

# Validation, Verification and Testing of Computer Software

W. Richards Adrion, Martha A. Branstad,  
John C. Cherniavsky

CS780 – Adv. Software Engineering

1

# Outline

- Introduction
- Verification Through The Life Cycle
- Validation Verification And Testing Techniques
- Conclusions And Research
- Questions

2

# Introduction

- Validation
  - Determination of correctness of final program
- Verification
  - Consistency, completeness and correctness at each stage and between each stage of the development life cycle.

3

# Introduction

- Testable software has 2 characteristics
  - Understandable
  - Measurable
- Attributes of the software are reliability, testability, usability, efficiency, transportability, and maintainability, but in practice efficiency often conflicts with the other attributes.

4

# Verification Through The Life Cycle

- The Requirements Definition Stage
- The Design Stage
- The Construction Stage
- The Operation and Management Stage

5

# Requirements Stage

- Determine verification approach
  - Tools available to aid the developer in requirements definition
    - Examples: Program Statement Language (PSL) and Software Requirements Engineering Program (SREP)
- Determine adequacy of requirements
  - Critical System
- Generate functional test data
  - Test method and test evaluation criteria
  - The developed product is the right one

6

## Design Stage

- Determine consistency of design with requirements
  - Correlates to the requirements of the system
- Determine adequacy of design
  - Different Design Techniques
  - Automated Design System

7

## Design Stage

- Generate structural and functional test data
  - Analyze and examine for errors
  - Design walk-through
  - Documentation
    - The most important aspect of the design stage.

8

## Construction Stage

- Determine consistency with design
  - Approach
- Determine adequacy of implementation
  - Depending on the need of the system
- Generate structural and functional test data
- Apply test data
  - Final testing done

9

## Operation and Management Stage

- Modification
  - New requirements
- Correction
  - Detection of bugs
- Regression Testing
  - Testing of previously verified programs

10

## Validation Verification And Testing Techniques

- Testing Fundamentals
- General Techniques
- Test Data Generation
- Functional Testing Techniques
- Structural Testing Techniques
- Test Data Analysis
- Static and Dynamic Analysis Techniques
- Combined Methods

11

## Testing Fundamentals

- Objects we test arise at software life cycle
- Oracle
- Exhaustive Testing
- Test Data Set

12

## General Techniques

- Traditional Manual Methods
  - Desk checking
  - Peer Review
- Walk-Through, Inspection and Review
  - Evolution of desk checking
  - Require a team
  - Should be performed at all stages

13

## General Techniques

- Proof of Correctness
  - Validating the consistency of an output
    - Input Assertion
    - Output Assertion
  - Approaches
    - Formal
    - Informal

14

## General Techniques

- Simulation
  - Very useful when “real-world” interface is critical
  - Several models both formal and computational needs to be constructed
  - Very important in the construction stage

15

## Test Data Generation

- Developing of test data sets involve two Aspects
  - The selecting of data Input
  - Determining the expected response
    - Very difficult though hand calculation and simulation can be used.
    - Solution could be executable specification languages.

16

## Test Data Generation

- Executable Specification languages
  - It is the actual implementation of the design, but a higher level than the final code.
  - Parnas refers it as an “abstract program”, representing in less detail the final implementation
  - Can be used as Oracles

17

## Test Data Generation

- Test Data derived in 2 ways
  - Black box
    - Is the functional analysis where test data are derived from the external specification of software behavior
  - White box
    - Is the structural analysis

18

## Functional Testing Techniques

- Exhaustive Testing
- Boundary Value Analysis
- Design-Based Functional Testing
- Cause Effect Graphing

19

## Functional Testing Techniques

- Exhaustive Testing
  - Infeasible
- Boundary Value Analysis
  - Partition the program domain into input classes
  - Partitioning is a problem since there is no easily stated procedure for forming the partition.
  - Array  $a[10, 10]$  should be tested as a single element array  $a[1, 1]$  and a full  $a[10, 10]$  array.

20

## Functional Testing Techniques

- Design-Based Functional Testing
    - Requirements functions
      - Overall functional capabilities of the program
    - Design function
      - Smaller functions to design the program
- Example: In a tree structure requirement function is considered as the root and the design functions as children.

