



# 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-7-08 1:48pm  
From: [soyawannaknow.blogspot.com](http://soyawannaknow.blogspot.com)

★★★★☆  Email



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 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."



Fallback solution also computer-controlled

### Autopilot Off

A "preliminary analysis" of the Qantas incident 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.

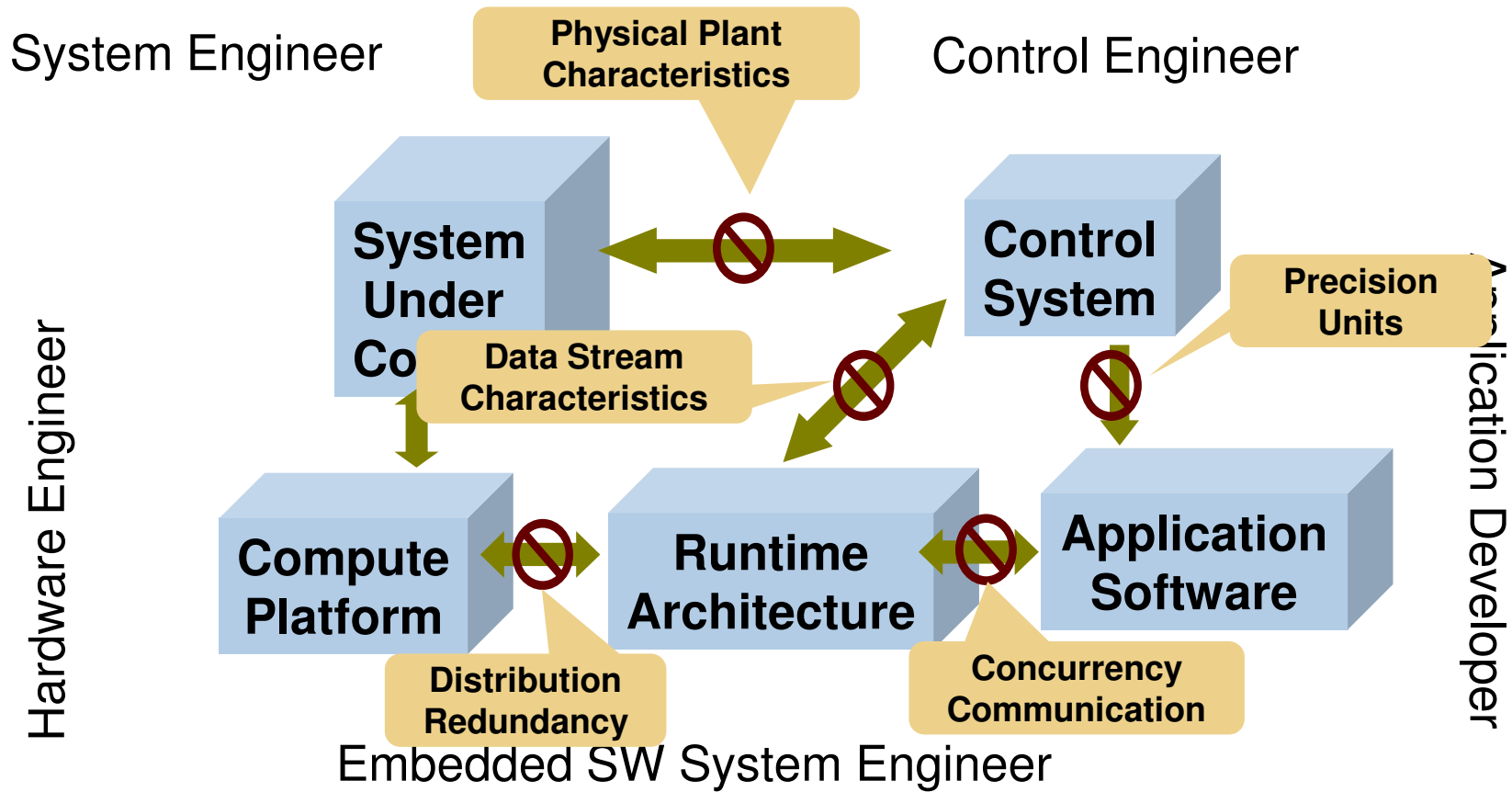
The unit continued to send false stall and speed warnings to the aircraft's primary computer and about 2 minutes after the initial fault "generated very high, random and incorrect values for the aircraft's angle of attack."

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.

# 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

**Data (stream) consistency**  
**End-to-end latency analysis**

- Stream miss rates, Mismatched data representation, Latency jitter & age

## Partitions as Isolation Regions

**Modeling of partitioned architectures**

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

## Virtualization of time & resources

**Fault propagation**  
**security analysis**  
**redundancy patterns**

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

## Inconsistent System States & Interactions

**Validation by model**  
**checking & proofs**

- Modal systems with modal components
- Concurrency & redundancy management
- Application level interaction protocols

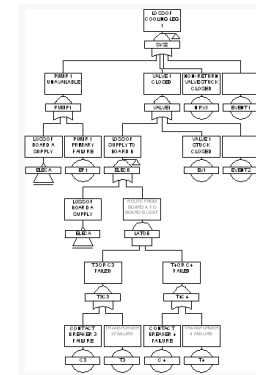
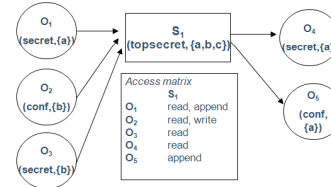
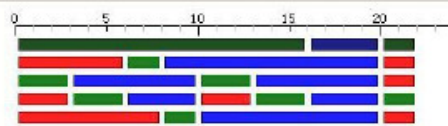


