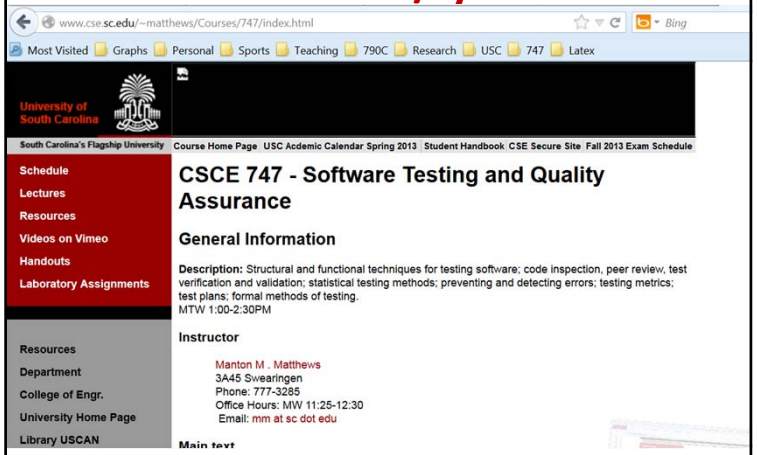
CSCE 747 Software Testing and Quality Assurance

Lecture 01 – Overview

Class website/syllabus



References on Slides

- Most of our slides a least for a while will be generated from our Text
- Jorgensen, Paul C. (2011-07-16). Software Testing Auerbach Publications. Kindle Edition.
- The abbreviated reference at the bottom of the slides will be
  - "Jorgensen, Paul C. Software Testing, Auerbach Publications" and sometimes maybe even a shortened version of this
  - "Software Testing-Jorgensen 2011"

Why test?

Showing they are correct?
Proving . . . . .

What Testing Does and Does Not Do

- "Program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence."

CSP

1957 - Compiler IBM John Backus

More Dijkstra's Quotes

- The use of COBOL cripples the mind; its teaching should, therefore, be regarded as a criminal offense.
- APL is a mistake, carried through to perfection. It is the language of the future for the programming techniques of the past: it creates a new generation of coding bums.
- FORTRAN, 'the infantile disorder', by now nearly 20 years old, is hopelessly inadequate for whatever computer application you have in mind today: it is now too clumsy, too risky, and too expensive to use.
- In the good old days physicists repeated each other's experiments, just to be sure. Today they stick to FORTRAN, so that they can share each other's programs, bugs included.
- It is practically impossible to teach good programming to students that have had a prior exposure to BASIC: as potential programmers they are mentally mutilated beyond hope of regeneration.
- Besides a mathematical inclination, an exceptionally good mastery of one's native tongue is the most vital asset of a competent programmer.
- Simplicity is prerequisite for reliability.
- Programming is one of the most difficult branches of applied mathematics; the poorer mathematicians had better remain pure mathematicians.
- We can found no scientific discipline, nor a hearty profession, on the technical mistakes of the Department of Defense and, mainly, one computer manufacturer.
- About the use of language: it is impossible to sharpen a pencil with a blunt axe. It is equally vain to try to do it with ten blunt axes instead.

Basic Definitions

- Error or mistake "person makes error"
- Fault or Defect fault appears in prog.
- Failure running software can have a failure
- Incident
- Test
- Test case

Preconditions
input
output
postcondition

System Under Test
SUT

Test Cases

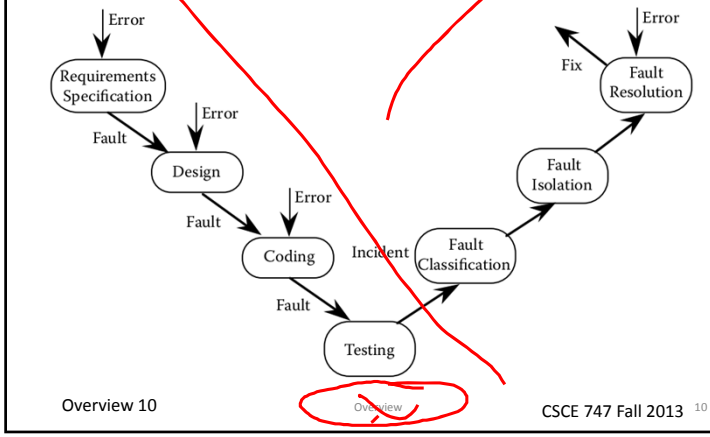
- Inputs to test cases:
  - preconditions
  - actual inputs
- Output
- Oracle

- Jane Austen (2010). Pride & Prejudice (Page 5). Amazon Digital Services, Inc.. Kindle Edition.

