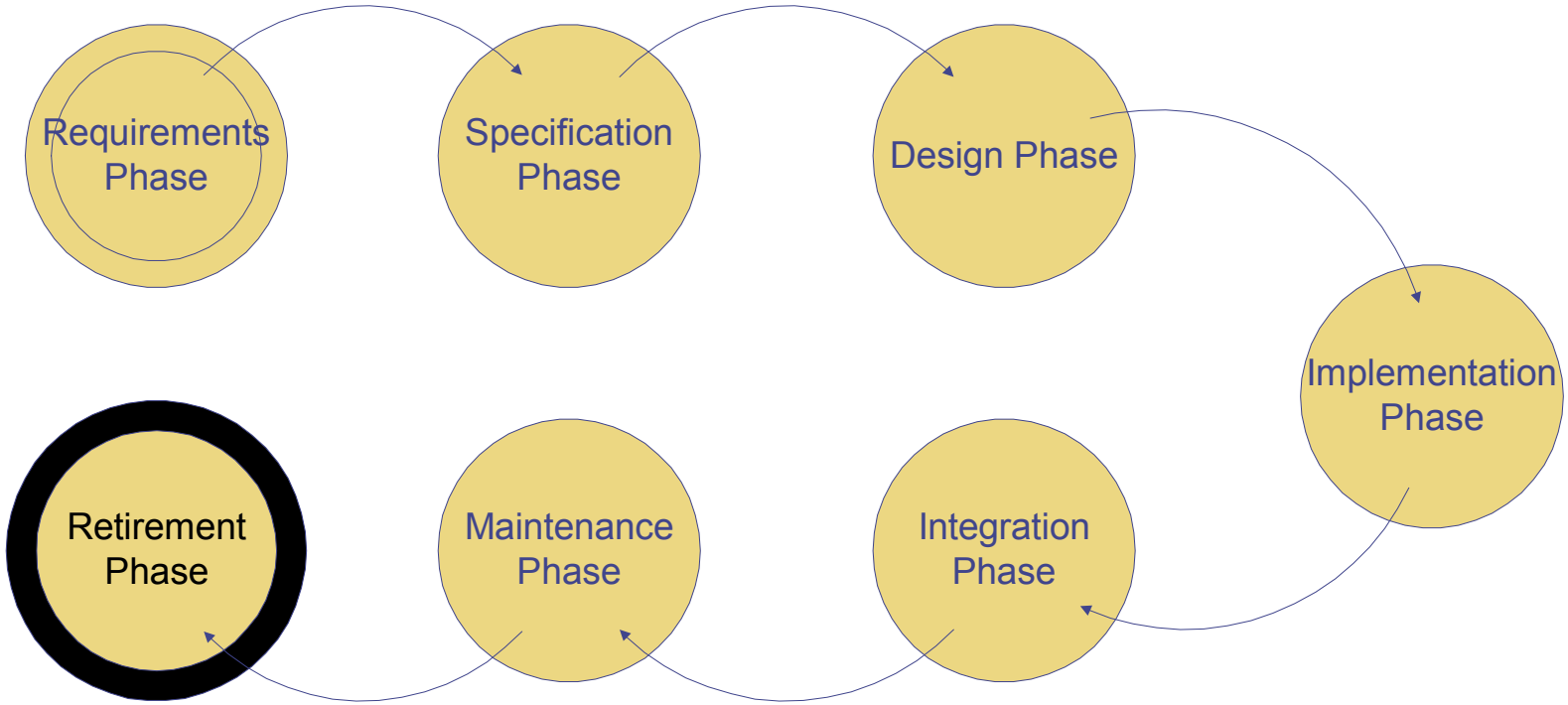
1.3 Software testing definition

- Supposing tester D tests W as a component of a larger application. Tester D does not have access to the code for W and hence uses only its interface, and interface mutation, to generate tests.
  - C1: black-box testing.



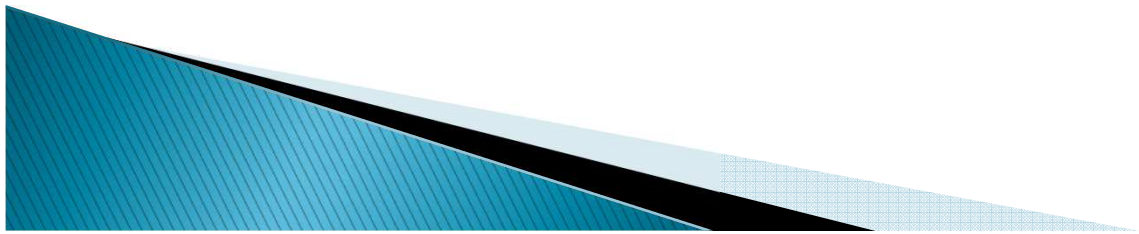
1.4 Software Testing Rules

Phases of the Software Process



1.4 Software Testing Rules

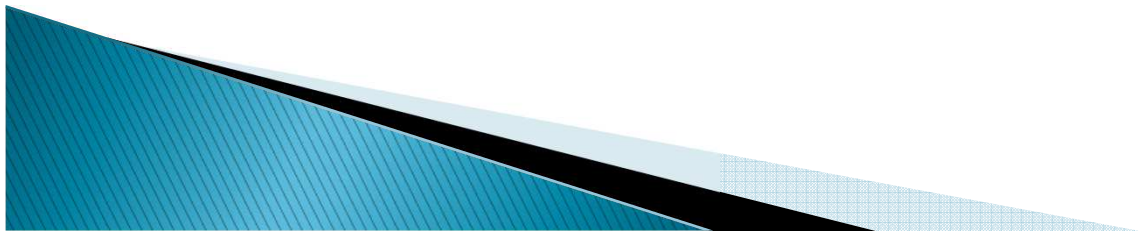
- Testing must be done at every phase.
- Testing of a phase must be build upon and checked against the results of the previous phase.
- Non-execution based testing is done in early phases (before executable code is produced).
- Execution-based testing can be done in later phases.



