Approach and Challenges in Qualification of Mission-critical Software-reliant Systems

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Peter H Feiler
June 15, 2010
Outline

Sources of unreliability in software-reliant systems
An architecture-centric approach to software-reliant system engineering
A framework for reliability validation & improvement
We Rely on Embedded Software Systems for Safe & Reliable System Operation

Quantas Airbus A330-300 Forced to make Emergency Landing – 36 Injured
Written by http://bloomberg.com From: voywagon from Singapore

Even with the autopilot off, flight control computers still "command control surfaces to protect the aircraft from unsafe conditions such as a stall," the investigators said.

The unit continued to send false stall and speed warnings to the aircraft's primary computer and about 2 minutes after the initial fault "generated very high, random and incorrect values for the aircraft's angle of attack."

Airbus Gives Alert as Autopilot Caused Plane's Plunge (Update3)

By Ed Johnson

Oct. 15 (Bloomberg) -- Airbus SAS issued an alert to airlines worldwide after Australian investigators said a computer fault on a Qantas Airways Ltd. flight switched off the autopilot and generated false data, causing the jet to nosedive.

The Airbus A330-300 was cruising at 37,000 feet (11,277 meters) when the computer fed incorrect information to the flight control system, the Australian Transport Safety Bureau said yesterday. The aircraft dropped 650 feet within seconds, slamming passengers and crew into the cabin ceiling, before the pilots regained control.

"This appears to be a unique event," the bureau said, adding that Toulouse, France-based Airbus, the world's largest maker of commercial aircraft, issued a telex late yesterday to airlines that fly A330s and A340s fitted with the same air-data computer. The advisory is "aimed at minimizing the risk in the unlikely event of a similar occurrence."

Autopilot Off

A "preliminary analysis" of the Qantas plunge showed the error occurred in one of the jet's three air-data inertial reference units, which caused the autopilot to disconnect, the ATSB said in a statement on its Web site.

The crew flew the aircraft manually to the end of the flight, except for a period of a few seconds, the bureau said.

Even with the autopilot off, flight control computers still "command control surfaces to protect the aircraft from unsafe conditions such as a stall," the investigators said.

The unit continued to send false stall and speed warnings to the aircraft's primary computer and about 2 minutes after the initial fault "generated very high, random and incorrect values for the aircraft's angle of attack."

The flight control computer then commanded a "nose-down aircraft movement, which resulted in the aircraft pitching down to a maximum of about 8.5 degrees," it said.

No "Similar Event"

"Airbus has advised that it is not aware of any similar event over the many years of operation of the Airbus," the bureau added, saying it will continue investigating.
Software Problems not just in Aircraft

Lexus GX 460 passes retest; Consumer Reports lifts "Don't Buy" label

Consumer Reports is lifting the Don't Buy: Safety Risk designation from the 2010 Lexus GX 460 SUV after recall work corrected the problem it displayed in one of our emergency handling tests. (See the original report and video: "Don't Buy: Safety Risk-2010 Lexus GX 460.")

We originally experienced the problem in a test that we use to evaluate what's called lift-off oversteer. In this test, as the vehicle is driven through a turn, the driver quickly lifts his foot off the accelerator pedal to see how the vehicle reacts. When we did this with our GX 460, its rear end slid out until the vehicle was almost sideways. Although the GX 460 has electronic stability control, which is designed to prevent a vehicle from sliding, the system wasn't intervening quickly enough to stop the slide. We consider this a safety risk because in a real-world situation this could cause a rear tire to strike a curb or slide off of the pavement, possibly causing the vehicle to roll over. Tall vehicles with a high center of gravity, such as the GX 460, heighten our concern. We are not aware, however, of any reports of injury related to this problem.

Lexus recently duplicated the problem on its own test track and developed a software upgrade for the vehicle's ESC system that would prevent the problem from happening. Dealers received the software fix last week and began notifying GX 460 owners to bring their vehicles in for repair.

We contacted the Lexus dealership from which we had anonymously bought the vehicle and made an appointment to have the recall work performed. The work took about an hour and a half.

Following that, we again put the SUV through our full series of emergency handling tests. This time, the ESC system intervened earlier and its rear did not slide out in the lift-off oversteer test. Instead, the vehicle understeered—or plowed—when it exceeded its limits of traction, which is a more common result and makes the vehicle more predictable and less likely to roll over. Overall, we did not experience any safety concerns with the corrected GX 460 in our handling tests.

Many appliances now rely on electronic controls and operating software. But it turned out to be a problem for the Kenmore 4027 front-loader, which scored near the bottom in our February 2010 report.

