FIT3142-Tutorial 6 - Solutions

2.1 Question 1 (25%)

Explain the difference between random failures and wearout failures. Why does this difference matter?

- **Random failures - exponentially distributed.**

$$R(t) = \exp(-\lambda t)$$

- **Wearout failures - normally distributed**

$$R_{wearout}(t) = \frac{1}{\sigma\sqrt{2\pi}} \int \exp(-\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)^2)dt$$

- **Infant Mortality**

2.2 Question 2 (25%)

Explain the difference between serial system and parallel system. What is Lusser's Product Law? What is a Cascade Failure?

Serial Systems

- Failure of single element takes out system.

Parallel Systems

- Failure of single element is survivable, but P[S] reduced

Lusser's product law

It states that the reliability of a series system is equal to the product of the reliability of its component subsystems, if their failure modes are known to be statistically independent.

- Discovered during A4/V2 missile testing in WW2
- Superceded dysfunctional 'weak link' model
- Describes behaviour of complex series systems.
- Theoretical basis of US Standards/methods defined in Mil-Hdbk-217 and Mil-Std-756
- The survival of the series system depends on the survival of every element in the system.
- Failure of single element takes out system.
- Use Lusser to quantify total λ and P[S] for some given time interval T.

$$R_s = \prod_{i=1}^{N} R_i \quad = \quad \exp(-\sum_{i=1}^{N} \lambda_i t)$$

Cascade Failure

- Cascade failures have become an area of research, following a number of such events arising in electricity grids;
- A cascade failure can occur in any system, in which there are a large number of interdependent nodes or subsystems;
- In a cascade failure event, loss of a node causes workload to be migrated to other nodes, if this workload is sufficient to saturate other nodes, these may also fail, resulting in a "cascade failure" as loss of nodes causes other nodes to fail;
- The well documented "congestion collapse" failures seen during early use of the Internet qualify as cascade failures;
- Any distributed application with automatic failover mechanisms is a candidate system for a cascade failure;
- Therefore, care must be taken with design to ensure this cannot happen – automation can produce unwanted effects!

2.3 Question 3 (25%)

Explain the six most common causes of software faults. What are the three most common methods for runtime detection of software faults?

- Numerical Failure - bad result calculated.
- Propagated Numerical Failure - bad result used in other calculations.
- Control Flow Failure - control flow of thread is diverted.
- Propagated Control Flow Failure – bad control flow propagates through code.
- Addressing Failure - bad pointer or array index.
- Synchronisation Failure - two pieces of code misunderstand each other's state.

Runtime Detection of software faults

- Consistency checks on values.
- Watchdog timers.
- Bounds checking

Consistency Checking

- Can identify a bad computational result.
- Exploit characteristics of data to identify problems.
- Protect data structures with checksums.
- Parallel dissimilar computations for result comparison.
- Recovery strategy required.

Watchdog Timers

- Require hardware support to interrupt tasks or processes.
- Watchdog timer periodically causes status check routine to be called.
- Status check routine verifies that code is doing what it should.
- Can protect against runaway control flow.
- Recovery strategy required.

Bounds Checking

- Compare results of computation with known bounds to identify bad results.
- Requires apriori knowledge of bounds upon results.
- Cannot protect against bad results which have ‘reasonable’ values.
- Recovery strategy required.

2.4 Question 4 (25%)

Explain the difference between the Dormant Fault Problem and the Complex System Problem. Why is testing often ineffective?

Dormant Fault Problem

- Statistical models used for hardware are irrelevant.
- Code may be operational for years with a fatal bug hidden somewhere.
- A set of conditions may one day arise which trigger the fault.
- If major disaster arises it may be impossible to recreate same conditions.

Complex System Problem

- Extremely complex system will be extremely difficult to simulate or test.
- Complexity may result in infeasible regression testing time.
- Components of system may interact in ‘unpredictable’ ways .
- Synchronisation failures may arise.
- Fault may be hidden and symptoms not easily detectable due complexity.