Act of Testing

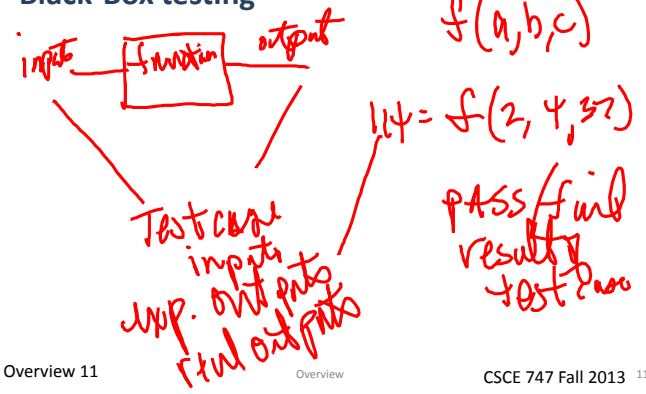
- Act of Testing
  - establishing preconditions
  - specifying inputs
  - observing outputs
  - comparing outputs with those expected
  - ensuring postconditions

Testing Life Cycle



Functional Testing

- Black-Box testing



Structural Testing

- White-box

Error and Fault Taxonomies

Faults classified by severity – B. Bezier

1. Mild	Misspelled word
2. Moderate	Misleading or redundant information
3. Annoying	Truncated names, bill for $0.00
4. Disturbing	Some transaction(s) not processed
5. Serious	Lose a transaction
6. Very serious	Incorrect transaction execution
7. Extreme	Frequent "very serious" errors
8. Intolerable	Database corruption
9. Catastrophic	System shutdown
10. Infectious	Shutdown that spreads to others

Table 1.1 Input/Output Faults

Type	*Instances*
Input	Correct input not accepted
	Incorrect input accepted
	Description wrong or missing
	Parameters wrong or missing
Output	Wrong format
	Wrong result
	Correct result at wrong time (too early, too late)
	Incomplete or missing result
	Spurious result
	Spelling/grammar
	Cosmetic

Table 1.2 Logic Faults

Missing case(s)
Duplicate case(s)
Extreme condition neglected
Misinterpretation
Missing condition
Extraneous condition(s)
Test of wrong variable
Incorrect loop iteration
Wrong operator (e.g., < instead of ≤)

Levels of Testing

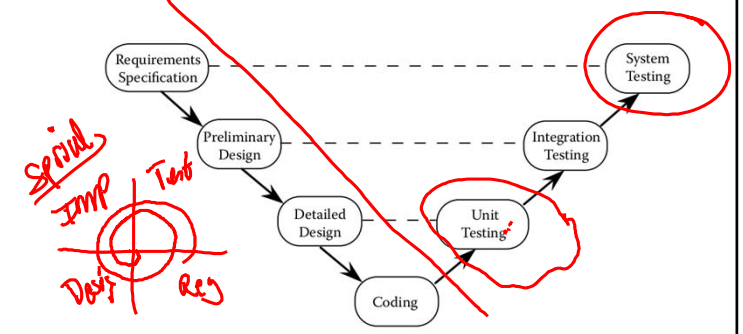


Figure 1.10 Levels of abstraction and testing in the waterfall model.

References

Beizer, B., *Software System Testing and Quality Assurance*, Van Nostrand Reinhold, New York, 1984.
IEEE Computer Society, *IEEE Standard Glossary of Software Engineering Terminology*, 1983, ANSI/IEEE Std. 729-1983.
IEEE Computer Society, *IEEE Standard Classification for Software Anomalies*, 1993, IEEE Std. 1044-1993.
Miller, E.F., Jr., Automated software testing: a technical perspective, *American Programmer*, Vol. 4, No. 4, April 1991, pp. 38–43.
Pirsig, R.M., *Zen and the Art of Motorcycle Maintenance*, Bantam Books, New York, 1973.
Poston, R.M., *T: Automated Software Testing Workshop*, Programming Environments, Inc., Tinton Falls, NJ, 1990.
Poston, R.M., A complete toolkit for the software tester, *American Programmer*, Vol. 4, No. 4, April 1991, pp. 28–37. Reprinted in CrossTalk, a USAF publication.