# Potential Model-based Engineering Pitfalls



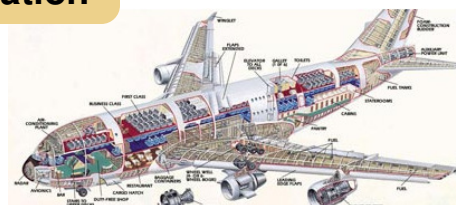
The system

Inconsistency between independently developed analytical models



System models

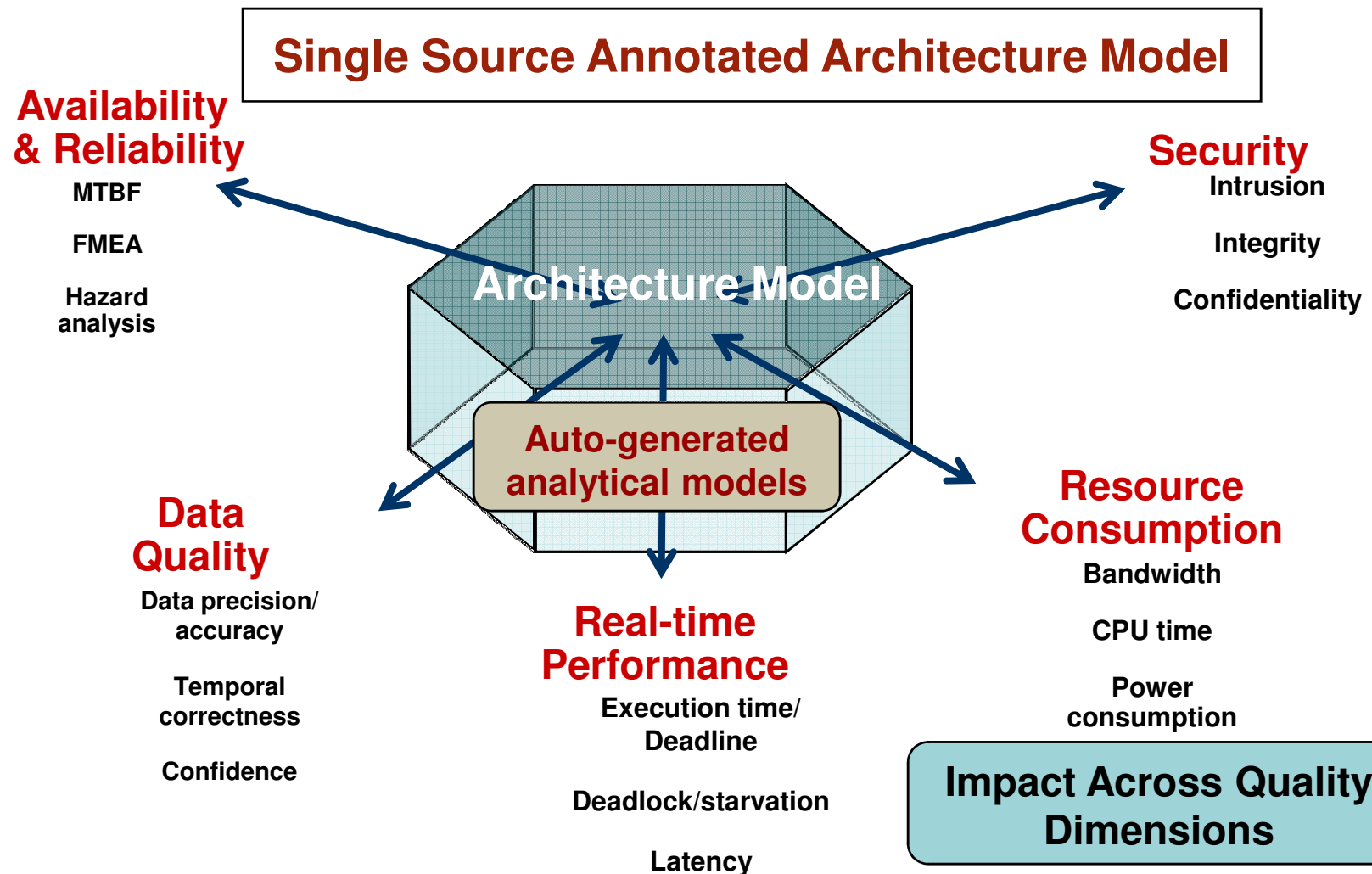
Confidence that model reflects implementation



System implementation

Models are more than Powerpoint pictures or block diagrams

# Architecture-Centric Modeling Approach



# Outline

---

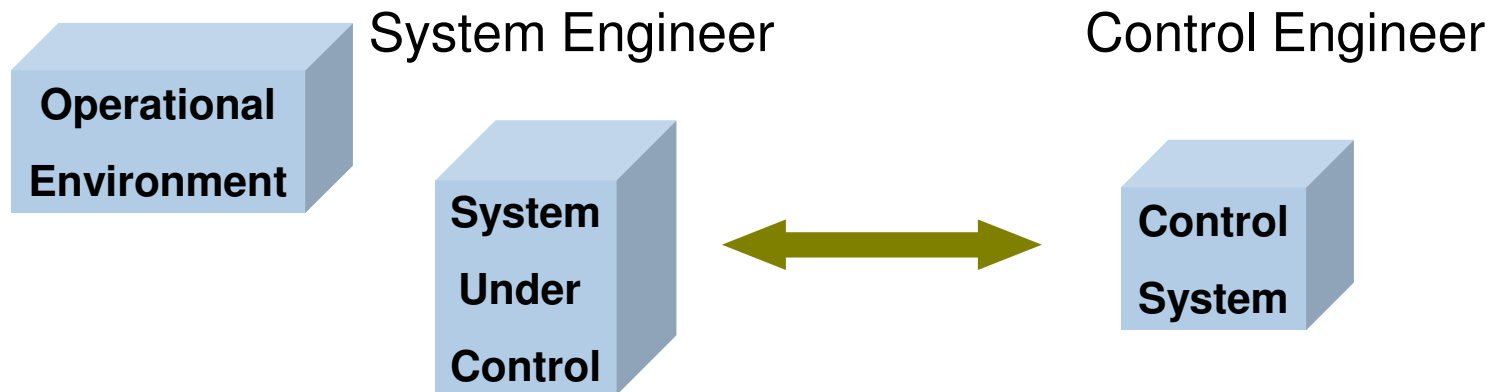
Architecture-Centric Model-based Engineering

