Model-based Architectural Verification & Validation

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Peter H Feiler
Feb. 2009
Outline

Architecture-Centric Model-based Engineering

Multi-fidelity Model-based Analysis

Validation of Implementations
Airbus Auto-Pilot Problem

Quantas Airbus A330-300 Forced to make Emergency Landing - 36 Injured

Written by htbw on Oct-7-09 1:40pm
From: soyawannahknow.blogspot.com

Thirty-six passengers and crew were injured, some seriously, in a mid-air drama that forced a Qantas jetliner to make an emergency landing, the Australian carrier and police said on Tuesday.

The terrifying incident saw the Airbus A330-300 issue a mayday call when it suddenly changed altitude during a flight from Singapore to Perth, Qantas said.

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 550 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 flight 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 9.5 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.
Mismatched Assumptions

Why do system level failures still occur despite fault tolerance techniques being deployed in systems?
System Level Fault Root Causes

Violation of data stream assumptions
- Stream miss rates, Mismatched data representation, Latency jitter & age

Partitions as Isolation Regions
- Space, time, and bandwidth partitioning
- Isolation not guaranteed due to undocumented resource sharing
- fault containment, security levels, safety levels, distribution

Virtualization of time & resources
- Logical vs. physical redundancy
- Time stamping of data & asynchronous systems

Inconsistent System States & Interactions
- Modal systems with modal components
- Concurrency & redundancy management
- Application level interaction protocols

Data (stream) consistency
End-to-end latency analysis
Modeling of partitioned architectures
Fault propagation
security analysis
redundancy patterns
Validation by model checking & proofs
Potential Model-based Engineering Pitfalls

The system

Inconsistency between independently developed analytical models

Confidence that model reflects implementation

System models

System implementation

Models are more than Powerpoint pictures or block diagrams
Architecture-Centric Modeling Approach

Single Source Annotated Architecture Model

Availability & Reliability
- MTBF
- FMEA
- Hazard analysis

Security
- Intrusion
- Integrity
- Confidentiality

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

Real-time Performance
- Execution time/Deadline
- Deadlock/starvation
- Latency

Impact Across Quality Dimensions

Auto-generated analytical models

Resource Consumption
- Bandwidth
- CPU time
- Power consumption

Architectural Verification & Validation
Feller, Feb 2009
© 2008 Carnegie Mellon University
Outline

Architecture-Centric Model-based Engineering
Multi-fidelity Model-based Analysis
Validation of Implementations
Latency Contributors

• Processing latency
• Sampling latency
• Physical signal latency
Impact of Sampling Latency Jitter

Impact of Scheduler Choice on Controller Stability

• A. Cervin, Lund U., CCACSD 2006

Sampling jitter due execution time jitter and application-driven send/receive
Flow Use Scenario through Subsystem Architecture

- Display -> IOProcessor -> Command -> Comm -> Nav -> IOProcessor -> Modem -> IOProcessor -> Nav -> Comm -> Command -> Display

Latency = Partition hops + processing + transfer
Independent clock per processor

Multiple rates and processors with independent clocks
Subsystem latency exceeds expected latency

Lower bound latency inherent to partition architecture
Managed Latency Jitter through Deterministic Sampling

From Partitions

• Nav signal data

• Nav sensor data

Navigation Sensor Processing

20Hz

Integrated Navigation

10Hz

Nav data

Nav sensor data

Guidance Processing

20Hz

Periodic I/O

To Partitions

• Nav data

• FP data

Flight Plan Processing

5Hz

• Performance data

Aircraft Performance Calculation

2Hz

Fuel Flow

Input-compute-output (ICO) AADL thread semantics

Immediate and delayed data port connections for deterministic sampling
Latency Revisited

Latency has increased
Software-Based Latency Contributors

Execution time variation: algorithm, use of cache
Processor speed
Resource contention
Preemption
Legacy & shared variable communication
Rate group optimization
Protocol specific communication delay
Partitioned architecture
Migration of functionality
Fault tolerance strategy
Outline

Architecture-Centric Model-based Engineering
Multi-fidelity Model-based Analysis
Validation of Implementations
Options

Implements model semantics

Validate generator vs. source code
From Customer Design Document

“The 200 Hz update rate was used because the MUX data needed to be processed at twice the rate of the fastest channel to avoid a race condition. Because channel 3 operates at 100 Hz, the IO processor had to operate at 200 Hz. The race condition has been fixed by double-buffering data, but the IO processor execution rate was left at 200 Hz to reduce latency of MUX data.”

