Analytical Architecture Fault Models

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Peter H. Feiler
Dec 4, 2012

© 2012 Carnegie Mellon University

Copyright 2012 Carnegie Mellon University and IEEE

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

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.

This material has been approved for public release and unlimited 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-0000087

Outline

Challenges in Safety-critical Software-intensive systems

SAE AADL and the Error Model Annex

Hazards & Fault Impact Analysis

Compositional Analysis

Operational and Failure Modes

The Intricacies of Desired Timing Behavior

Summary and Conclusion

© 2012 Carnegie Mellon University

We Rely on Software for Safe Aircraft Operation

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.''

Embedded software systems introduce a new class of problems not addressed by traditional system modeling & analysis

Quantas
Landing

Written by htbw

From: soyawan

mayday call when it suddenly changed altitude during a flight from Singapore to Perth, Qantas said.

lunge

lwide

irways

Ltd. flight ... causing the jet to nosedive.

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

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.

© 2012 Carnegie Mellon University

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.

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.

Expert • Independent • Nonprofit

ConsumerReports.org

This article appeared in May 2010 Consumer Reports Magazine.

Many appliances now rely on electronic controls and operating softw ... 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.

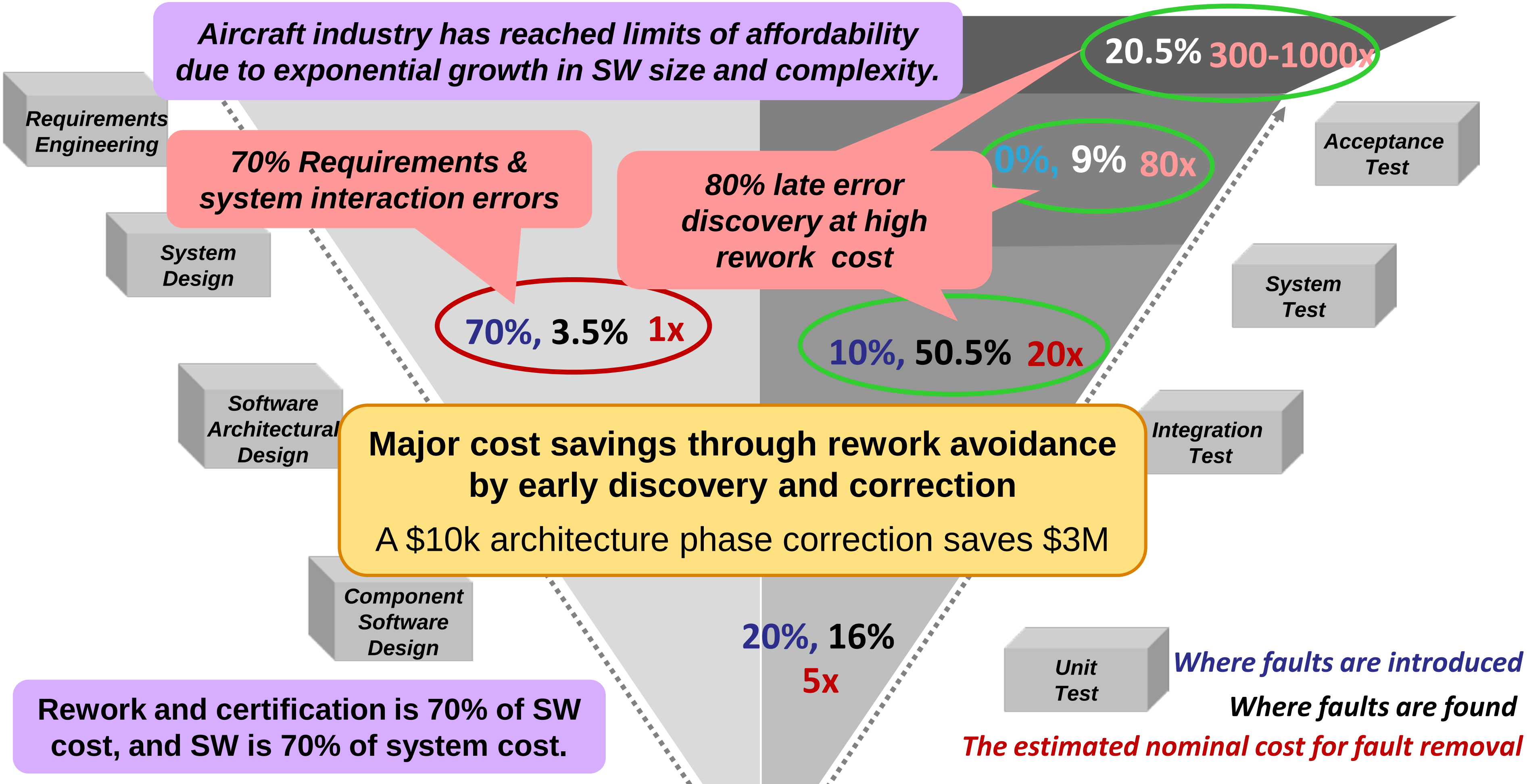
Our retests of the reprogrammed Kenmore 4027 found that the cycles now worked properly, and the machine excelled. It now tops our Ratings (available to subscribers) of more than 50 front-loaders and we've made it a CR Best Buy.

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?

© 2012 Carnegie Mellon University

High Fault Leakage Drives Major Increase in Rework Cost



<u>Sources</u>:

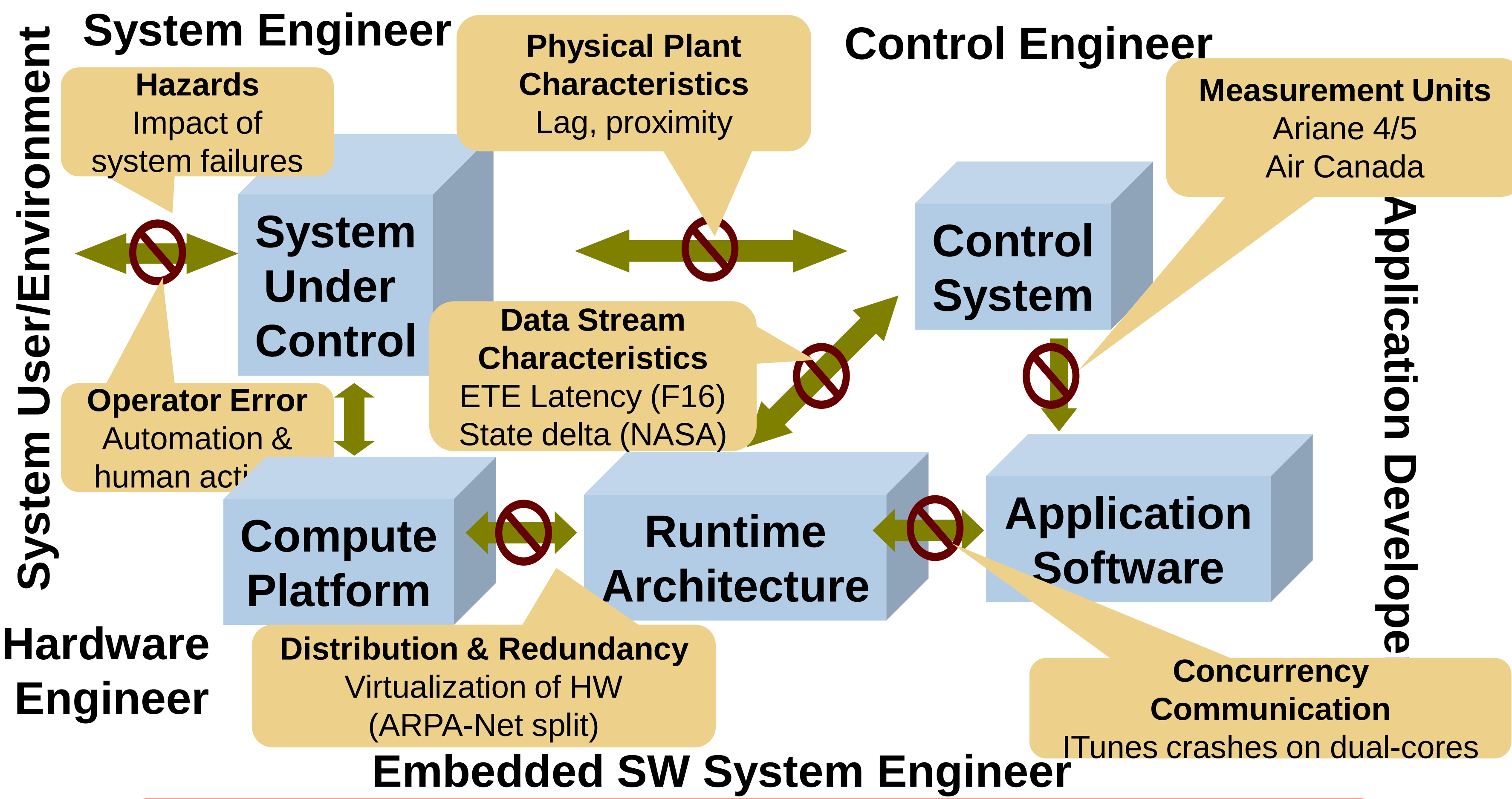
NIST Planning report 02-3, *The Economic Impacts of Inadequate Infrastructure for Software Testing*, May 2002.

D. Galin, *Software Quality Assurance: From Theory to Implementation*, Pearson/Addison-Wesley (2004)

B.W. Boehm, *Software Engineering Economics*, Prentice Hall (1981)

© 2012 Carnegie Mellon University

Mismatched Assumptions in Embedded SW



Why do system level failures still occur despite fault tolerance techniques being deployed in systems?

