CSE 4283 / 6283
Software Testing and QA

Dr. Tanmay Bhowmik

tbhowmik@cse.msstate.edu

Special thanks to Dr. Nan Niu & Dr. Byron Williams

Agenda

Last Topic:
Intro to Testing

This Topic:
Types of Testing

Next Topic:
Testing the Requirement
(Creativity and RBT)

Defect Resolution (overall)

- Actions agreed upon and impact / priority determined
- Defect logging & tracking
- Consistent defect interpretation & tracking
- Timely defect reporting (used to monitor and control projects)
- Learn from past problems (locations in the code, defect types, developer issues)
- Developer Issues - conceptual mistakes, unfamiliarity with domain, inexperience with methods

Ad-Hoc Testing

- Ad-hoc testing
  - "run-and-observe"
  - Implicit checklists may be involved
- Drawbacks
  - Lack of structure
  - Likely to miss
  - Likely to repeat oneself
  - In general, the whole process is hard to repeat
- One way to structure is to build a checklist

Systematic Testing: Checklists

- "Systematic" ➔ process is explicitly defined
  - Recall: how to achieve software quality (in Intro to SE)
- Testing with checklists
  - List of items that must be tested → Each item is "checked off" → When list is complete, testing is done
- Examples
  - Functional (black-box)
  - System elements (white-box)
  - Structures (implementation/white-box)
  - Properties (black-box or white-box)

Picture source: Internet

Functional Checklists: Exercise

- Function/feature (external) checklists
  - Black-box in nature
  - List of major functions that are expected

- An example high-level functional checklist for an ATM

- Card insertion & rejection
- Password management
- Envelope and printing
- Abnormal termination
- Installation and re-installation
- Backup and restore
- Commit and rollback
- Locking
- Logging and recovery
- Migration
- Stress
- ...

Implementation Checklists: Example

- Different forms of implementation checklists
  - White-box in nature but at varying levels of abstraction
    - E.g., lists of modules/components/etc. - Module interaction patterns
- Example: coding standard
  - Naming conventions: to improve software maintainability
    - Functional (black-box)
  - e.g., standard items (in concurrency control)
    - ACID (atomicity, consistency, isolation, durability)
    - Locking (e.g., read-lock, write-lock, two-phase)
    - Serialization (timestamp ordering, commit ordering, etc.)

Picture source: Internet

Partition Testing

Testing for Partition Coverage

- Sensitize test cases
  - i.e., defining specific input variables and associated values to exercise certain parts of the program in the white-box view or to perform certain functions in the black-box view
  - e.g., function *add(int a, int b)*
  - considering valid/invalid input values of *a* and *b*
    - How many cases are in an exhaustive test?

Test Case	Condition		Input	
	int a	*int b*	*a*	*b*
1	False	False	3.2	-0.4
2	False	True	"MSU"	2
3	True	False	7	3/4
4	True	True	-9	-2

Partitions: Formal Definitions

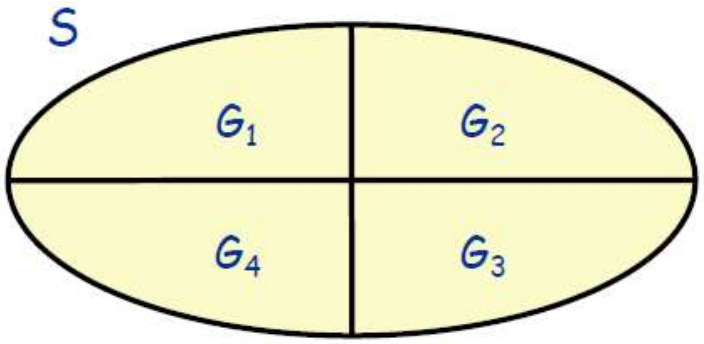
- A set S contains a list of unique elements
- A partition of S creates subsets G1, G2, … Gn such that
  - Sets are mutually exclusive $\forall i, j,\ i \neq j \Rightarrow G_i \cap G_j = \emptyset$
  - Sets are collectively exhaustive $\bigcup_{i=1}^{n} G_i = S$
- Each G1...Gn in a partition is called an **equivalence class**, where the specific relation that is used to define the subsets is:
  - Reflexive - holds on every member
  - Symmetric - holds if order is change
  - Transitive - holds in a relation chain

Better efficiency (eliminate "duplicates")

Better coverage (eliminate "holes")

Partitions

- What's a partition? Can you give an example?



$$\forall i, j,\ i \neq j \Rightarrow G_i \cap G_j = \emptyset$$

$$\bigcup_{i=1}^{n} G_i = S$$

x is a natural number AND x<=100 //x represents "grade"

If we're dealing with "A" students most of the time...

