Ingineria programării

Curs 10 – 27 Aprilie

Cuprins

- Recap – Design Patterns
  - Creational Patterns
  - Structural Patterns
  - Behavioral Patterns
  - Other Design Patterns
- Quality Assurance
  - Software Testing
  - Testing Methodologies
  - Testing process
  - Manual Testing vs. Automatic Testing

Recapitulare

- GOF = ?
- Creational Patterns
- Structural Patterns
- Behavioral Patterns

Recapitulare – CP

- Abstract Factory
- Builder
- Factory Method
- Prototype
- Singleton

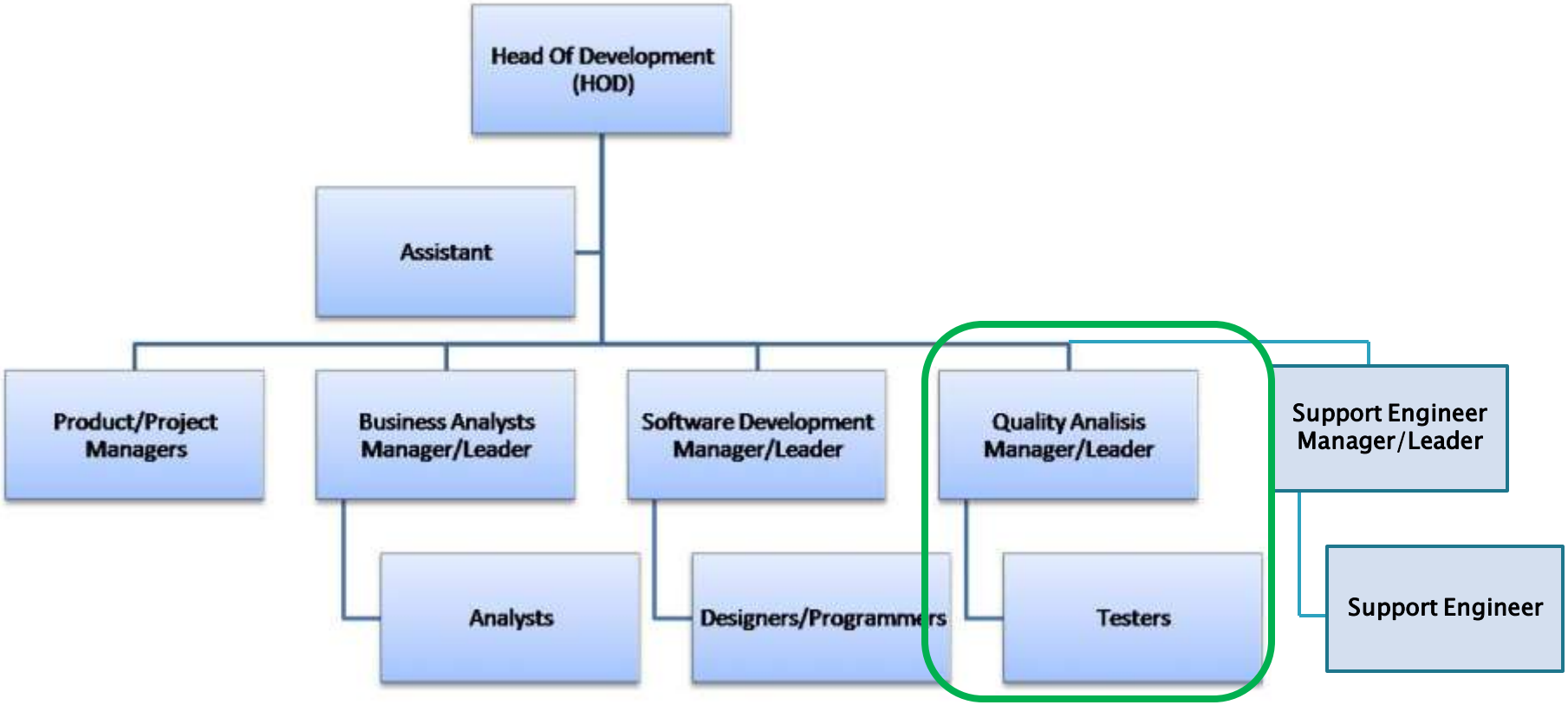
Recapitulare – SP

- Adapter
- Bridge
- Composite
- Decorator
- Façade
- Flyweight
- Proxy

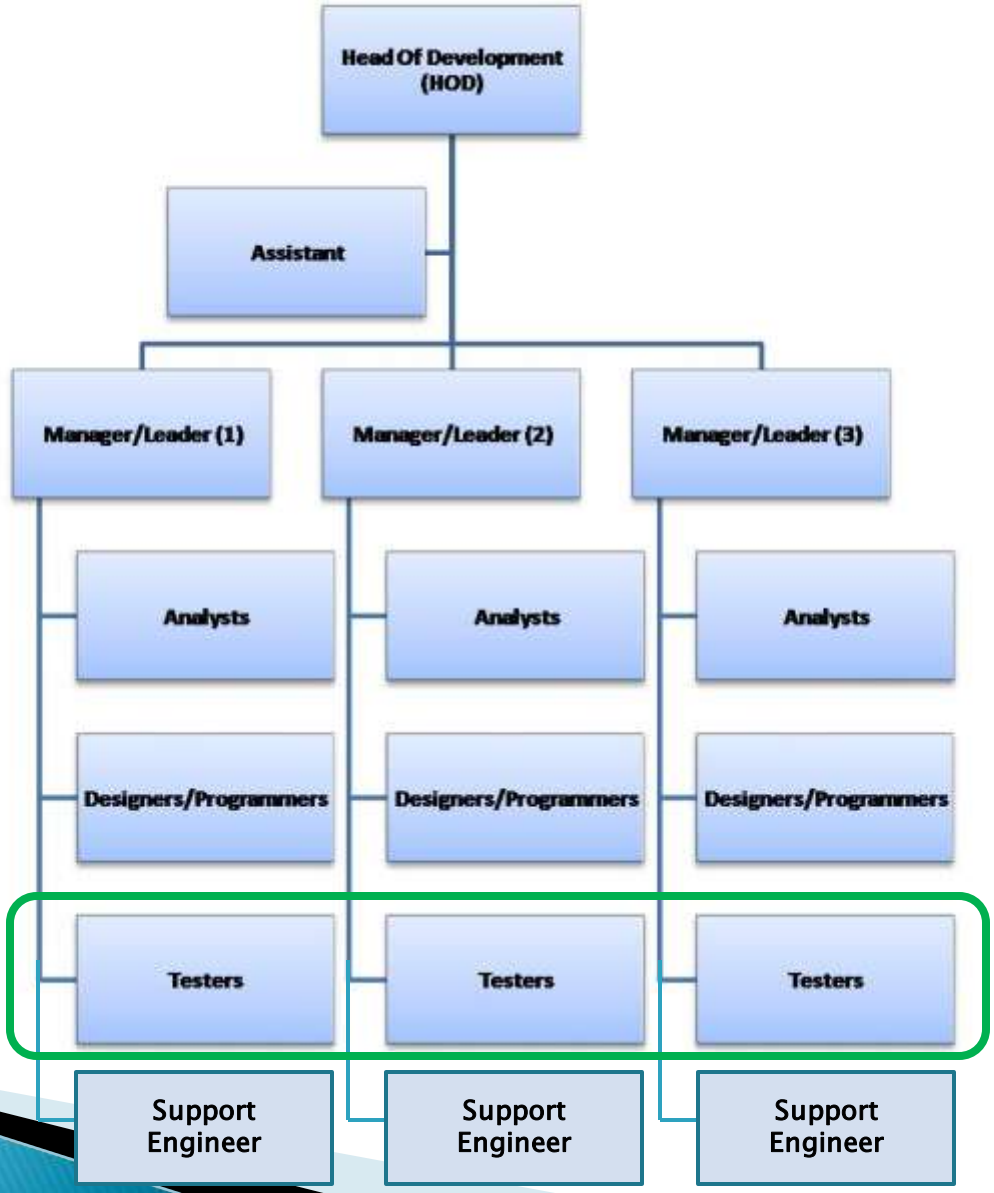
Recapitulare – BP

- Chain of Responsibility
- Command
- Interpreter
- Iterator
- Mediator
- Memento
- Observer
- State
- Strategy
- Template Method
- Visitor