1.4 Software Testing Rules

Cutting Testing Costs can Increase other Costs

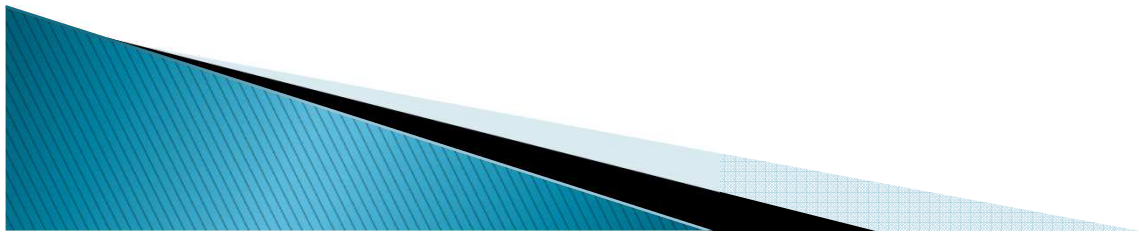
- Customer support can be very expensive. Less bugs = less calls.
- Customers will look for more reliable solutions.
- Software organizations must perform cost benefit analysis' to determine how much to spend on testing.



1.4 Software Testing Rules

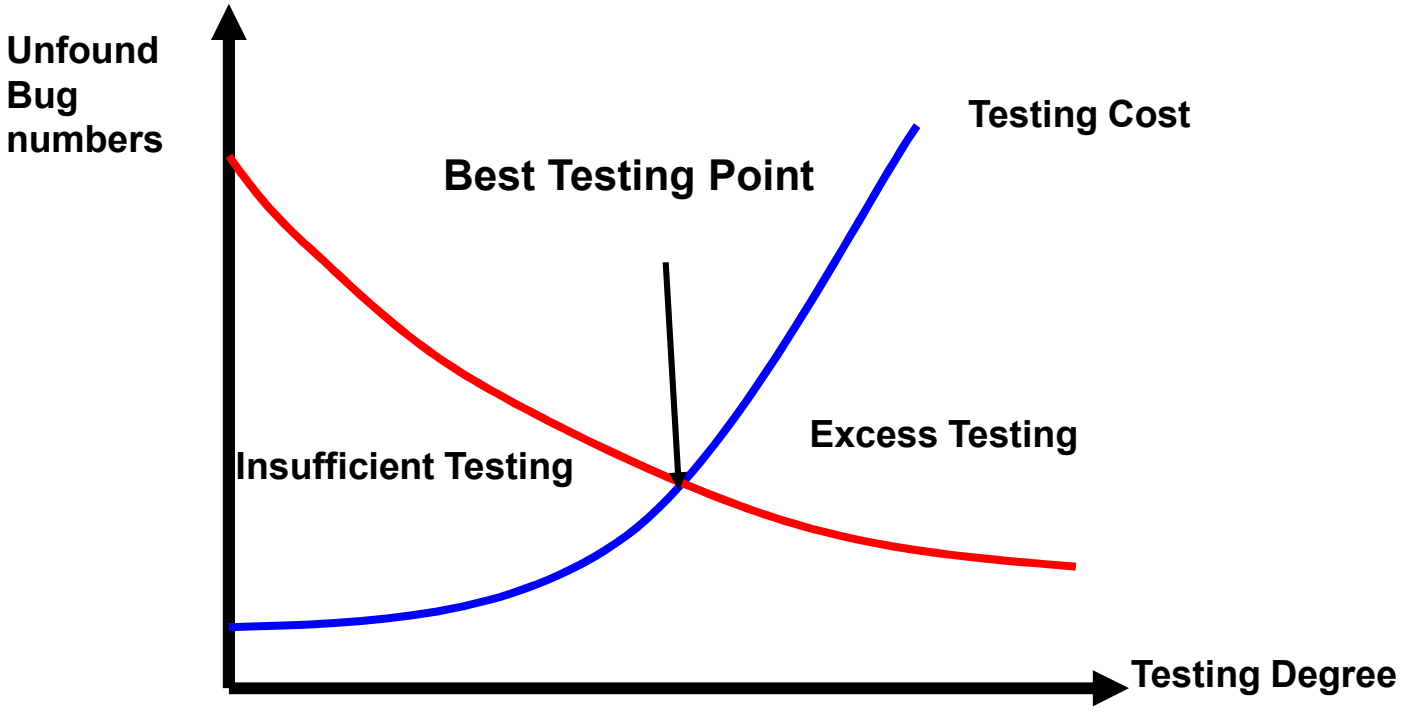
Cost of Delaying the Release of a Software Product

- Timing is another important factor to consider.
- New products: The first to the market often sells better than superior products that are released later.



