Verification in the Lifecycle

EEE492A 2008
References:[HvV §13.2, 13.9]

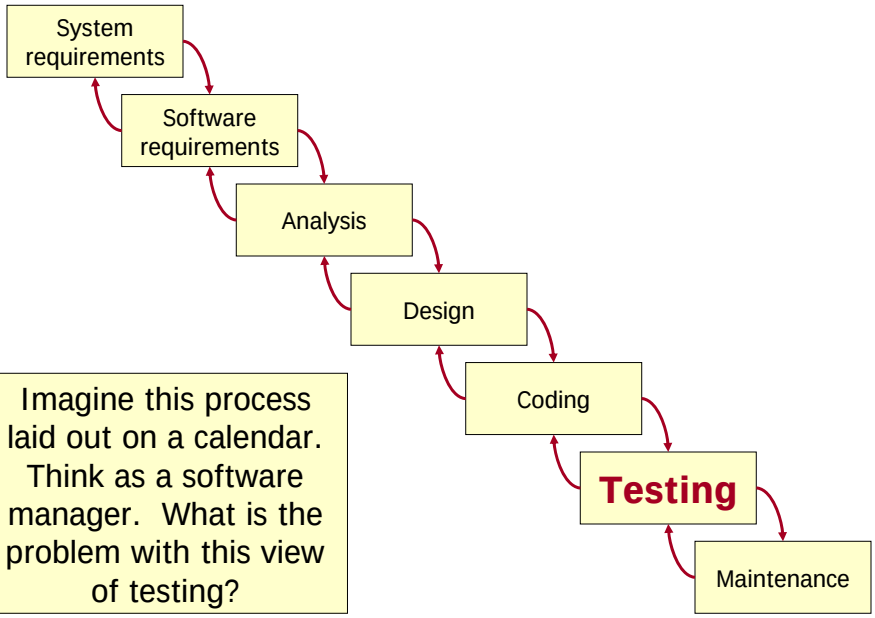
Sylvain P. Leblanc
Royal Military College of Canada
Electrical and Computer Engineering
sylvain.leblanc@rmc.ca
tarpit.rmc.ca/leblanc

Outline

- Verification in each phase of development
  - Requirements
  - Design
  - Implementation
  - Maintenance
- Test Stages
  - User Tests
  - Unit Tests
  - Integration Tests
  - Stress Tests
  - Acceptance Tests
  - System Tests

2

Testing in the software lifecycle



Imagine this process laid out on a calendar. Think as a software manager. What is the problem with this view of testing?

Requirements & Verification

- Activities
  - design a test strategy - a plan
  - determine test requirements (resources, tools, ...)
  - prepare functional test cases
- Verify requirements against
  - **completeness** - self-explanatory but difficult to verify, user scenarios may help to identify omissions
  - **consistency** - essentially a check that no requirements contradict each other or any external interface
  - **feasibility** - risk analysis to determine cost effectiveness against key factors (safety, speed, reliability,...)
  - **testability** - requirements must be specific, unambiguous and quantitative to be able to be tested

4

Design & Verification

- Activities
  - check consistency between requirements and design
  - prepare more detailed structural and functional tests
  - verify the architecture and design
- Verify the architecture against
  - change - maintainability and flexibility are measures of how easily the design foundation may be changed
- Verify design against
  - completeness, consistency, feasibility and testability
    - similar to requirements verification
  - modern tools may support verification through executable designs

5

Implementation & Verification

- Activities
  - check consistency between design and implementation
  - generate structural and functional test data
  - verify implementation; execute tests
- Verify implementation against
  - the design (and ultimately the requirements)
- Verification may be
  - static - code inspections and walkthroughs or
  - dynamic - executable tests
- Test tools and automation are critical
  - test generators, stubs, drivers, comparators (JUnit)

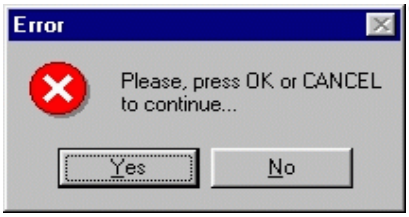
6

Maintenance & Verification

- Activities
  - maintain the development tests and tools
  - regression testing
- Verify changes against
  - new or changed requirements
  - a previously working system
- Well designed regression testing is necessary so as to avoid a “retest-all approach”
- In reality maintenance of code begins during development on any project
  - therefore be ready to support maintenance verification

