Safety Analysis with AADL

Julien Delange <jdelange@sei.cmu.edu>
Jerome Hugues <jerome.hugues@isae.fr>

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

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.

DM-0002583

Objectives

Introduce the AADL Error-Model v2 (EMV2)

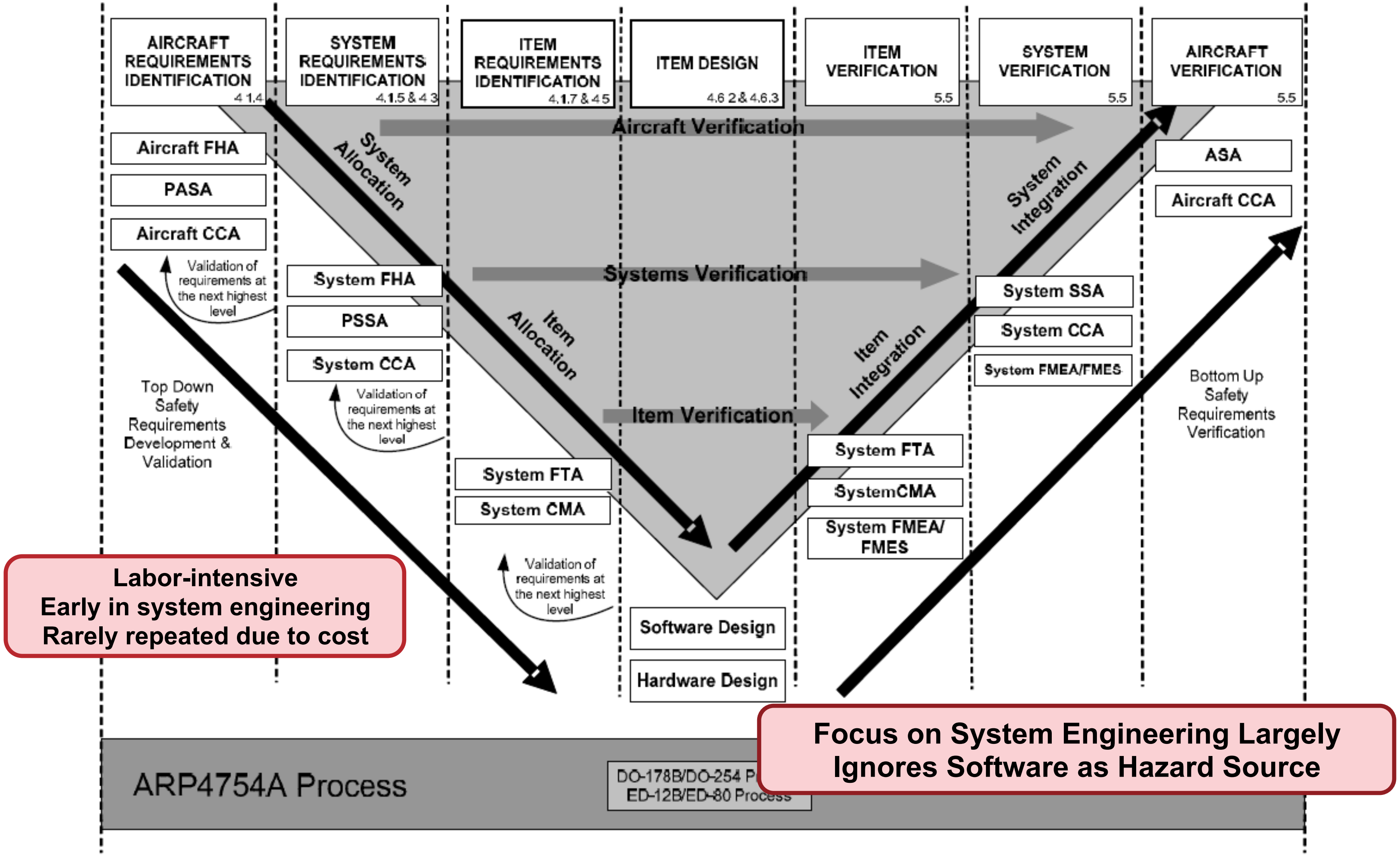
Explain main concepts (errors sources and propagation)

Present safety analysis tools

Exercise safety analysis on the ADIRU system

Introduction to the AADL Error Model Annex v2

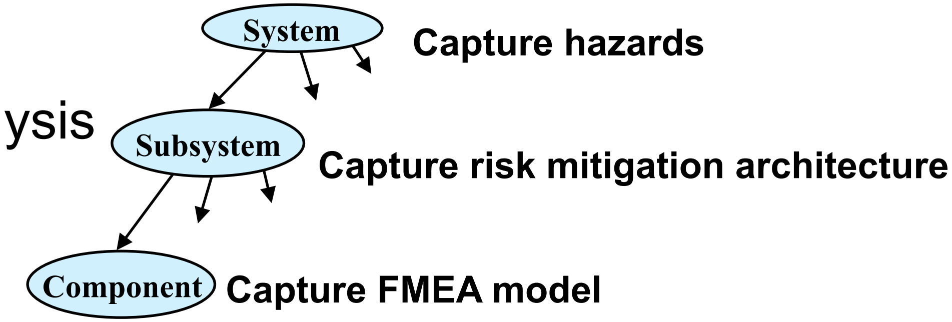
Safety Practice in Development Process Context



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

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

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

Error Model V2: 4 levels of abstraction

1. Focus on fault interaction with other components
2. Focus on fault behavior of components
3. Focus on fault behavior in terms of subcomponent
4. Types of malfunctions and propagations

