

# Regression Testing



**BY MATTHEW MOLONEY, MARK GARSIDE AND  
JASON LONGDON**

# Regression Testing Definition



*“Selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements” [IEEE 610.12-90.]*

1 *IEEE Standard Glossary of Software Engineering Terminology - 1990*

# Why Conduct Regression Testing?



Software Development is continually modifying and extending code which can lead to:

- Hidden Bugs Discovered By New Tests
- Old Bugs Re-Introduced
- Revised Code Causes New Bugs
- Integration of Individual Modules Impacts Existing Functions

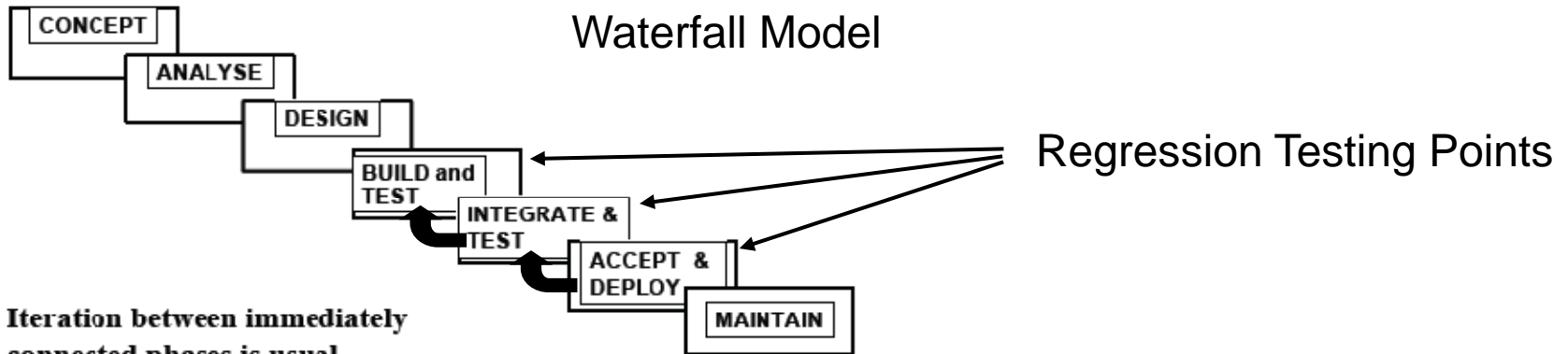
# Where in the Lifecycle?



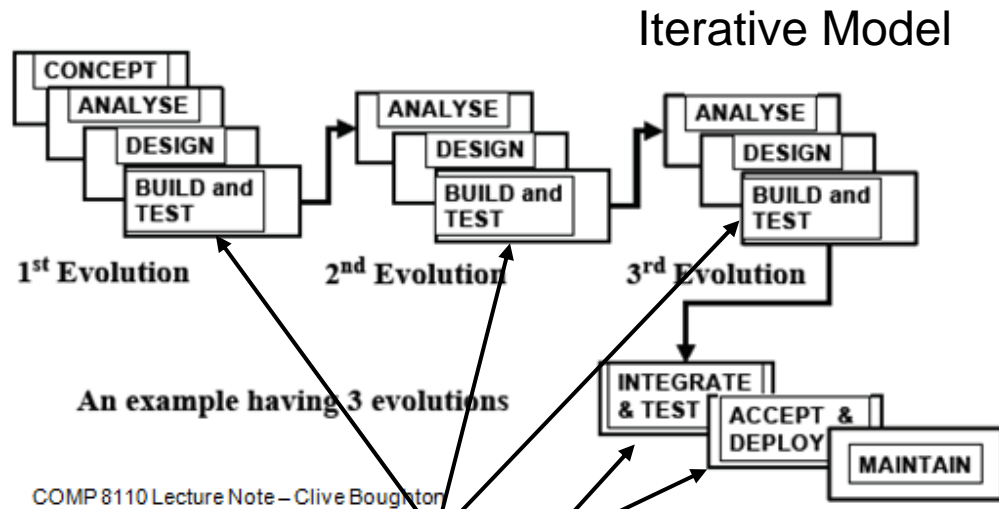
## Regression Testing Is An Activity Within The Following Lifecycle Processes and Activities:

- **Development and Maintenance Processes**
  - As defined by IEEE 12207
- **Unit Testing Activity**
  - Amendments to be validated post initial release
- **Integration Testing Activity**
  - Demonstrate component functionality within a larger module
- **System Testing Activity**
  - Demonstrate that the corners of the system still function
- **Acceptance Testing Activity**
  - Amendments have not degraded previously accepted elements