**Multi-fidelity Model-based Analysis**

Validation of Implementations



# Latency Contributors



- Processing latency
- Sampling latency
- Physical signal latency

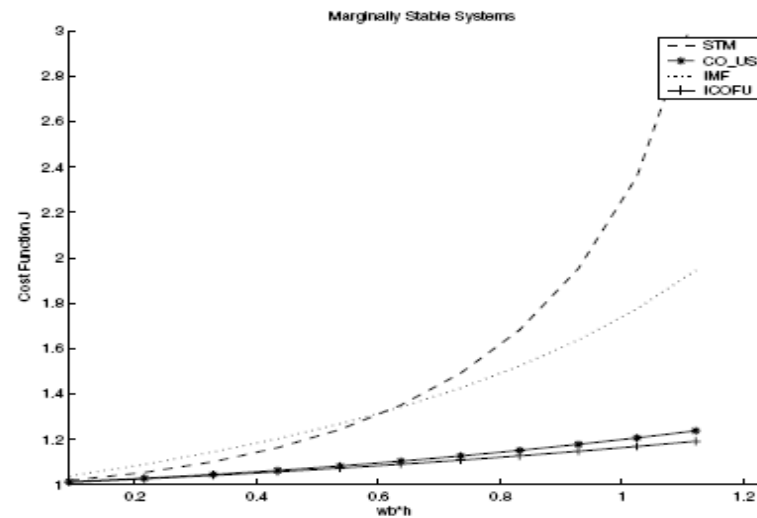
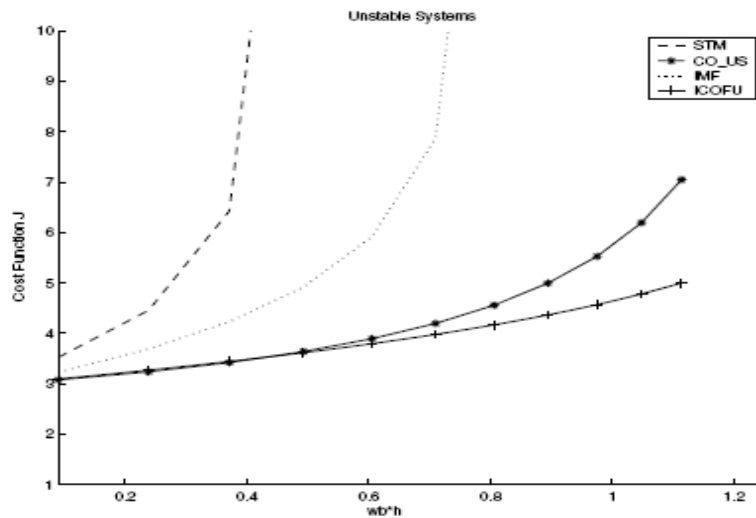
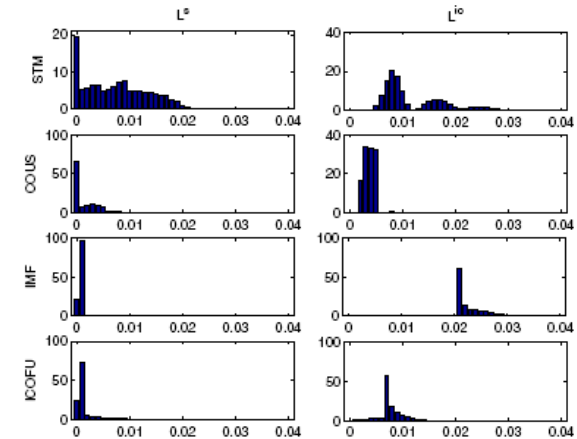


# Impact of Sampling Latency Jitter

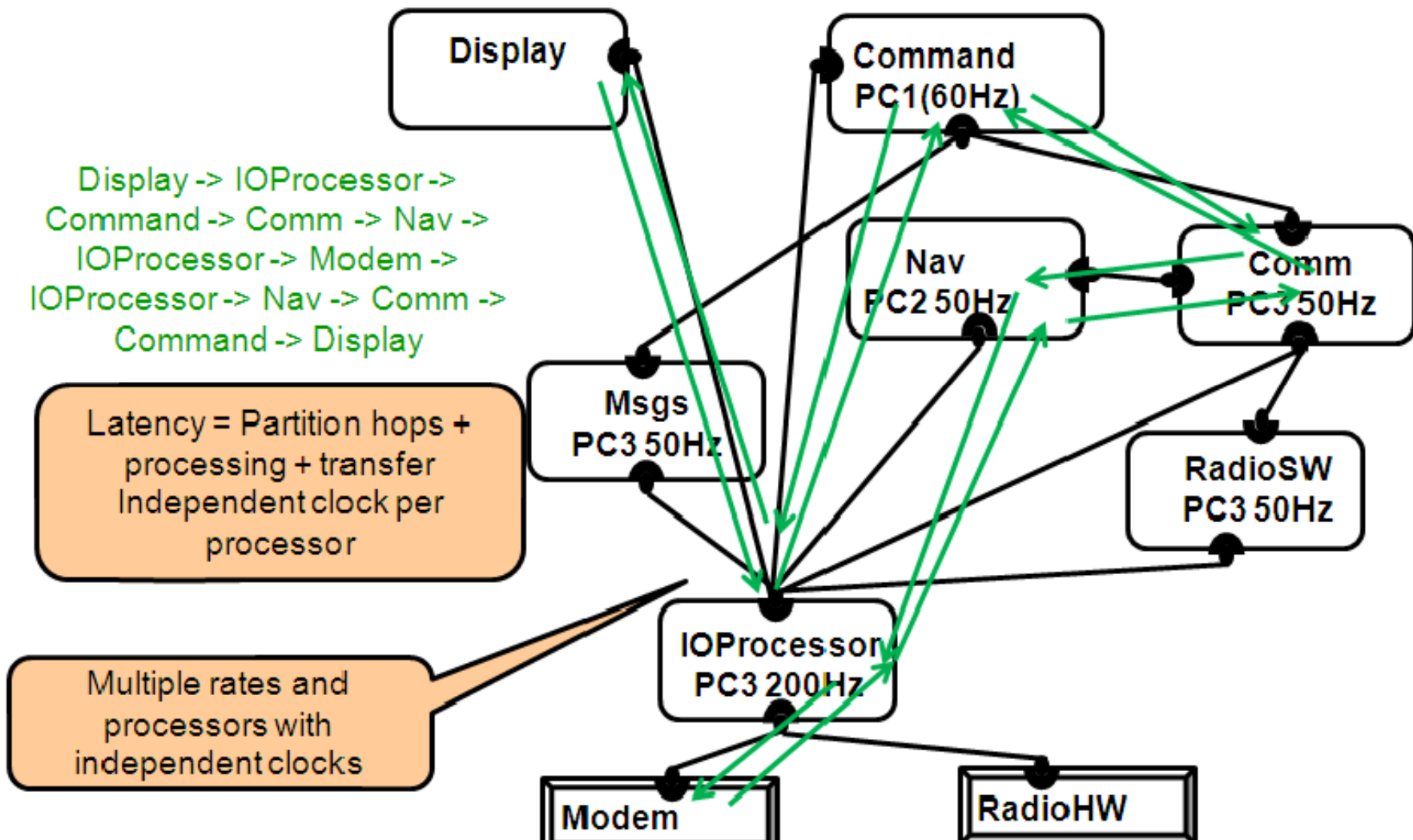
## Impact of Scheduler Choice on Controller Stability