Automation of SAE ARP4761 System Safety Assessment Practice

AADL & EMV2

FHA

Spreadsheet

Uses error sources

Component	Error	Hazard Description	Crossrefer	Functional Failure	Operational P
StabilatorPositionSe	"ServiceOmission o	"No stabilator position readings due to s	"1.1.3"	"Loss of sensor readings"	"all"
StabAct1	"ServiceOmission o	"Failure to move stabilator into desired	"1.1.2"	"Loss of actuator functionalit	"all"
StabAct2	"ServiceOmission o	"Failure to move stabilator into desired	"1.1.2"	"Loss of actuator functionalit	"all"
StabilatorController	"null on ActCmd"	"Absence of computed data should signa	"1.1.1"	"Loss of guidance values"	"Approach"
StabilatorController	"null on ActCmd"	"Absence of computed data should signa	"1.1.1"	"Loss of guidance values"	"Approach"
StabilatorController	"null on ActCmd"	"Absence of computed data should signa	"1.1.1"	"Loss of guidance values"	"Approach"

FMEA

Spreadsheet

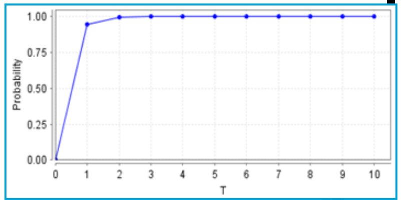
Uses error flows & propagations

B	C	D	E	F	G
			Three CPU FMEA		
	Date:	12/1/2010	Author:		
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