Did double buffering solved the problem or do we need to do more buffering?
Application-based Send and Receive (ASR)

\[(\tau_P | \tau_C)^*\]

3 buffers for ICO guarantee

\[T_P \leq \alpha_P \leq S \leq \Omega_P \leq D_P\]
\[T_C \leq \alpha_C \leq R \leq \Omega_C \leq D_C\]

\(\alpha\) : actual execution start time

\(\Omega\) : actual completion time

\[\alpha_p - \Omega_P \cap \alpha_C - \Omega_C \neq \emptyset \Rightarrow \text{non-deterministic sampling (S/R) order}\]
### Periodic Task Communication Summary

<table>
<thead>
<tr>
<th>Periodic</th>
<th>ASR</th>
<th>DSR</th>
<th>DMT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IMT</td>
<td>IMT</td>
<td></td>
</tr>
<tr>
<td>Same period</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \tau_P ; \tau_C )</td>
<td>MF:1B</td>
<td>PD:2B</td>
<td>PD:2B</td>
</tr>
<tr>
<td></td>
<td>( S_{\lor} X \lor R )</td>
<td>( R )</td>
<td>( S_{\lor} X/R )</td>
</tr>
<tr>
<td>( \tau_C ; \tau_P )</td>
<td>PD:1B</td>
<td>PD:1B</td>
<td>PD:1B</td>
</tr>
<tr>
<td>( \tau_P \neq \tau_C )</td>
<td>ND:1B</td>
<td>PD:2B</td>
<td>PD:2B</td>
</tr>
<tr>
<td></td>
<td>( X )</td>
<td>( R )</td>
<td>( X/R )</td>
</tr>
<tr>
<td>( \tau_P \mid \tau_C )</td>
<td>ND:3B</td>
<td>PD:2B</td>
<td>PD:2B</td>
</tr>
<tr>
<td></td>
<td>( S/X_C )</td>
<td>( R )</td>
<td>( X/R )</td>
</tr>
</tbody>
</table>

- **MF**: Mid-Frame
- **PD**: Period Delay
- **ND**: Non-Deterministic
- **NDI**: No Data Integrity
- **1B**: Single buffer
- **2B**: Two buffers
- **3B**: Three buffers
- **4B**: Four buffers
- **S, X, R**: data copy
- **S/X**: IMT combined send/xfer
- **S/X/R**: DMT combined S, X, R
- **X/R**: DSR/PMT combined X, R
- **\( \lor \)**: One operation copy
Dual Redundant Flight Guidance System


NASA/CR-2005-213912

A Methodology for the Design and Verification of Globally Asynchronous/Locally Synchronous Architectures

Steven P. Miller and Mike W. Winfield
Rockwell Collins, Inc., Cedar Rapids, Indiana

Dan O’Brien, Nazi P. Hamidabadi, and Ayilli Joobi
University of Minnesota, Minneapolis, Minnesota
To validate:
1) At least one output
2) Exactly one output
3) Two outputs in critical mode

Property2 := (\text{Left\_FGS\_Pilot\_Flying} = \neg\text{Right\_FGS\_Pilot\_Flying});

However, instead of AG(Property2), the CTL property that we need to prove is

\[
AG (\neg Property2 \land \neg Left\_FGS\_Clock) \rightarrow (A (\neg LR\_Channel\_Clock \lor (AX A (\neg Right\_FGS\_Clock \lor Property2)))) | (\neg Property2 \land Right\_FGS\_Clock) \rightarrow (A (\neg LR\_Channel\_Clock \lor (AX A (\neg Left\_FGS\_Clock \lor Property2))))
\]

Increased complexity of property
Observation of events by sampling state

- Corrected asynchronous solution ignores pilot input events
- Push button requires *complete event stream*

Distributed processing of mode state machine

- Central vs. distributed logic
- Fail safe coordination of state transitions
- Properties during mode transition

Clock drift in asynchronous system

- Acceptable drift: bounded vs. fault
- Built-in drift bound through period: period wrap-around
- Loss of data stream element due to wrap around

Modeling in AADL helped identify issues

Button input as event to data port

Reduced property complexity by mode transition as state

Synchronization domains & fault conditions

Event processing vs. data sampling
Quantified Out-of-sync Modes
Subtle Errors – LM Aero UAV Sensor Voting

OFP Triplex Voter

- 96 Simulink Subsystems
- 3 Stateflow Diagrams
- \(6 \times 10^{13}\) Reachable States

Formal Verification

- 25 Informal Requirements
- 57 Formal Properties

Resulting In

- 24 Counterexamples
- 10 Design Modifications

Formal verification has found subtle errors that would likely be missed by traditional testing.

- Lockheed Martin
Towards Architecture Centric Engineering

Build on architecture tradeoff analysis (e.g., SEI ATAM)
- Provides focused evaluation method
- MBE/AADL provides quantitative analysis & starter models to build on

Facilitate pattern-based technical architecture root cause analysis
- Identify systemic risks in technology migration and refresh
- AADL provides semantic framework to reason about technical problem areas and potential mitigation strategies

Scalability through architecture extraction
- Leverage existing design data bases
- Challenge: abstract away from design details
- Focus on “what” instead of “how”

Support system and software assurance
- Provides structured approach to safety/dependability assurance
- MBE/AADL provides evidence based on validated models