© 2012 Carnegie Mellon University

Outline

Challenges in Safety-critical Software-intensive systems

SAE AADL and the Error Model Annex

Hazards & Fault Impact Analysis

Compositional Analysis

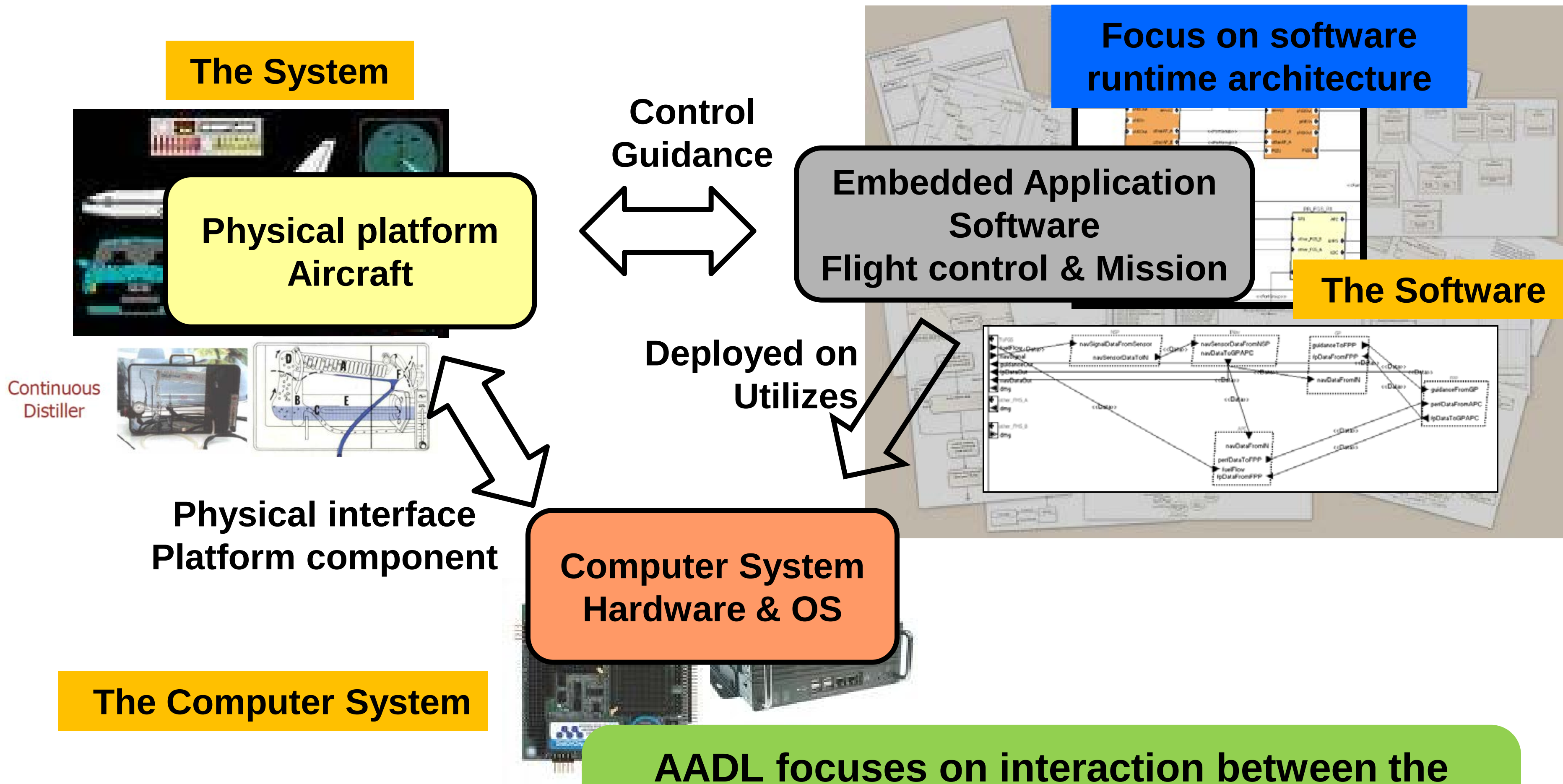
Operational and Failure Modes

The Intricacies of Desired Timing Behavior

Summary and Conclusion

© 2012 Carnegie Mellon University

SAE Architecture Analysis & Design Language (AADL) for Software-reliant Systems



AADL focuses on interaction between the three elements of a software-intensive system based on architectural abstractions of each.

© 2012 Carnegie Mellon University

System Level Fault Root Causes

Violation of data stream assumptions

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

End-to-end latency analysis
Port connection consistency

Partitions as Isolation Regions

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

Partitioned architecture models
Model compliance

Virtualization of time & resources

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

Virtual processors & buses
Synchronization domains

Inconsistent System States & Interactions

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

Fault propagation
Security analysis
Architectural redundancy patterns

Performance impedance mismatches

- Processor, memory & network resources
- Compositional & replacement performance mismatches
- Unmanaged computer system resources

Resource budget analysis & task roll-up analysis
Resource allocation & deployment configurations

© 2012 Carnegie Mellon University

Architecture-Centric Modeling Approach

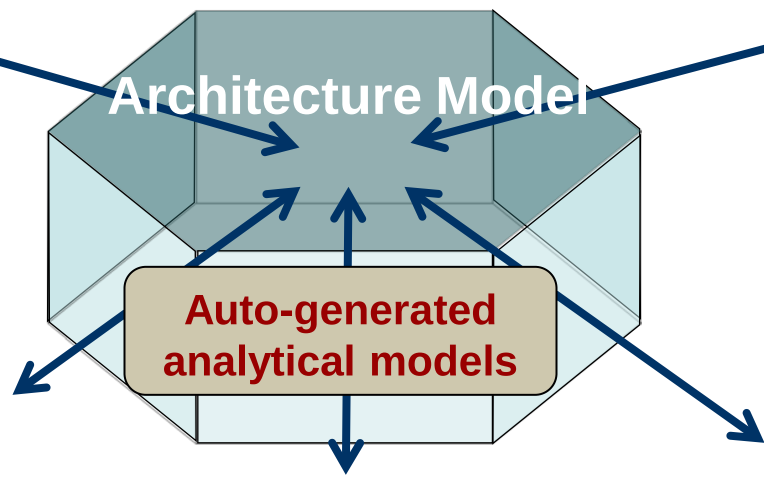
Single Annotated Architecture Model Addresses Impact Across Non-Functional Properties

Safety & 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

Resource Consumption

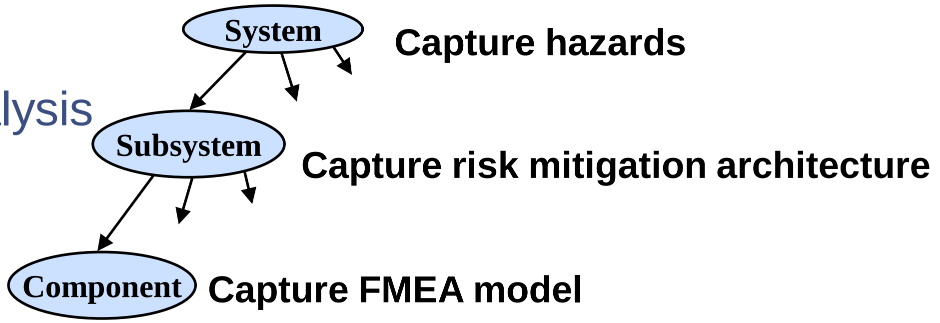
- Bandwidth
- CPU time
- Power consumption

© 2012 Carnegie Mellon University

AADL Error Model Scope and Purpose

System safety process uses many individual methods and analyses, e.g.

- hazard analysis
- failure modes and effects analysis
- fault trees
- Markov processes



SAE ARP 4761 *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*

Related analyses are also useful for other purposes, e.g.

- maintainability
- availability
- Integrity

Goal: a general facility for modeling fault/error/failure behaviors that can be used for several modeling and analysis activities.

Annotated architecture model permits checking for consistency and completeness between these various declarations.

© 2012 Carnegie Mellon University

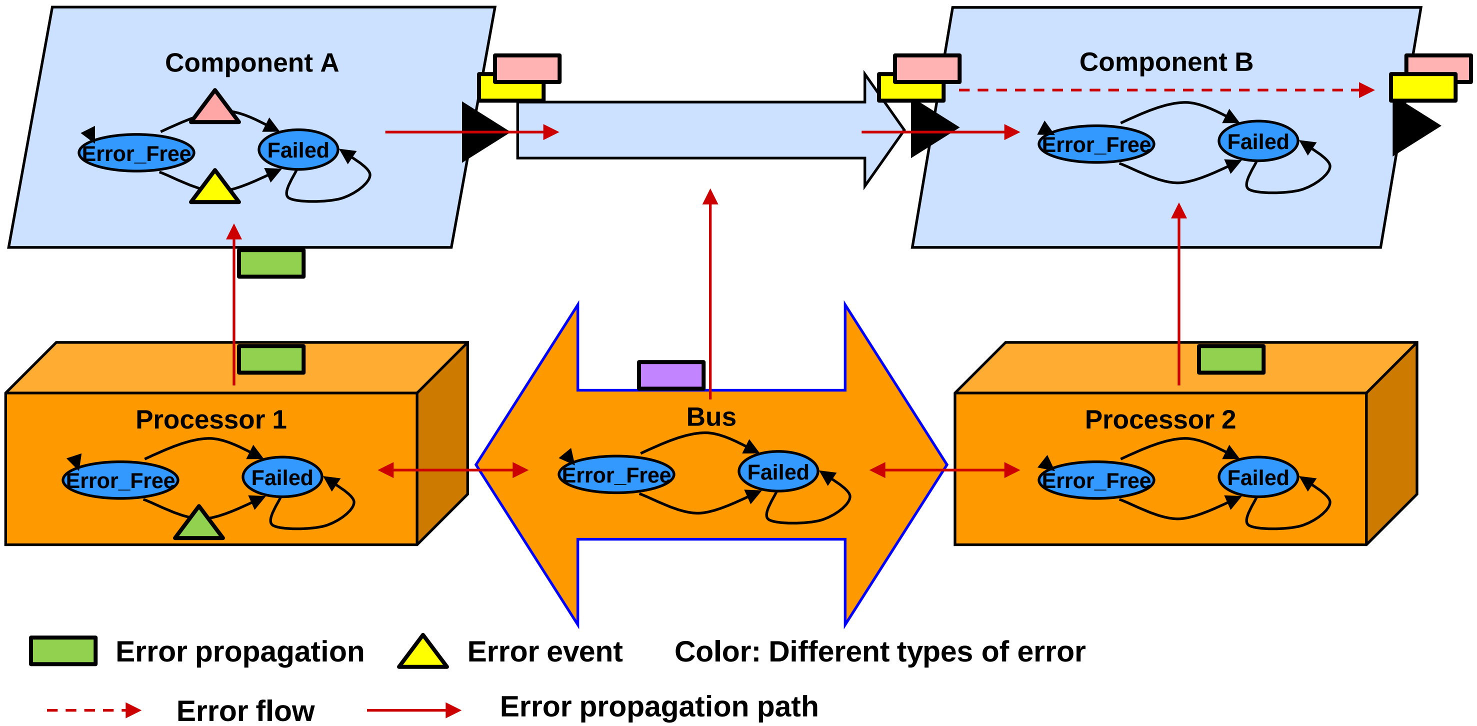
Error Model and the Architecture

Propagation of *errors* of different *types* from *error sources* along *propagation paths* between architecture components.

Error flows as abstractions of propagation through components.