Markov Chain

PRISM

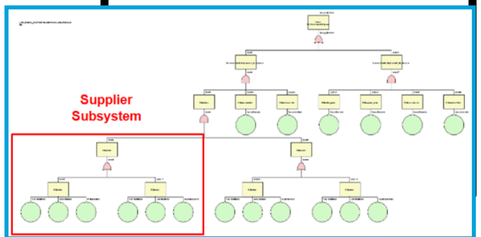
Uses error flows & behavior



FTA

CAFTA, OpenFTA

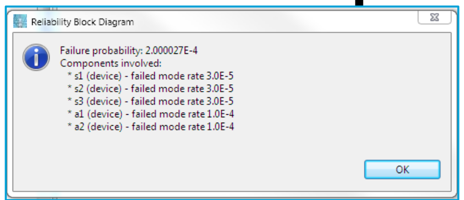
Uses composite error behavior



RBD/DD

OSATE plugin

Uses composite error behavior



Value of Automated Architecture-led Safety Analysis

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

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

Providing different views

EMV2-like Compositional Fault Behavior Specification for Simulink Models

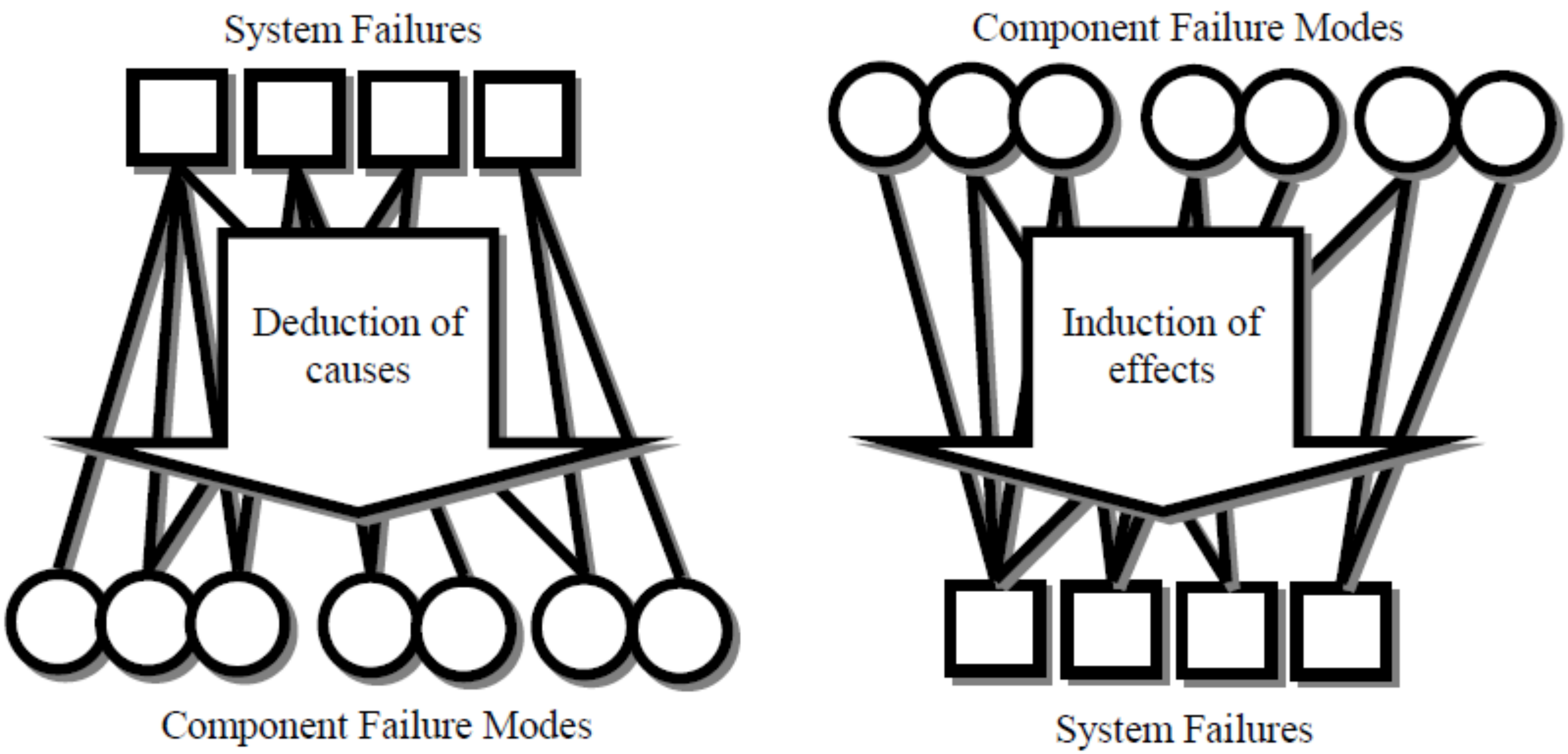
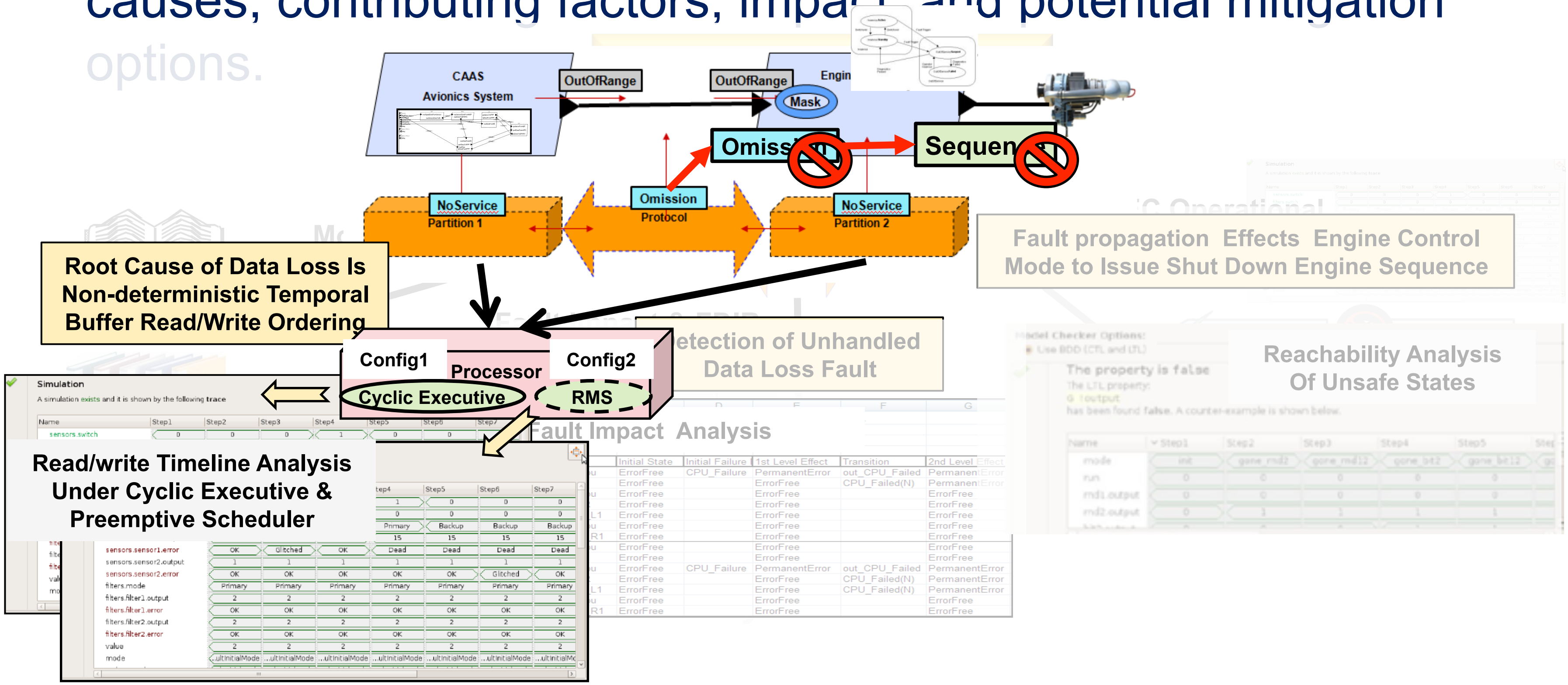


Figure 9 - Inverse relationship between fault trees (left) and FMEA (right)

Understanding the Cause and Effects of Faults

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



Safety-Criticality Requirements

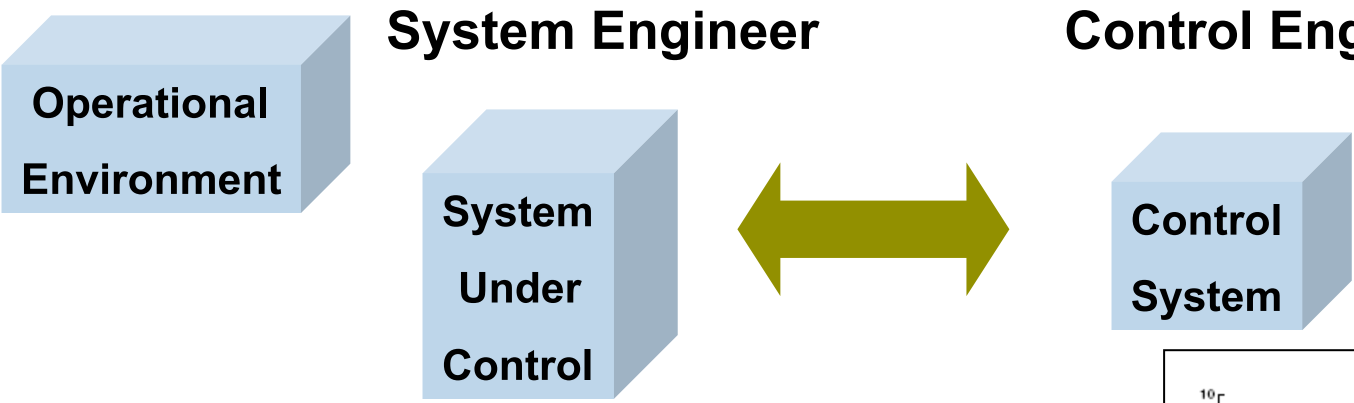
Exceptional conditions, anomalies and hazards