Hierarchy of an IT Company – Compartments



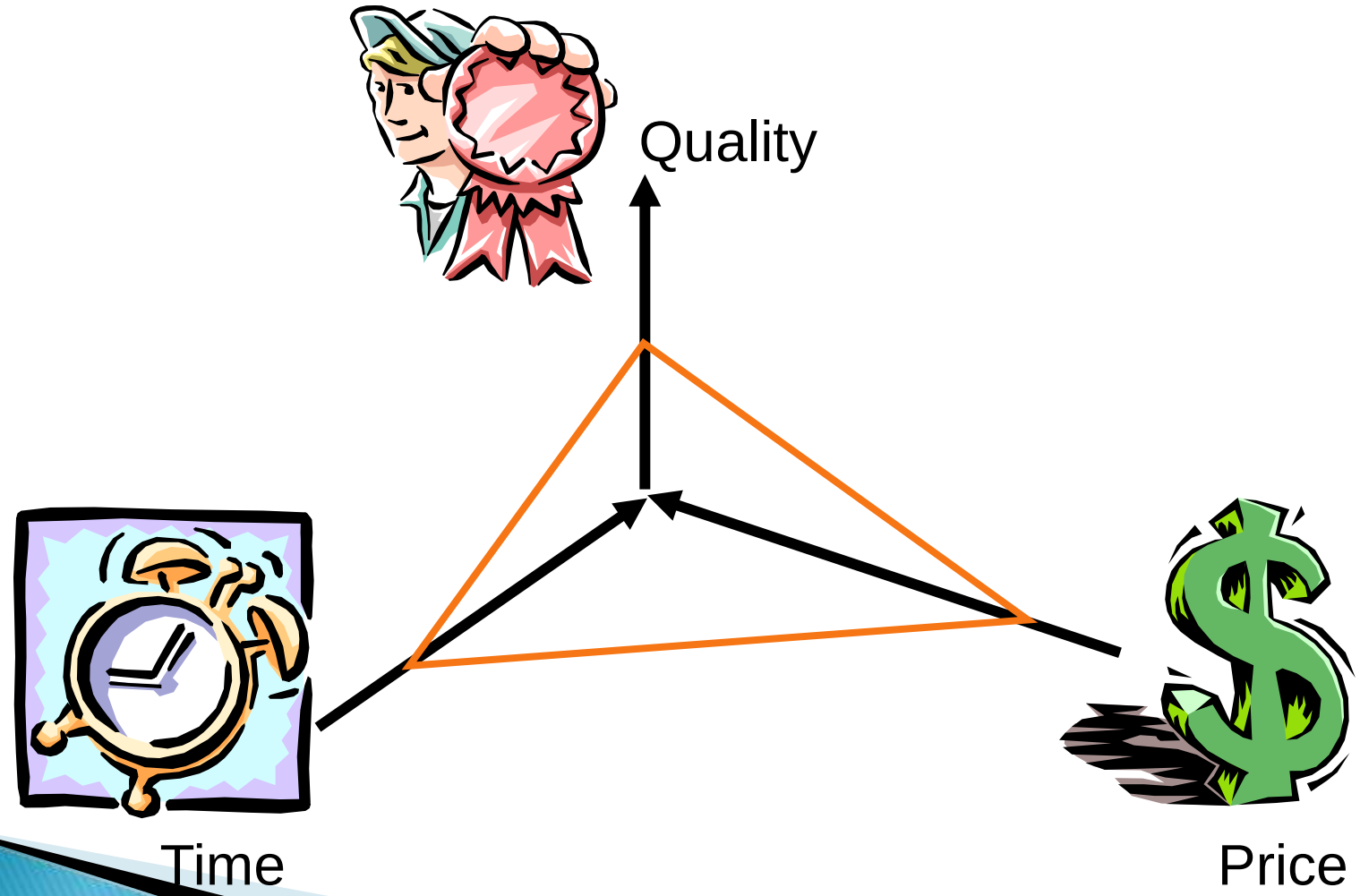
Hierarchy of an IT Company – Projects



Quality Assurance

- **Planned and systematic production processes** that provide confidence in a product's suitability for its intended purpose.
- A set of activities intended to ensure that **products satisfy customer requirements**
- *QA cannot absolutely guarantee the production of quality products but makes this more likely*
- Two key principles of QA:
  - "fit for purpose" – the product should be suitable for the intended purpose, and
  - "right first time" – mistakes should be eliminated

Quality Assurance Dilemma



Quality Assurance – Definition

"The process of exercising or evaluating a system by manual or automated means to verify that it satisfies specified <u>requirements</u> or to identify differences between expected and actual results."

(IEEE Standard Glossary, 1983)

Software Quality Assurance

- (SQA) consists of a means of **monitoring the software engineering processes and methods used to ensure quality**
- May include ensuring conformance to one or more standards, such as ISO 9000 or CMMI

Software Quality Assurance

- SQA encompasses the entire software development process, which includes processes such as
  - *software design,*
  - *coding,*
  - *source code control,*
  - *code reviews,*
  - *testing,*
  - *change management,*
  - *configuration management, and*
  - *release management*

ISO 9000

- ISO 9000 is a family of standards for quality management systems
- Some of the requirements in ISO 9001 (from ISO 9000 family) include
  - a set of procedures;
  - monitoring processes;
  - keeping adequate records;
  - checking output for defects;
  - regularly reviewing individual processes;
  - facilitating continual improvement

Software Quality Assurance

- Quality assurance is concerned with prevention
  - Defect Prevention
  - Processes
  - Continuous improvement of this processes
- Quality control is concerned with correcting
  - Software testing is a subset of Quality control

Image credit test-institute.org

Software Testing – Introduction

- An empirical investigation conducted to provide information about the *quality of the product or service under test, with respect to the context in which it is intended to operate*.
- Allow the business to *appreciate and understand* the risks at implementation of the software
- Test techniques include the process of executing a program or application with **the intent of finding software bugs**
- The process of **validating and verifying** that a software program/application/product meets the business and technical requirements that guided its design and development

Software Testing – When?

- Can be implemented at any time in the development process
- However the most test effort is employed after the requirements have been defined and coding process has been completed

Software Testing

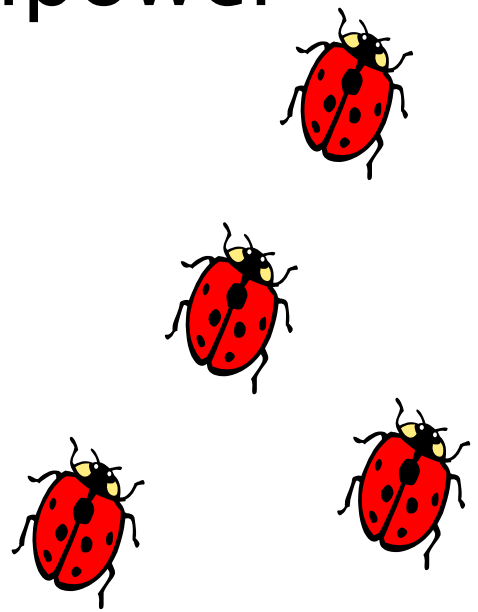
- Software Testing is NOT a phase
- It is integrated in all phases of software development
- Each development step has an attached testing documentation

What is the purpose of testing?

- We need not only to find bugs but also to prevent them (which is better)
- To know when to stop because effectiveness and economics of the process is essential.
- To know that not all system requires the same level of quality (mission critical against IT).
- Testing is not only for the SOFTWARE it is for all DELIVERABLES

Why are there bugs in software?

