CS 6150: RELIABLE COMPUTING

Semester Hours: 3.0  
Contact Hours: 3
Coordinator: Ray Kresman
Text: TBD
Author: TBD
Year: TBD

SPECIFIC COURSE INFORMATION

Catalog Description:

Techniques for writing reliable software including n-version programming, fault-tolerant data structures and formal proofs of correctness. Rollback and recovery methods. Fault-tolerant hardware and methods of hardware error detection and correction. Prerequisites: Full Admission to MS in CS program, or consent of department.

Course type: ELECTIVE

SPECIFIC COURSE GOALS

• I can articulate why empirical software testing does not provide 100% guarantee on software correctness.
• I am able to write the specification/predicates, that should hold, at various points for simple programs.
• I understand how to use axiomatic techniques to prove correctness of simple programs, both partial and total.
• I am able to define/give examples of groups, rings and vector spaces.
• I can explain the relationship between minimum Hamming distance and error detection/correction capability.
• I can construct basis, or G matrix, to derive codewords for messages.
• I can construct H matrix and detect/correct received data.
• I can explain the application of memory error detection/correction techniques using Hamming code.
• I can construct fault tolerant data structures, for example, modify a linked list to permit error detection and correction.
• I understand how to derive test points that can detect a variety of linear domain errors.
• I can explain the tradeoff between memory and CPU in masking hardware faults.

LIST OF TOPICS COVERED

• Fault-Tolerant Hardware
  o Tandem computer architecture(*)
  o Stratus computer architecture
  o The (4,2) computer architecture
  o Hardware error detection and correction through coding(*)
  o Redundant array of inexpensive disks (RAID)(*)
• Fault-Tolerant Software
  o Formal proofs of correctness(*)
    ▪ Axiomatic semantics and proof rules
    ▪ Weakest precondition
    ▪ Strongest post condition
    ▪ Invariants and assertions
  o Formal specification – an overview
    ▪ VDM or Z
    ▪ Algebraic specification and data types
  o Roll back and recovery, check pointing(*)
  o Software Safety
  o N-version techniques(*)
  o Fault tolerant data structures and scrubbing(*)
  o User of error detection codes in software
  o Data integrity in distributed transactions
    ▪ Validation protocols for transactions
    ▪ Distributed check pointing
• Estimation of Mean Time Between Failures (MTBF)
  o Numerical aspects of software testing
  o Domain testing
  o Effect of redundant components
  o Effect of scrubbing
  o Standards for software fault-tolerance

(*) These topics are core material to be covered every time the course is taught.