- Mode confusion (reported state vs. observed state vs. actual state)
- Unexpected fault conditions and fault impact
- Inclusion/exclusion of pilot in system
- Fault Detection, Isolation, and Recovery (FDIR)
  - Safety system architecture, security system architecture

Certification impact

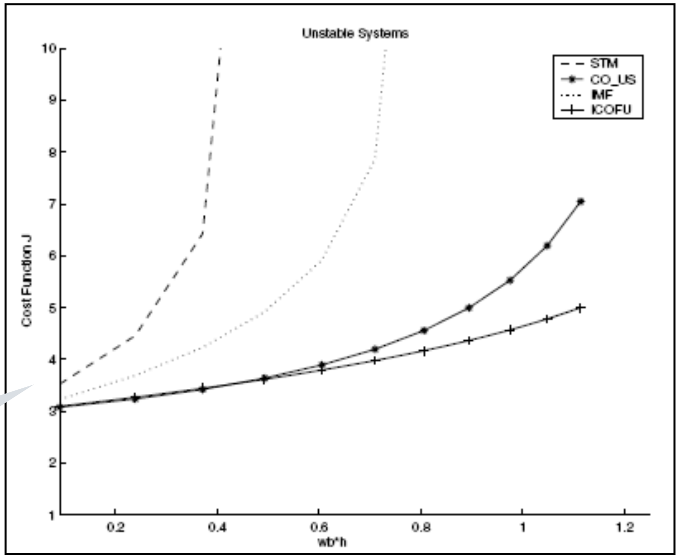
- Criticality levels, design assurance levels and verification implications
- Partition allocations (isolation) and avoidable certification cost
- Understanding change impact to achieve proportional recertification

Latency Sensitivity in Control Systems



Common latency data from system engineering

- Processing latency
- Sampling latency
- Physical signal latency



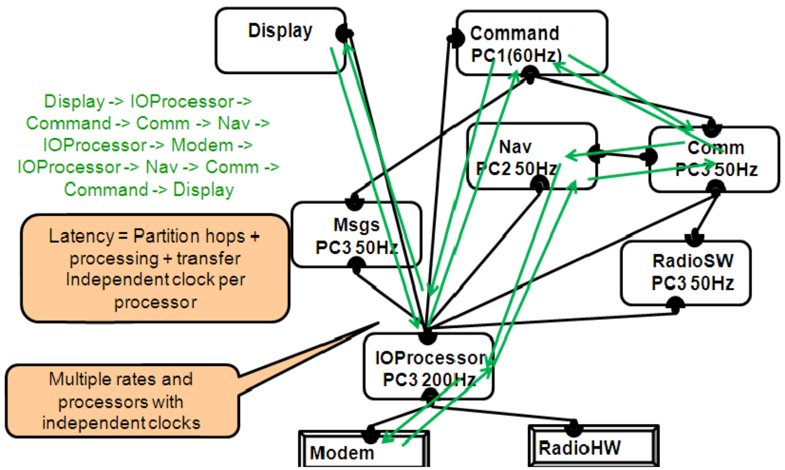
Impact of Scheduler Choice on Controller Stability

A. Cervin, Lund U., CCACSD 2006

Software-Based Latency Contributors

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

Flow Use Scenario through Subsystem Architecture



The Symptom: Missed Stepper Motor Steps

Software modeled and verified in SCADE

Full reliance on SCADE of SM & all functionality

Problems with missing steps not detected

Stepper motor (SM) controls a valve

- Commanded to achieve a specified valve position
  - Fixed position range mapped into units of SM steps
- New target positions can arrive at any time
  - SM immediately responds to the new desired position

Software tests did not discover the issue

Time sensitive systems are hard to test for.

Safety hazard due to software design

- Execution time variation results in missed steps
- Leads to misaligned stepper motor position and control system states
- Sensor feedback not granular enough to detect individual step misses

Two Customer Proposed Solutions

Sending of data at 12ms offset from dispatch

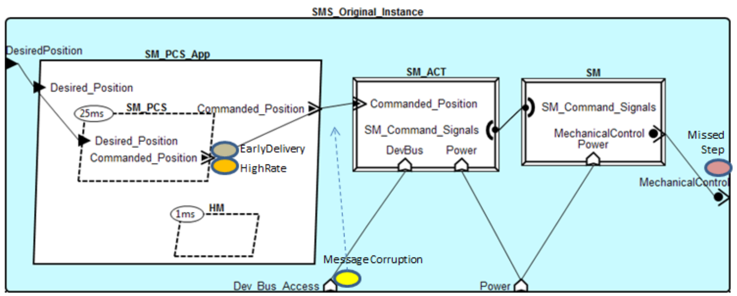
Buffering of command by SM interface

No analytical evidence that the problem will be addressed

Analysis Results and Solution

Architecture Fault Model Analysis

- Fault impact analysis identifies <u>multiple sources</u> of missed steps
  - Early arrival of step increment commands
  - Step increment command rate mismatch
  - Transient message corruption or loss
- Understanding of error cause
  - When is early too early
  - Guaranteed delivery assumption for step increment commands