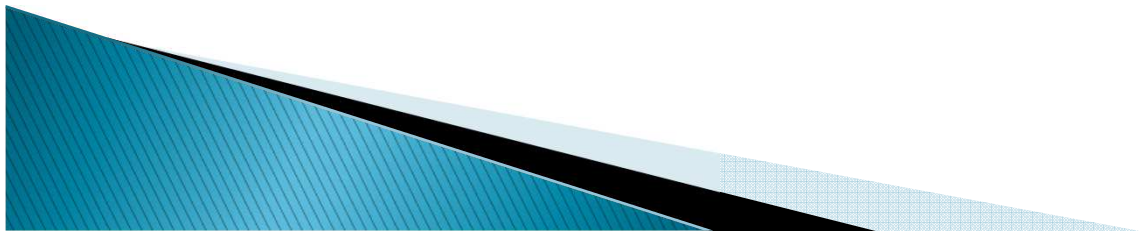
1.4 Software Testing Rules

- Curved line Testing and Cost



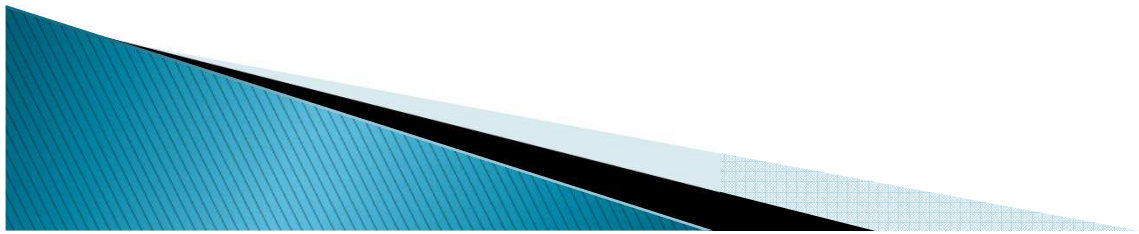
1.4 Software Testing Rules

- Good-enough
- 80-20



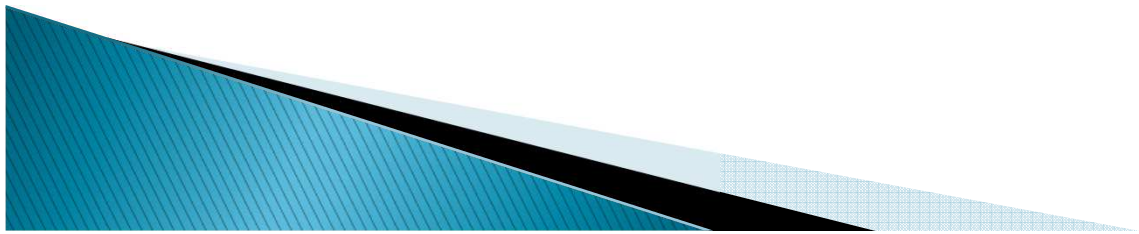
1.4 Software Testing Rules

- Assign your best people to testing
- Ensure that testability is a key objective in your software design
- Never alter the program to make testing easier
- Testing, like almost every other activity, must start with objectives



1.4 Software Testing Rules

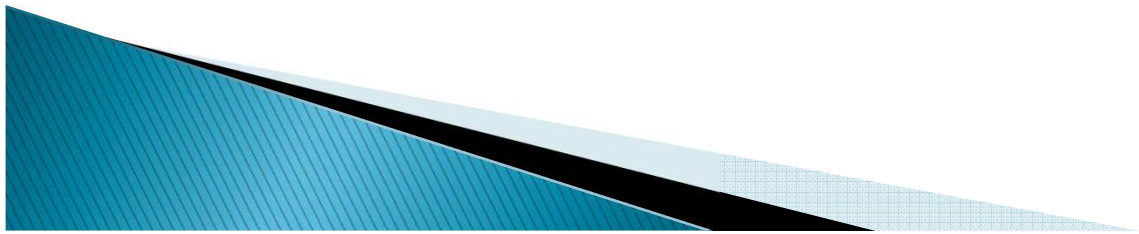
- Define your expected results.
- Understand the business reason behind the application.
- Use multiple levels and types of testing.
- Review and inspect the work, it will lower costs.
- Don't let your programmers check their own work.



1.4 Software Testing Rules

Testing Plan

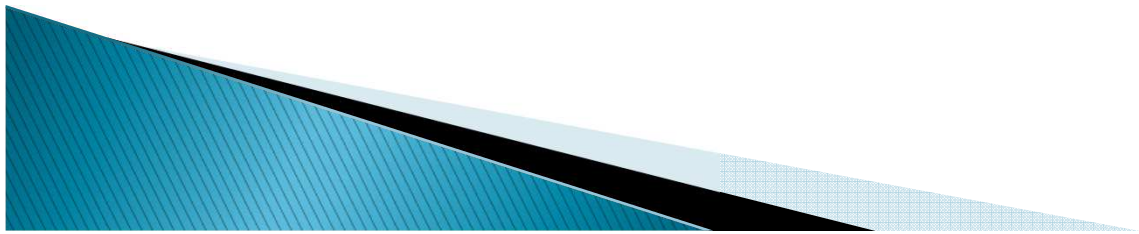
- Define the functions, roles and methods for all test phases.
- Test planning usually start during the requirements phase.
- Major test plan elements are:
  1. Objectives for each test phase
  2. Schedules and responsibilities for each test activity
  3. Availability of tools, facilities and test libraries.
  4. Set the criteria for test completion