Component *error behavior* as *transitions* between *states* triggered by *error events* and *incoming propagations*.

Composite error behavior in terms of component error behavior states.

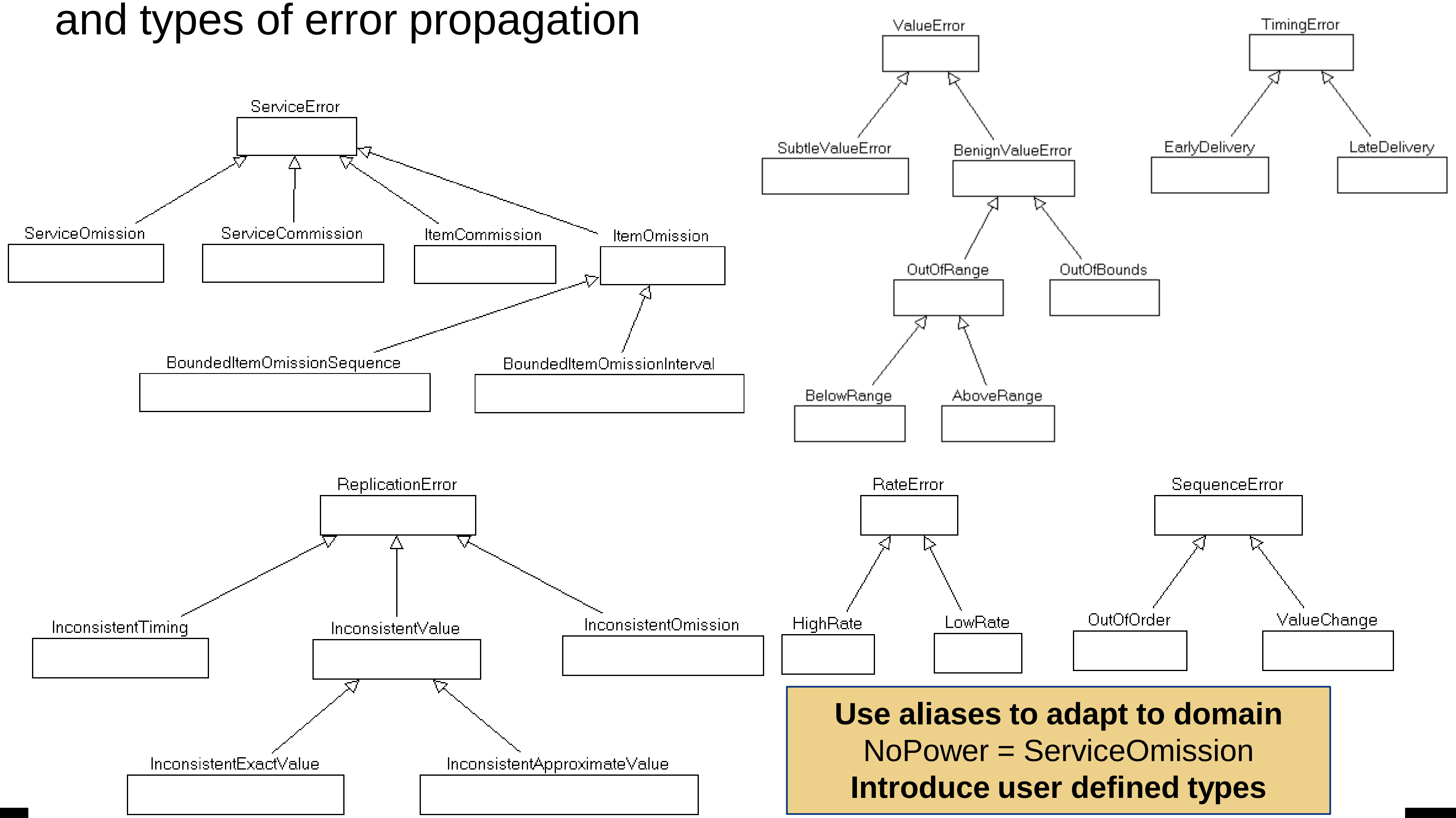


© 2012 Carnegie Mellon University

A Common Set of Error Propagation Types

Independent hierarchies of error types

Can be combined to characterize failure modes, resulting error states, and types of error propagation



Use aliases to adapt to domain
NoPower = ServiceOmission
Introduce user defined types

© 2012 Carnegie Mellon University

Formal Error Type Specifications

Service errors with respect to the service as a whole rather than individual service items

- *Service Omission* is perceived as a permanent fault in that no service items are provided after the point of failure.
  Service Omission: $\exists\ s_i \in S' \subseteq S:\ (st_i = \infty)$
- *Service Commission* is perceived as an impromptu service in that service items are provided before the point service is expected.
  Service Commission: $\exists\ si \in S' \supseteq S:\ (\ st_i \notin ST\ )$
- Other forms of service error can be defined: early service start, late service start, early service termination, late service termination.

Value errors with respect to the value of an individual service item

- *Out Of Range* error indicating that the value is outside the expected range of values, a detectable error.
  Out Of Range error: $s_i:\ sv_i \notin SV$

© 2012 Carnegie Mellon University

Error Types & Error Type Sets

Error type declarations

```
ServiceError: type ;
Omission: type extends ServiceError;
Commission: type extends ServiceError;
Early: type extends TimingError ;
Late: type extends TimingError ;
```

Error Type Set as Constraint
{T1, T2} power set of two error type hierarchies
{T1} error type set of one type
{T1+T2} product type (one error type from each error type hierarchy)
{T1,*} error type set with at least one element from type (hierarchy) T1
{**NoError**} represents the empty set

An *error type set* represents the combination of error types that can occur or be propagated simultaneously.

- An error type set is defined as the product of error types.
- Example: an error propagation may involve both a late value and an incorrect value.

```
InputOutputError : type set {TimingError, ValueError};
StreamError : type set {TimingError, ValueError, SequenceError,
RateError};
```

An *error tuple* represents a typed token instance

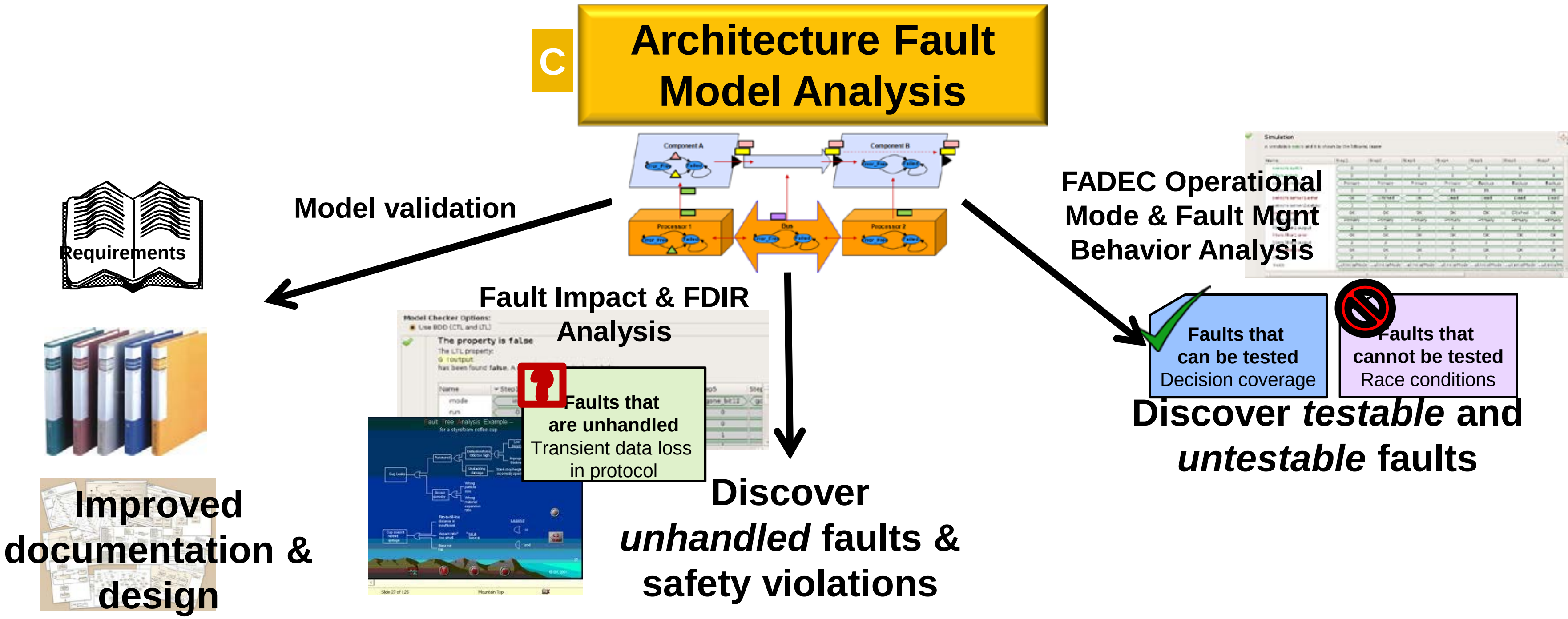
- Represents actual event, propagation, or state types

```
{LateValue + BadValue} or {LateValue}
```


© 2012 Carnegie Mellon University

Analyzable Architecture Fault Models

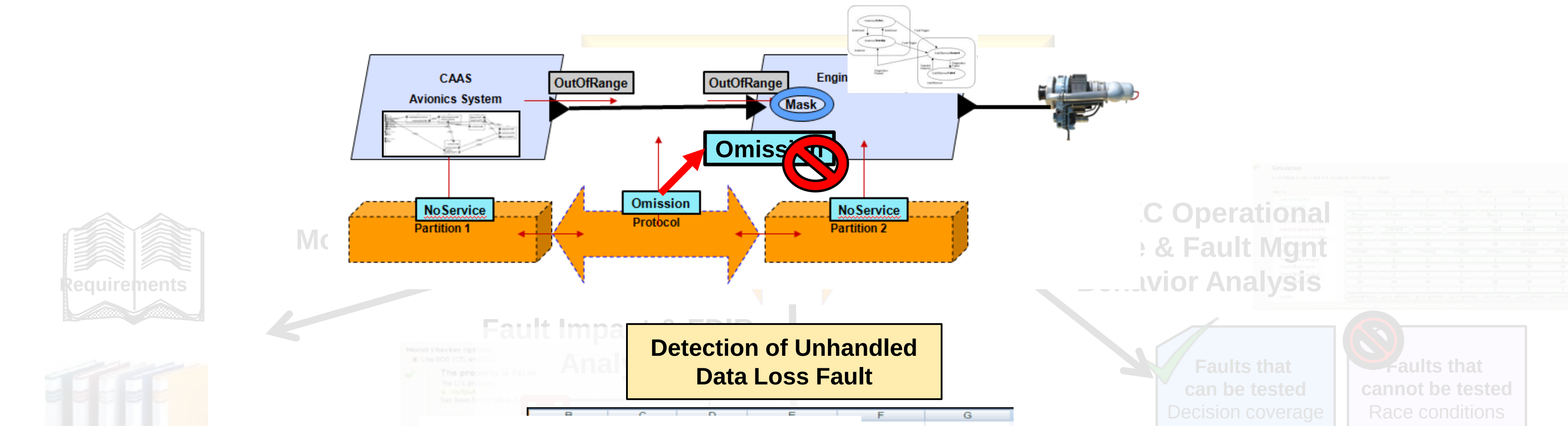
Through model-based analysis identify architecture induced unhandled, testable, and untestable faults and understand the root cause, impact, and potential mitigation options.