Java Testing Tools

- Junit
- Eclipse
- Maven
- Mockito
- Hamcrest
- Jenkins

Running Examples

1. The triangle problem - a venerable example in testing circles;
2. NextDate - a logically complex function, and
3. The commission problem - an example that typifies Management Information Systems (MIS) applications

More Examples

- The simple ATM (SATM) system;
- The currency converter, an event-driven application typical of graphical user interface (GUI) applications; and
- The windshield wiper control device from the Saturn™ automobile.
- o-oCalendar, an object-oriented version of NextDate

Triangle Problem

- Most widely used example in software testing literature.
- Problem Statement
- Simple version: The triangle program accepts three integers, a, b, and c, as input. These are taken to be sides of a triangle.
- The output of the program is the type of triangle determined by the three sides: Equilateral, Isosceles, Scalene, or Not A Triangle.

Triangle Improved

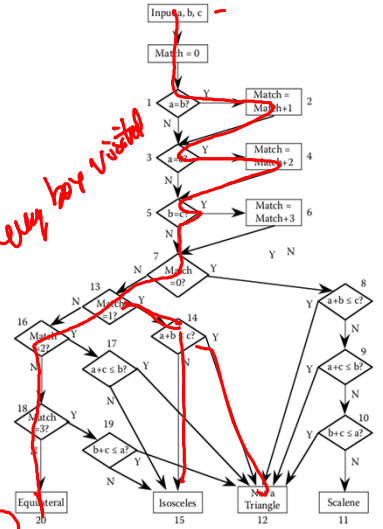
- Improved version: The triangle program accepts three integers, a, b, and c, as input. These are taken to be sides of a triangle. The integers a, b, and c must satisfy the following conditions:

c1. $1 \le a \le 200$ c4. $a < b + c$
c2. $1 \le b \le 200$ c5. $b < a + c$
c3. $1 \le c \le 200$ c6. $c < a + b$

- The output of the program is the type of triangle determined by the three sides: Equilateral, Isosceles, Scalene, or NotATriangle. If an input value fails any of conditions c1, c2, or c3, the program notes this with an output message, for example, "Value of b is not in the range of permitted values."
- If values of a, b, and c satisfy conditions c1, c2, and c3, one of four mutually exclusive outputs is given:

1. If all three sides are equal, the program output is Equilateral.
2. If exactly one pair of sides is equal, the program output is Isosceles.
3. If no pair of sides is equal, the program output is Scalene.
4. If any of conditions c4, c5, and c6 is not met, the program output is NotATriangle.

Flowchart for the traditional triangle program implementation



Dataflow Diagram

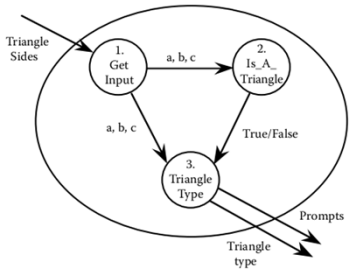


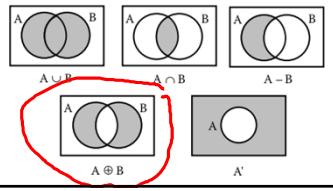
Figure 2.2 Dataflow diagram for a structured triangle program implementation.

