CISC 327
Software Quality Assurance

Lecture “review2”

Review for Mini-Exam #2

Likely topics on mini-exam #2

- Testing vs Debugging
- Verification vs Validation
  - Give you example scenario, and tell if it is verification/validation
- Systematic Testing (P2-1)
  - You can do anything to test
  - BUT is it a systematic testing method??
  - Required element/criterion to be considered `systematic`

Likely topics on mini-exam #2

- Level of Testing
  - Unit/Integration/System
- Proper use of Testing
  - Key elements: automation, repetition, evolution
- Test Adequacy
- Testing Techniques
  - Regression Testing (three configurations)
  - Failure Testing
  - *-ility Testing (capability, reliability etc.)
- Kind of Testing (Black vs. White)

Likely topics on mini-exam #2

- Given a text description of the problem, what are the possible testing techniques you can use?
  - A dev team is going to implement a driver for a water pump of a nuclear reactor. The driver is supposed to be well-protected and working properly all the time. List three testing techniques can be used:
    - Reliability Testing
    - Regression (configuration: retest all for reliability guarantee).
    - Failure Testing (make sure failure will not happen)
    - Security Testing (make sure adversarial cannot manipulate it)

Testing methods so far

- Functionality testing
- Input Overage:
  - Exhaustive Testing
  - Input Partition Testing
  - Shotgun Testing
  - Input Partition + Shotgun Testing
  - Boundary Testing
- Robustness Testing
  - Boundary Testing
  - Shotgun Testing
- Output Coverage
  - Exhaustive Testing
  - Output Partition
- Are they systematic approach? What is the stopping criteria (completeness criteria)
- Advantage/Disadvantage

Likely topics on mini-exam #2

- Functionality Testing
  - Partition the functional specification into different clauses and create test case for each of them
  - Example:
    - The class *Remover* has one instance variable (attribute), the character dial. Method *Remover.apply* will take one string and remove the letter that corresponds to the dial setting. Create functionality test cases for *Remover.apply.*

    - R1: will take one string
      - Input: a string e.g. ‘a string’ and a dial setting of ‘a’. Expected output: no exception/errors throwing out
    - R2: remove the letter that corresponds to the dial setting
      - Input: a string e.g. ‘a string’ and a dial setting of ‘a’. Expected output:
      - ‘ string’

Likely topics on mini-exam #2

- Exhaustive Input Coverage
  - Example: return the XOR result of two Boolean numbers
  - Can we do the same test for:
    - Return the XOR result of two 32bit unsigned integers ?
- Input Partition
  - Think about what is considered a partition and where we can come up the partition information (specifications!)
  - Example: the absolute value of an integer
    - Is 2 and 3 in the same partition? Why
    - Is 0 and 1 in the same partition? Why
  - For a partition, there are many choices of the test case, which would you pick (simplicity first)
- Same applied for output coverage exhaustive/partition testing
  - But more difficult for output coverage, why?

Likely topics on mini-exam #2

- Exhaustive Input Coverage
  - Example: return the XOR result of two Boolean numbers
  - Can we do the same test for:
    - Return the XOR result of two 32bit unsigned integers ?
- Input Partition
  - Think about what is considered a partition and where can can come up the partition information (specifications!)
  - Example: the absolute value of an integer
    - Is 2 and 3 in the same partition? Why
    - Is 0 and 1 in the same partition? Why
  - For a partition, there are many choice of the test case, which would you pick (simplicity first)
- Same applied for output exhaustive/partition testing
  - But more difficult, why?

Likely topics on mini-exam #2

- Input/Output Partitioning with Multiple variables/streams
  - Example: the program accepts a string and a file path as arguments. If the string is empty, return None. If the file does not exist, return -1. If the file exists but it is empty, return 0. If the file is not empty, return 1
  - Partitions:
    - String is empty; file does not exist
    - String is empty; file exists but empty
    - String is empty; file exists but not empty
    - String is not empty; file does not exist
    - String is not empty; file exists but empty
    - String is not empty; file exists but not empty

Likely topics on mini-exam #2

- Input/Output Partitioning with Multiple variables/streams
- What if I have too many variables as input/output?
  - Full combination is computationally infeasible
    - Test adequacy
  - Separate each variable as a partition and then do sub-partition for that variable. Using the last example:
    - String is empty;
    - String is not empty;
    - file does not exist;
    - file exists but empty
    - file exists but not empty
    - …
- For output coverage methods, what happen for a specific analyzed output value/partition if we cannot find an input?

Likely topics on mini-exam #2

- Input Boundary Testing
  - **As a Blackbox approach, where we can get those boundaries?**
  - How to come up test cases?
  - Example:
    - Return 1 if x is not larger than 1000 else -1
    - Test case: input 1000 and expected output is -1

Likely topics on mini-exam #2

- Black Box Testing can be applied on all levels of testing.

- Model-based approach:
  - Still we use as a black box approach here.
  - Better reliability but computationally expensive