© 2012 Carnegie Mellon University

Analyzable Architecture Fault Models

Through model-based analysis identify architecture induced unhandled, testable, and untestable faults and understand the root cause, impact, and potential mitigation options.

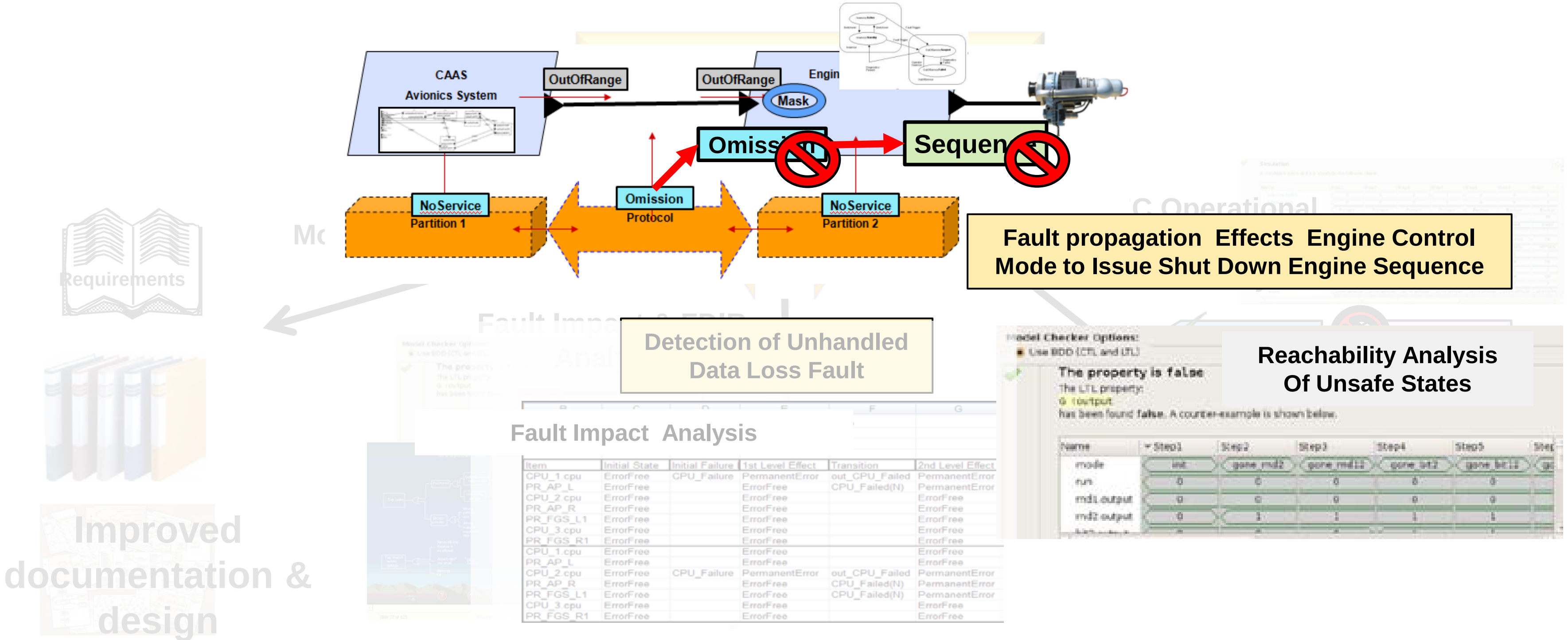


Item	Initial State	Initial Failure	1st Level Effect	Transition	2nd Level Effect
CPU_1.cpu	ErrorFree	CPU_Failure	PermanentError	out_CPU_Failed	PermanentError
PR_AP_L	ErrorFree		ErrorFree	CPU_Failed(N)	PermanentError
CPU_2.cpu	ErrorFree		ErrorFree		ErrorFree
PR_AP_R	ErrorFree		ErrorFree		ErrorFree
PR_FGS_L1	ErrorFree		ErrorFree		ErrorFree
CPU_3.cpu	ErrorFree		ErrorFree		ErrorFree
PR_FGS_R1	ErrorFree		ErrorFree		ErrorFree
CPU_1.cpu	ErrorFree		ErrorFree		ErrorFree
PR_AP_L	ErrorFree		ErrorFree		ErrorFree
CPU_2.cpu	ErrorFree	CPU_Failure	PermanentError	out_CPU_Failed	PermanentError
PR_AP_R	ErrorFree		ErrorFree	CPU_Failed(N)	PermanentError
PR_FGS_L1	ErrorFree		ErrorFree	CPU_Failed(N)	PermanentError
CPU_3.cpu	ErrorFree		ErrorFree		ErrorFree
PR_FGS_R1	ErrorFree		ErrorFree		ErrorFree

© 2012 Carnegie Mellon University

Analyzable Architecture Fault Models

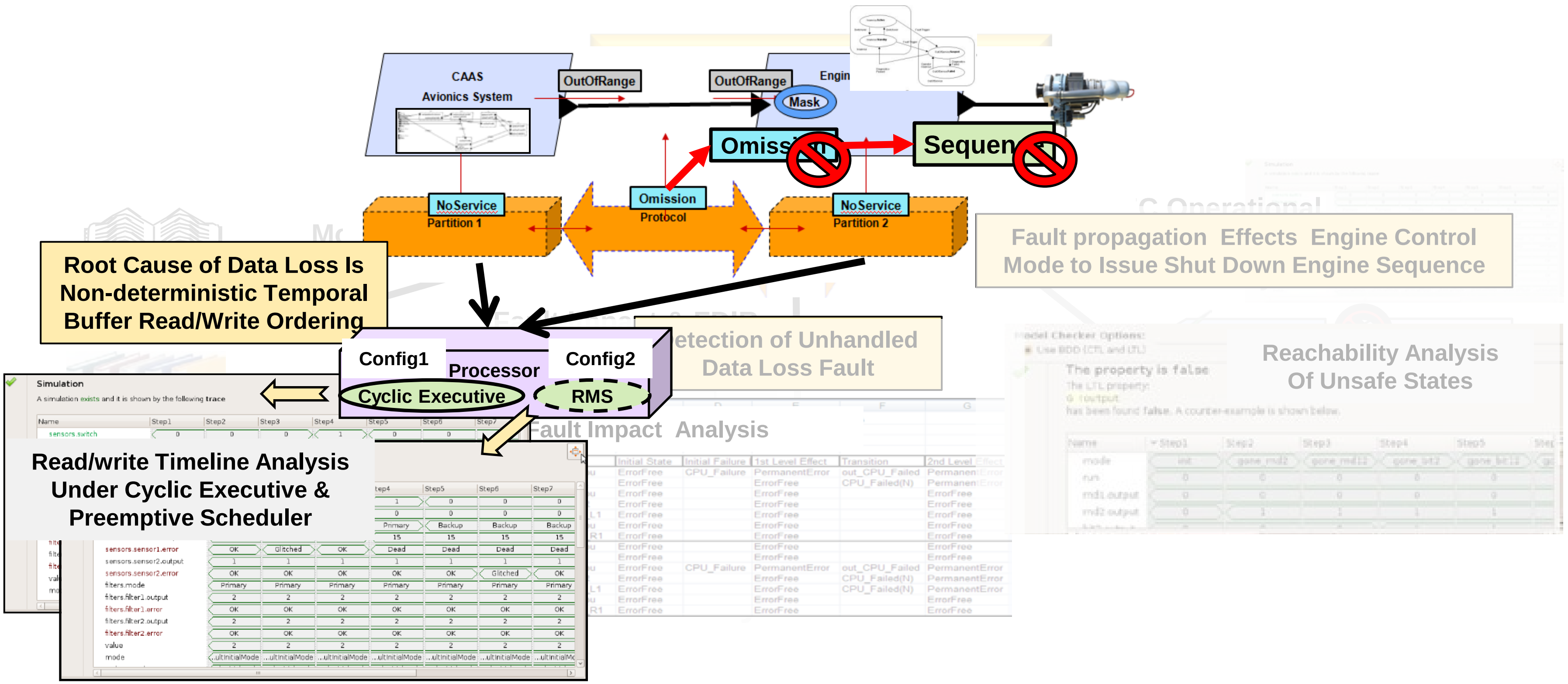
Through model-based analysis identify architecture induced unhandled, testable, and untestable faults and understand the root cause, impact, and potential mitigation options.



© 2012 Carnegie Mellon University

Analyzable Architecture Fault Models

Through model-based analysis identify architecture induced unhandled, testable, and untestable faults and understand the root cause, impact, and potential mitigation options.