1.4 Software Testing Rules

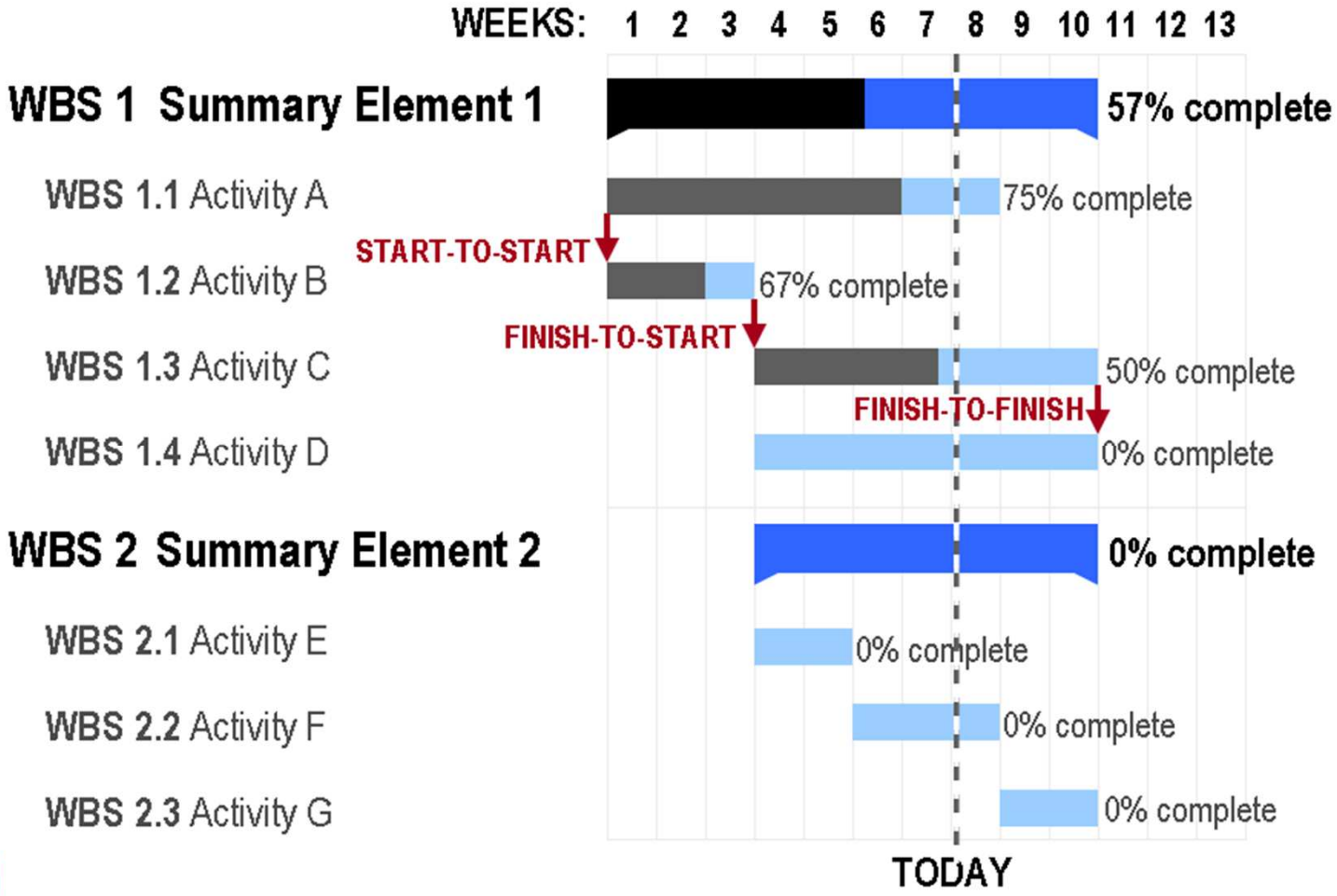
Schedules

- The goals of scheduling are to know:
  - What work needs to be completed?
  - How much work is left to do?
  - When will the work be finished?
  - Who will finish each task?
  - Other measurable queries.
- A **Gantt chart** is a popular type of bar chart that illustrates a project schedule.



1.4 Software Testing Rules

Schedules using Gantt chart



1.4 Software Testing Rules

- Test early and test often.
- Integrate the application development and testing life cycles.
- Formalize a testing methodology.
- Develop a comprehensive test plan.
- Use both static and dynamic testing.

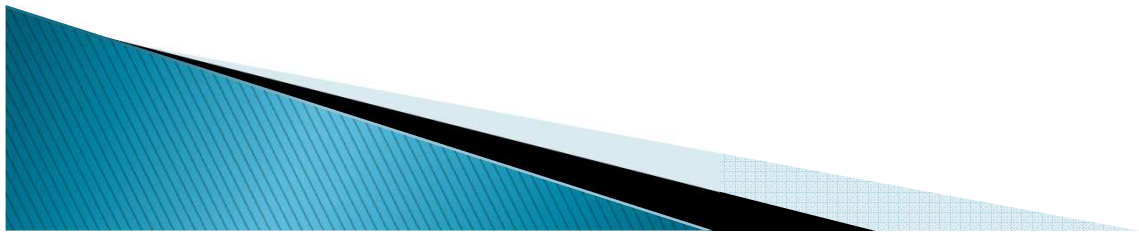
- Static testing

- Dynamic testing

1.4 Software Testing Rules

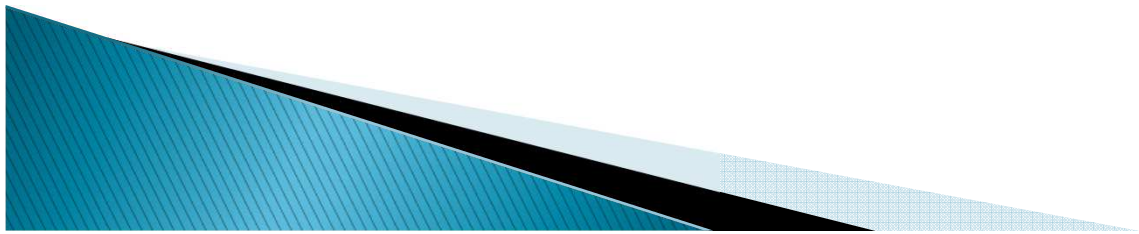
Enough Test cases

- What is test case ?
  - Inputs to test the system and the predicted outputs from these inputs if the system operates according to its specification.