- A. Cervin, Lund U., CCACSD 2006

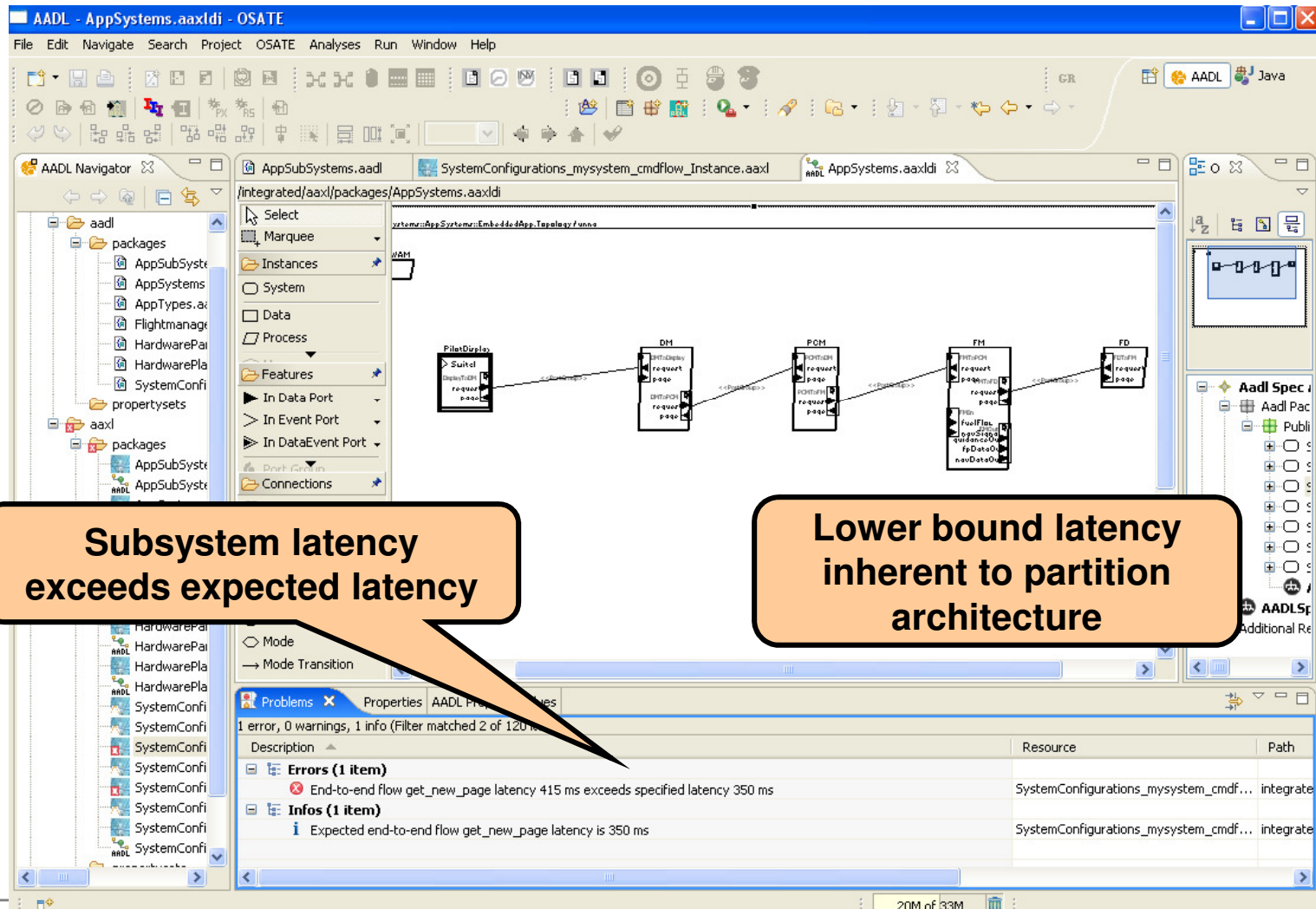
Sampling jitter due execution time jitter and application-driven send/receive



# Flow Use Scenario through Subsystem Architecture



# Partition-Level Flow Latency



The screenshot shows the AADL IDE interface with a flow graph. The graph consists of several components: PilotDisplay, DM, PCM, FM, and FD. Each component has a 'request' and 'page' port. The flow is: PilotDisplay (request) -> DM (request) -> PCM (request) -> FM (request) -> FD (request). The FM component also has a 'newData0' output port.