© 2012 Carnegie Mellon University

Outline

Challenges in Safety-critical Software-intensive systems

SAE AADL and the Error Model Annex

Hazards & Fault Impact Analysis

Compositional Analysis

Operational and Failure Modes

The Intricacies of Desired Timing Behavior

Summary and Conclusion

© 2012 Carnegie Mellon University

Hazard Information in AADL Error Model

Hazards modeled as error propagations in the AADL Error Model

```
error model HazardList
features
  Guidance_Loss: out error propagation {hazard =>
   (reference => "1.1.1",
    failure => "Loss of guidance values",
    phase => "approach",
    description => "Presence of no computed data should signal FD and AP disconnect.",
    criticality => "minor",
    comment => "Becomes major hazard, equivalent to incorrect guidance, if disconnect fails.")};

  Guidance_Incorrect: out error propagation {hazard =>
   (reference => "1.1.2",
    failure => "Incorrect guidance values",
    phase => "approach",
    description => "Gradual departure from references until detected by flight crew during check of primary flight data resulting in manual disconnect and
    criticality => "major",
    comment => "No difference to the AP between loss of guidance and incorrect guid

  Transfer_Control_Loss: out error propagation {hazard =>
   (reference => "4.1.1",
    failure => "Loss of transfer control of flight guidance data to AP",
    phase => "all",
    description => "Flight crew unable to change 'Pilot Flying' side FGS. Manual disconn
    criticality => "minor",
    comment => "-")};
```

```
  SEI::PowerBudget => 2000.0 W applies to FG_Power.Backbone;
  -- Select components for inclusion in FHA
  Safety::doFHA => true applies to PR_FGS_L1;
  --Safety::doFHA => true applies to PR_FGS_R1;
 annex Error_Model {**
  model => FGSHazards::HazardList applies to PR_FGS_L1;
  model => FGSHazards::HazardList applies to PR_FGS_R1;
 **};
end FlightGuidance.subsystems;
```


© 2012 Carnegie Mellon University

Sample FHA in Spreadsheet View

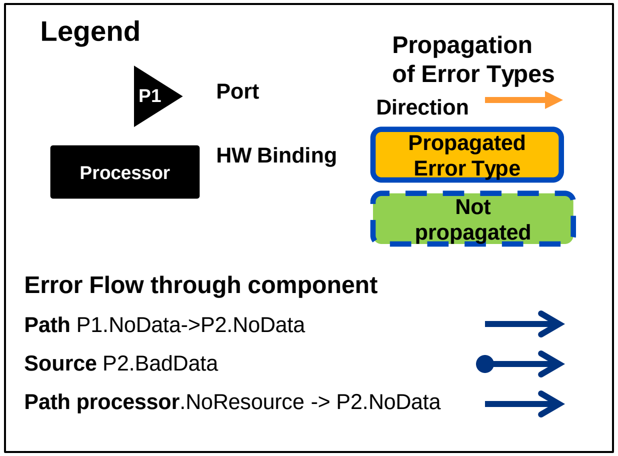
Hazard information exported to spreadsheet format

	A	B	C	D	E	F	G	H
1	Component	Error	Reference	Functional Failure (Hazard)	Critical Operational Phase	Aircraft Manifestation	Criticality	Comment
2	FlightGuidance_FlightGuidance_subsystems_Instance.PR_FGS_L1	Guidance_Loss	1.1.1	Loss of guidance values	approach	Presence of no computed data should signal FD and AP disconnect.	minor	Becomes major hazard, equivalent to incorrect guidance, if disconnect fails.
3		Guidance_Incorrect	1.1.2	Incorrect guidance values	approach	Gradual departure from references until detected by flight crew during check of primary flight data resulting in manual disconnect and manual flying.	major	No difference to the AP between loss of guidance and incorrect guidance.
4		Transfer_Control_Loss	4.1.1	Loss of transfer control of flight guidance data to AP	all	Flight crew unable to change 'Pilot Flying' side FGS. Manual disconnect and manual flying	minor	-

Architecture Fault Model allows hazards to be recorded for use throughout all development phases

© 2012 Carnegie Mellon University

Component Error Propagation

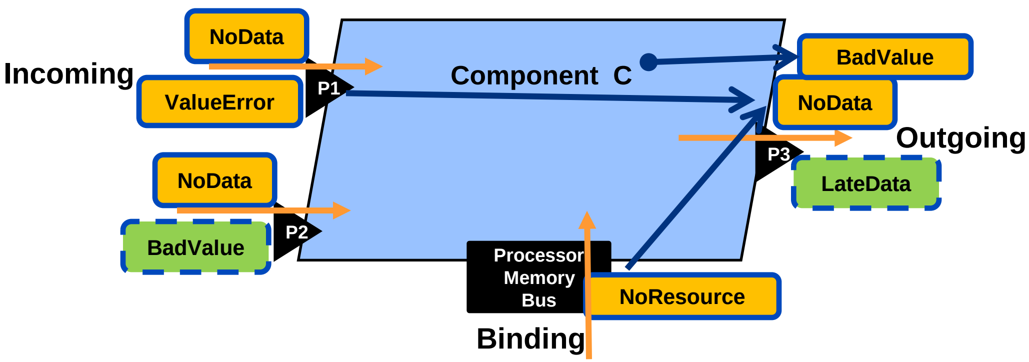


Error Flow:

Path P1.NoData->P3.NoData

Source P2.BadData;

Path processor.NoResource -> P2.NoData



"Not" indicates that this error type is not intended to be propagated.

This allows us to determine whether propagation specification is complete.

Incoming/Assumed

- Propagated errors
- Errors not propagated

Outgoing/Intention

- Propagated errors
- Errors not propagated

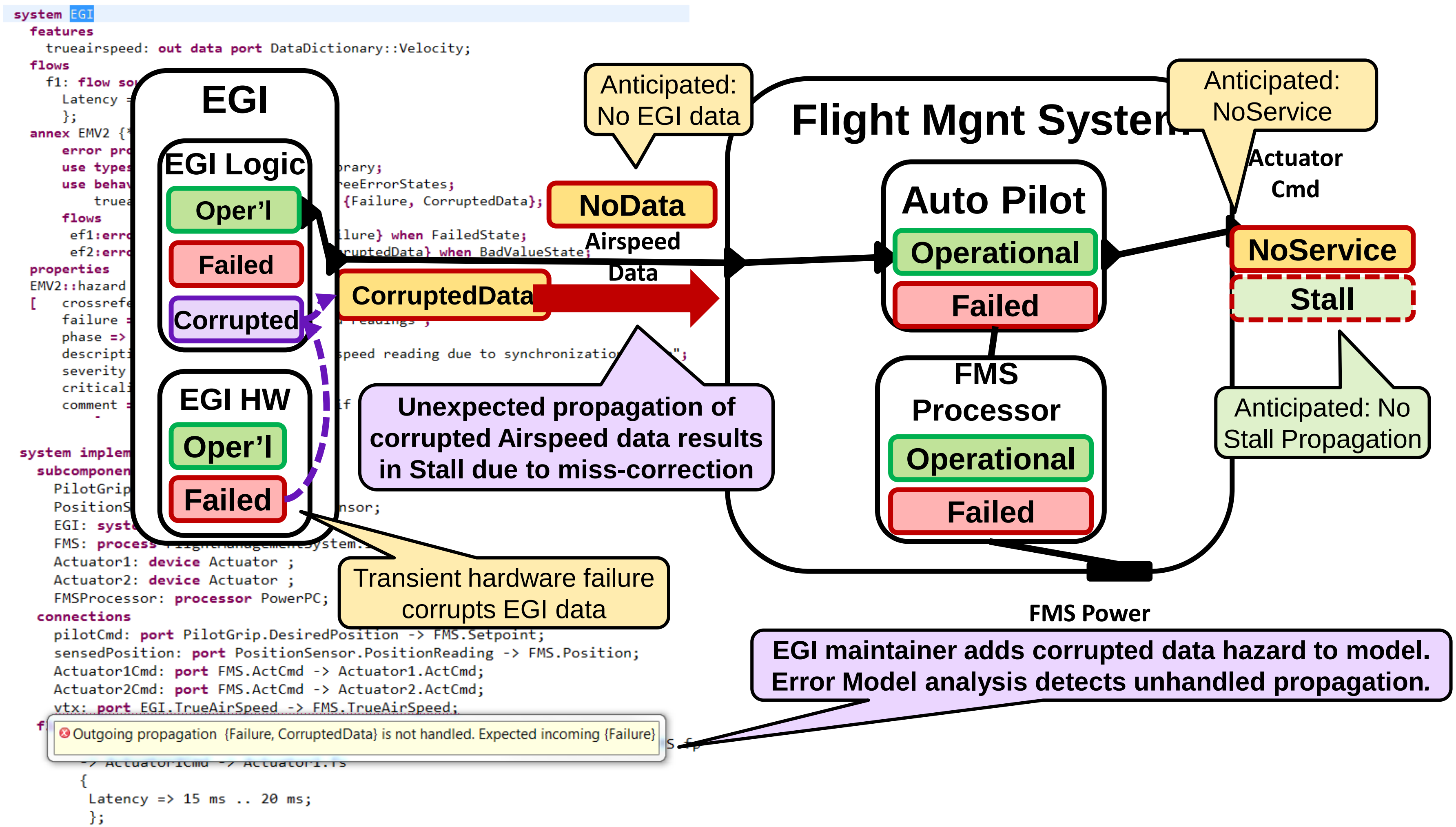
Error propagation and flow specification supports fault impact analysis based on a Fault Propagation and Transformation Calculus (FPTC)

Bound resources

- Propagated errors
- Errors not propagated
- Propagation from/to resource

© 2012 Carnegie Mellon University

Discovery of Unexpected PSSA Hazard



© 2012 Carnegie Mellon University

Recent 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

ID	Item	Initial State	Initial Failure Mode	1st Level Effect	Transition	2nd Level Effect	Transition	3rd Level Effect	Severity	M
1	Sat_Bus	Working	Failure	Failed		Failed	Recovery	Working		Workir
1	Sat_Payload	Working		Working	Bus failure causes payload transition	Standby		Standby	Bus Recovery Causes Payload Transition	Workir
2	Sat_Bus	Working		Working		Working	5			
2	Sat_Payload	Working	Failure	Failed	Recovery	Working	5			

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)

Myron Hecht, Aerospace Corp.
Safety Analysis for JPL, member of DO-178C committee

© 2012 Carnegie Mellon University

Outline

Challenges in Safety-critical Software-intensive systems

SAE AADL and the Error Model Annex

Hazards & Fault Impact Analysis

Compositional Analysis

Operational and Failure Modes

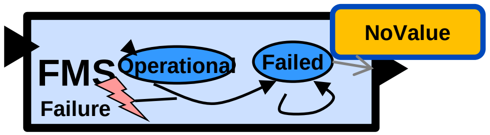
The Intricacies of Desired Timing Behavior

Summary and Conclusion

© 2012 Carnegie Mellon University

Error Model at Each Architecture Level

- Abstracted error behavior of FMS
  - Error behavior and propagation specification