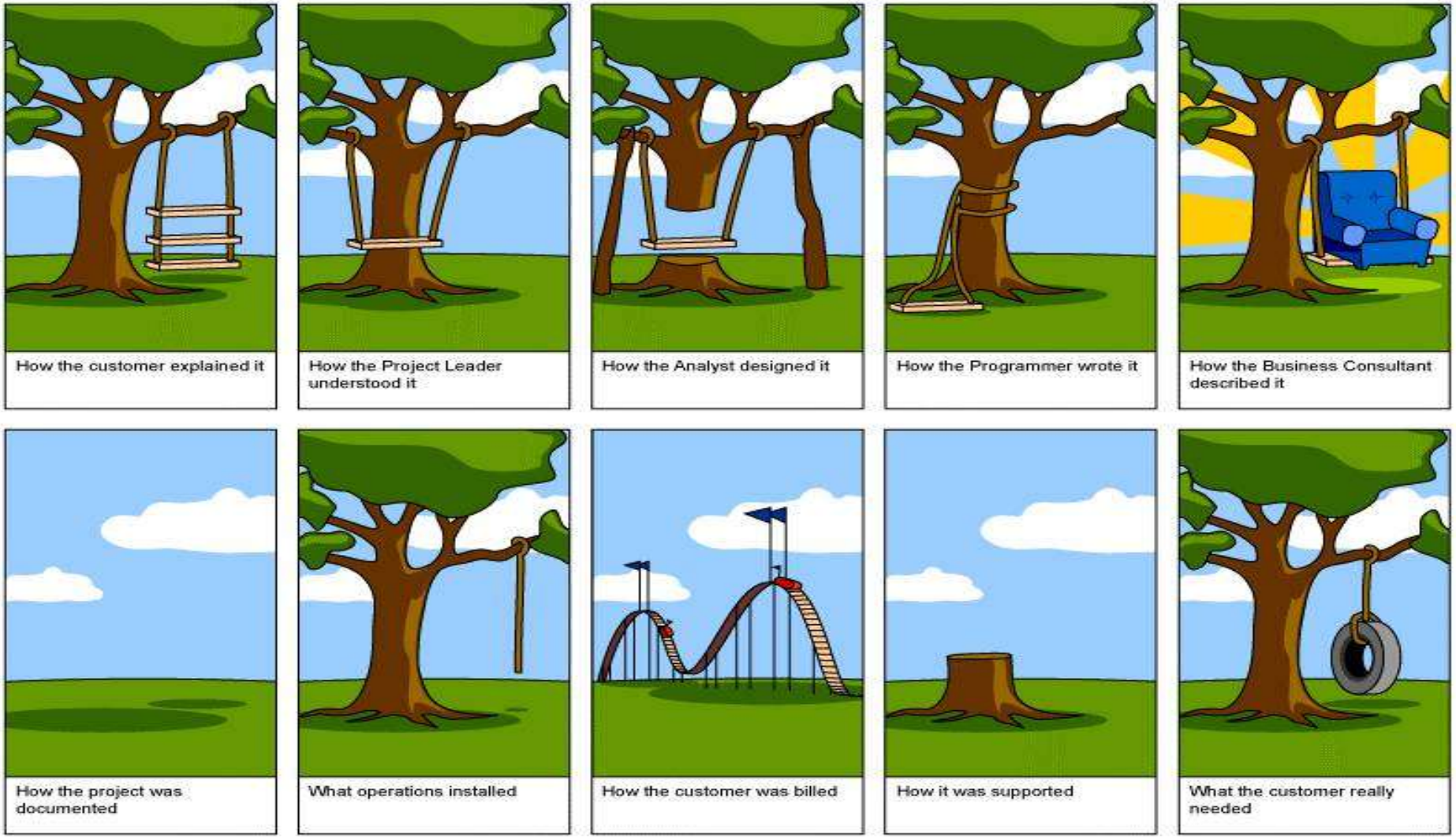
- Miscommunication
- Misunderstanding
- Low professional manpower
- Time pressures



Miscommunication

"Didn't you get my e-mail?"

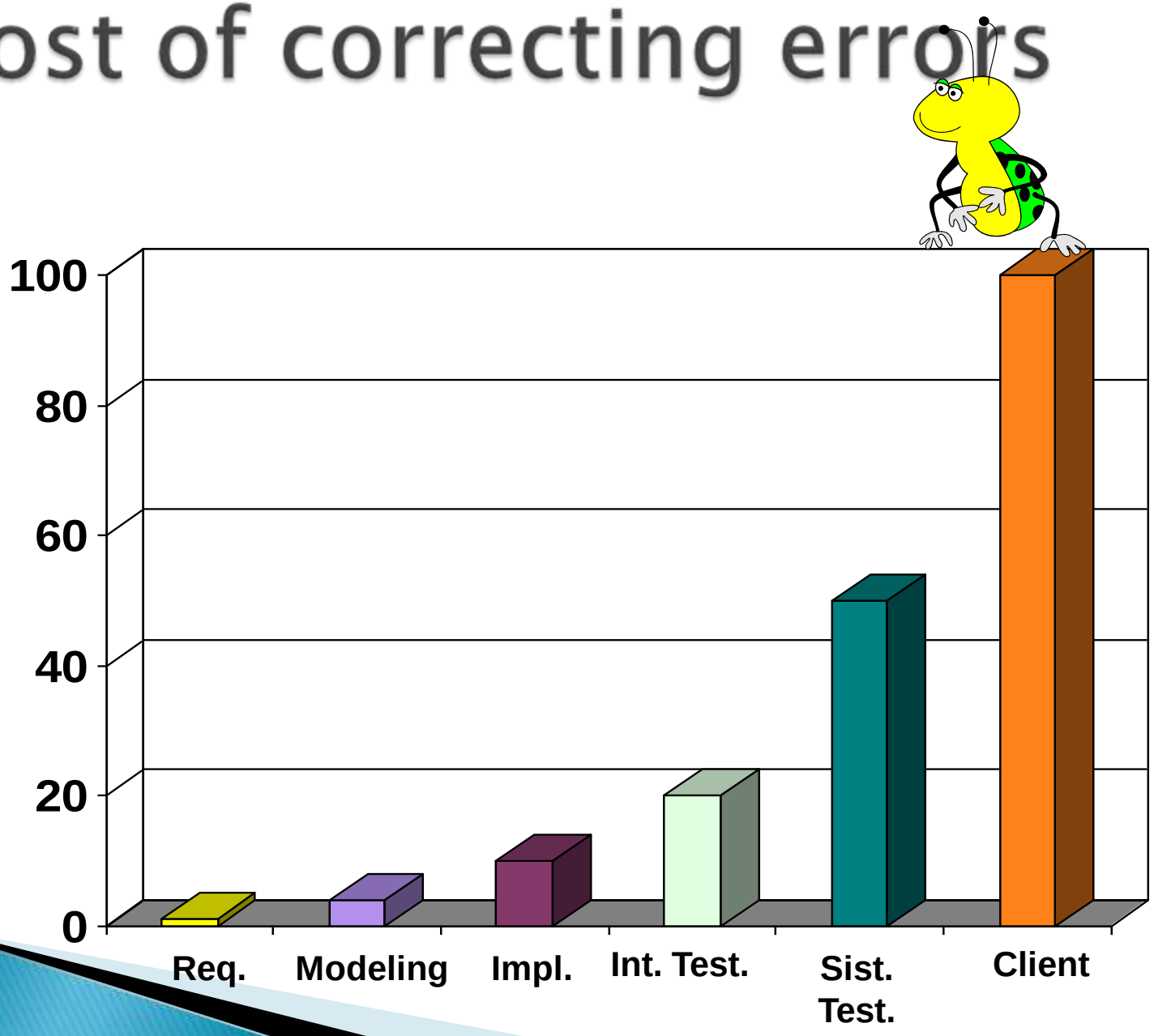
Misunderstandings



What genenerates most errors?

