From Virtual System Integration to Incremental Lifecycle Assurance

Peter H. Feiler

Software Solutions Conference 2015
November 16–18, 2015
From Virtual System Integration to Incremental Lifecycle Assurance

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.

[Distribution Statement A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material 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.


DM-0003034
Agenda

Challenges in Software Reliant Systems
Four Pillar Improvement Strategy
Virtual System Integration
Incremental Lifecycle Assurance
We Rely on Software for Safe Aircraft Operation

Oct. 15 (Bloomberg) -- Airbus SAS issued an alert to airlines on Monday after Australian investigators said a computer fault on a Qantas Airways Ltd. flight switched off the autopilot and generated false altitude data, sending 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 plane dropped 650 feet within seconds, slamming passengers and crew against the ceiling, before the pilots regained control.

“This appears to be a unique event,” the bureau said, adding that Airbus, the world’s largest maker of passenger aircraft, issued a telex late yesterday to airlines that fly A300s fitted with the same air-data computer. The advisory is aimed at minimizing the risk in the unlikely event of a similar occurrence.

The Federal Aviation Administration says a software problem with Boeing 787 Dreamliners could lead to one of the advanced jetliners losing electrical power in flight, which could lead to loss of control.

The FAA notified operators of the airplane Friday that if a 787 is powered continuously for 248 days, the plane will automatically shut down its alternating current (AC) electrical power.
Software Problems not just in Aircraft

May 7, 2010

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.

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.

How do you upgrade washing machine software?

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.
High Fault Leakage Drives Major Increase in System Cost

Aircraft industry has reached limits of affordability due to exponential growth in SW size and complexity.

70% Requirements & system interaction errors

80% late error discovery at high rework cost

Major cost savings through rework avoidance by early discovery and correction
A $10k architecture phase correction saves $3M

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

Sources:

Software as % of total system cost
1997: 45% → 2010: 66% → 2024: 88%

Post-unit test software rework cost
50% of total system cost and growing
Mismatched Assumptions in System Interactions

System Engineer

- Hazards
- Impact of system failures

Control Engineer

- Measurement Units, value range
- Boolean/Integer abstraction
- Air Canada, Ariane, 7500 Boolean variable architecture

System Under Control

- Physical Plant Characteristics
- Lag, proximity

Data Stream Characteristics

- Latency jitter affects control behavior
- Potential event loss

Operator Error

- Automation & human actions

System User/Environment

- Operator Error

Hardware Engineer

- Distribution & Redundancy
- Virtualization, load balancing, mode confusion

Runzheimer Architecture

- Concurrency
- Communication
- iTunes crashes on dual-cores

Application Developer

- Concurrency

Embedded SW System Engineer

- Embedded software system as major source of hazards

Why do system level failures still occur despite fault tolerance techniques being deployed in systems?
Model-based Engineering Pitfalls

Inconsistency between independently developed analytical models

Confidence that model reflects implementation

This aircraft industry experience has led to the System Architecture Virtual Integration (SAVI) initiative

The system

System models

System implementation
Awareness of Requirement Quality

Textual requirement quality statistics

- Current requirement engineering practice relies on stakeholders traceability and document reviews resulting in high rate of requirement change

Managed awareness of requirement uncertainty reduces requirement changes by 50%

- 80% of requirement changes from development team
- Expert assessment of change uncertainty
- Focus on high uncertainty and high importance areas
- Engineer for inherent variability

<table>
<thead>
<tr>
<th>Requirements error</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incomplete</td>
<td>21%</td>
</tr>
<tr>
<td>Missing</td>
<td>33%</td>
</tr>
<tr>
<td>Incorrect</td>
<td>24%</td>
</tr>
<tr>
<td>Ambiguous</td>
<td>6%</td>
</tr>
<tr>
<td>Inconsistent</td>
<td>5%</td>
</tr>
</tbody>
</table>

NIST Study

Rolls Royce Study

![Figure 8: Precedence measurement scale](image)
Agenda

Challenges in Software Reliant Systems
Four Pillar Improvement Strategy
Virtual System Integration
Incremental Lifecycle Assurance
Assurance & Qualification Improvement Strategy

Assurance: Sufficient evidence that a system implementation meets system requirements

---

**Architecture-led Requirement Specification**

**Architecture-centric Virtual System Integration**

**Static Analysis & Compositional Verification**

**Incremental Assurance Plans & Cases throughout Life Cycle**

---

Early Problem Discovery through Virtual System Integration & Analysis

Improved Assurance through Better Requirements & Automated Verification
Improved Cost, Time and Quality

- **Assure the System**
- **Build the System**

**Improved Quality through Better Requirements & Evidence**
- **Defect Introduction**
- **Post Unit Test Discovery**

**Reduced Cost and Time through Early Discovery**
- **80% Post Unit Test Discovery**
- **70% Defect Introduction**

- Improved Cost, Time and Quality
- 80% Post Unit Test Discovery
- 70% Defect Introduction
- Reduced Cost and Time through Early Discovery
- Improved Quality through Better Requirements & Evidence
Agenda