Two callout boxes highlight key findings:

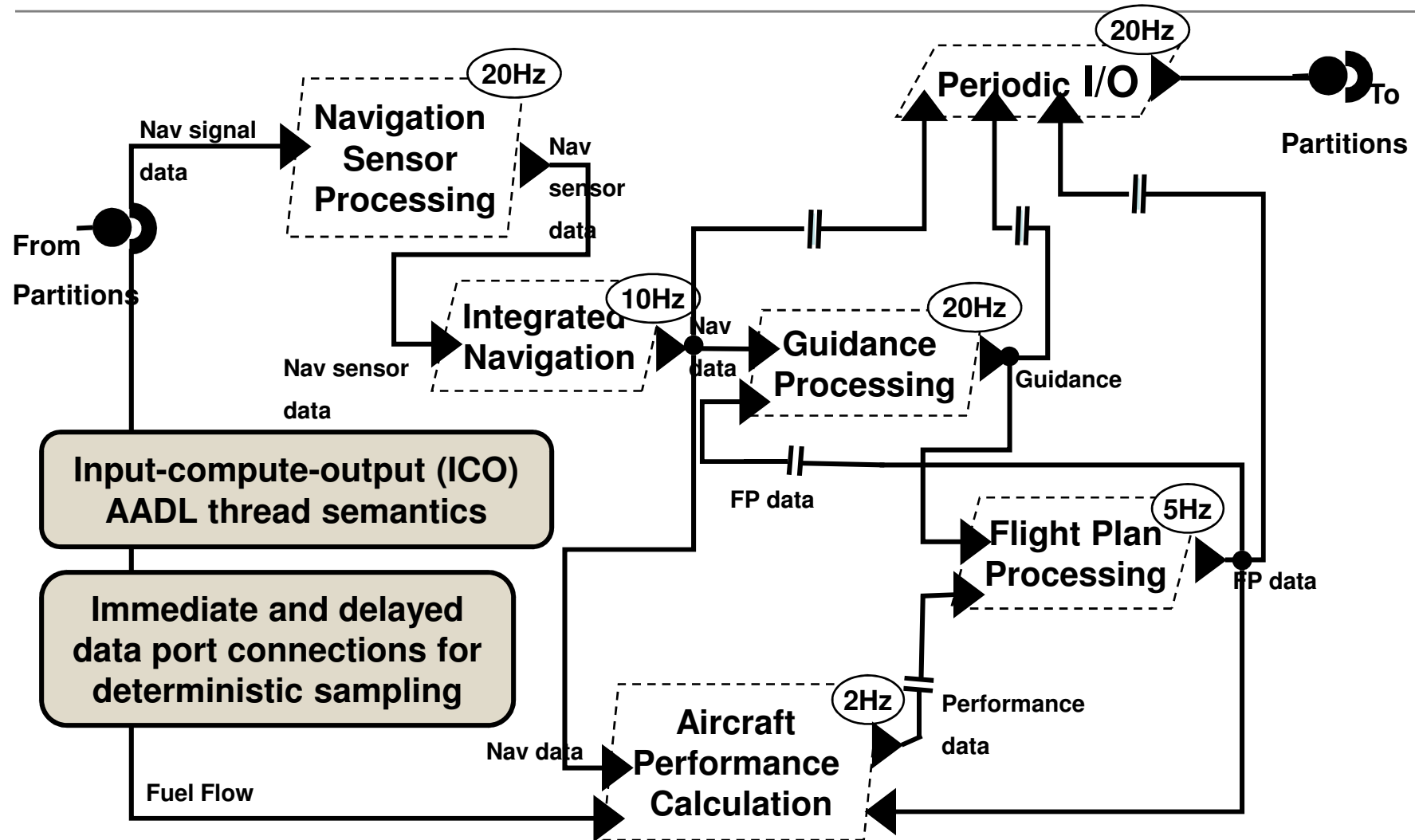
- Subsystem latency exceeds expected latency**: This callout points to a specific error in the Problems window.
- Lower bound latency inherent to partition architecture**: This callout points to the overall flow graph structure.

The Problems window at the bottom shows the following error:

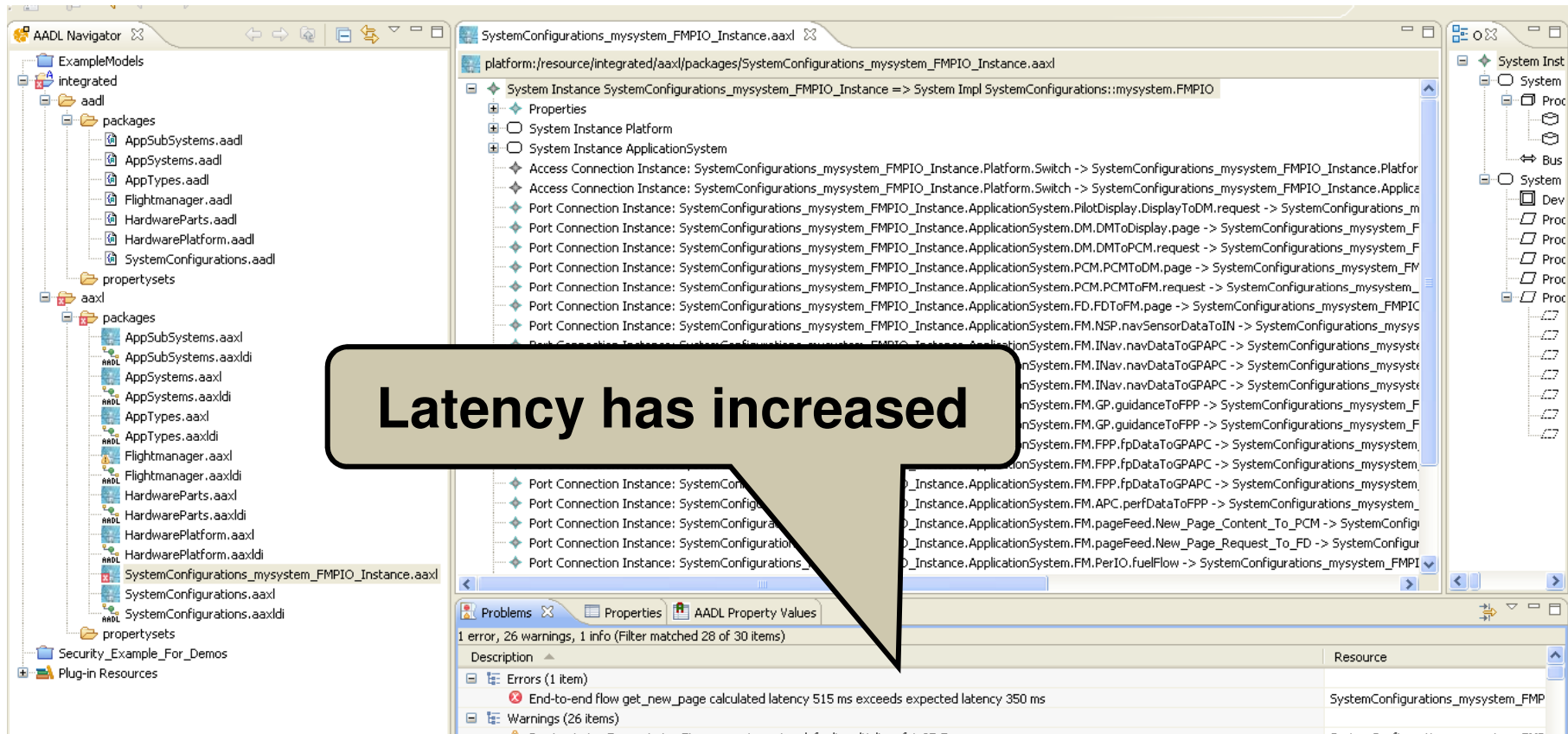
Description	Resource	Path
<b>Errors (1 item)</b>		
End-to-end flow get_new_page latency 415 ms exceeds specified latency 350 ms	SystemConfigurations_mysystem_cmdf...	integrate
<b>Infos (1 item)</b>		
Expected end-to-end flow get_new_page latency is 350 ms	SystemConfigurations_mysystem_cmdf...	integrate