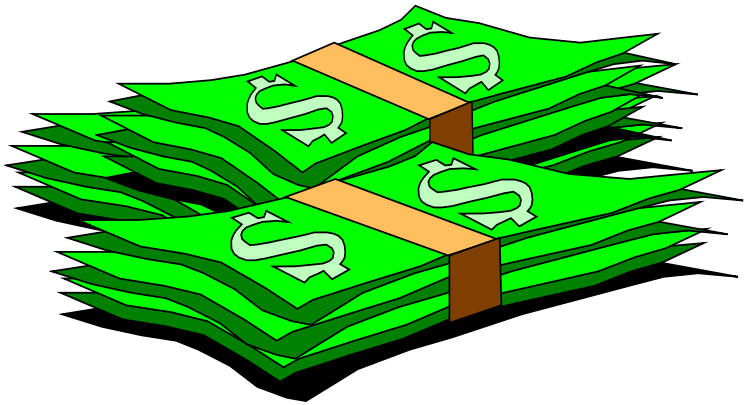
- Incorrect or incomplete requirements 50%
- Ambiguous or incomplete modeling 30%
- Programming errors 20%

Cost of correcting errors

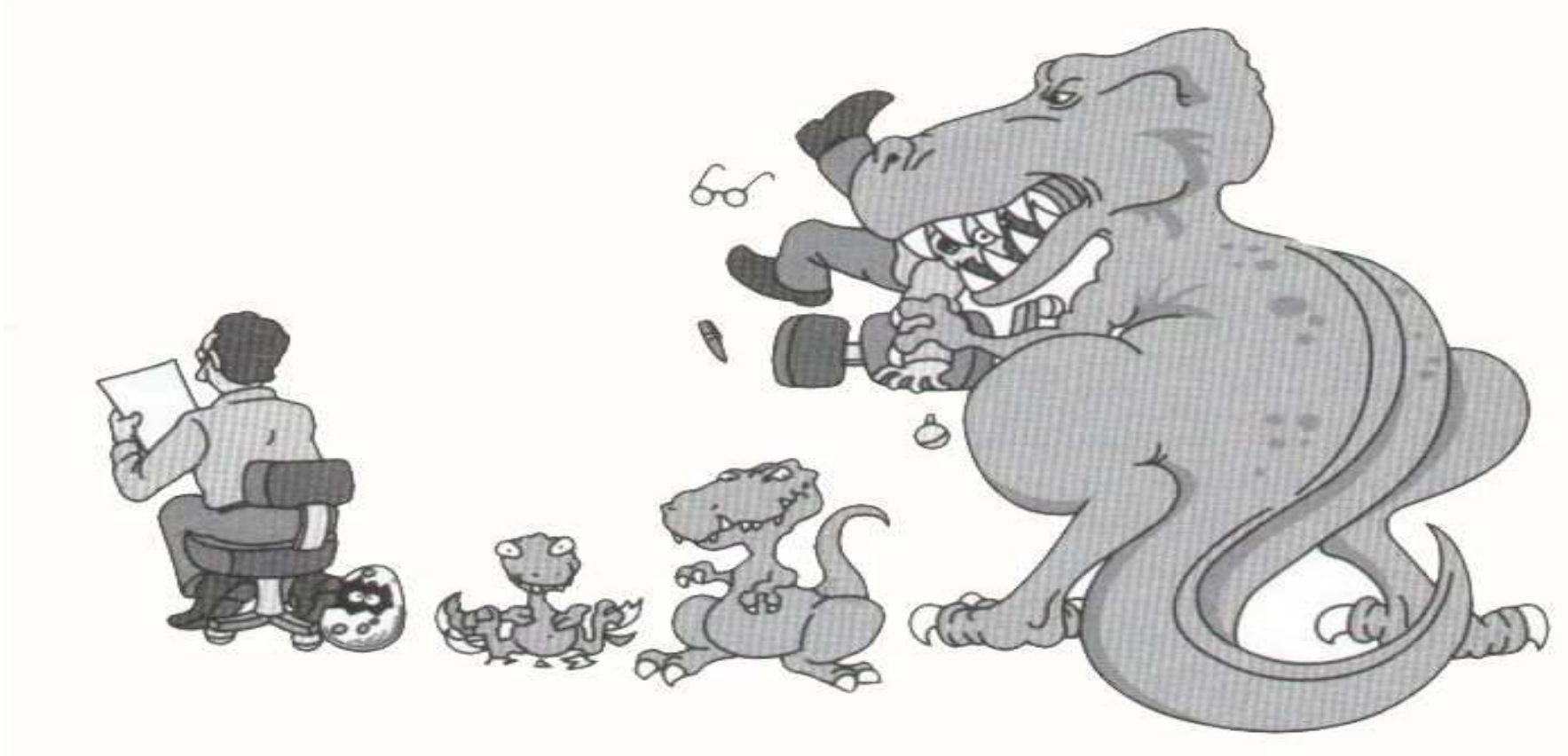


Note

Late error detection⇒ greater repair cost



Error must be repaired as soon as possible



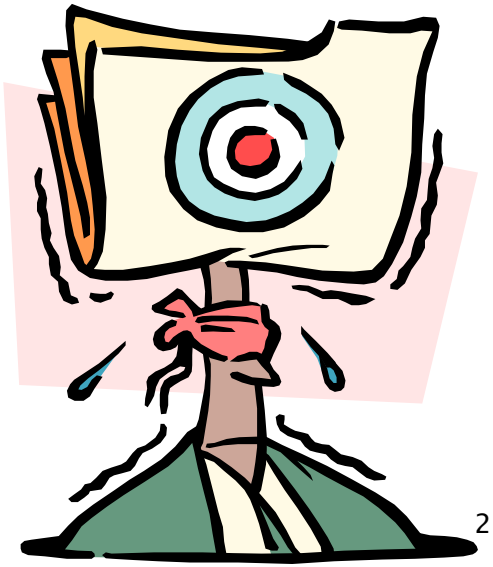
REQ. MODELING IMPLEM. CLIENT TESTING

Professional Testing

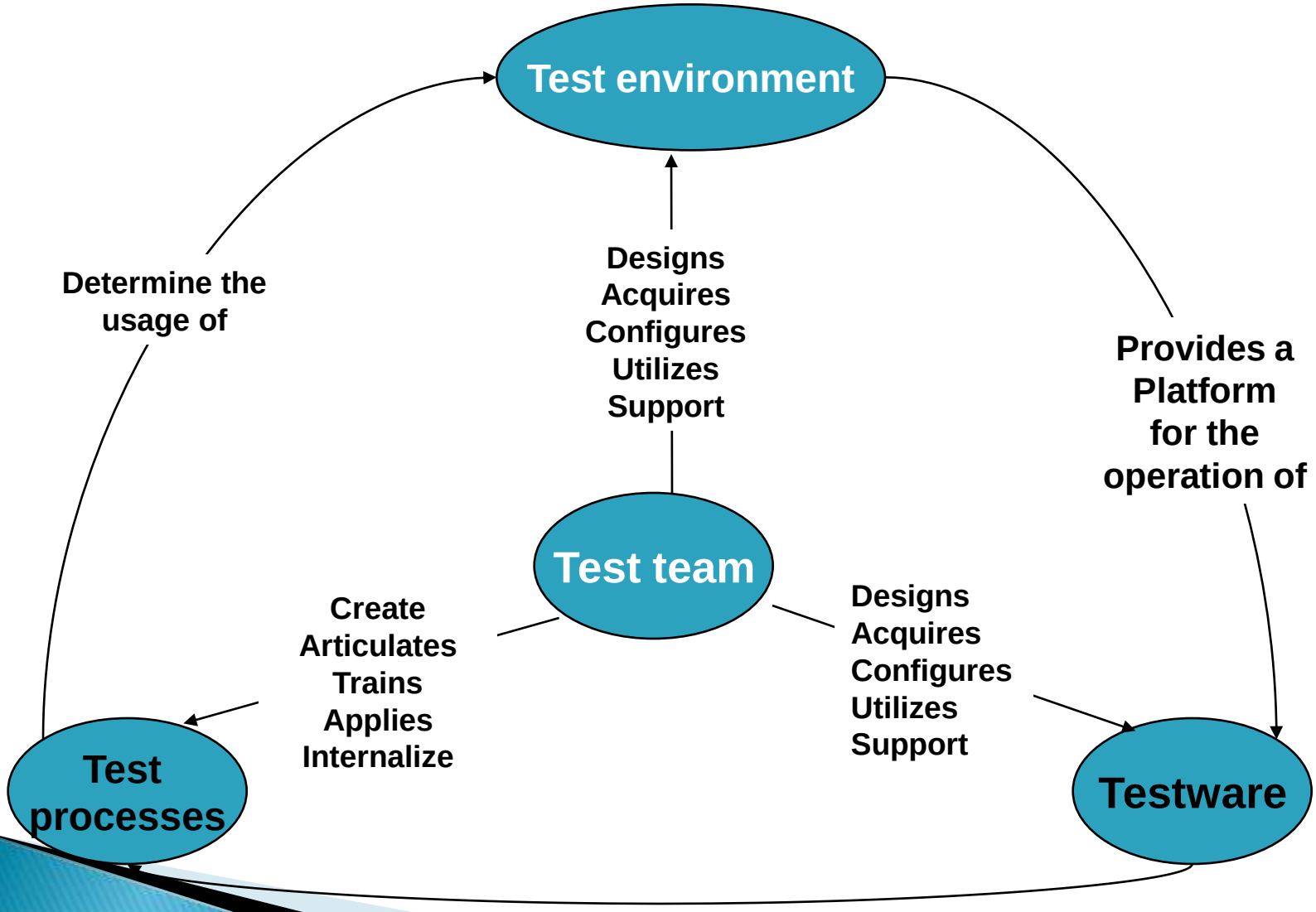
Professional testing means finding the least amount of test cases which will check the most amount of system features.

When does testing stop?

- Never
- When the number of errors found in a test cycle is lower than a set amount
- When no more critical faults are found
- When we run out of time



Schema of a Testing System



Testing Methodology

- Differences between testing and debugging
- Layers of testing
- Testing methods
- Testing content
- Manual vs Automatic Testing

Testing vs. Debugging

Testing

- Check compliance to requirements
- Normally carried out by an external and neutral party