21

## Functional Testing Techniques

- Cause-effect Graphing
  - Is a technique for developing test cases for programs from high-level specifications.
  - Output domain partitioned into various classes called “effects”
  - Create a limited entry decision table
    - Choose test cases to exercise each column of the table

22

## Structural Testing Techniques

- A testing method where the test data are derived solely from the program structure.
- Two Types
  - Coverage-Based Testing
  - Complexity-Based Testing

23

## Structural Testing Techniques

- Coverage-Based Testing
  - Based on number of statements, branches, or paths in the program that are exercised by test data
  - Any program can be represented by a graph
  - Node represent statements or collection of sequential statements

24

## Structural Testing Techniques

- Coverage-Based Testing

To test the program completely, should cause the execution of all paths.

Since complete path coverage is infeasible, the measure of the software quality is done by evaluating the proximity of ideal coverage.

25

## Complexity-Based Testing

- Test are derived in proportional to the software complexity
- There are a number of complexity based metrics but they are unsuited or have not been applied to problem stating.
- Depends on the graph

26

## Test Data Analysis

- To determine the “goodness” of the test data set.
- Two Types
  - Statistical Analysis and Error Seeding
  - Mutation Analysis

27

## Test Data Analysis

- Statistical Analysis and Error Seeding
  - Most common type of analysis is statistical
  - Mills developed this technique to “seed” known errors into the code so that their placement is statistically similar to that of actual errors.
  - He later developed confidence levels for his techniques

28

## Test Data Analysis

- estimate  $E = IS/K$   
where S is the number of seeded errors, K is the number of discovered seeded errors, and I is the number of discovered unseeded errors

29

## Test Data Analysis

- Mutation Analysis
  - The program that is to be tested is seeded with errors.
  - Several mutations of original program are generated, each with different sets of errors
  - Mutants and programs are ran interpretively on the test set.

30

## Test Data Analysis

- Assumptions
  - “Competent Programmer assumption”, stating that an incorrect program will not differ much from the desired program
  - “Coupling Effect”, is the conjecture that tests that uncover simple errors will also uncover deeper and more complex errors.

31

## Test Data Analysis

- Mutation Score  $ms(P, T) = \frac{|DM(P, T)|}{|M(P) - E(P)|}$  where
  - P is a program, T is the test set
  - M(P) is the finite set of mutant programs
  - E(P) is the set of functionally equivalent programs to P
  - DM(P, T) is the set of programs in M(P) differentiated from P by the test set T
- If score approaches 1 the test case is adequate

32

## Static and Dynamic Analysis Techniques

- Static Analysis
  - Does not involve the execution of actual code.
  - Many of the general techniques discussed above, such as formal proof
  - Performed by parsers and associated translators residing in the memory
- Dynamic Analysis Technique
  - Involves the execution of the actual code

33

## Symbolic Execution

- Symbolically defining data that forces program paths to be executed
- During assignment replace the value on the left hand side by the unevaluated expression on the right hand side.

34

## Example of Symbolic Execution

- IN a, b;
- a := a \* a;
- x := a + b;
- IF x=0 THEN x:=0
- ELSE x := 1;
- The symbolic execution of the program will result in the following expression:
- If a \* a + b = 0 then x :- 0
- else if a\*a+b <> 0 then x:=1

35

## Symbolic Execution

- The large, complex expression can be composed and viewed as a tree structure
- Each leaf represents a path through the program
- Branch points in the tree represent the decision point in the tree
- Every program path is represented in the tree and every branch path is taken.

36

## Combination Methods

- Several Techniques have been combined
  - Standard Test with formal verification
  - Symbolic Execution with formal verification
  - Mutation Analysis and formal proof techniques
  - Bounded Analysis and Standard Test

37

## Conclusion

- Most Successful has been the manual techniques such as walk-through, reviews and inspection
- Because the uniform applicability to requirements, design and coding phases
- Good management and careful planning are key to the success of manual techniques

38

## Conclusion

- Many of the other techniques has not been used
- High Cost in use
- Flaw in theory
- Hard to use and adapt

39

## Conclusion

- Combination of two methods might lead to extra cost and errors beyond the necessity to acquire them
- Choosing of tools according to the requirements
- Different tools at different stages
- More research on basic foundation of verification

40

## Materials not discussed in the paper

- Model Checking
  - Create a finite state model of the software
- Automated Testing
  - Formal specification based testing for a given bound

41

## Questions

- Complexity-Based Testing: How will you know the complexity of the system?
- How can you do the placement of seeded errors statistically similar to the actual errors?
- Do you think efficiency always conflicts with other attributes?

42

# Thank You

- It is too much testing!!!



- Manual Techniques rock.