# Managed Latency Jitter through Deterministic Sampling



# Latency Revisited



**Latency has increased**

1 error, 26 warnings, 1 info (Filter matched 28 of 30 items)

Description	Resource
End-to-end flow get_new_page calculated latency 515 ms exceeds expected latency 350 ms	SystemConfigurations_mysystem_FMP
Bus Injection Transmission Time property uses default multiplier of 1.0E-5	SystemConfigurations_mysystem_FMP



# 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**



# Double Buffering

---

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.”*

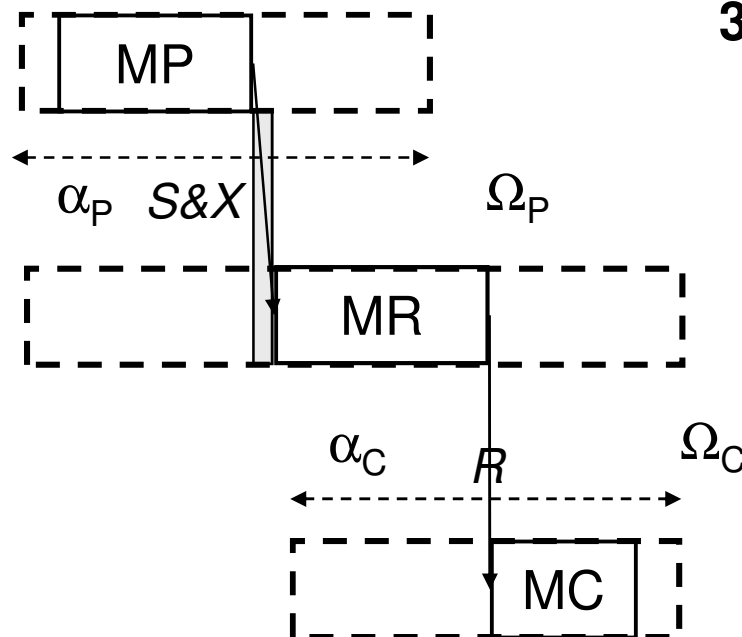
Did double buffering solved the problem  
or do we need to do more buffering?



# 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

$\alpha_P - \Omega_P \cap \alpha_C - \Omega_C \neq \emptyset \Rightarrow$  non-deterministic sampling (S/R) order



# Periodic Task Communication Summary

Periodic Same period	ASR		DSR		DMT
	IMT   PMT		IMT   PMT		
$\tau_P ; \tau_C$	MF:1B	PD:2B SvXvR	PD:2B R	PD:2B SvX/R	MF:1B
$\tau_C ; \tau_P$	PD:1B	PD:1B	PD:1B	PD:1B	PD:1B
$\tau_P \neq \tau_C$	ND:1B	PD:2B X	PD:2B R	PD:2B X/R	ND:1B
$\tau_P   \tau_C$	ND:3B S/X <sub>C</sub> R <sub>C</sub>	PD:2B X	PD:2B R	PD:2B X/R	NDI:2B S/X/R <sub>C</sub>

