CS245 – Lecture 8

Embedded Software Reliability

Tony Givargis

Software

- Unlike mechanical or electrical devices, software never “breaks”
  - Intuitively software stays as is
  - Unless problems in hardware effecting compute/storage
- Software has no shape, color, and mass
- Software does not age, rust, or deteriorate
  - But it has real existence and is critical component
- Unless ***proven*** correct, software is likely to be buggy

Software Tragedies

- Computer-controlled radiation-therapy machine of 1986 failed due to software not detecting a race condition
- The British destroyer Sheffield was sunk because the radar system identified an incoming missile as friendly
- Golf War, the math error that missed 0.000000095 second in precision every 10th of a second, accumulating for 100 hours, made the Patriot missile fail to intercept a scud missile and 28 lives were lost
- In 1991, after changing three LOC in a program with millions of LOC, the telephone systems in CA crashed

Should we use Software?

- ATM machine miscalculates your money
  - 50% chance you'll be happy
- Airplane software makes a mistake
  - Long way down
- Your heart pace-maker or radiation-therapy machine fails due to software error
  - matter of life and death
- Are we embedding potential disasters while we embed software into systems?

Software Reliability

- The probability of failure-free software operation for a specified period of time in a specified environment
  - Probabilistic function with the notion of time
- But, Software Reliability is not a direct function of time
  - As in age-related failure
- Software Reliability is an attribute
  - Similar to functionality, usability, performance, serviceability, capability, installability, maintainability
- Software Reliability is hard to achieve
  - Complexity of software

Software Complexity

- Software is easy to generate, thus more is generated
- Easy to implement features in software
  - More functionality is pushed onto software
- Software is easy to augment
  - Software components grow over time
- Examples
  - Aircraft over 5 million LOC, International Space Station over 5 million LOC + 10 million LOC on ground support, modern car has over 2 million LOC, etc.

Software Failures

- Software failures may be due to
  - Programming errors
  - Ambiguities
  - Oversight or misinterpretation of the specification
  - Carelessness or incompetence in writing code
  - Inadequate testing
  - Incorrect usage
  - Unexpected usage
- Hardware vs. Software
  - Physical vs. design

Design Faults

- Design faults differ from physical faults
  - Hard to visualize
  - Hard to classify
  - Thus, hard to detect & correct
- Design faults are closely related to human factors and design methodologies
- Design faults can not be address by duplication of components
  - Can’t be masked by voting
- Design faults are not manufacturing faults

Reliability Models

- Over 500 models developed
  - But still no good model for software reliability in existence
  - No single model is complete or can be applied at all times
- Most models are based on measurement metrics
  - Complexity of software == reliability of sotware
- Is LOC a measure of software complexity?
  - How do you count LOC?
- Functionality measurements
  - Count the number of functions delivered to the user

Reliability Models …

- Control-oriented complexity measurement
  - Reduce the code to its control structure
  - Eliminate data paths
  - Obtain a control-flow graph
- Data-oriented complexity measurement
  - Reduce the code to its data structure
  - Eliminate control paths
  - Obtain a data-flow graph
- Control/Data-oriented complexity measurement
  - Combine the above

Reliability Models …

- Coverage metric is to measure the amount of software that is executed correctly
  - Simulation based test
  - Under some input assumption
  - Under some expected behavior
- Project management metric
  - Better project management lead to more reliable software
- Process metrics
  - Better software development methodologies lead to more reliable software
  - ISO-9000 certification

Reliability Models …

- Fault and failure metrics is to measure the rate at which bugs are discovered
  - Keep testing until you the rate at which errors are discovered decreases
- Formal models
  - Generate (automatically) proofs or counter proofs that some software has certain property
    - What software?
    - What property?
    - How costly is the evaluation?

Redundancy

- Software requires different redundancy
  - Airplane might have two identical engines as a measure of redundancy and failure recovery
  - Can’t do the same with Software
- Software redundancy requires
  - Duplicate effort
    - Write the software using a different algorithm, process, programmer, etc.
  - Voting system
    - Must be much less complex than the duplicated functions
    - Formally proven to be correct