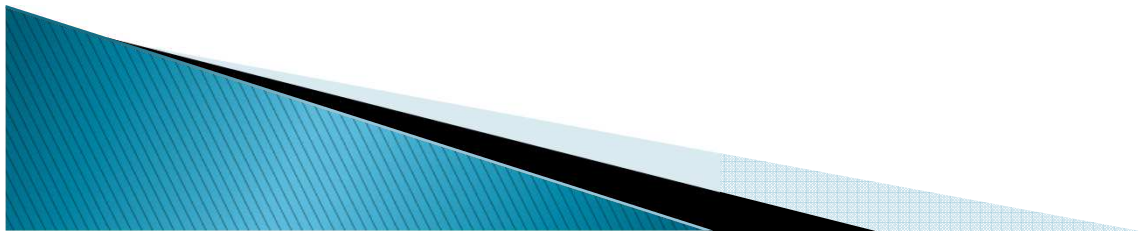
1.4 Software Testing Rules

- The basic component of testing is a **Test Case**
- In its most general form: (*inputs, expected-result)*
  - *inputs* include system state, user commands and data values to be processed
  - *expected result* includes visible/audible interface changes or changes in the system state
- Test cases are organized into **Test Suites**
  - functionality, security, performance, …



1.4 Software Testing Rules

- Test Case must be designed follow these characteristics :
  - Validity
  - Reusability
  - Easy organized
  - Evaluate
  - Manageability

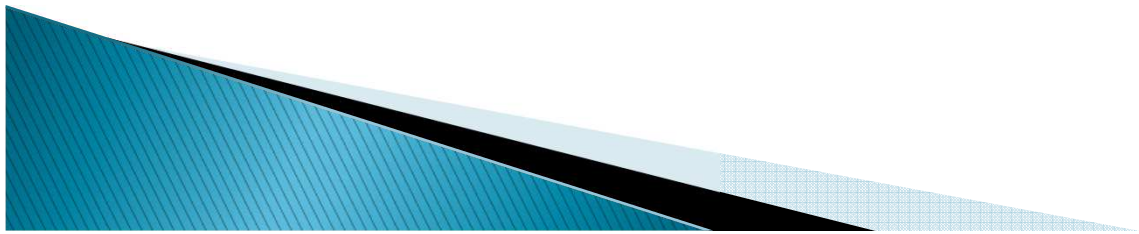


1.4 Software Testing Rules

- Example Test case - 1

  System Test of input of numeric month into data field

Ref.	Field /Button	Action	Input	Expected Result	Pass/ Fail
001	Month	Enter Data	0	Data rejected. Error Message 'Invalid Month'	Fail
002	Month	Enter Data	1	Data Accepted, January Displayed	Pass
003	Month	Enter Data	06	Data Accepted, June Displayed	Pass
004	Month	Enter Data	12	Data Accepted, December Displayed	Pass
005	Month	Enter Data	13	Data rejected. Error Message 'Invalid Month'	Fail



1.4 Software Testing Rules

Example Test case - 2

Search Researcher Page						
Test Ref.	**Reqs Ref.**	**Function**	**Inputs**	**Expected Result**	**Actual Result**	**Pass/ Fail**
2.001	REF003	Search Researchers	1. Forenames = John 2. Surname = <Blank> 3. eMail = <Blank>	All UCL researchers with forenames starting Pete displayed in alphabetic order, 23 records per page List comprises Name, Department, Occupation Type All data items hyperlinked	**427** matches - paging working correctly, data displayed correctly and in reasonable time (5 secs)	Pass
2.002	REF003	Search Researchers	1. Forenames = <Blank> 2. Surname = Smith 3. eMail = <Blank>	All UCL researchers with surnames starting Smith displayed in alphabetic order, 23 records per page List comprises Name, Department, Occupation Type All data items hyperlinked	**61** matches - paging working correctly, data displayed correctly and in reasonable time (5 secs)	Pass

1.4 Software Testing Rules

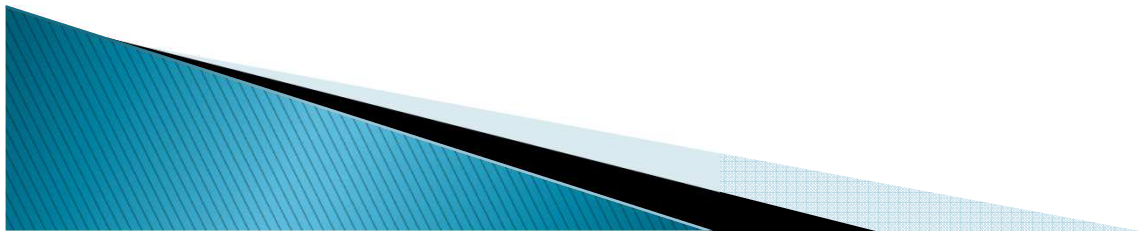
Example Test case - 3

No.	Type	Scenario	Test	To test?	New/ Change?	Covered (Y/N)	Pass/ Fail	Who	Date tested
39	**Admissions**	Basic Licence Admissions	Basic Licence Admissions - Setup	Y	Y	Y	P	SM	06/01/08
40			Basic Licence Admissions - Criterion Setup	Y	Y	Y	P	SM	06/01/08
41			Basic Licence Admissions - Admissions Policy	Y	Y	Y	P	SM	06/01/08
42			Basic Licence Admissions - ADT Import from EMS	Y	Y	Y	P	SM	06/01/08
43			Basic Licence Admissions - ASL Export to EMS	Y	Y	Y	F	SM	06/01/08