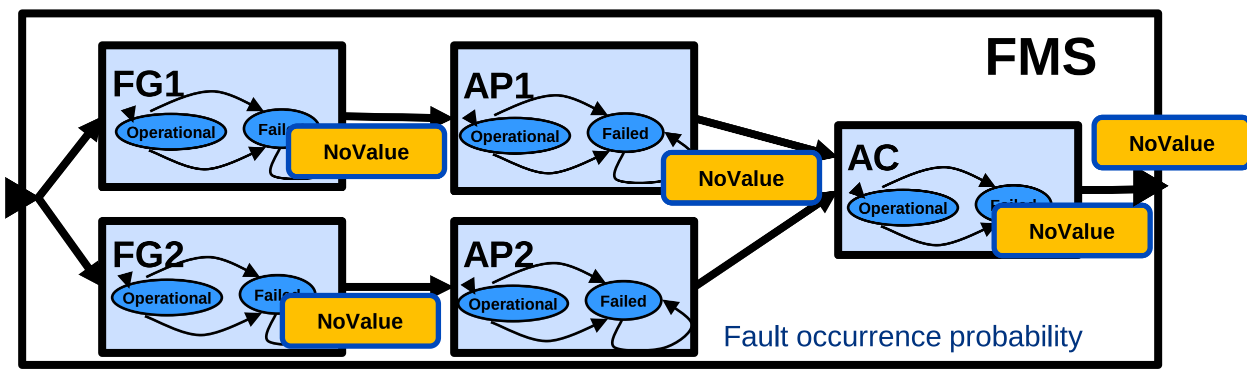
Fault occurrence probability

Composite error models lead to fault trees and reliability predictions

- Composite error behavior specification of FMS
  - State in terms of subcomponent states

[1 **ormore**(FG1.Failed **or** AP1.Failed) **and**

1 **ormore**(FG2.Failed **or** AP2.Failed) **or** AC.Failed]->Failed



Consistency Checking Across Levels of the Hierarchy

© 2012 Carnegie Mellon University

Impact of Deployment Configuration Changes

FMS Failure on 2 or 3 processor configuration (CPU failure rate = 10^{-5})

FMS Failure Rate	0	$5*10^{-6}$	$5*10^{-5}$
MTTF – One CPU operational	112,000	67,000	14,000
MTTF – Two CPU operational	48,000	31,000	7,000

Side effects of design and deployment decisions in reliability predictions

Workload balancing of partitions later in development affects reliability

3 processor configuration can be less reliable than 2 processor configuration

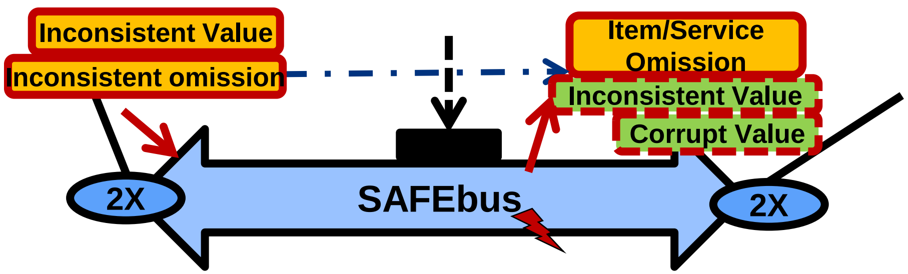
Example: AP and FG distributed across two processors

© 2012 Carnegie Mellon University

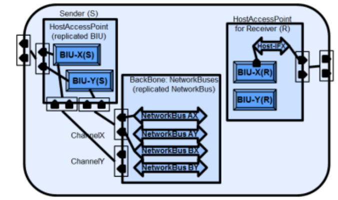
SAFEbus with Dual Communication Channel

- Network is aware of dual host nature
  - Reflected in replication factor properties for connections and bus access
  - Dual channel communication as abstraction reflected in Replication Errors
- Integrity gate keeper for host system
  - Maps inconsistent value/omission errors into item omissions
  - *Expressed by error path from incoming to outgoing binding propagation*

SAFEbus as Application Gate Keeper and Source of Error



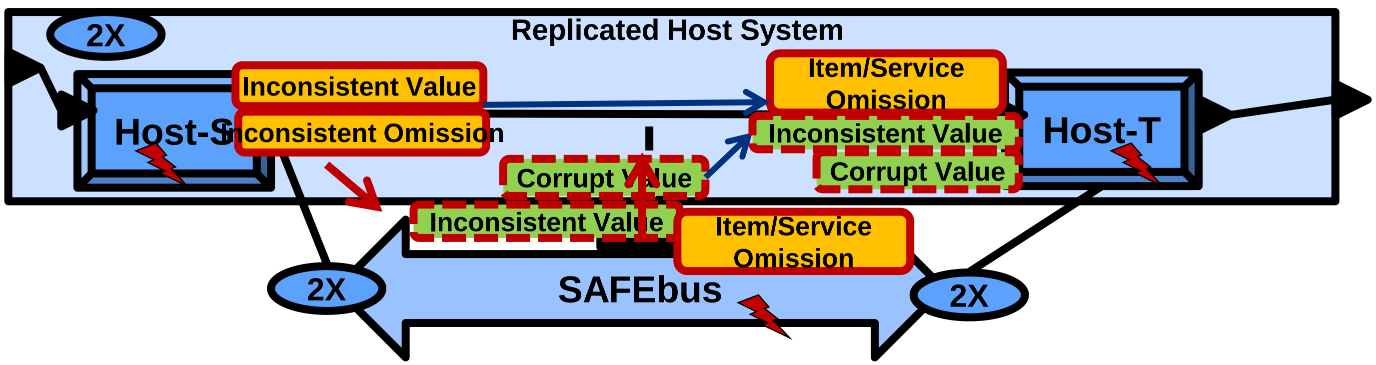
- Fail-op/fail-stop tolerance of SAFEbus faults
  - Operational, SingleErrorOp, Failed states: SAFEbus error events cause state change
  - Operational: gate keeper mappings
  - SingleErrorOp: no error source & full gate keeper mappings
  - Failed: service omission as sole outgoing propagation
  - *Modeled by error behavior state machine and component behavior conditions*



© 2012 Carnegie Mellon University

Does SAFEbus Meet Application Needs?

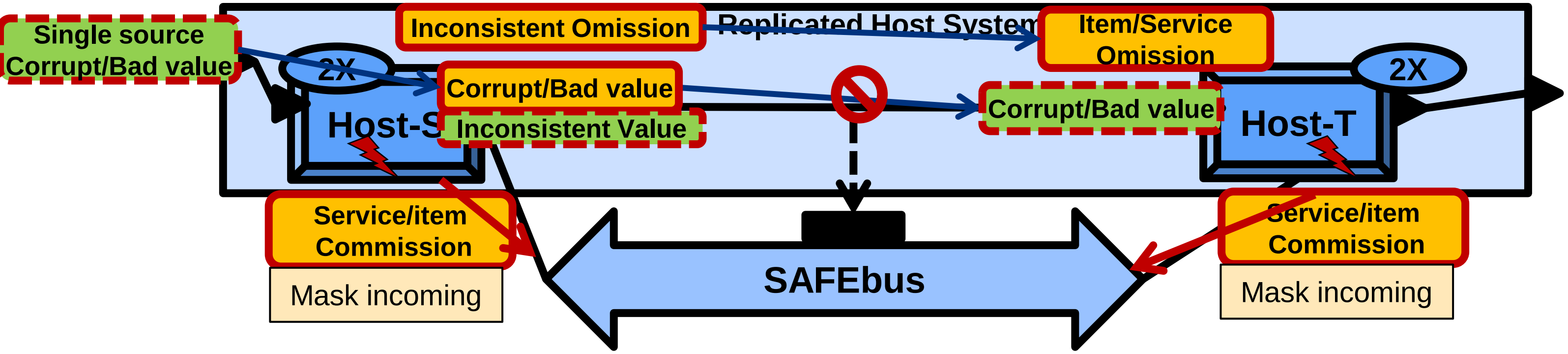
- SAFEbus as Integrity Gate Keeper
  - Transform outgoing propagations to match incoming application integrity assumptions
- SAFEbus communication transport mechanism as error source
  - Map SAFEbus error events into incoming error propagations that must meet integrity assumptions



© 2012 Carnegie Mellon University

Assumptions & Hazards in Use of SAFEbus

- Assumption: no replicated Host stand-by operation
  - Item/service omission on one Host-S channel => Item/service omission for both Host-T channels
  - *Modeled as Inconsistent Omission propagation in stand-by operational mode*
- Assumption: no identical corrupt/bad value propagation from sender
  - Need to ensure no replication of corrupt/bad value (e.g., sensor fan-out)
  - *Modeled as no outgoing symmetric value error from sending host*
- SAFEbus as shared resource
  - Manage babbling host (item/service commission)
  - *Modeled as propagated commission that is expected to be masked*



© 2012 Carnegie Mellon University

Outline

Challenges in Safety-critical Software-intensive systems

SAE AADL and the Error Model Annex

Hazards & Fault Impact Analysis

Compositional Analysis

Operational and Failure Modes

The Intricacies of Desired Timing Behavior

Summary and Conclusion

© 2012 Carnegie Mellon University

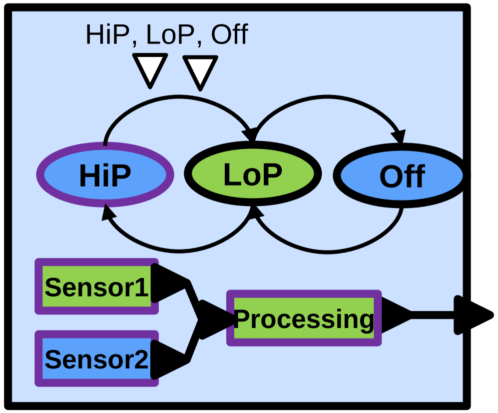
Operational Modes and Failure Modes

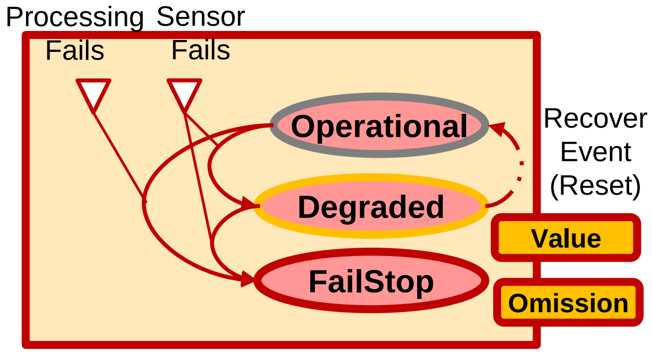
- Nominal system behavior
  - Operational modes
  - Functional behavior
  - Represented by AADL modes and Behavior Annex
- Anomalous behavior reflecting failure modes
  - Deviation from nominal behavior
  - Due to design error
  - Due to physical failure (operational error)
  - Represented by AADL Error Model Annex
- System awareness of anomalous behavior
  - Detection, reporting/recording
  - Transformation, masking
  - Represented by AADL Error Model Annex