- Is planned and controlled

Debugging

- Check validity of program sections
- Run by the developer
- Is a random process

Layers of testing

- Unit testing or debugging
- Module/Sub-System
- Integration
- System
- Acceptance

Unit Testing

- Testing a function, program screen, feature
- Run by programmers
- Predefined
- Results must be documented
- Input and Output simulators are used

Integration testing

- Testing of several modules at the same time
- Testing coexistence
- Run by programmers or testers
- Pre-planned testing
- Results must be documented

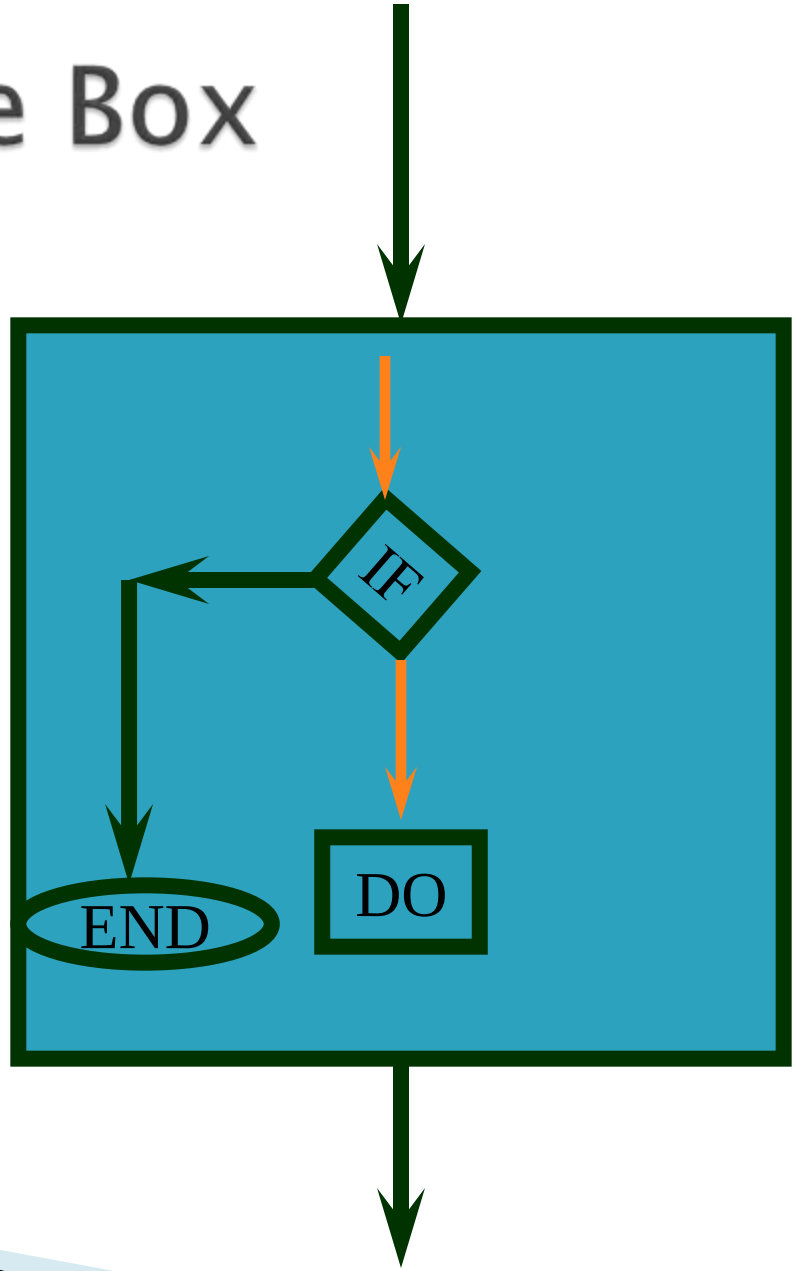
System Testing

- System testing of software or hardware is testing conducted **on a complete, integrated system** to evaluate the system's compliance with its specified requirements.
- System testing falls within the scope of **black box testing**
- System testing is a more limiting type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

Testing methods

- White Box
- Black Box
- Gray Box
- Graphical user Interface Testing
- Acceptance Testing
- Regression Testing