Are the following partitions?

[0,50) and [50,100]

[0,60), [60,70), [70,80), [80,90), [90,100]

Subset-even={0,2,4,…,100} and Subset-odd={1,3,5,…99}

Partitions-Based Testing

- Basic ideas
  - Members in equivalence class are treated "equivalent" → Defining meaningful partitions
  - Sampling from partitioned subsets for different types of partitions
  - Coverage of partitions: uniform
- Different types of partitions
  - Pure membership based partitions:
    - e.g., components in a subsystem, negative numbers, positive numbers
    - direct sampling, e.g., one component from each subsystem for coverage
  - Properties/relations used in definitions:
    - operations on numerical variables, e.g., x<=100
  - Combinations
    - e.g., **non-negative** integers less than 21

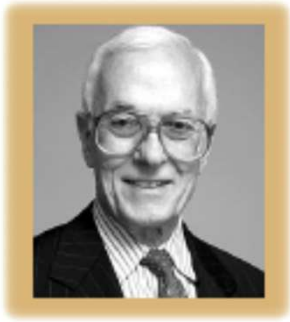
Reliability

- ***Reliability***: Probability of failure-free operation for a specific period or a given set of input under a specific environment
- Accomplished through
  - availability, latency, performance, efficiency, change management, monitoring, emergency response, and capacity planning
- **Key Measures - System Quality**
  - Mean-time-to-Failure (MTTF) - how often does the thing stop working
  - Mean-time-to-Repair (MTTR) - once it stops working, how long does it take to fix it

UBST (Usage-Based Statistical Testing)

- UBST ensures reliability
- Reliability: Customer view of quality
  - Probability: statistical modeling
  - Time / input / environment
- **OP: Operational Profile**
  - Quantitative characterization of the way a system will be used
  - Generate/execute test cases for UBST
  - Realistic reliability assessment - development decisions/priorities

OP (Operational Profile)



- John D. Musa
  - Giant in SRE (SW Reliability Eng.)
- OP
  - ***Definition***: a list of disjoint set of operations and their associated probabilities of occurrence
  - A quantitative way of characterizing the way a software system is or will be used
  - Operations: multiple possible test cases or multiple runs
    - Each operation corresponds to an individual sub-domain in domain partitions, thus representing a whole equivalence class.

Picture source: Internet

Comparing BBT with WBT

	BBT	WBT
Perspective	external behavior (functional)	internal implementation (structural)
Defect Focus	failures	faults
Scale	large software (as a whole)	small objects (looking inside)
Timeline	later (e.g., acceptance testing)	earlier (e.g., unit testing)
Tester	IV&V	developers themselves

Usage-Based (Statistical) Testing

- **Usage-based statistical testing (UBST)**
  - Actual usage and scenario/information
  - Captured in operational profiles (OPs)
  - Simulated in testing environment (problem?)
    - (too numerous ➔ random sampling)
  - Example: Canvas / beta-testing: add a course; delete a course; produce reports…
- **Applicability**
  - Final stages of testing
  - Particularly system/acceptance testing
  - Use with software reliability engineering
- **Termination criteria:** reliability goals

Coverage-Based Testing

- Coverage-based testing (CBT)
  - Systematic testing based on formal models and techniques
  - Testing models based on internal details or external expectations
  - Coverage measures defined for models - Testing measured by coverage goals
  - Example: Canvas / unit-testing: post announcement & set receiver role(s); set up submission deadline; …
- Applicability
  - All stages of testing - Particularly unit and component testing
  - Later phases at high abstraction levels
- Termination criteria: coverage goals

Comparing UBST with CBT

	UBST / BBT	CBT / WBT
Perspective	external behavior (functional)	internal implementation (structural)
Stopping Criteria	reliability goals	coverage goals
Scale	large software (as a whole)	small objects (looking inside)
Timeline	later (e.g., acceptance testing)	earlier (e.g., unit testing)
Tester	IV&V	developers themselves

Summary

- Types of Testing
  - Ad-hoc
  - Systematic
  - Partition-based
  - Usage-based
  - Coverage-based

- Next
  - Testability

THANK YOU