Our tests found that the rinse cycles on some models worked improperly, resulting in an unimpressive cleaning.

When Sears, which sells the washer, saw our February 2010 Ratings (available to subscribers), it worked with LG, which makes the washer, to figure out what was wrong. They quickly determined that a software problem was causing short or missing rinse and wash cycles, affecting wash performance. Sears and LG say they have reprogrammed the software on the models in their warehouses and on about 65 percent of the washers already sold, including the ones we had purchased.

Our retests of the reprogrammed Kenmore 4027 found that the cycles now worked properly, and the machine excelled. It now tops our Ratings (available to subscribers) of more than 50 front-loaders and we've made it a CR Best Buy.

If you own the washer, or a related model such as the Kenmore 4044 or Kenmore Elite 4051 or 4219, you should get a letter from Sears for a free service call. Or you can call 800-733-2299.

How do you upgrade washing machine software?
State of Current Practice

Reliance on engineering process
  • Safety culture and build-then-test practice
  • Primarily text-based requirements documentation & traceability

Separation of system engineering & software engineering
  • Leads to requirements gaps for software

Reliability engineering treats hardware & software reliability the same
  • Reliability engineering has its roots in hardware
    – focuses on physical fault history of slowly evolving systems
    – assumes independence of events
  • Software is often assumed to be perfectible
    – software faults are due to design errors and are often systemic
    – rapid SW evolution and changing context result in reliability degradation over time

False assumption that reliable and secure systems are safe systems
Late Discovery of System-Level Problems

80% of accidents due to operator error
High recertification cost of design error corrections leads to 75% of operator time spent in work-arounds

Where faults are introduced
Where faults are found
The estimated nominal cost for fault removal

60% of errors in fault management software

Requirements & system interaction errors

80% late error discovery at high rework & recertification cost

20%, 16%
5x

20.5%, 110x

0%, 9%, 40x

Sources:
New Levels of System Interaction Complexity & Mismatched Assumptions

System User
- Human
  - System Under Control
    - System Engineer
      - Compute Platform
        - Operator Error
          - Driver lockout
    - Physical Plant Characteristics
      - Lag, proximity
      - Model recalibration
    - Data Streams
      - Unstable control & inconsistent state
        - due to jitter and loss
  - Runtime Architecture
    - Distribution & Redundancy
      - Loss of redundancy & other hazards due to HW Virtualization
  - Control System
    - Control Engineer
      - Data Representation
        - Ariane 4/5: 16-bit data
        - Air Canada: gal vs. l
      - Concurrency
        - Race conditions crash applications designed for single-core on multi-cores
  - Application Software
    - Application Developer
      - Data Representation
        - Ariane 4/5: 16-bit data
        - Air Canada: gal vs. l

Software runtime system impacts safety-critical software & system properties
Modeling Pitfall: Low Confidence due to Multiple “Truths”

Aircraft Industry Experience

System evolving over time

Inconsistency of independently maintained analysis models

Power budget  Timing model  Security model  Fault model  Redundancy model

Lack of confidence that implementation is consistent with model

System implementation
Outline

Sources of unreliability in software-reliant systems

An architecture-centric approach to software-reliant system engineering

A framework for reliability validation & improvement
SAE Architecture Analysis & Design Language (AADL) Captures Architecture for Analysis

**The System**
- Physical platform
  - Aircraft

**The Computer System**
- Hardware & OS

**The Software**
- Embedded Application Software
  - Flight control & Mission

**Focus on software runtime architecture**

**Deployed on**

Physical connectivity

**AADL** focuses on software runtime architecture and its interaction with physical systems & computer systems. Includes execution & communication, timing, behavior, error, partition (incl. ARINC653), data modeling, deployment semantics.
AADL: The Language

Precise execution semantics for components
- Thread, process, data, subprogram, system, processor, memory, bus, device, virtual processor, virtual bus

Continuous control & event response processing
- Data and event flow, synchronous call/return, shared access
- End-to-End flow specifications

Operational modes & fault tolerant configurations
- Modes & mode transition

Modeling of large-scale systems
- Component variants, layered system modeling, packaging, abstract, prototype, parameterized templates, arrays of components and connection patterns

Accommodation of diverse analysis needs
- Extension mechanism, standardized extensions
Single Truth through Consistency Across Analysis Models

Achieved by Single Source Annotated Architecture Model with Well-defined Semantics