**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**  
**o1vo2: One operation copy**



# Dual Redundant Flight Guidance System

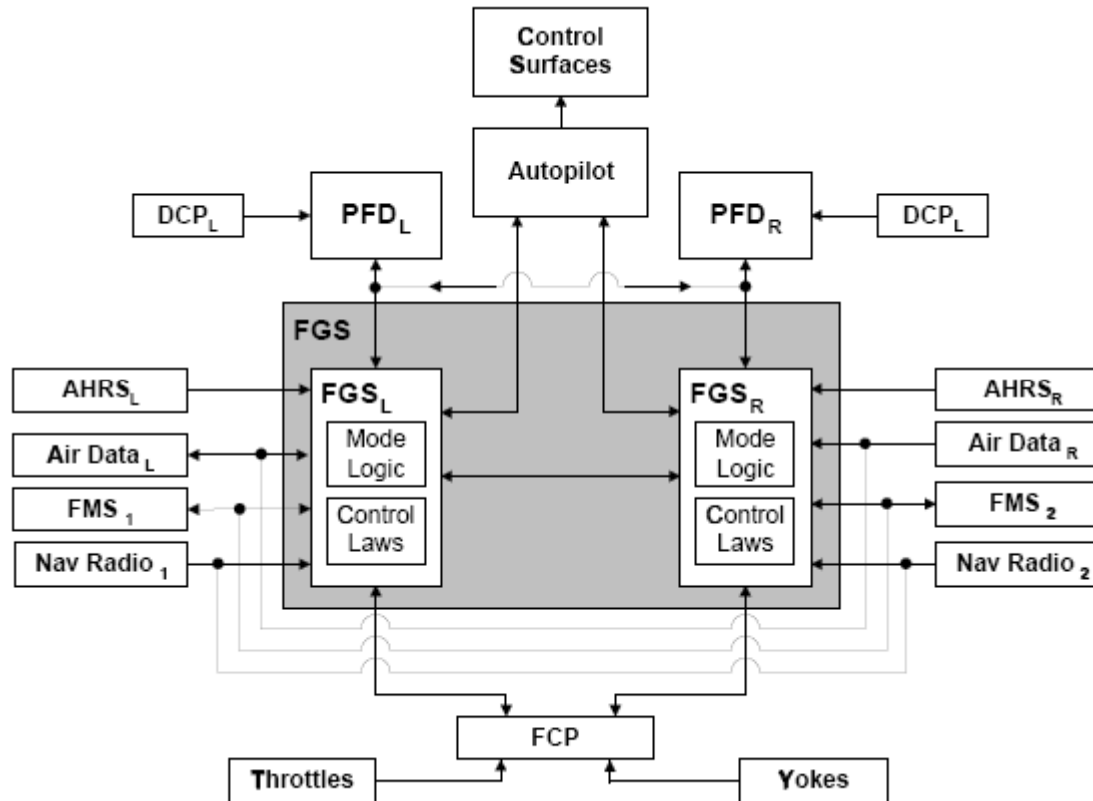
NASA/CR-2005-213912



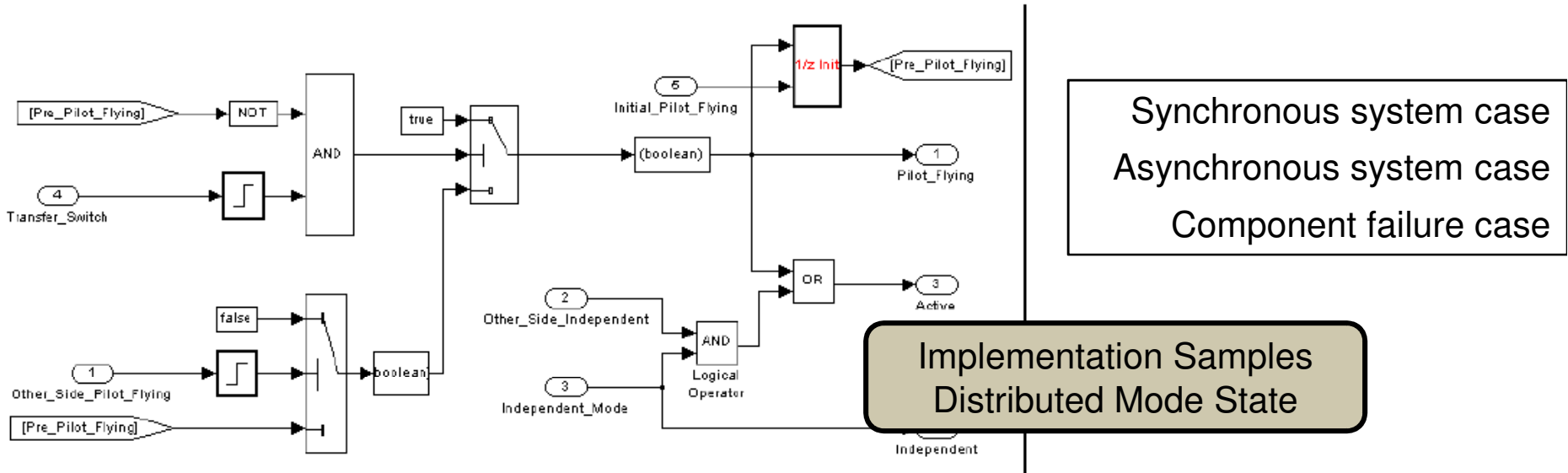
A Methodology for the Design and Verification of Globally Asynchronous/Locally Synchronous Architectures

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

*Dan O'Brien, Mats P. Heimdahl, and Anjali Joshi  
University of Minnesota, Minneapolis, Minnesota*



# Mode Logic Specification



- To validate:
- 1) At least one output
  - 2) Exactly one output
  - 3) Two outputs in critical mode

Property2 := (Left\_FGS\_Pilot\_Flying = !Right\_FGS\_Pilot\_Flying);

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

```
AG( ((!Property2 & Left_FGS_Clock) ->
      (A [!LR_Channel_Clock U (Property2 |
        (AX A [!Right_FGS_Clock U Property2]))])) |
      ((!Property2 & Right_FGS_Clock) ->
        (A [!RL_Channel_Clock U (Property2 |
          (AX A [!Left_FGS_Clock U Property2]))])) );
```