MissedStep	Original Design	Fixed Send Time	Buffered Command	Position Command
SMS logical failures	EarlyDelivery HighRate	HighRate	HighRate	
SMS mechan-ical failures	ActuatorFailure StepperMotorFailure	ActuatorFailure StepperMotorFailure	ActuatorFailure StepperMotorFailure	ActuatorFailure StepperMotorFailure
Transient comm failures	MessageCorruption MessageLoss	MessageCorruption MessageLoss	MessageCorruption MessageLoss	
Mechanical failures in Op Environment	ECUFailure PowerLoss ValveFailure	ECUFailure PowerLoss ValveFailure	ECUFailure PowerLoss ValveFailure	ECUFailure PowerLoss ValveFailure

Time-sensitive Auto-brake Mode Confusion

Auto-brake mode selection by push button

- Three buttons for three modes
- Each button acts as toggle switch

Event sampling in asynchronous system setting

- Dual channel COM/MON architecture
- Each COM, MON unit samples separately
  - Button push close to sampling rate results in asymmetric value error
  - COM/MON mode discrepancy votes channel out
  - Repeated button push does not correct problem
  - Operational work around (1 second push) is not fool proof

Avoidable complexity design issue

- Concept mismatches: desired state by event and sampled event

Error Model Annex v2
Main Concepts

Error Type Libraries

```
Package myerrortypes
public
Annex emv2{**
error types
    AxleFailure: type;
    Fracture: type extends axlefailure;
    Fatigue: type extends axlefailure;
end types;
**};
End myerrortypes;
```

Error Type libraries and AADL Packages

- An AADL package can contain one Error Model library declaration
- The **error types** clause represents the Error Type library within the Error Model library
- The Error Type library is identified and referenced by the package name

Error Type library represents a namespace for error types and type sets

- Error type and type set names must be unique within an Error Type library
- An Error Type library can contain multiple error type hierarchies

Error Types & Error Type Sets

Error type declarations

```
TimingError: type ;
EarlyValue: type extends TimingError;
LateLate: type extends TimingError;
ValueError: type ;
BadValue: type extends ValueError;
```

Error Type Set as Constraint
{T1} tokens of one type hierarchy
{T1, T2} tokens of one of two error type hierarchies
{T1*T2} type product (one error type from each error type hierarchy)
{**NoError**} represents the empty set
Constraint on state, propagation, flow, transition condition, detection condition, outgoing propagation condition, composite state condition

An *error type set* represents a set of type instances

- Elements in a type set are mutually exclusive
- An error type with subtypes includes instances of any subtype
- A *type product* represents a simultaneously occurring types
  - Combinations of subtypes

```
InputOutputError : type set {TimingError, ValueError,
TimingError*ValueError};
```