1.4 Software Testing Rules

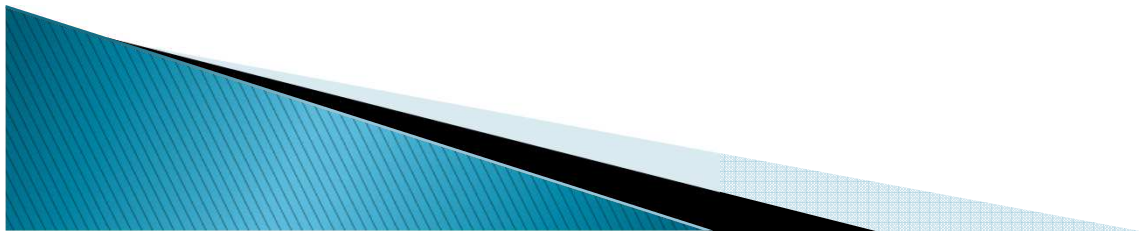
Test case execution

- A running of the software (under test) that provides the inputs specified in the test case and observing the results and comparing them to those specified by the test case
  - If the actual result varies from the expected result, then a failure has been detected



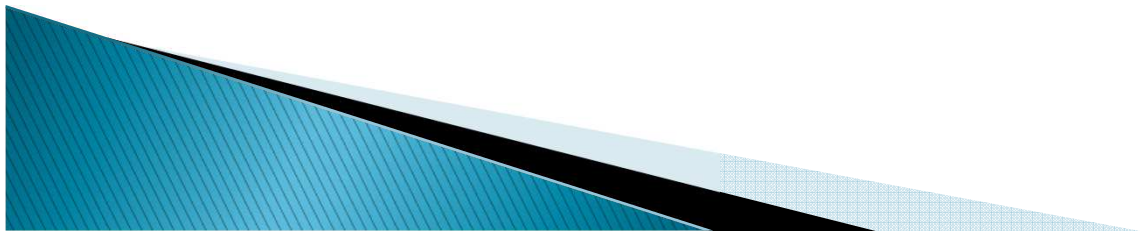
1.4 Software Testing Rules

- How to design a test case?
  - Understand the design of the product, specification, scenario.
  - A test cases should try to cover one or more situation.
  - Do not copy the specification.
  - Localized software testing
  - State
  - To decompose the test cases



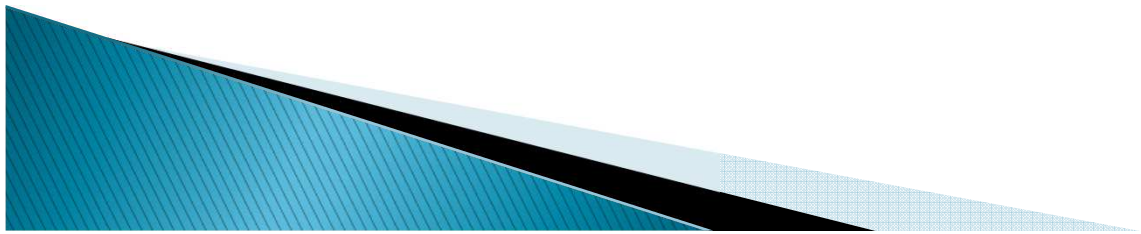
1.4 Software Testing Rules

- Test case design rules
  - Avoid vague test case.
  - Similar functions should be abstracted and classified.
  - Avoid complicated test case.



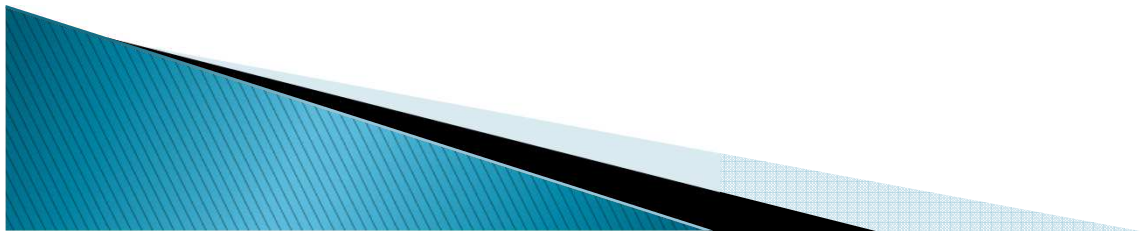
1.4 Software Testing Rules

- A good test case is one that has a high probability of detecting an undiscovered defect, not one that shows that the program works correctly
- It is impossible to test your own program
- A necessary part of every test case is a description of the expected result



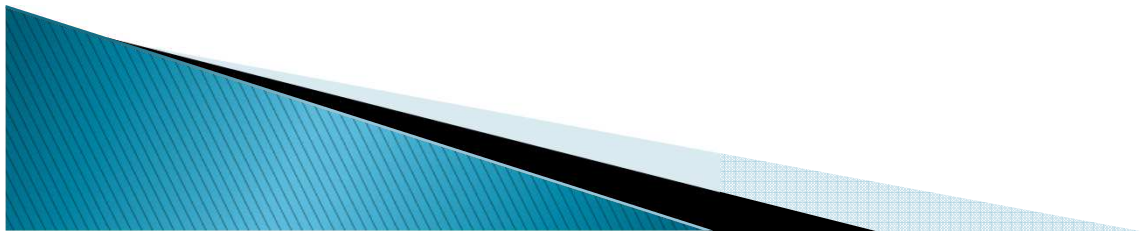
1.4 Software Testing Rules

- Avoid non-reproducible or on-the-fly testing
- Write test cases for valid as well as invalid input conditions.
- Thoroughly inspect the results of each test
- As the number of detected defects in a piece of software increases, the probability of the existence of more undetected defects also increases



1.4 Software Testing Rules

- Test passing rules
  - Whether all test cases are executed.
  - Whether function design is finished.
  - Whether we get enough bugs.