7

User Testing



- early testing of prototype systems
- designed to
  - validate User Interfaces
  - elicit or improve understanding of requirements
- must occur **early enough** in the process that the results can be incorporated into the product

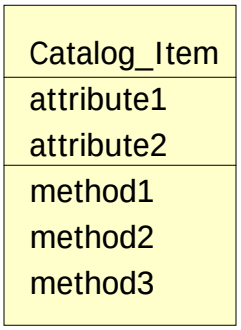
  or

  often enough to properly steer iterative developments (XP)

8

Unit Testing

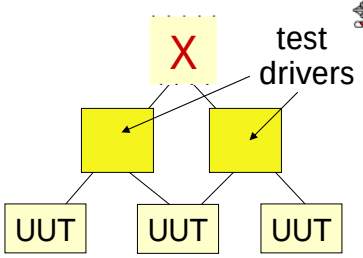
- testing of an individual module, class, or unit - based on the module specification
- normally performed by the developer responsible
- tests should include
  - data flow across module interface
  - local data structures and access to global data structures
  - selective coverage of execution paths
  - error handling code
  - boundary tests
    - first and last elements, first and last iterations, etc.



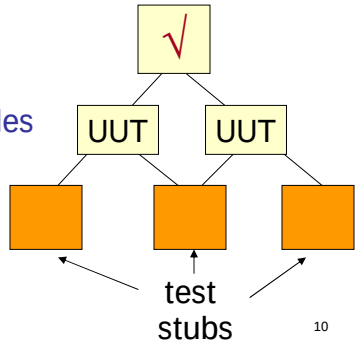
9

Integration Testing

- bottom-up
  - test the “bottom” modules using drivers to represent higher modules
  - collect modules from the bottom up into “clusters”



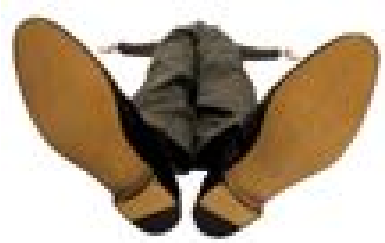
- top-down
  - test the “top” module using stubs to represent lower modules
  - as lower modules become available, stubs are removed
  - lower modules may be added depth-first or breadth-first



10

Stress Testing

- performance-under-load tests
  - 100 hits per second
  - 50 simultaneous database accesses
  - release all real-time tasks at once to observe scheduleability



- systems with load requirements must be tested under load
  - simulated load scenarios must be designed and supported
  - frequently requires significant test code and equipment to adequately support

11

Stress Testing

- performance-under-load tests
  - 100 hits per second
  - 50 database accesses simultaneously
  - release all real-time tasks at once to observe scheduleability

- systems with load requirements must be tested under load
  - simulated load scenarios must be designed and supported
  - frequently requires significant test code and equipment to adequately support

Alternate Slide 1

12

Stress Testing

- performance-under-load tests
  - 100 hits per second
  - 50 database accesses simultaneously
  - release all real-time tasks at once to observe scheduleability

- systems with load requirements must be tested under load
  - simulated load scenarios must be designed and supported
  - frequently requires significant test code and equipment to adequately support

Alternate Slide 2

13

Acceptance Testing

- complete when “the software performs in a manner that can reasonably be expected by the customer”
  - who gets to decide what’s “reasonable”?
- acceptance tests demonstrate conformity with requirements
  - failures are recorded on deficiency lists and must ultimately be addressed
- where requirements are unclear or the customer base is broad, may use alpha and beta testing
  - customers may be willing to use and report faults in exchange for early access to capabilities

14

Test Stages - System Testing

- software is only one element of the system.
  - Also includes, people, data, procedures, documentation and hardware

- ultimately, the entire system must be tested for effectiveness

15

Supplemental References

Roger S. Pressman. ***Software Engineering - A Practitioner's Approach 5th Edition, Chapter 18***. McGraw-Hill, 2001. ISBN 0-07-365578-3

16

Next Class:

Inspections and Walkthroughs