Challenges in Software Reliant Systems
Four Pillar Improvement Strategy
Virtual System Integration
Incremental Lifecycle Assurance
SAE Architecture Analysis & Design Language (AADL) to the Rescue

- **Physical system**
- **Command & Control**
- **Task & Communication Architecture**
- **Deployed on Physical interface**
- **Distributed Computer Platform**

AADL Addresses Increasing Interaction Complexity and Mismatched Assumptions
Analysis of Virtually Integrated Software Systems

Single Annotated Architecture Model Addresses Impact Across Operational Quality Attributes

Safety & Reliability
- MTBF
- FMEA
- Hazard analysis

Security
- Intrusion
- Integrity
- Confidentiality

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

Affects temporal correctness

Potential new hazard

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

Increased latency

Resource Consumption
- Bandwidth
- CPU time
- Power consumption

Change of Encryption from 128 bit to 256 bit

Higher CPU demand

Auto-generated analytical models
Towards an Architecture-Centric Virtual Integration Practice (ACVIP)

Army and other Government Shadow Projects

Future Vertical Lift

JMR TD: ACVIP Shadow Projects

Common Avionics Architecture System

Apache Block III ATAM

JPL Mission Data System

CH47F Health Monitor

System Architecture Virtual Integration (SAVI) Software & Systems Engineering

SAE AADL Standard & AADL Workbench: Research Transition Platform

US & European Research Initiatives

Other Standards and Regulatory Guidance

Towards an Architecture-Centric Virtual Integration Practice (ACVIP)
Finding Problems Early

**Issue:** Contractor could not assess integration risk early enough.

**Action:** 6 Week Virtual Integration identified 20 major issues.

**Result:** Adjusted CDR Schedule to remediate.

- Prevented 12 month delay in a 2 year project.

The current method would not have identified the issues until 3 months before delivery.

---

**System Architecture Virtual Integration (SAVI) 2008-**

Proof of concept with AADL led to ten year commitment

**SAVI ROI Study (2009/10)**

$2B savings on $10B aircraft through 33% early detection

---

**Architecture-centric Virtual Integration Practice (ACVIP)**

2014/15 Virtual Integration Shadow led to early discovery of 85+ potential integration issues

Led to acceleration of adoption by JMR contractors and inclusion in RFP for FY16/17 projects
Incremental Multi-Tier Assurance in SAVI

- Aircraft: (Tier 0)
- Aircraft system: (Tier 1)
  - Engine, Landing Gear, Cockpit, ...
  - Weight, Electrical, Fuel, Hydraulics, ...
- LRU/IMA System: (Tier 2)
  - Hardware platform, software partitions
  - Power, MIPS, RAM capacity & budgets
  - End-to-end flow latency
- System & SW Engineering:
  - Mechatronics: Actuator & Wings
  - Safety Analysis (FHA, FMEA)
  - Reliability Analysis (MTTF)
- Subcontracted software subsystem: (Tier 3)
  - Tasks, periods, execution time
  - Software allocation, schedulability
  - Generated executables
- OEM & Subcontractor:
  - Subsystem proposal validation
  - Functional integration consistency
  - Data bus protocol mappings
- Repeated Virtual Integration Analyses:
  - Power/weight
  - MIPS/RAM, Scheduling
  - End-to-end latency
  - Network bandwidth

**Proof of Concept Demonstration and Transition by Aerospace industry initiative**

- Architecture-centric model-based software and system engineering
- Architecture-centric model-based acquisition and development process
- Multi notation, multi team model repository & standardized model interchange

- Multi-tier system & software architecture (in AADL)
- Incremental end-to-end verification of system properties
Automated FMEA Experience

Failure Modes and Effects Analyses are rigorous and comprehensive reliability and safety design evaluations

• Required by industry standards and Government policies
• When performed manually are usually done once due to cost and schedule
• If automated allows for
  • multiple iterations from conceptual to detailed design
  • Tradeoff studies and evaluation of alternatives
  • Early identification of potential problems

Largest analysis of satellite to date consists of 26,000 failure modes

• Includes detailed model of satellite bus
• 20 states perform failure mode
• Longest failure mode sequences have 25 transitions (i.e., 25 effects)
Agenda

Challenges in Software Reliant Systems
Four Pillar Improvement Strategy
Virtual System Integration
Incremental Lifecycle Assurance
Incremental Lifecycle Assurance Objectives

Measurably improve critical system assurance through

• Better coverage and managed uncertainty
• Incremental analytical verification throughout lifecycle
• Focus on high payoff areas
Requirements & Architecture Design Constraints

Textual Requirements for a Patient Therapy System

1. The patient shall never be infused with a single air bubble more than 5ml volume.
2. When a single air bubble more than 5ml volume is detected, the system shall stop infusion within 0.2 seconds.
3. When piston stop is received, the system shall stop piston movement within 0.01 seconds.
4. The system shall always stop the piston at the bottom or top of the chamber.