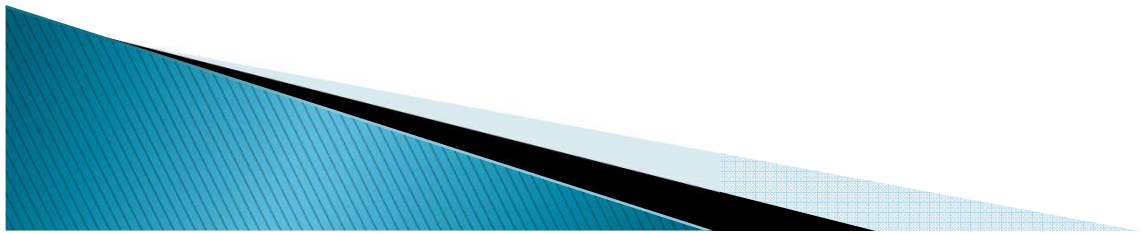
1.4 Software Testing Rules

Test documents

- Test plan
  - Quality objectives, resource needs, schedules, assignments, methods, etc.
- Test cases
  - Inputs and expected outputs.
- Bug reports
  - E.g., the Bugzilla web-based bug tracker.
- Test tools and automation
- Metrics, statistics, and summaries
  - Number of unresolved bugs, mean time to repair a bug, etc.

1.4 Software Testing Rules

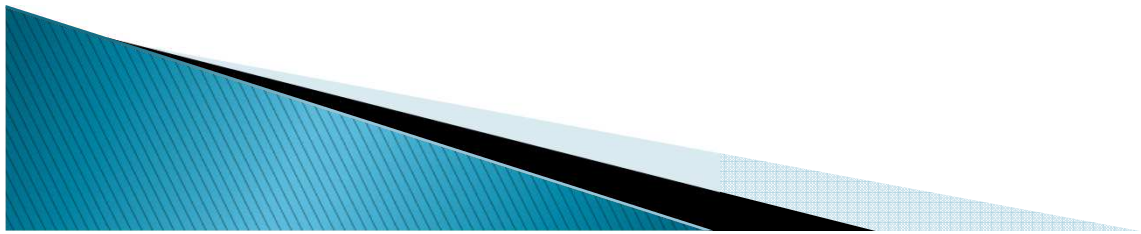
- Test Execution & Reporting
  - Testing should be treated like an experiment.
  - Testing require that all anomalous behavior be noted and investigated.
  - Big companies keep a special library with all copies of test reports, incident forms, and test plans



1.4 Software Testing Rules

Software Project Staff

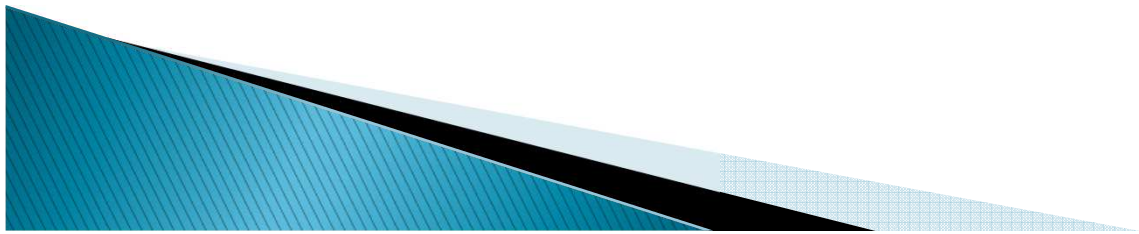
- Project managers
  - Write product specification, manage the schedule, critical decision tradeoffs
- Software architects, system engineers
  - Design the software, work closely with programmers
- Programmers, developers, coders
  - Write code, fix bugs
- Testers, quality assurance staff
  - Find bugs, document bugs, track progress of open bugs
- Technical writers
  - Write manuals, on line documentation
- Configuration managers, builders
  - Packaging and code, documents, and specifications



1.4 Software Testing Rules

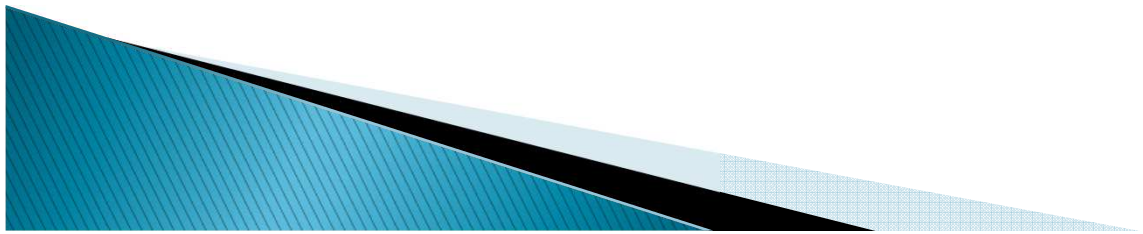
Who's involved in testing?

- Requirements Analysts – Inspections, Peer Reviews
- Developers – Code Inspection, Unit Testing
- Testers – System & Integration Testing
- Trainers – Training materials production
- Users – User Acceptance Testing
- Project Managers – Scheduling, Resourcing, Risks, Issues, Defect Stats
- Everybody is responsible for quality - NASA



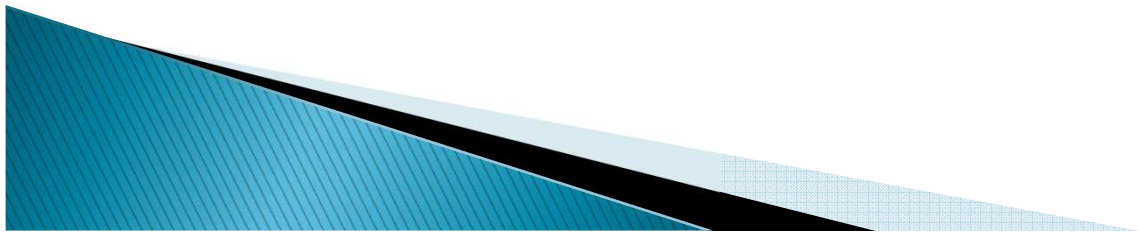
1.4 Software Testing Rules

- Capability for Software Tester
  - Technique ability
  - Communication ability
  - Suspicion
  - Confidence
  - Patience
  - Analysis ability
  - Cooperation