Safety & Reliability
- MTBF
- FMEA
- Hazard analysis

Data Quality
- Data precision/accuracy
- Temporal correctness
- Confidence

Security
- Intrusion
- Integrity
- Confidentiality

Resource Consumption
- Bandwidth
- CPU time
- Power consumption

Embedded System Architecture Model in SAE AADL

Interaction Behavior
- Operational modes
- Protocols

Auto-generation of analytical models & implementation

Real-time Performance
- Execution time/Deadline
- Deadlock/starvation
- Latency
AADL Standard Suite & Industry Initiatives

www.aadl.info
wiki.sei.cmu.edu/aadl

AADL UML MARTE Profile 2009

OSATE Toolset
SEI STOOD ElliDiss

AADL Behavior Annex 2009

AADL ARINC653 Annex 2009

AADL Data Modeling Annex 2009


AADL Error Annex Standard June 2006

OpenGroup Real-Time Forum EU + US partners

IST ARTIST2 Embedded Systems Center of Excellence 2007-2011

OMG MARTE 2005-2009

EAST ADL Consortium AutoSAR


ITEA SPICES Model-Driven Embedded Systems Engineering 15 partners €16M 2006-2009

AVSI SAVI Analysis-based System Validation 12+ partners $80+M 2008-2014

Software Engineering Institute Carnegie Mellon
The JPL Mission Data System

A reference architecture autonomous space systems
- To be instantiated for different applications

An embedded systems architecture
- Consists of physical system, computing hardware, application software

A control systems architecture
- Feedback loops in application architecture
- Feedback loops in data management architecture

A multi-layered architecture
- From low-level control loops to goal-oriented planning and plan execution
Mission Data System (MDS) Architecture*

Closed loop
Goal-Directed
Explicit models
Separation of Concerns
Integral Fault Protection

* M. Bennett, R. Knight, R. Rasmussen, M. Ingham, “State-Based Models for Planning and Execution, 2006-08-11.
Excerpt from the Textual Specification:

process implementation MDSControlSystem.basic

subcomponents

GoalPlanner: thread group ControlSoftware::GoalPlanner;
GoalExecutive: thread group ControlSoftware::GoalExecutive;
GoalMonitor: thread group ControlSoftware::XGoalMonitor;
StateEstimation: thread group ControlSoftware::estimator;
StateControl: thread group ControlSoftware::controller;
OperatorConsole: thread group ControlSoftware::OperatorConsole;

Textual & Graphical Representations

Focus on Information Flow
Reference Architecture Instantiation

**Instantiation of reference architecture through refinement of AADL model**

**Deployment on different computing hardware platforms**
Temperature Control Loop Latency & Jitter

Use of immediate & delayed connections to achieve deterministic sampling

flow path
Latency Contributors

System & control engineering view:
- Processing latency, sampling latency, physical signal latency

Embedded software systems engineering view:
- Processor speed, scheduling protocol, sampled & queued message processing, resource contention, communication protocol & physical communication delays, legacy shared variable communication, rate group optimization, partitioned architecture, deployment changes
MDS Mission Planning & Plan Execution

State Analysis framework for mission modeling (CalTech/JPL)
Represent planning & plan execution tasks
Represent goal-based fault management

See also
Workload Analysis of Goal Network

Goal network interpretation & analysis in terms of modes

- Starting set: tasks with no predecessor
- Active set between synchronization points
- Generate System Operation Mode (SOM) sets
- Perform mode specific scheduling analysis

Diagram:

- Camera Temperature
- Heater Switch & Health
- External Temperature

Transition to temp between 10 and 20 degrees C
Healthy and switching between both on and off
Unconstrained
Outline

Sources of unreliability in software-reliant systems
An architecture-centric approach to software-reliant system engineering
A framework for reliability validation & improvement
SEI researching approach to establish an industry standard practice of reliability validation and qualification for software-reliant mission-critical systems by:

- Establishing an engineering framework for reliability validation & improvement
  - Integrates state-of-the-art technologies including formalized functional and non-functional requirements
  - System-focused safety analysis
  - Architecture-centric model-based analysis
  - Assurance cases
  - Formalized static analysis to complement current build-then-test practice
- Demonstrating its feasibility of reliability validation via model-based architecture analysis
- Proposing a set of metrics for cost-effective reliability evaluation & improvement

In collaboration with AVSI System Architecture Virtual Integration (SAVI) Initiative (Thursday talk)
Reliability Validation & Improvement Framework

Incremental System Requirements Validation and Verification

From System Requirements to Software Requirements
Formalized requirements
Focus on safety-criticality requirements

Mission Requirements
Function
Behavior
Performance

Safety-criticality Requirements
Reliability
Safety
Security

Software Assurance
Justified confidence that mission & safety-criticality requirements (claims) are met
Evidence through reviews, analysis, testing, and validated assumptions

Model Repository

Architecture-centric Model-based Engineering
Architecture model with well-defined semantics (AADL)
Consistency across analysis dimension