Same Requirements Mapped to an Architecture Model

We have effectively specified a system partial architecture.

Importance of understanding system boundary

U Minnesota Study
Three Dimensions of Incremental Assurance

Incremental assurance through virtual system integration for early discovery

Priority focused architecture design exploration for high payoff

Compositional verification and partitions to limit assurance impact

Early Discovery leads to Rework Reduction

Build the System

Assure the System

Requirements Engineering
- Requirements Validation
  - Architecture Modeling
    - Analysis & Generation
  - Deployment Build
    - Acceptance Test

System Design
- System Architecture Validation
  - Integration Build
    - Integration Test
  - System Test

Software Architectural Design
- Architecture Validation
  - Design Build
    - Design Validation
  - Code Development
    - Unit Test

Component Software Design
- Design Build
  - Unit Test

Requirements Coverage
- Requirement Coverage
  - Compositional Verification
    - Design & Req Refinement
      - Compositional Verification
Three Dimensions of Requirement Coverage

- **System interactions, state, behavior**
  - Environment
  - Constraints/Controls
  - System
  - Behavior
  - Output
  - Input
  - State
  - Resources
  - Implementation constraints
  - Exceptional conditions

- **Design & operational quality attributes**
  - Security
    - Data confidentiality
    - Data integrity
  - Availability
    - H/W failure
    - COTS S/W failures
  - Modifiability
    - Change COTS
    - Change Web user interface in < 8 person-weeks
  - Performance
    - Data Latency
    - Transaction Throughput
  - (L,M)
    - Reduce storage latency on customer DB to < 200 ms.
  - (H,H)
    - Deliver video in real time.
  - (M,M)
    - Add CORBA middleware in < 20 person-months.
  - (H,L)
    - Change Web user interface in < 8 person-weeks.
  - (H,H)
    - Power outage at site 1 requires traffic redirected to site 2 in < 3 seconds.
  - (H,H)
    - Network outage in < 1.5 minutes.
  - (H,H)
    - Credit card transactions are secure 99.999% of the time.
  - (H,L)
    - Customer DB authorization works 99.999% of the time.

- **Guarantees**
  - Assumptions
  - Invariants

- **Fault impact & contributors**
  - Omission errors
  - Commission errors
  - Value errors
  - Sequence errors
  - Timing errors
  - Replication errors
  - Rate errors
  - Concurrency errors
  - Authentication errors
  - Authorization errors

- **Fault Propagation Ontology**

---

From Virtual System Integration to Incremental Lifecycle Assurance
Nov 18, 2015
© 2015 Carnegie Mellon University
Automated Incremental Assurance Workbench

Identify Assurance Hotspots throughout Lifecycle

Stakeholder Goals

High Abstraction

Tier 0

Model

Ver Plan

Req

Tier 1

Model+1

Ver Plan

Req+1

Tier 2

Model+2

Model+2'

Ver Plan

Req+2

Assurance Case

Low Level

Close to Implementation
Secure Mathematically-Assured Composition of Control Models

Key Problem
Many vulnerabilities occur at component interfaces. How can we use formal methods to detect these vulnerabilities and build provably secure systems?

16 months into the project
Draper Labs could not hack into the system in 6 weeks
Had access to source code

Accomplishments
- Created AADL model of vehicle hardware & software architecture
- Identified system-level requirements to be verified based on input from Red Team evaluations
- Developed Resolute analysis tool for capturing and evaluating assurance case arguments linked to AADL model
- Developed example assurance cases for two security requirements
- Developed synthesis tool for auto-generation of configuration data and glue code for OS and platform hardware

Technical Approach
- Develop a complete, formal architecture model for UAVs that provides robustness against cyber attack
- Develop compositional verification tools driven from the architecture model for combining formal evidence from multiple sources, components, and subsystems
- Develop synthesis tools to generate flight software for UAVs directly from the architecture model, verified components, and verified operation system
Agenda

Challenges in Software Reliant Systems
Four Pillar Improvement Strategy
Virtual System Integration
Incremental Lifecycle Assurance
Benefits of Virtual System Integration & Incremental Lifecycle Assurance

Reduce risks
  • Understand system wide impact early
  • Verify assumptions across system

Increase confidence
  • Verified models to complement integration tests
  • System design evolved from verified models

Reduce cost
  • Fewer system integration problems
  • Less assurance related rework
References

AADL Website www.aadl.info and AADL Wiki www.aadl.info/wiki
Blog entries and podcasts on AADL at www.sei.cmu.edu
AADL Book in SEI Series of Addison-Wesley
On AADL and Model-based Engineering
http://www.sei.cmu.edu/library/assets/ResearchandTechnology_AADLandMBE.pdf
On an architecture-centric virtual integration practice and SAVI
http://www.sei.cmu.edu/architecture/research/model-based-engineering/virtual_system_integration.cfm
On an a four pillar improvement strategy for software system verification and qualification