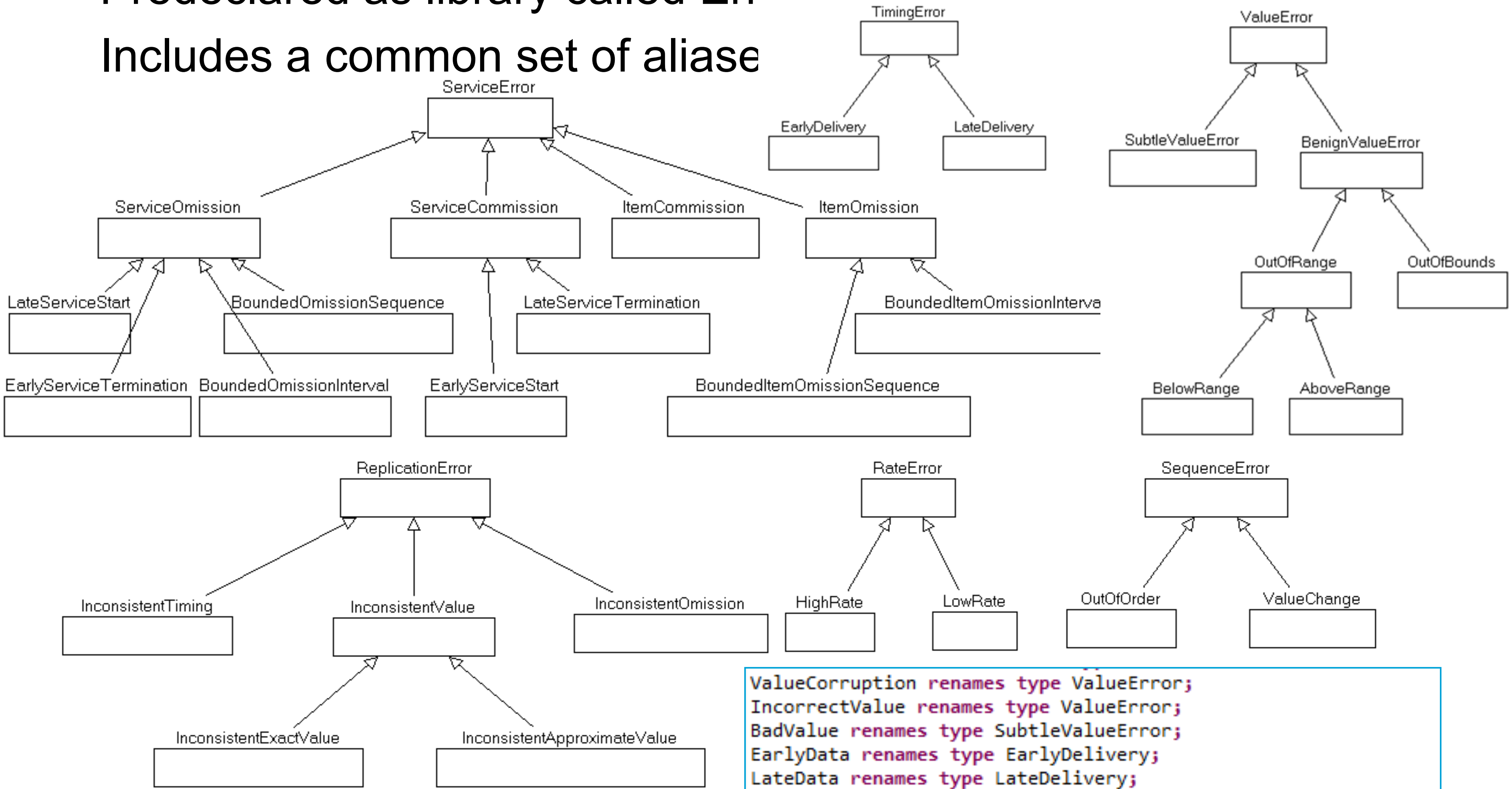
An *error type instance*

- Represents the error type of an actual event, propagation, or state

A Standard Set of Error Propagation Types

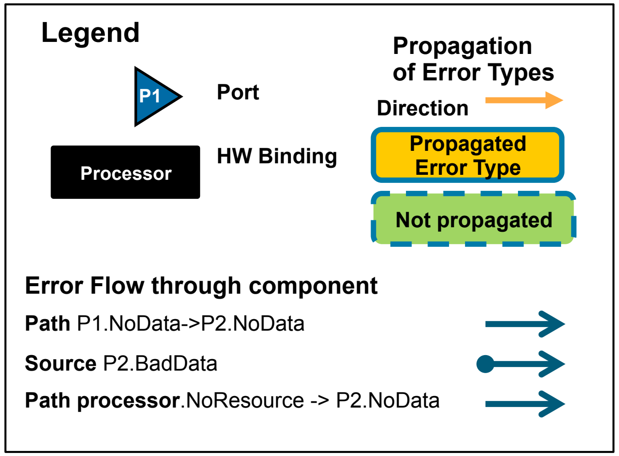
Predeclared as library called ErrorLibrary

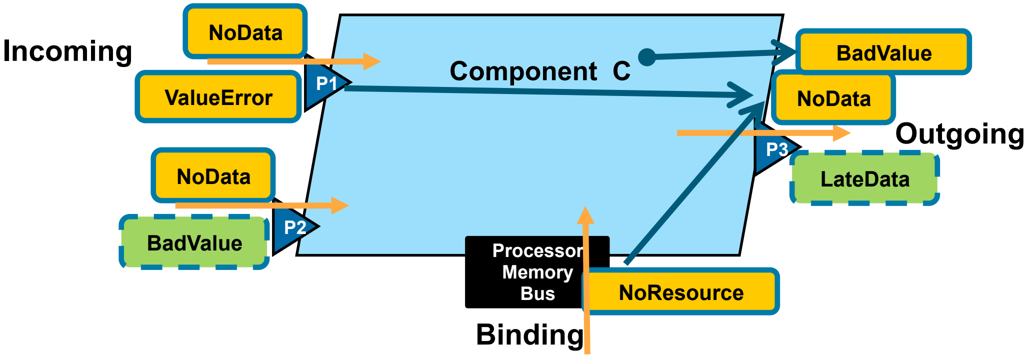
Includes a common set of aliase




```
ValueCorruption renames type ValueError;
IncorrectValue renames type ValueError;
BadValue renames type SubtleValueError;
EarlyData renames type EarlyDelivery;
LateData renames type LateDelivery;
AsymmetricValue renames type InconsistentValue;
SymmetricValue renames type ValueError;
AsymmetricItemOmission renames type InconsistentItemOmission;
```

Component Error Propagation





“Not“ on propagated indicates that this error type is intended to be contained.

This allows us to determine whether propagation specification is complete.

Incoming/Assumed

- **Error Propagation**
  Propagated errors
- **Error Containment:**
  Errors not propagated

Outgoing/Contract

- **Error Propagation**
- **Error Containment**

Bound resources

- **Error Propagation**
- **Error Containment**
- **Propagation to resource**

Supports Fault Propagation & Transformation Calculus (FPTC) by York University

Also origin of safety cases

Error Propagation Declarations

```
system Subsystem
features
  P1: in data port;
  P2: in data port;
  P3: out data port;
annex EMV2 {**
  use types ErrorLibrary;
  error propagations
   P1: in propagation {NoData, ValueError} ;
   P2: in propagation {NoData};
   P2: not in propagation  {BadValue};
   P3: out propagation {NoData, BadValue};
   P3: not out propagation {LateData};
  processor:  in propagation {NoResource};
end propagations; **};
```

Binding Related Propagation Specifications

Processor, Memory, Connection, Binding, Bindings

Path follows predeclared Binding properties

Error Flows

Error flow specifies the role of a component in error propagation

- The component may be a source or sink of a propagated error types
- The component may pass incoming types through as outgoing types
- The component may transform an incoming type into a different outgoing type
- By default all incoming errors of any feature flow to all outgoing features

```
annex EMV2 {**
  error propagations
  .. .. ..
  flows
    es1: error source P3{BadData} ;
    es2: error source P3{NoData} ;
    es3: error sink P2{NoData};
    ep1: error path P2{BadData}->P3;   -- same type as incoming type
    ep2: error path P1{ValueError} -> P3{ItemOmission};  -- all value errors xformed into ItemOmission
    ep3: error path processor -> P3
         mapping MyErrorModelLibrary::MyMapping; -- use a  type mapping table
  end propagations ; **};
```

The same propagation may be part of a flow source/sink and flow path.

