Prediction of Reliability and Availability for Ground Systems

Myron Hecht
SoHaR Incorporated, Beverly Hills, CA

Presented to
Ground System Architectures Workshop
The Aerospace Corporation
El Segundo, California
March 13 - 15, 2002
Outline

- Motivation
- Message
- Conceptual Framework for Integrated Hardware and Software Reliability Prediction
- Empirical Observations
- Example Model and Measurement-based approach
- Questions that can Now be Answered
Motivation

- Ground systems are mission critical
- Availability and Reliability are important performance metrics
- Traditional reliability prediction techniques are not directly applicable to software
- Software is the major cause of failures
- An integrated systems approach is needed
Motivation (continued)

- Existing approaches are inadequate for dealing with system reliability
  - Nearly all software reliability prediction models deal with reliability growth
    - can not provide results at levels needed for mission critical systems
    - deal with individual modules, not systems
  - Improper to consider software as component in series with hardware
    - Many different kinds of software (application, kernel, detection and recovery)
    - Need to consider special operating modes
Stochastic reliability/availability modeling techniques can be applied to Ground Control

- Can provide significant non-obvious answers to questions
- Architectural tradeoffs
Conceptual Framework for integration of software and hardware models

- Systems View
- Failure Model
- Controllable Failures and Non-controllable failures
Characteristics of mission critical systems

- Redundancy
- Conservative programming practices (e.g., no dynamic memory allocation)
- Disciplined development process
- Narrower scope of I/O
- COTS/NDI and developed hardware and software

Failures occur in a systems context

- Many failures not related to software at all
- Frequently not possible to determine whether software was involved
Failure Model

Development

Requirements/Design Mistake (Root Cause)

Manufacture Process Problem (Root Cause)

Operation & Maintenance

Triggering Event (Immediate Cause)

System Defect (Fault)

Precursor Event (Contributing Cause)

Failure (Error)

Effects
Same Model and Failure Properties for Hardware and Software

- Occurrence Time
- Duration
- Clustered (yes/no)
- Effect (Local, intermediate and system level)
- Site (CPU, interfaces, sensors, actuators)
- Cause (Root cause, Precursor Event, Triggering Event)
  
  • Manufacturing defects less likely for software (but could still occur, e.g., CM, undetected copies, etc.)
Unified Approach to Hardware and Software Reliability Analysis

- Same Model and Failure Properties Means that Similar Reliability Analysis Methods Can be Used
- Many operational software failures are not distinguishable from transient hardware failures
  - not reproducible ("random"), related to timing, unusual sequences, not repeatable, difficult to verify removal
- Some software failures are distinguishable
  - related to functionality, repeatable, relatively easy to verify removal
Conceptual Framework

Controllable Software Failures

• Characteristics
  – Known Root Cause under control of developer or user
    • Incorrect/ambiguous requirement
    • Configuration Management
    • Coding error
    • Installation
    • Maintenance
  – Means of determining that the failure has been mitigated

• Example Failure Mode
  – Consistently Incorrect response

• Addressed by Development, V&V and SQA processes
Conceptual Framework

**Random Failures**

- **Characteristics**
  - Related to timing, sequencing
  - Difficult to characterize and reproduce
    - Predominant failure mechanisms are due to randomly arriving inputs in a stable operational environment that interact with residual defects in the code
  - Generally not possible to tell whether transient failure was hardware or software related
  - Many failures fixed by reset
  - Act in a manner which can be modeled as random failures
Random Failures (continued)

- Example Failure Modes
  - Hang
  - Crash
  - Late response
  - Early response
- Not well addressed by traditional V&V and SQA processes
- Can be modeled as random events
Dealing with Random Software Failures

- Subject to measurement
  - Event logging (often integrated with operating system)
    - Record failures
    - Record time
- Parameters can be estimated using measurement techniques analogous to hardware
  - Determine failure rates and recovery times
  - Determine recovery probabilities (if fault tolerance is used)
- Confidence intervals can be estimated
Stochastic Approach is not well suited to controllable failures
- Traditional disciplined software development methodology is

Traditional V&V and SQA approaches do not address random failures
- Measurement based approach does

Measurement based approach should be used in systems sufficient testing and/or operational experience such that non-controllable failures are not important contributors to failure behavior
Empirical Observations on Failure Behavior

- Observations in earlier Research
- Terminal Doppler Weather System
Empirical Observations on Failure Behavior

Observations in earlier research

- Mature fault tolerant and other high integrity systems have residual software faults whose failure behavior can be characterized by an MTBF [Nagle82, Adams84, Hsueh88]

- A majority of such failures could be recovered from by the use of physical redundancy [Gray90, Lee95, Tang95].
Empirical Observations on Failure Behavior

ATC failure experience

Terminal Doppler Weather Radar

- 47 systems (not all currently installed)
- C-band Doppler weather radar to detect changes in wind speed and direction.
- Data processing subsystem (Dual Redundant Harris Nighthawk computers) provides warning messages on wind shear
- RS/6000 CPUs with real time kernel
Empirical Observations on Failure Behavior

TDWR Failure Distribution

74 failures were transient (solved by reset); 1 related to a specific software bug

Total TDWR Failures by Repair Action
171 failures occurring between March and June, 1996

- Reset: 43%
- Replace: 27%
- Auto recovery: 6%
- Not Stated: 21%

74 failures were transient (solved by reset); 1 related to a specific software bug
Analysis of a Hardware/Software System
Model

Top Level Model (Block Dgm)

- Backbone Model (Markov)
- Gateway Model (Markov)
- Front end Server (Markov)
- DBMS Server (Block Dgm)
  - Server (Block Dgm)
  - Hardware (Markov)
  - Solaris_OS (Markov)
  - Software (Markov)

Hardware (Markov)
Solaris_OS (Markov)
Software (Markov)
Top Level Model
Front End Server Subsystem

Diagram Front_End_Servers
Initial State: S0
Failure State: S7
Individual Server Model
Screen Shot from ME 2.0
Results

Failure Rates by Subsystem
Results (continued)

Impact of WAN Restoration Time

![Graph showing the impact of WAN Restoration Time on Yearly-Downtime (hours) with Service Provider Repair Rate (per hour).]
Economic Impact

Graph showing the relationship between WAN Service Provider Backbone Restoration Time (in minutes) and Monthly Revenue Loss ($). The graph indicates a linear increase in revenue loss as restoration time increases.
Conclusion: Questions that can be answered

• What architectural changes do I need to increase reliability?

• How much redundancy and what service provider guarantees do I need in order to support a given transaction level with a given availability?

• Which are the biggest contributors to the downtime of my mission critical information services?
Conclusion: Questions that can be answered

- How can I measure the combined availability and performance benefit of an a system investment?
- How can my budget be spent most effectively to increase availability and performance?
- What are the highest impact items to negotiate in my service level agreement?
- Is the benefit of replacing a single server with a cluster and/or a RAID subsystem worth the acquisition and ongoing support cost?