# Development Lifecycle Types

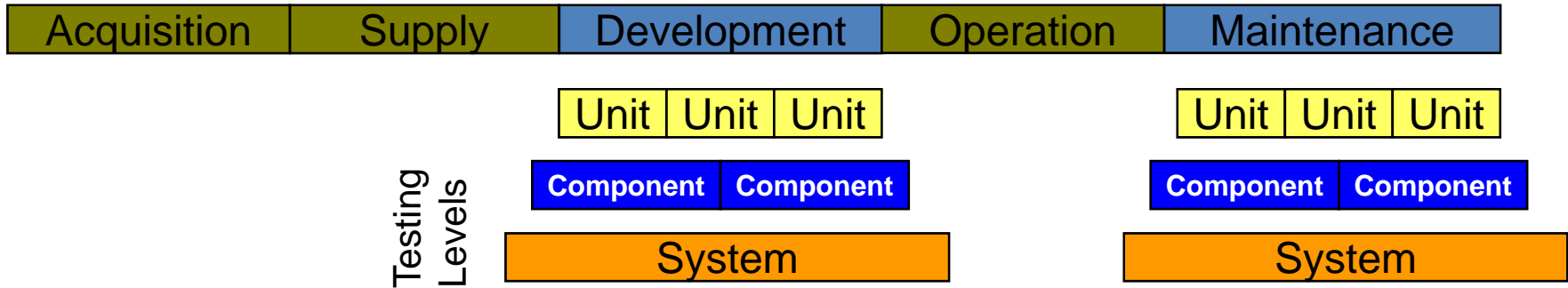


COMP 8110 Lecture Note – Clive Boughton



COMP 8110 Lecture Note – Clive Boughton

# Regression Testing and Lifecycle Relationships



Unit | Unit | Unit

*Design, Code, Test*

- Fault – Fix, Test Fix, Re-Test Complete Unit
- Re-Run All Tests

Component | Component

*Integrate, Test*

- Fault – Fix, Test Fix, Re-Test Selective Units/Related Components
- Re-run Failed Integration Test and Selective Additional Integration Tests

System

*Integrate, Test*

- Fault – Fix Code, Test Fix, Re-Test Selective Units/Related Components
- Re-run Failed System Test and Selective Additional System Tests

# Regression Test Selection



## Retest all

- selects all

## Random / Ad-Hoc

- selects a random sample / selects tests based on testers knowledge and experience

## Minimization

- selects the minimum set required to cover all modified or affected parts of the program

## Safe

- selects all tests which execute at least one modified statement

# Cost and Benefit Model



## Costs

- Management overhead in database maintenance and tool development
- Cost of selecting tests, executing tests, and analysis of results
- Cost of undetected faults

## Savings

- Costs saved by not running superfluous test cases

Only beneficial if  $\text{Cost} < \text{Savings}$

# Which parts are affected?



## Methods of detection

- **Using Control Flow Graphs**
- **UML diagrams**
- **Activity Diagrams**
- **Traceability matrixes**
- **Execution Traces**

# At what level of granularity?



- **Finer granularity may result in fewer chosen tests to rerun but will be more expensive to compute and maintain**
- **i.e. an algorithm that attempts to run through all components of the program to obtain a complete execution trace would be more expensive to run than all of the tests.**

# Further refinement of tests



- Once a set of tests have been identified as impacted by the change, it may still be too large to economically execute all of these selected tests.
- **Simple Risk Model**
  - Risk Exposure = Probability of fault \* Cost of fault
  - Rank tests according to risk exposure, execute the highest tests until an acceptable risk exposure is reached
  - Often important features are included in the test selection just in case

# Using Metrics



- Bug metrics charts are used in assessing system stability and readiness for release. Importantly a stable system should see a levelling off of discovered Bugs. A spike in a number of bugs found after a change can suggest either the change caused new bugs or uncovered existing bugs. It is important to know which because new bugs implies poor changes, old bugs implies system instability.

# Test Automation



- **What is Test Automation?**
  - The process of using software to control the administering of tests, comparing output, and reporting results.
  - It is an addition to manual testing
- **An example: HP WinRunner**
- **Upcoming software**
  - IBM Jazz
  - Microsoft Camano

# Test Automation



- The **Good** Points About Test Automation:
  - Confidence
  - Eliminate risk of manual errors
  - Easily & quickly retest after small changes
  - Tests can be performed out of office hours
  - In expensive in the long run

# Test Automation



- The **Bad** Points About Test Automation:
  - Expensive to setup (staff training, etc)
  - LAWST stats:
    - ✦ Regression testing finds around 15% of bugs
    - ✦ Cost is around 80% of testing budget
  - Can still miss errors as the tests may be miss directed

# Changing Regression Tests



- The definition of Regression testing shows us that its open to change
- Testers add tests as more functionality is added to the project
- Tests should be made in relation to Traceability Matrix
- Tests should be made to cover newly found errors