SE 2AA4, CS 2ME3 (Introduction to Software Development)

Winter 2018

30 Introduction to Verification Continued (Ch. 6)

Dr. Spencer Smith

Faculty of Engineering, McMaster University

March 26, 2018

30 Introduction to Verification Continued (Ch. 6)

- Partially based on slides by Dr. Wassyng, Ghezzi et al
- Administrative details
- Approaches to verification
- Goals of testing
- Test plan
- Types of test - white box, versus black box, manual versus automated, etc.

Administrative Details

- A3
  - Part 2 - Code: due 11:59 pm Mar 26
- A4
  - Due April 9 at 11:59 pm
  - Might be tempted to write code first
    - Recommend starting with syntax of module(s), state variables
    - Maybe program a little before semantics part

Properties of Verification

- May not be binary (OK, not OK)
  - Severity of defect is important
  - Some defects may be tolerated
  - Our goal is typically acceptable reliability, not correctness
- May be subjective or objective - for instance, usability, generic level of maintainability or portability
  - How might we make usability objective?
- Even implicit qualities should be verified
  - Because requirements are often incomplete
  - For instance robustness, maintainability, performance
- What is better than implicitly specified qualities?

Approaches to Verification

- What are some approaches to verification?
- How can we categorize these approaches?

Approaches to Verification

- Experiment with behaviour of product
  - Sample behaviours via testing
  - Goal is to find "counter examples"
  - Dynamic technique
  - Examples: unit testing, integration testing, acceptance testing, white box testing, stress testing, etc.
- Analyze product to deduce its adequacy
  - Analytic study of properties
  - Static technique
  - Examples: Code walk-throughs, code inspections, correctness proof, etc.

Does our Engineering Analogy Fail?

- If a bridge can hold 512 kN, can it hold 499 kN?
- If our software works for the input 512, will it work for 499?

Verification in Engineering

- Example of bridge design
- One test assures infinite correct situations
- In software a small change in the input may result in significantly different behaviour
- There are also chaotic systems in nature, but products of engineering design are usually stable and well-behaved

Modified Version Works for 512, but not 499

```
procedure binary-search (key: in element;
                         table: in elementTable; found: out Boolean) is
begin
    bottom := table'first; top := table'last;
    while bottom < top loop
        if (bottom + top) rem 2 ≠ 0 then
            middle := (bottom + top - 1) / 2;
        else
            middle := (bottom + top) / 2;
        end if;
        if key ≤ table (middle) then
            top := middle;
        else
            bottom := middle + 1;
        end if;
    end loop;
    found := key = table (top);
end binary-search
```

if we omit this the routine works if the else is never hit! (i.e. if size of table is a power of 2)

Testing and Lack of "Continuity"

- Testing samples behaviours by examining "test cases"
- Impossible to extrapolate behaviour of software from a finite set of test cases
- No continuity of behaviour - it can exhibit correct behaviour in infinitely many cases, but may still be incorrect in some cases

Goals of Testing

- If our code passes all test cases, is it now guaranteed to be error free?
- Are 5000 random tests always better than 5 carefully selected tests?

Goals of Testing

- To show the presence of bugs (Dijkstra, 1972)
- If tests do not detect failures, we cannot conclude that software is defect-free
- Still, we need to do testing - driven by sound and systematic principles
  - Random testing is often not a systematic principle to use
  - Need a test plan
- Should help isolate errors - to facilitate debugging

Goals of Testing Continued

- Should be repeatable
  - Repeating the same experiment, we should get the same results
  - Repeatability may not be true because of the effect of the execution environment on testing
  - Repeatability may not occur if there are uninitialized variables
  - Repeatability may not happen when there is nondeterminism
- Should be accurate
  - Accuracy increases reliability
  - Part of the motivation for formal specification
- Is a successful test case one that passes the test, or one that shows a failure?

Test Plan

- Given that no single verification technique can prove correctness, the practical approach is to use ALL verification techniques. Is this statement True or False?

Test Plan

- Testing can uncover errors and build confidence in the software
- Resources of time, people, facilities are limited
- Need to plan how the software will be tested
- You know in advance that the software is unlikely to be perfect
- You need to put resources into the most important parts of the project
- A risk analysis can determine where to put your limited resources
- A risk is a condition that can result in a loss
- Risk analysis involves looking at how bad the loss can be and at the probability of the loss occurring

Test Plan

- Risks cannot be eliminated, but the development process can reduce the probability of loss associated with risks to an "acceptable" level
- One approach to risk analysis is FMEA - Failure Mode Effect Analysis
- Consider the capstone project of the autonomous rescue robots (A3)
  - The final grade is mostly based on the final documentation and the final demonstration/presentation in front of an industry panel
  - What are the most significant risks?
  - How do you test the robot?
  - How should calibration be handled?

Test Plan For Autonomous Rescue Robot

- Consider the capstone project of the autonomous rescue robots
  - Largest risk, robot fails during final demonstrations
  - Test to improve reliability
  - Test results of great interest to IBM judges
  - Think about test cases, think about testing environment versus final environment

White Box Versus Black Box Testing

- Do you know (or can you guess) the difference between white box and black box testing?
- What if they were labelled transparent box and opaque box testing, respectively?

White Box Versus Black Box Testing

- White box testing is derived from the program's internal structure
- Black box testing is derived from a description of the program's function
- Should perform both white box and black box testing
- Black box testing
  - Uncovers errors that occur in implementing requirements or design specifications
  - Not concerned with how processing occurs, but with the results
  - Focuses on functional requirements for the system
  - Focuses on normal behaviour of the system

White Box Testing

- Uncovers errors that occur during implementation of the program
- Concerned with how processing occurs
- Evaluates whether the structure is sound
- Focuses on abnormal or extreme behaviour of the system