© 2012 Carnegie Mellon University

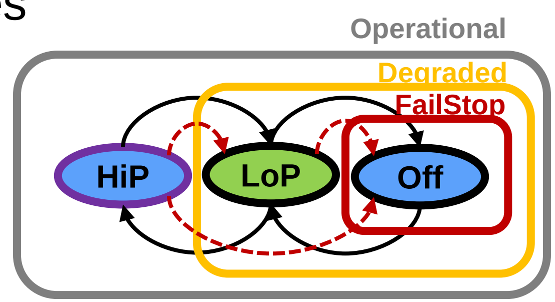
Example: GPS

- Operational modes: Hi-Precision, Lo-Precision, Off
  - User initiated transitions
- Failures: Sensor failure, Processing failure
- Error states of GPS:
  - Dual Sensor Op(operational)
    - Sensor1{NoError} and Sensor2{NoError} and Processing{NoError}
  - Single Sensor Op (Degraded)
    - 1 orless(Sensor1{Failed}, Sensor2{Failed})
  - Dual sensor failure or processing failure (FailStop)
    - 1 ormore(Sensor1{Failed} and Sensor2{Failed}, Processing{Failed})



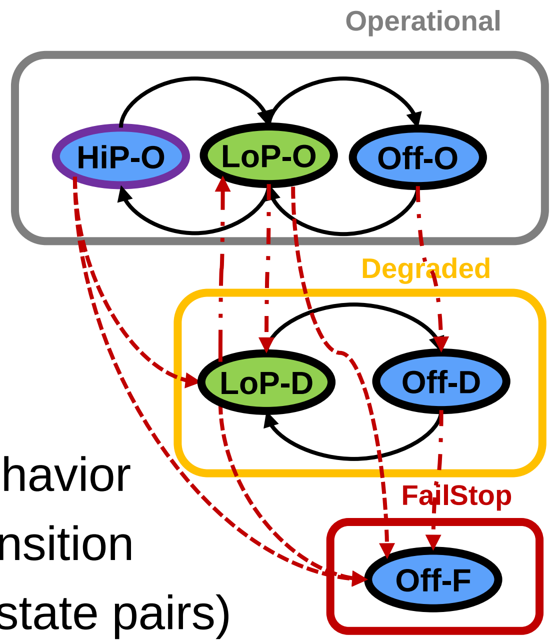


- Mapping of Error States onto Operational Modes



© 2012 Carnegie Mellon University

Combined GPS Behavior Model



- Error states constrain operational modes
  - Degraded supports LoP and Off
  - FailStop shows Off behavior
- Forced mode transitions
  - New error state that excludes current mode
  - Explicit reflection of forced transition in mode behavior
    - Detection event and “Emergency” mode transition
- Behavioral record of error state (Mode state/Error state pairs)
  - Specify detection and reporting of error state
  - Allows for detection and reporting of limits on user initiated transitions

Expected mode	Operational	Degraded	Fail Stop
HiP	HiP	Force transition to LoP {ValueError}	Force transition to Off {Omission}
LoP	LoP	LoP Ok: LoP <-> Off	Force transition to Off {Omission}
Off	Off	Off	Off Perceived {NoError}

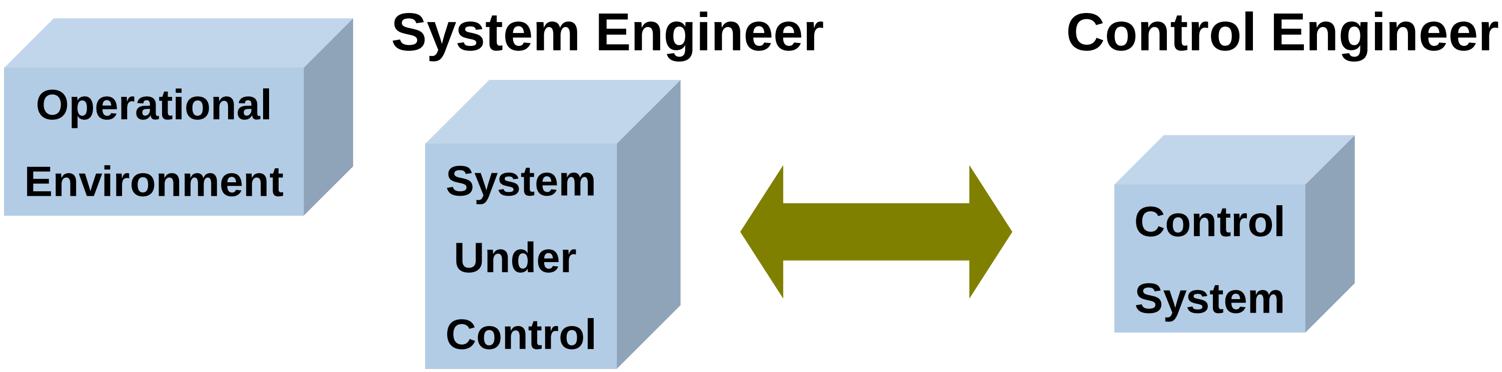
© 2012 Carnegie Mellon University

Outline

Challenges in Safety-critical Software-intensive systems
SAE AADL and the Error Model Annex
Hazards & Fault Impact Analysis
Compositional Analysis
Operational and Failure Modes
The Intricacies of Desired Timing Behavior
Summary and Conclusion

© 2012 Carnegie Mellon University

End-to-end Latency in Control Systems

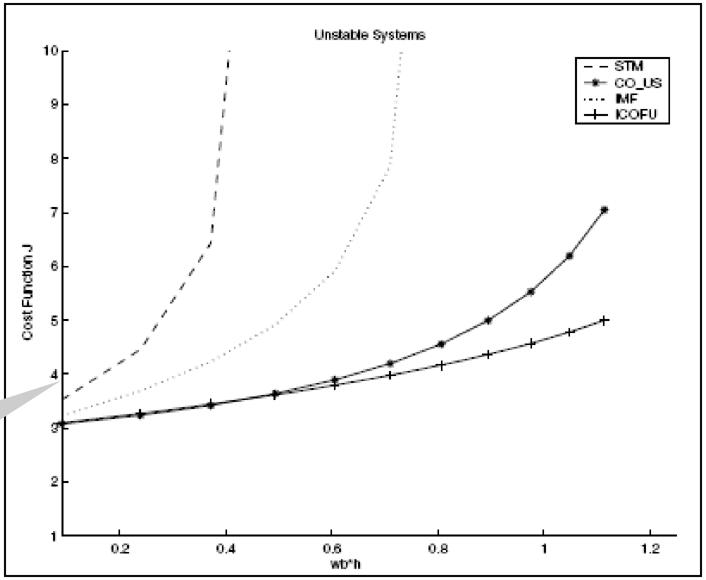


- Processing latency
- Sampling latency
- Physical signal latency

Impact of Software Implemented Tasks

Jitter is typically managed by Periodic I/O

AADL immediate & delayed connections specify deterministic sampling



Impact of Scheduler Choice on Controller Stability

A. Cervin, Lund U., CCACSD 2006

© 2012 Carnegie Mellon University

Software-Based Latency Contributors

Execution time variation: algorithm, use of cache

Processor speed

Resource contention

Preemption

Legacy & shared variable communication

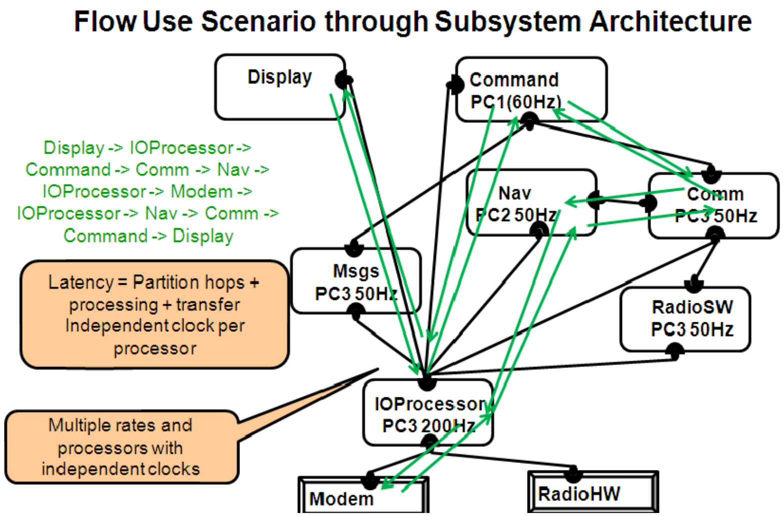
Rate group optimization

Protocol specific communication delay

Partitioned architecture

Migration of functionality

Fault tolerance strategy



AADL supports modeling of latency contributor timing behavior

Input sampling by application code virtualizes time

© 2012 Carnegie Mellon University

Migration of Dual Fault Tolerant Control System

Highly unstable system being controlled

Control software fault tolerance

- Simplex: Baseline, experimental, recovery controllers
- Monitor experimental, recover to baseline control

Dual redundancy to address hardware failures

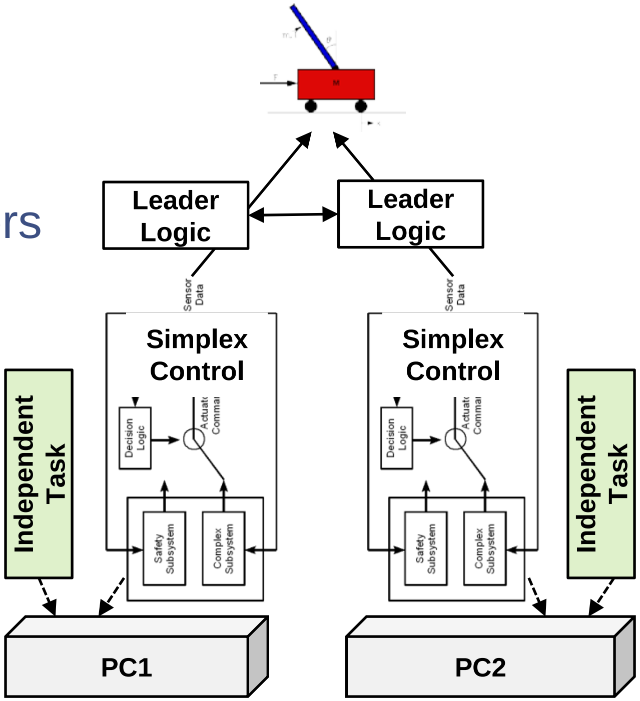
- Two instances of Simplex control system

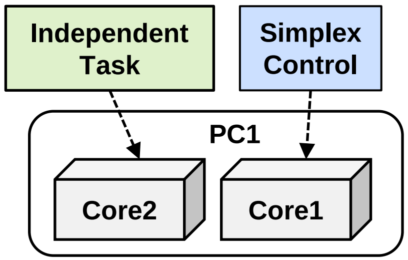
Asynchronous dual processor hardware

- Bounded clock drift
- Distributed leadership decision making
- Each processor shared with unrelated higher priority task