Architecture Model
Component Models
System Implementation

Resource & Performance Analysis
Reliability & Safety Analysis
Mode & Interaction Behavior Analysis
Static Analysis
Formal methods to complement testing
End-to-end V&V of mission and safety-criticality requirements

Software Engineering Institute | Carnegie Mellon
S5 Challenges & Approach to Software-reliant System Qualification
June 15, 2010
© 2010 Carnegie Mellon University
Confidence through Argument and Evidence

An assurance case requires claims, evidence, and an argument linking evidence to claims:

- **Claim**
  - Assertion that a design or implementation meets a requirement

- **Evidence**
  - Results of observing, analyzing, testing, simulating, and estimating the properties of a system

- **Argument**
  - Explanation of how the decomposition of claims and available evidence demonstrates compliance with mission & safety-criticality requirements

- **Confidence through credible evidence and effective verification methods**
  - Effectiveness of reviews, testing, analysis, simulation
  - Reuse of claim & evidence patterns

**Roots in Safety Cases (Kelly)**
UK Defense Std 00-56: Safety Mgmt Requirements for Defense Systems

**Compositional certification (Rushby)**
Composability, compositionality, emergence of properties
Partitioning as key to compositional reasoning

**Automation demonstrated in System Verification Manager project by B. Krogh et.al. (MoBIES)**
Requirements for Operational Systems

Operational system process: a set of behaviors by execution of functions to transform input into output utilizing state, respecting constraints/controls, requiring resources, to meet a defined mission in a given environment.

[Association Française d'Ingénierie Système (AFIS)]

Requirement specification consists of:

Mission requirements that specify required behavior under nominal conditions for a given time (functionality, behavior, and performance)

Safety-criticality requirements that specify the ability to operate under anomalous conditions (reliability, safety, and security).

Requirements and Architecture
Traceable decomposition of system requirements to system & software components to assure coverage

Formalized specification of error propagation and fault management via AADL Error Model Annex
Archetype-based Fault & Hazard Identification

Application interaction architecture patterns
- Feedback control system
- Data, event, message, command streams
- State-based interaction protocols
- Multi-tier service layers

Fault management architecture patterns
- Redundancy
- Monitoring & recovery
- Partitions

Pre-analyzed architecture patterns enable analysis of potentially high-risk safety-criticality areas

Example: Partitions limit error propagation to input/output errors [Rushby]

Fault & hazard types in common architecture patterns as starting point for FHA, FTA, FMEA, root cause analysis, and IV&V

Example: Potential hazard areas in feedback loop [Leveson]
Continuous V&V through Virtual Integration

Predictive validation

Verification Evidence

Sensitivity analysis for uncertainty

Confidence in implementation

Requirements Engineering

System Design

Software Architectural Design

Component Software Design

Top-Level Verification Items

High-level AADL Model

Detailed AADL Model

Specify Model-Code Interfaces

Model-driven artifact generation

Conformance of models and systems

Integration Test

System Test

Acceptance Test

Unit Test

→ generation of test cases

← updating models with actual data

Continuous V&V through Virtual Integration

Analysis-based validation & verification to support and complement testing.

S5 Challenges & Approach to Software-reliant System Qualification
June 15, 2010
© 2010 Carnegie Mellon University
Metrics for Cost-effective Reliability Improvement

Metrics for identifying high-risk areas: unreliability contributors
- High complexity architecture interaction patterns & root cause problem areas
- Faults, hazards, assumption coverage & their impact potential
- Independently maintained overlapping models
- New faults & hazards due to technology insertion and paradigm shifts

Metrics for identifying effective V&V: reliability contributors
- Mission & safety-criticality requirement decomposition & coverage
- Effectiveness of reviews, analysis, testing, … for a given claim
- Effectiveness of fault management & residual fault monitoring
- Evidence of assumption validation & end-to-end V&V

Maximize risk reduction through cost-effective reliability validation
- Focus on high risk areas first – taking into account criticality & impact severity
- Take into account cost & effectiveness of V&V methods and fault management
Summary

Software-reliant system engineering and qualification requires a paradigm shift from build-then-test to model-based engineering.

Architecture-centric modeling with strong semantics as found in the SAE AADL standard is key to analysis and validation of system-level mission and safety-critical properties throughout development life cycle.

Qualification requires assurance through justified confidence in evidence that mission and safety-criticality requirements are met.

Aerospace industry is advancing a virtual system architecture integration approach centered around the AADL Meta model & semantics and a model repository & model bus concept.
NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this presentation is not intended in any way to infringe on the rights of the trademark holder.

This Presentation may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.