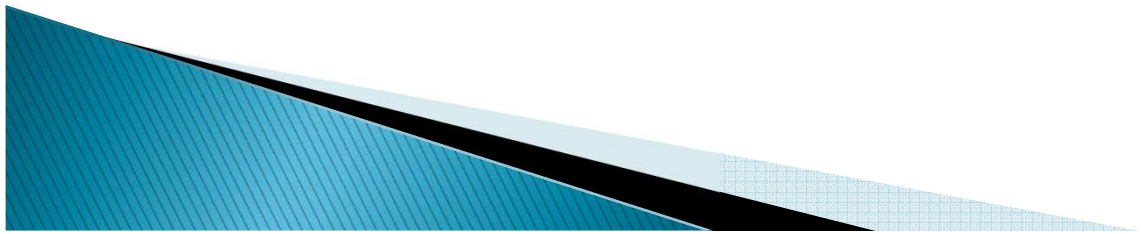
What to look for when interviewing someone for the position of software tester

- Are they explorers?
- Are they troubleshooters?
- Are they relentless?
- Are they creative?
- Are they perfectionists (within reason)?
- Do they exercise good judgment?
- Are they tactful and diplomatic?
- Are they persuasive?



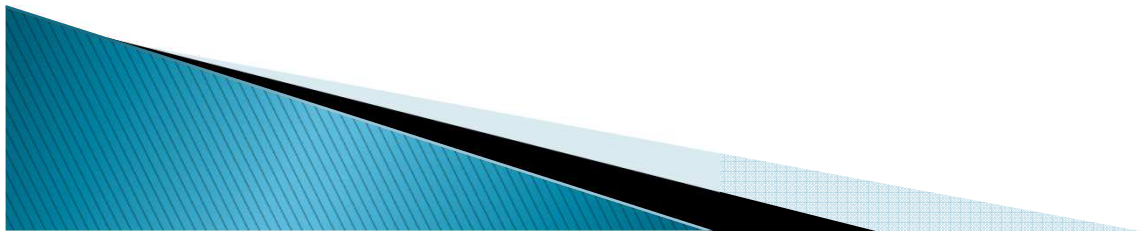
You now know …

- … what is a bug
- … the relationship between specification and bugs
- … the cost of a bug relative to when it is found
- … the unattainable goal of perfect software
- … the goal of the software tester
- … valuable attributes of a software tester



You now know …

- … what is software
- … what is software engineering
- … what is the composition of a software development organization
- … what are the major phases of a software development project
- … how major phased are organized

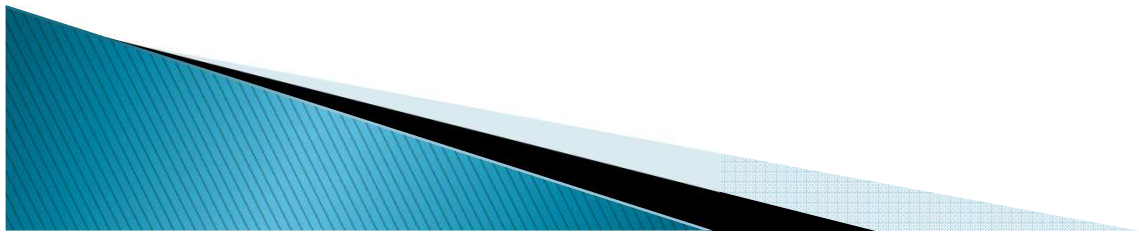


Thank you !

Test Automation

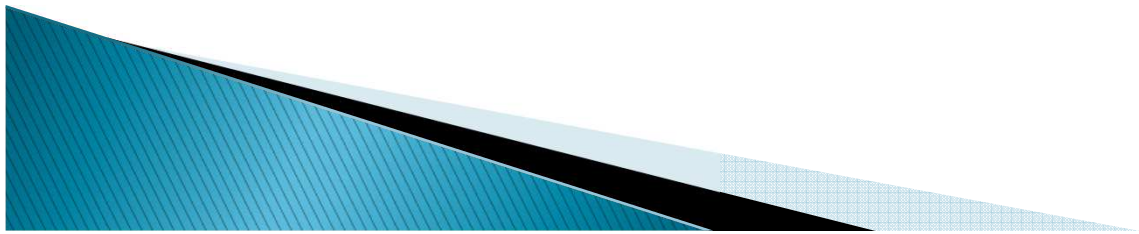
e.g.:

- If a manual test costs $X to run the first time, it will cost $X to run every time thereafter.
- An automated test can cost 3 to 30 times $X the first time, but will cost about $0 after that.



Designing Test Cases

- Dynamic testing relies on good test cases
- Testing techniques are characterized by their different strategies how to generate test cases
- Testing effectiveness is not maximized by arbitrarily selected test cases since they may expose an already detected fault by some other test case and waste effort
  - Number of test cases do not determine the effectiveness
  - Each test case should detect different faults



The goal of software testing

- The process of uncovering evidence of defects in software systems
  - Does not include efforts associated with tracking down bugs and fixing them
- No amount of testing will improve the quality of a computer program
  - The more testing we do of a system, the more convinced we might be of its correctness
  - Testing cannot in general prove a system works 100% correctly