Increased complexity of property

# Modeling & Validation Issues

Modeling in AADL helped identify issues

## Observation of events by sampling state

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

Button input as event to data port

## Distributed processing of mode state machine

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

Reduced property complexity by mode transition as state

## Clock drift in asynchronous system

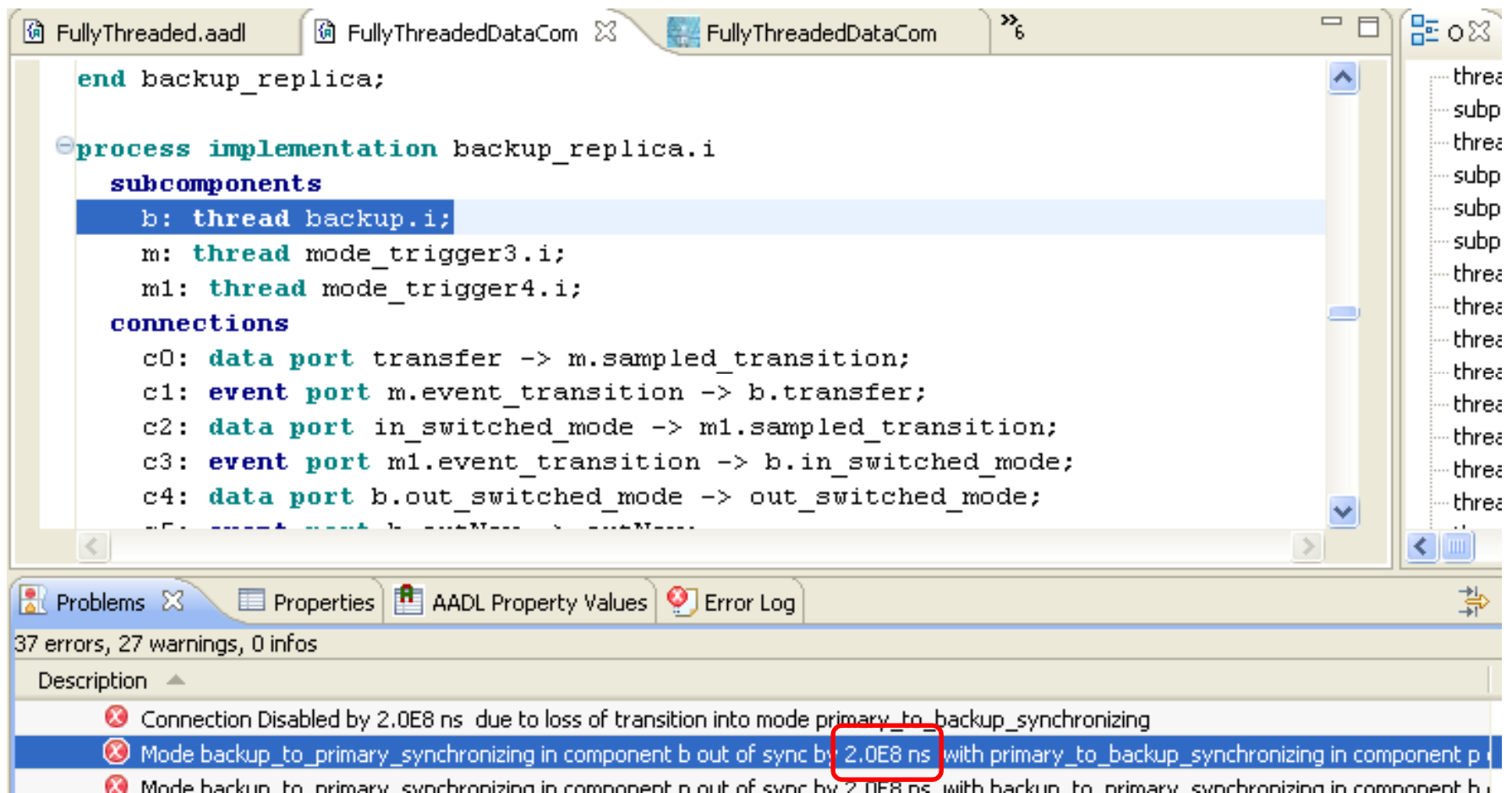
- Acceptable drift: bounded vs. fault
- Built-in drift bound through period: period wrap-around
- Loss of data stream element due to wrap around

Synchronization domains & fault conditions

Event processing vs. data sampling



# Quantified Out-of-sync Modes



```

end backup_replica;

process implementation backup_replica.i
  subcomponents
    b: thread backup.i;
    m: thread mode_trigger3.i;
    m1: thread mode_trigger4.i;
  connections
    c0: data port transfer -> m.sampled_transition;
    c1: event port m.event_transition -> b.transfer;
    c2: data port in_switched_mode -> m1.sampled_transition;
    c3: event port m1.event_transition -> b.in_switched_mode;
    c4: data port b.out_switched_mode -> out_switched_mode;
  end connections
end backup_replica.i
  
```

Problems 37 errors, 27 warnings, 0 infos

- Connection Disabled by 2.0E8 ns due to loss of transition into mode primary\_to\_backup\_synchronizing
- Mode backup\_to\_primary\_synchronizing in component b out of sync by 2.0E8 ns with primary\_to\_backup\_synchronizing in component p
- Mode backup\_to\_primary\_synchronizing in component p out of sync by 2.0E8 ns with backup\_to\_primary\_synchronizing in component b



# Subtle Errors – LM Aero UAV Sensor Voting

## OFP Triplex Voter

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

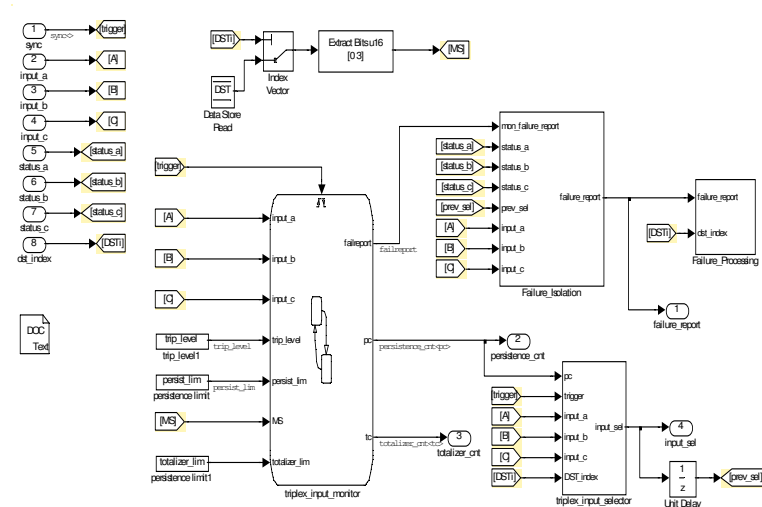
## Formal Verification

- 25 Informal Requirements
- 57 Formal Properties

**$10^{100+}$  reachable states  
in Rockwell examples**

## 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





**Software Engineering Institute**

**Carnegie Mellon**

Peter H Feiler

[phf@sei.cmu.edu](mailto:phf@sei.cmu.edu)