Eclipse And Java For Total Beginners

- "Eclipse And Java For Total Beginners" video tutorial, which is available at http://eclipsetutorial.sourceforge.net/
- Install Java: Select "Eclipse IDE for Java Developers".
- Install Eclipse: www.eclipse.org/downloads
- Implement Triangle program (base problem)
- Dropbox report on progress/code by Midnight Tuesday 8/27/2013

Discrete Math for Testers

- Set Theory: ∅, Venn diagrams,
- Subsets: ⊂ ⊆ ⊄ element of ∈ ∉
- Set operations
  - Union, intersection, complement
  - Relative complement: A-B
  - Symmetric difference: A ⊕ B

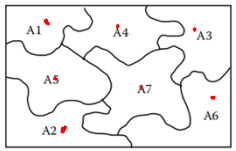


Relations

- Cartesian Product –
  - A × B = { <x,y>: x ∈ A and y ∈ B}
  - Ordered pairs usually (x,y) in this text <x,y>
  - | A × B | = |A| * |B|
- Relations
- Formally a relation R:A→B is just a subset of A × B
  - i.e., a set of ordered pairs first coordinate from A and second from B
  - function

Partitions

- A partition of a set A is a set of disjoint subsets $A_1, A_2, A_3, \ldots A_n$, such that
  - $A_1 \cup A_2 \cup A_3 \cup \ldots \cup A_n = A$
  - Note disjoint means if $i \neq j$ then $A_2 \cap A_3 = \varnothing$
- As in partition up test space



Set Identities

Name	Expression
Identity laws	$A \cup \varnothing = A$ $A \cap U = A$
Domination laws	$A \cup U = U$ $A \cap \varnothing = \varnothing$
Idempotent laws	$A \cup A = A$ $A \cap A = A$
Complementation laws	$(A')' = A$
Commutative laws	$A \cup B = B \cup A$ $A \cap B = B \cap A$
Associative laws	$A \cup (B \cup C) = (A \cup B) \cup C$ $A \cap (B \cap C) = (A \cap B) \cap C$
Distributive laws	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
DeMorgan's laws	$(A \cup B)' = A' \cap B'$ $(A \cap B)' = A' \cup B'$

Functions

- Definition
- Domain and Range f: D → R
- Types of functions
  - Onto
  - Into
  - One-to-one
  - Many-to-one
- Composition

Fig 3.4 Causal Flows in DF diagram

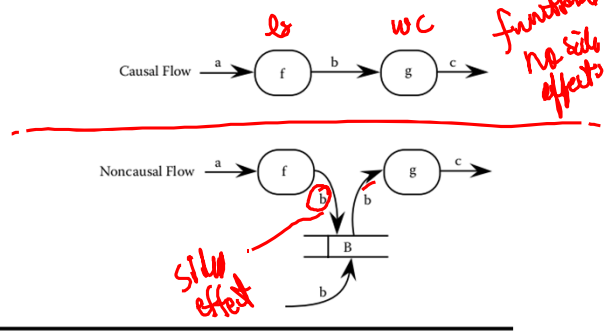
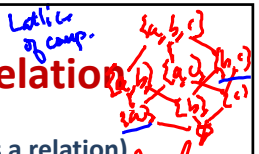


Figure 3.4 Causal and noncausal flows in a dataflow diagram.

Participation of a Relation



- Properties of functions (a function is a relation)
  - One-to-many, … many-to-many
- Definition Given two sets A and B, a relation R ⊆ A × B, the participation of relation R is:
  - Total iff every element of A is in some ordered pair in R
  - Partial iff some element of A is not in some ordered pair in R
  - Onto iff every element of B is in some ordered pair in R
  - Into iff some element of B is not in some ordered pair in R
- Jorgensen, Paul C. (2011-07-16). Software Testing (Page 43). Auerbach Publications. Kindle Edition.

Properties of Relations

- A relation R ⊆ A × A is:
  - Reflexive iff for all a ∈ A, <a, a> ∈ R
  - Symmetric iff <a, b> ∈ R ⇒ <b, a> ∈ R
  - Antisymmetric iff <a, b>, <b, a> ∈ R ⇒ a = b
  - Transitive iff <a, b>, <b, c> ∈ R ⇒ <a, c> ∈ R
- Ordering relation is reflexive, antisymmetric, and transitive
  - Partial order
  - Common in software: predecessor, successor, ancestor…
- Jorgensen, Paul C. (2011-07-16). Software Testing (Page 44). Auerbach Publications. Kindle Edition.

Equivalence Relation

- Definition
- A relation R ⊆ A × A is an equivalence relation if R is reflexive, symmetric, and transitive.
- Equivalence class
  - .
- Induces a partition
  - .