Migration to dual core processors

- One core dedicated to Simplex control system
- Unrelated task on second core
- Failure to provide control





© 2012 Carnegie Mellon University

Outline

Challenges in Safety-critical Software-intensive systems

SAE AADL and the Error Model Annex

Hazards & Fault Impact Analysis

Compositional Analysis

Operational and Failure Modes

The Intricacies of Desired Timing Behavior

Summary and Conclusion

© 2012 Carnegie Mellon University

Architecture Fault Modeling Summary

Architecture Fault Modeling with AADL

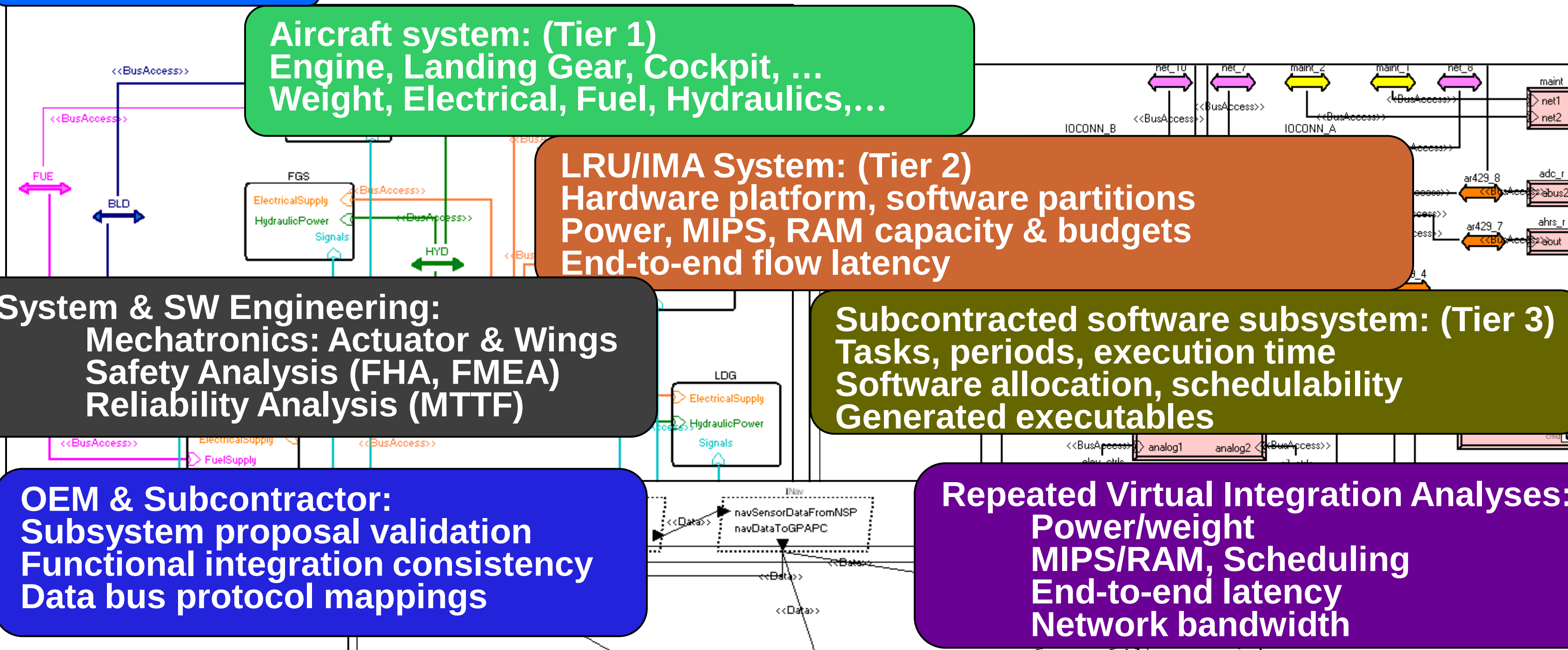
- SAE AADL V2.1 published in Sept 2012 (V1 published in 2004)
- Error Model Annex was published in 2006
  - Supported in AADL V1 and AADL V2
- Error Model concepts and ontology not specific to AADL, can be applied to other modeling notations
- Revised Error Model Annex (V2) based on user experiences currently in review

Safety Analysis and Verification

- Error Model Annex front-end available in OSATE open source toolset
  - Allows for integration with in-house safety analysis tools
- Multiple tool chains support various forms of safety analysis (Honeywell, Aerospace Corp., AVSI SAVI, ESA COMPASS)
- FHA, FMEA, fault tree, Markov models, stochastic Petri net generation from AADL/Error Model
  - Open source implementation as part of Error Model V2 publication

© 2012 Carnegie Mellon University

Early Discovery and Incremental V&V through Virtual Integration (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

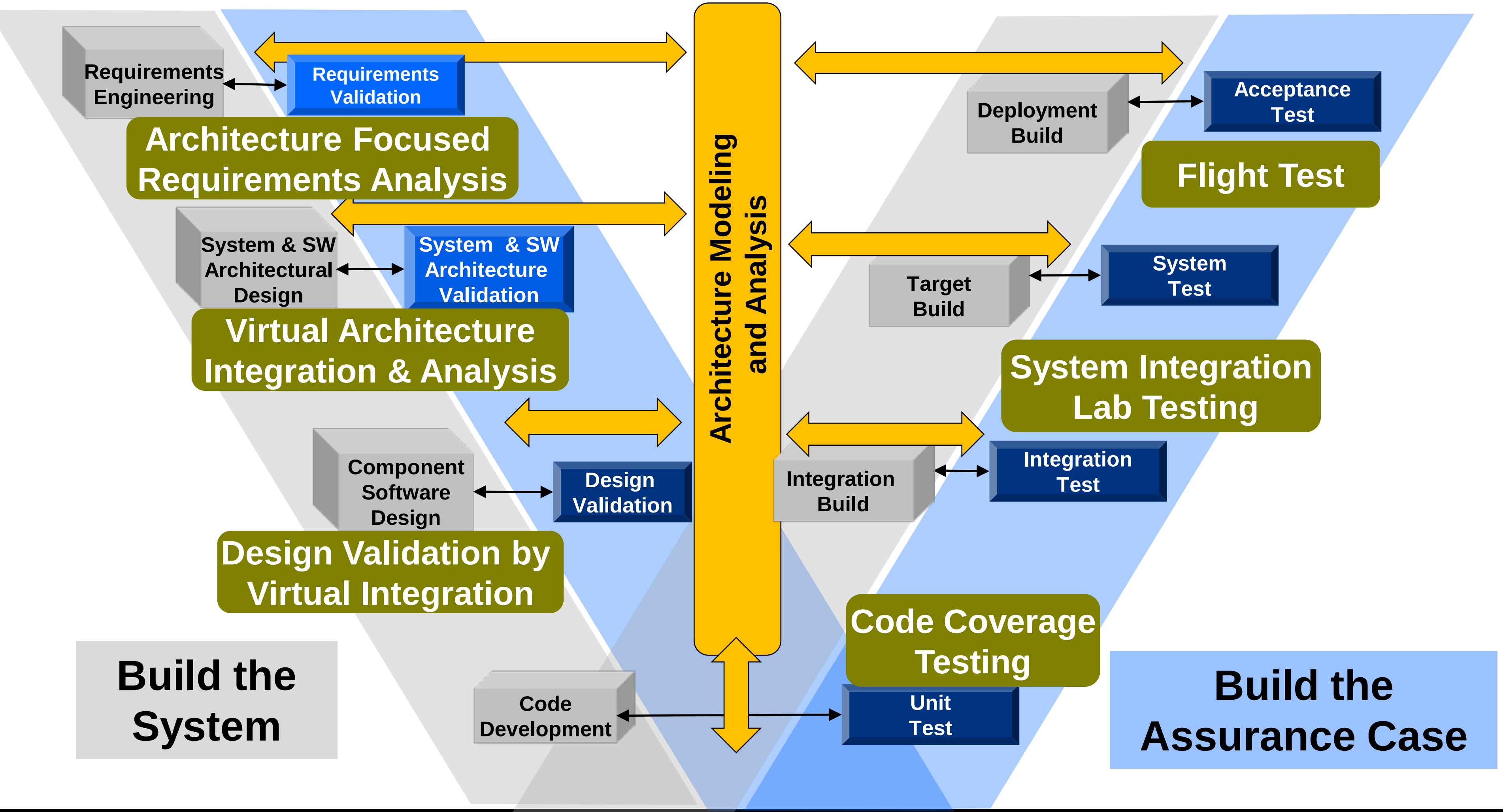
Repeated Virtual Integration Analyses:
- **Power/weight**
- **MIPS/RAM, Scheduling**
- **End-to-end latency**
- **Network bandwidth**

Proof of Concept Demonstration and Transition by Aerospace industry initiative
- **Propagate requirements and constraints**
- **Higher level model down to suppliers' lower level models**
- **Verification of lower level models satisfies higher level requirements and constraints**

- Multi-tier system & software architecture (in AADL)
- Incremental end-to-end validation of system properties

© 2012 Carnegie Mellon University

Increased Confidence through Model-based Analysis and Testing Evidence throughout Life Cycle

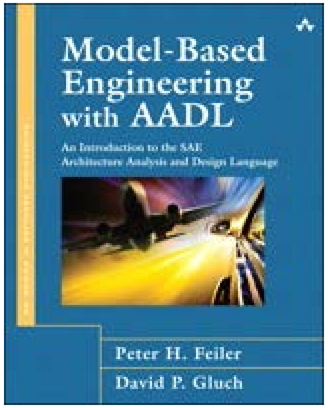


© 2012 Carnegie Mellon University

References

Website www.aadl.info

Public Wiki https://wiki.sei.cmu.edu/aadl



AADL Book in SEI Series of Addison-Wesley
http://www.informit.com/store/product.aspx?isbn=0321888944

© 2012 Carnegie Mellon University

Contact Information

Peter H. Feiler

Senior MTS

RTSS

Telephone: +1 412-268-7790

Email: phf@sei.cmu.edu

U.S. Mail

Software Engineering Institute

Customer Relations

4500 Fifth Avenue

Pittsburgh, PA 15213-2612

USA

Web

www.sei.cmu.edu

www.sei.cmu.edu/contact.cfm

Customer Relations

Email: info@sei.cmu.edu

SEI Phone: +1 412-268-5800

SEI Fax: +1 412-268-6257

© 2012 Carnegie Mellon University