White Box

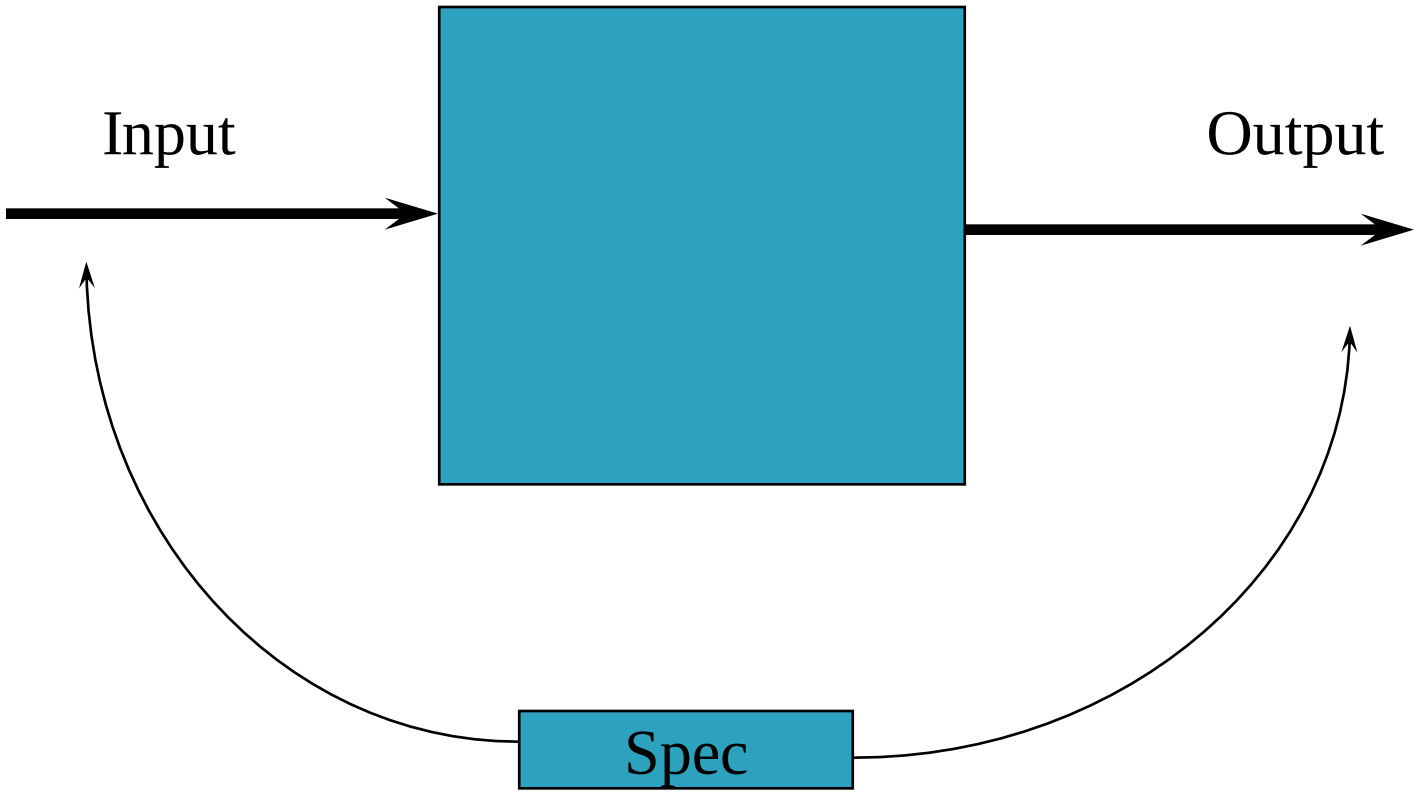


White Box (2)

- The tester has access to the internal data structures and algorithms

- Types of white box testing
  - api testing – Testing of the application using Public and Private APIs
  - code coverage – creating tests to satisfy some criteria of code coverage
  - fault injection methods
  - mutation testing methods
  - static testing – White box testing includes all static testing

Black Box



- "like a walk in a dark labyrinth without a flashlight,"

Black Box (2)

- Specification-based testing

- Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzzy testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.

Grey Box

- This involves having access to internal data structures and algorithms for purposes of designing the test cases, but testing at the user, or black-box level
- Manipulating input data and formatting output do not qualify as "grey-box," because the input and output are clearly outside of the "black-box" that we are calling "the software under test"

GUI Testing

- In computer science, GUI software testing is the process of testing a product that uses a graphical user interface, to ensure it meets its written specifications.
- The variety of errors found in GUI applications:
  - Data validation, Incorrect field defaults, Mandatory fields, not mandatory, Wrong fields retrieved by queries, Incorrect search criteria
  - Field order, Multiple database rows returned, single row expected
  - Currency of data on screens, Correct window modality?
  - Control state alignment with state of data in window?

Acceptance Testing

- **A black-box testing performed on a system** prior to its delivery
- In software development, acceptance testing by the system provider is often distinguished from acceptance testing by the customer (the user or client) prior to accepting transfer of ownership.
- In such environments, acceptance testing performed by the customer is known as **user acceptance testing** (UAT).
- This is also known as **end-user testing**, site (acceptance) testing, or field (acceptance) testing.

Regression Testing

- Regression testing is **any type of software testing** which seeks to uncover software regressions.
- Such regressions occur whenever software functionality that was previously working correctly, stops working as intended.
- Typically regressions **occur as an unintended consequence of program changes**.
- Common methods of regression testing include **re-running previously run tests and checking whether previously fixed faults have re-emerged**

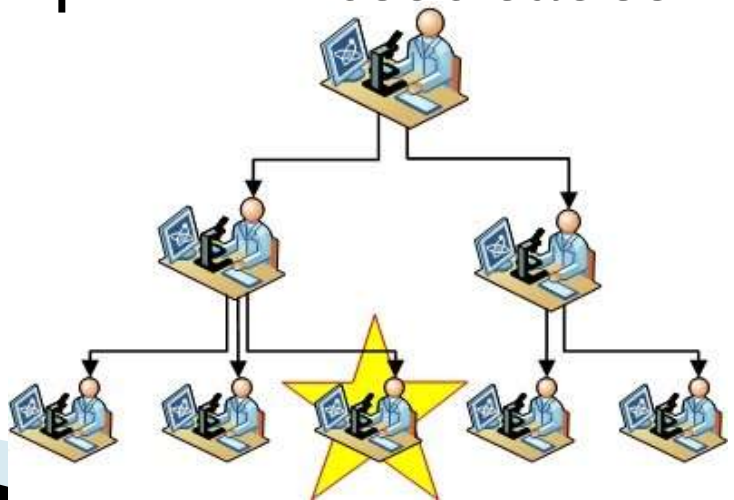
Automatic vs Manual Testing

- *Problems are found quickly*
- *Cheap to repeat*
- *The process of writing code is more flexible*
- *Less manual testing*
- *Software development becomes predictable and can be planned*

- *Solves interface issues: correctness of text, messages, page layout, element order, visibility etc.*
- *Writing test scenarios can be difficult and implies technical knowledge of the entire system*

Manual Testing

- *Manual testing is the process of manually testing software for defects*
- It requires a **tester** to play the role of an end user, and use most of all features of the application to ensure correct behavior
- To ensure completeness of testing, the tester often follows a written **test plan** that leads them through a set of important **test cases**



Definitions

- **Test Strategy** is developed by the "Project manager" which contains what type of technique to follow and which module to test
- **Test Plan** is developed by the Test Lead, which contains "what to test", "how to test", "when to test", "who to test"
- **Test Scenario** is a name given to test case. It is dealt with by the Test Engineer
- **Test Case** specifies a testable condition to validate functionality. The test cases are deal by Test Engineer