A propagation may be a sink for one type and not for another

```
type mappings MyMapping
use types ErrorLibrary;
{BadData} -> {NoData} ;
{NoService} -> {NoData} ;
end mappings;
```

Functional Hazard Assessment

Hazard property

- Tailoring for safety standards (ARP4761, MIL-STD-882)
- Associated with error state, error source, outgoing propagation, error type

```
Hazards: list of record
  (
   crossreference      : aadlstring; -- cross refere
   hazardtitle         : aadlstring; -- short descri
   description         : aadlstring; -- description
   failure             : aadlstring; -- description
   failureeffect       : aadlstring; -- description
   phases              : list of aadlstring; -- oper
   environment         : aadlstring; -- description

   risk                : aadlstring; -- description
   failurecondition    : aadlstring; -- description
   severity            : ARP4761::SeverityLabels;
   likelihood          : ARP4761::LikelihoodLabels;
   targetseverity      : ARP4761::SeverityLabels; -
   targetlikelihood    : ARP4761::LikelihoodLabels;
   developmentassurancelevel : EMV2::DALLabels; --

   verificationmethod : aadlstring; -- verificatio
   safetyreport        : aadlstring; -- capturing th
   comment             : aadlstring; -- additional i
  )
```

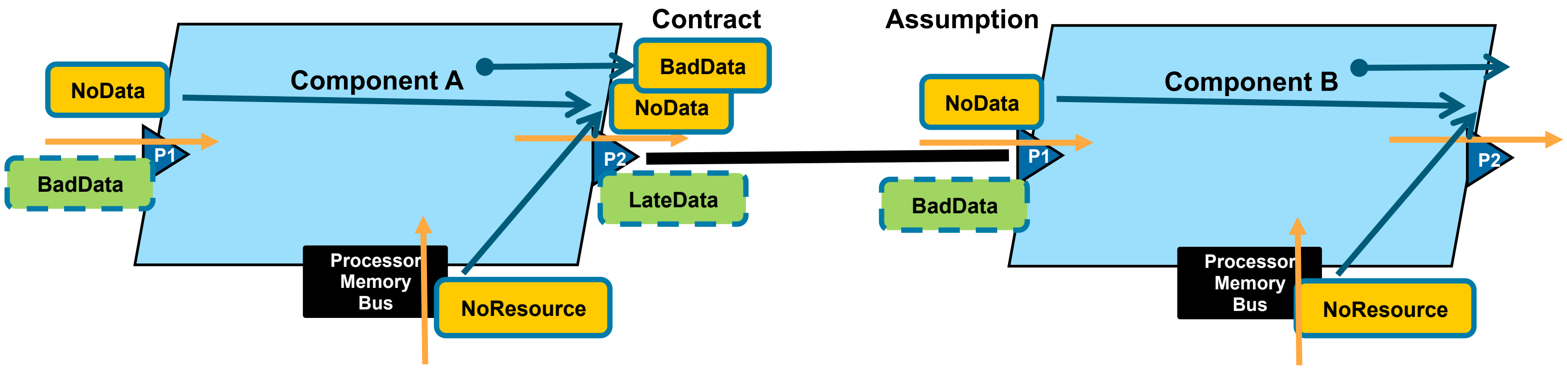
```
device PositionSensor
  features
    PositionReading: out data port DataDictionary::Position;
  flows
    f1: flow source PositionReading {
      Latency => 2 ms .. 3 ms;
      };
  annex EMV2 {**
      use types ErrorLibrary, FHAErrorLibrary;
      use behavior ErrorModelLibrary::Simple;
  error propagations
          PositionReading: out propagation  {ServiceOmission};
      flows
       ef1:error source PositionReading {ServiceOmission} when Failed;
  end propagations;
  properties
  EMV2::hazards =>
  ([  crossreference => "1.1.3";
      failure => "Loss of sensor readings";
      phases => ("all");
      severity => MILSTD882::Critical;
      likelihood => MILSTD882::remote;
      description => "No stabilator position readings due to sensor failure";
      comment => "Becomes major hazard, if no reundant sensor";
          ])
          applies to ef1.Failed;
  **};
end PositionSensor;
```

				Functiona	Operational Phase	Effects of	Severit		
10	power/battery1	Depleted	TBD	Battery D	all	No more	Major		
11	power/battery1	Explode	TBD	Battery E	all	Battery E	Catastrop	Extremel	Have a physical impact on the surrounding components
12	power/battery1	NoPower on socket	ARP4761 page 277 figure 9	Loss of or	Landing/RTO	Loss of El	Major	Probable	Major hazard if both power are lost
13	power/battery2	Depleted	TBD	Battery D	all	No more	Major	Probable	Can be an issue if redundant battery is failing also
14	power/battery2	Explode	TBD	Battery E	all	Battery E	Catastrop	Extremel	Have a physical impact on the surrounding components
15	power/battery2	NoPower on socket	ARP4761 page 277 figure 9	Loss of or	Landing/RTO	Loss of El	Major	Probable	Major hazard if both power are lost
16	blue_pump	HydraulicError	ARP4761 page 275 figure L9	Hydraulic	TBD	Loss of or	Major	Probable	Major hazard if both pumps are lost
17	green_pump	HydraulicError	ARP4761 page 275 figure L9	Hydraulic	TBD	Loss of or	Major	Probable	Major hazard if both pumps are lost
18	accumulator	HydraulicError	ARP4761 page 275 figure L9	Hydraulic	TBD	Loss of or	Major	Probable	Major hazard if both pumps are lost