Propositional Logic

- Logical operators
  - three basic logical operators are
  - and (∧),
  - or (∨), and
  - not (¬);
- Propositional symbols
- Expressions, Truth Tables
- Well formed formulas (wffs)
- R ∧ A ∧ ¬(B ∨ C)

Truth Table Equivalence "Proofs"

- Definition
- Two propositions p and q are logically equivalent (denoted p ⇔ q) iff their truth tables are identical.

p	q	p → q	q → p	(p → q) ∧ (q → p)	¬((p → q) ∧ (q → p))
T	T	T	T	T	F
T	F	F	T	F	T
F	T	T	F	F	T
F	F	T	T	T	F

Jorgensen, Paul C. (2011-07-16). Software Testing Auerbach Publications. Kindle Edition.

Graph Theory for Testers

- Definition
- A graph G = (V, E)

Graph Definitions

- degree, in-degree, out-degree
- adjacency matrix
- incidence matrix
- paths
- connectedness
  - Connectedness is an equivalence relation
- component

condensation graph

- Definition Given a graph G = (V, E), its condensation graph is formed by replacing each component by a condensing node.

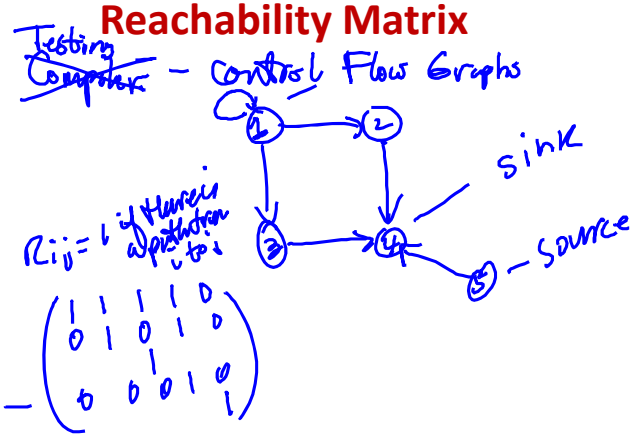
cyclomatic number

- The cyclomatic number of a graph G is given by V(G) = e – n + p,
  - where e is the number of edges in G,
  - n is the number of nodes in G, and
  - p is the number of components in G.

Directed Graphs

- sources, sinks, transfer node
- A (directed) path is a sequence of edges such that, for any adjacent pair of edges e in the sequence, the terminal node of the first edge is the initial node of the second edge.
- A cycle is a directed path that begins and ends at the same node.
- A (directed) semipath is a sequence of edges such that, for at least one adjacent pair of edges in the sequence, the initial node of the first edge is the initial node of the second edge, or the terminal node of the first edge is the terminal node of the second edge.

Reachability Matrix



Connectedness in Digraphs

Program Graphs

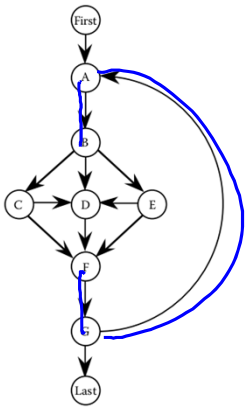


Figure 4.5 Digraphs of the structured programming constructs.

Finite State Models

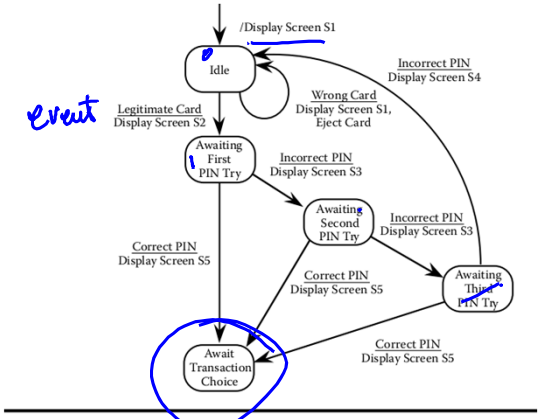


Figure 4.6 Finite state machine for PIN tries.

Petri Nets

- Petri nets are the accepted model for protocols and other applications involving concurrency and distributed processing.