Test Strategy vs. Test Plan

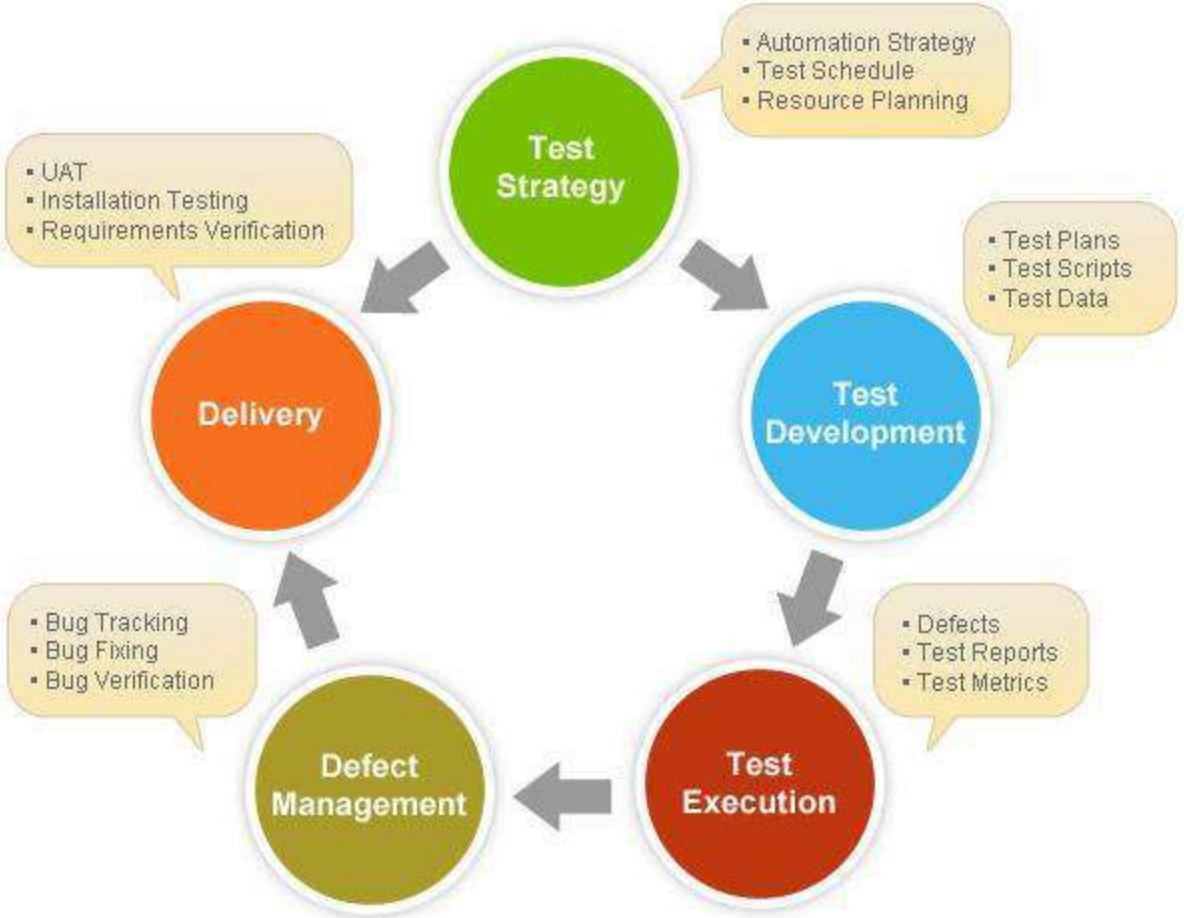


Image credit iconma.com

Test Plan

- **A systematic approach to testing a system**
- Contains a detailed understanding of what the eventual workflow will be
- Documents the strategy that will be used to verify and ensure that a product or system meets its design specifications and other requirements
- Is usually prepared by or with significant input from Test Engineers

Test Plan Components

- May include one or more of the following:
  - *Design Verification or Compliance test*
  - *Manufacturing or Production test*
  - *Acceptance or Commissioning test*
  - *Service and Repair test*
  - *Regression test*

Test Plan Structure (IEEE 829-1998)

- Test plan identifier
- Introduction
- Test items
- Features to be tested
- Features not to be tested
- Approach
- Item pass/fail criteria
- Suspension criteria
- Test deliverables
- Testing tasks
- Environmental needs
- Responsibilities
- Staffing and training needs
- Schedule
- Risks and contingencies
- Approvals

Test Plan – Life Cycle

Test Case

- **A set of conditions or variables** under which a tester will determine whether an application or software system meets specifications
- **A sequence of steps** to test the correct behavior/functionalities, features of an application
- In order to fully test that all the requirements of an application are met, there must be *at least one test case for each requirement* (*two recommended*)

Test Case Format

- Test case ID
- Test case Description
- Expected Output
- Actual Output
- Pass/Fail
- Remarks
- Test step or order of execution number
- Related requirement(s)
- Depth
- Test category
- Author
- Check boxes for whether the test is automatable and has been automated.

Large Scale Engineering Projects

- Need a systematic approach:
  1. Choose a high level test plan
  2. Write detailed test cases
  3. Assign the test cases to testers, who manually follow the steps and record the results.
  4. Author a test report, detailing the findings of the testers.

- The report is used by managers to determine whether the software can be released

Test Automation

- **A process of writing a computer program to do testing** that would otherwise need to be done manually
- **The use of software to control the execution of tests**, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other test control and test reporting functions
- Commonly, test automation involves automating a manual process already in place that uses a formalized testing process

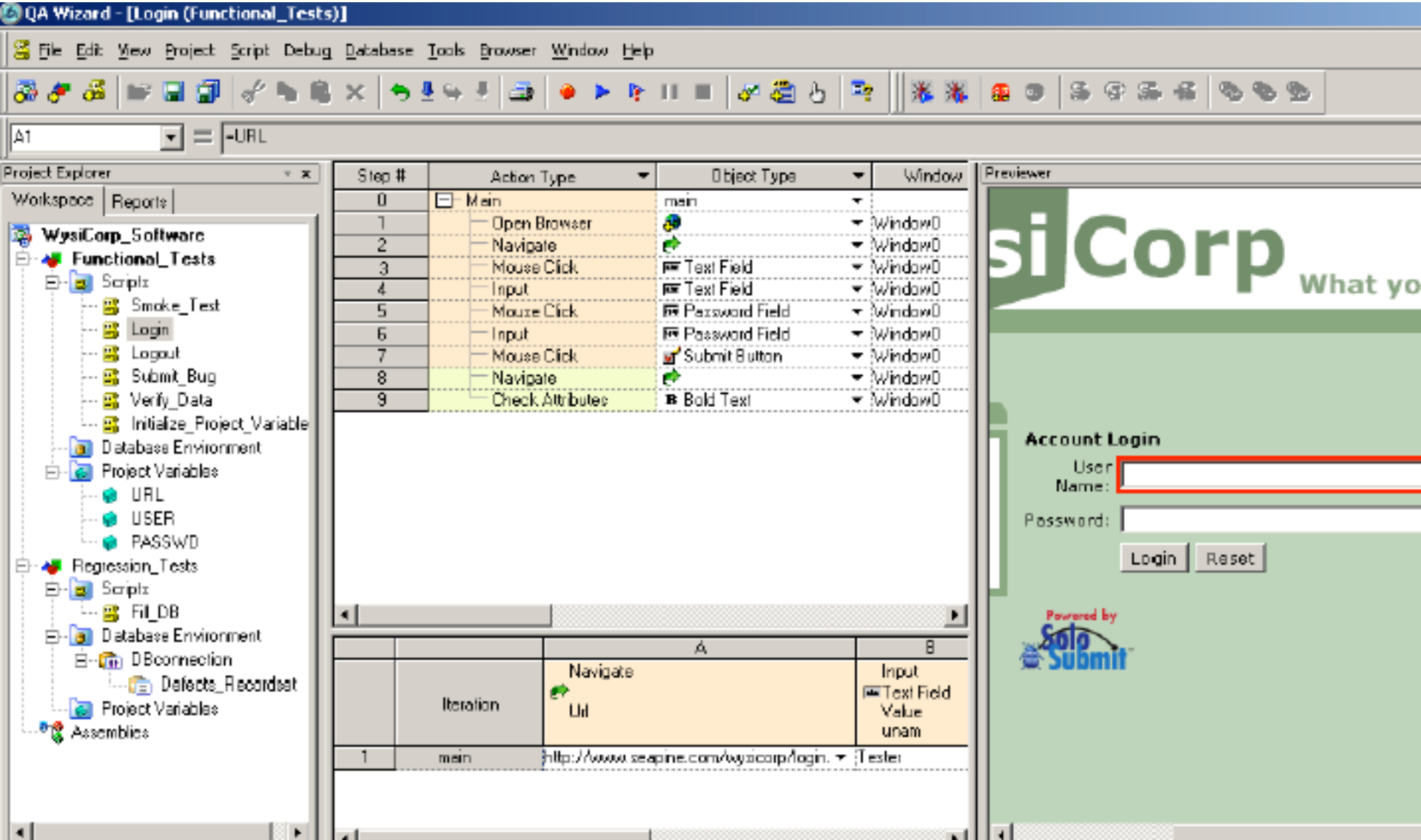
Test Automation – Approaches

- **Graphical user interface testing**. A testing framework generates user interface events such as keystrokes and mouse clicks, and **observes the changes** that result in the user interface, to validate that the observable behavior of the program is correct
- **Code-driven testing**. The public (usually) interface to classes, modules, or libraries are tested with a variety of input arguments to validate that the results that are returned are correct

TA – What to test

- Testing tools can help automate tasks such as *product installation, test data creation, GUI interaction, problem detection, defect logging*, etc.
- Important points when thinking about TA:
  - Platform and OS independence
  - Data driven capability (Input Data, Output Data, Meta Data)
  - Customizable Reporting (DB Access, crystal reports)
  - Email Notifications
  - Easy debugging and logging
  - Version control friendly
  - Extensible & Customizable
  - Support distributed execution environment
  - Distributed application support

Test Automation – Example



Test Automation with Scripting

- Manual Test Case Steps:
  - Launch Calculator
  - Press **2**
  - Press **+**
  - Press **3**
  - Press **=**
  - The screen should display **5**.
  - Close Calculator.

Example credit softwaretestinghelp.com

Test Automation with Scripting

```
//the example is written in MS Coded UI using c# language.
[TestMethod]
public void TestCalculator()
{

//launch the application
var app = ApplicationUnderTest.Launch("C:\\Windows\\System32\\calc.exe");

//do all the operations
Mouse.Click(button2);
Mouse.Click(buttonAdd);
Mouse.Click(button3);
Mouse.Click(buttonEqual);

//evaluate the results
Assert.AreEqual("5", txtResult.DisplayText,”Calculator is not showing 5);

//close the application
app.Close();
}
```

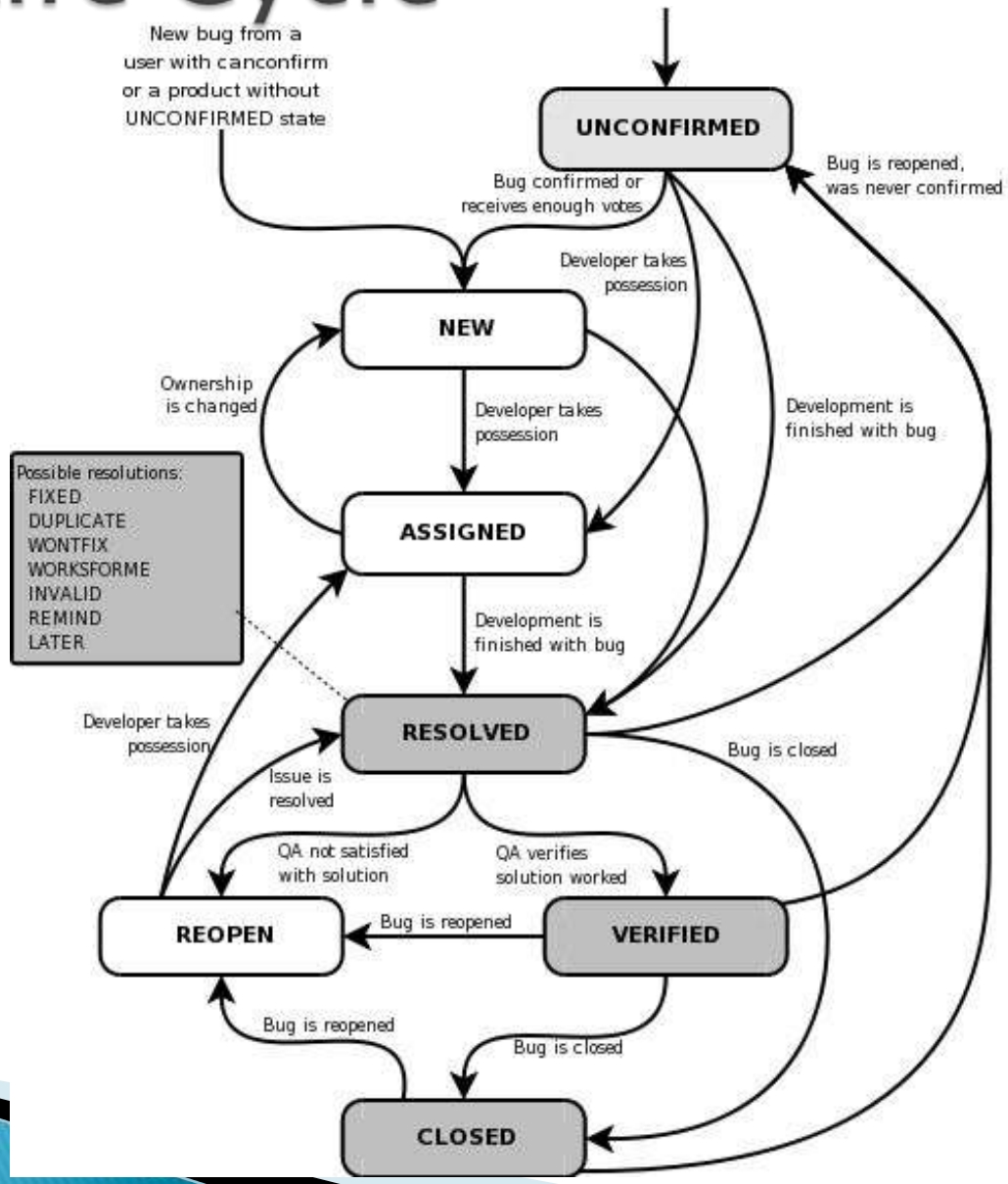
Software Bug

- **A software bug is an error, flaw, mistake, failure, or fault in a computer program**
- Most bugs arise from mistakes and errors made by people (in program or in its design), and a few are caused by compilers
- Reports detailing bugs in a program are commonly known as **bug reports**, **fault reports, problem reports, trouble reports, change requests**, and so forth.

Bugs Effects

- Bugs in the code controlling the Therac-25 radiation therapy machine were directly responsible for some patient deaths in the 1980s.
- Smart ship USS Yorktown was left dead in the water in 1997 for nearly 3 hours after a divide by zero error.
- Eve Online's deployment of the Trinity patch erased the boot.ini file from several thousand users' computers, rendering them unable to boot.
- Valve's Steam client for Linux could accidentally delete all the user's files in every directory on the computer.
- In 2002 were responsible for losses of $59 billion annually, or about 0.6 percent of the gross domestic product.

Bug Life Cycle



Bug Prevention

- Programming style
- Programming techniques
- Development methodologies
- Programming language support
- Code analysis
- Instrumentation

Coding Style – Motivation

- Coding conventions are important because:
  - 80% of the work on software is maintenance
  - Usually, a product is not maintained by the persons who created it
  - Code conventions improve code readability and allow a software developer to quickly understand new programs

Coding Style – General Reuquirements

- Use comments: what do functions do, what do variables represent, explain steps of algorithms, etc.
- Use suggestive names for variables and functions
- Develop modular code
- Use dual pairs: set/get, start/stop, add/remove, save/load

Correctness

- Correctness of an algorithm is asserted when it is said that **the algorithm is correct with respect to a specification**
- **Functional correctness** refers to the input-output behavior of the algorithm (i.e., for each input it produces the correct output)
- A distinction is made between **total correctness**, which additionally requires that the algorithm terminates, and **partial correctness**, which simply requires that *if* an answer is returned it will be correct.

Links

- Software Quality Assurance: http://satc.gsfc.nasa.gov/assure/agbsec3.txt
- Software Testing: http://en.wikipedia.org/wiki/Software_testing
- GUI Software Testing: http://en.wikipedia.org/wiki/GUI_software_testing
- Regression Testing: http://en.wikipedia.org/wiki/Regression_testing
- Junit Test Example: http://www.cs.unc.edu/~weiss/COMP401/s08-27-JUnitTestExample.doc
- https://www.test-institute.org/What_is_Software_Quality_Assurance.php

Coding Style – Links

- C++:
  - http://www.chris-lott.org/resources/cstyle/
  - http://geosoft.no/development/cppstyle.html
- Java:
  - http://java.sun.com/docs/codeconv/
  - http://geosoft.no/development/javastyle.html