Other Predeclared EMV2 Properties

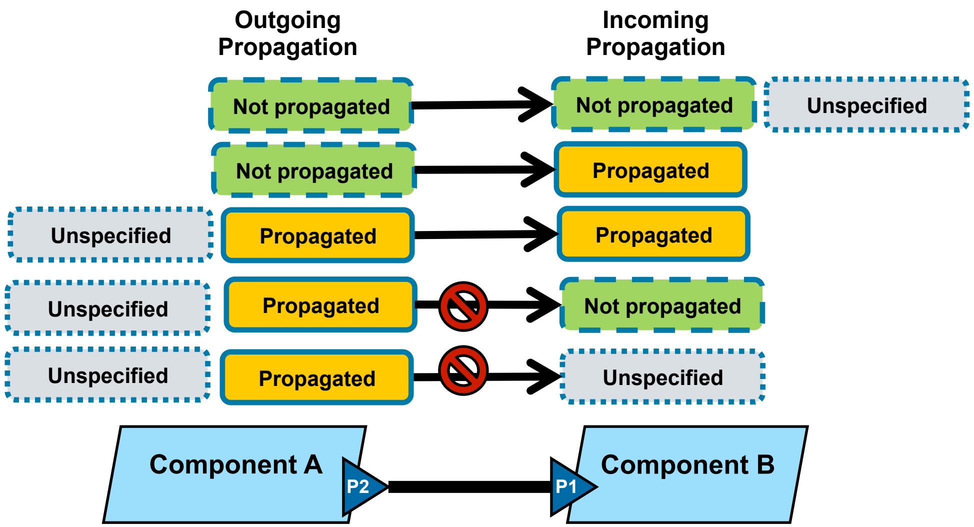
- Occurrence distribution
  - Distribution functions: Fixed, Poisson/Exponential, Normal/ Gauss, Weibull, Binominal
- Persistence: Permanent, Transient, Singleton
- Duration distribution
- Fault kind: design, operational
- State kind: working, nonworking
- Detection mechanism

Consistency in Error Propagation

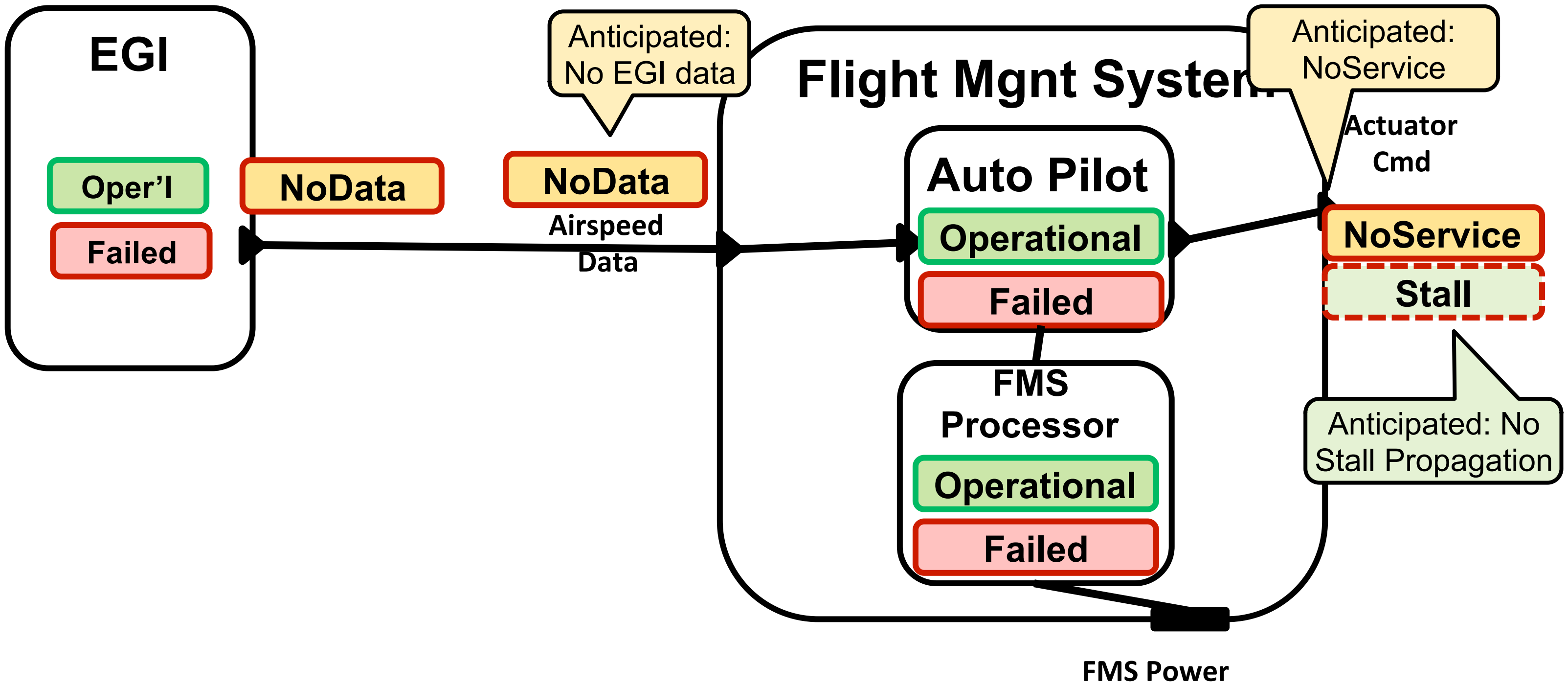


Mismatched fault propagation and containment assumptions

Discovery of unhandled error propagations.

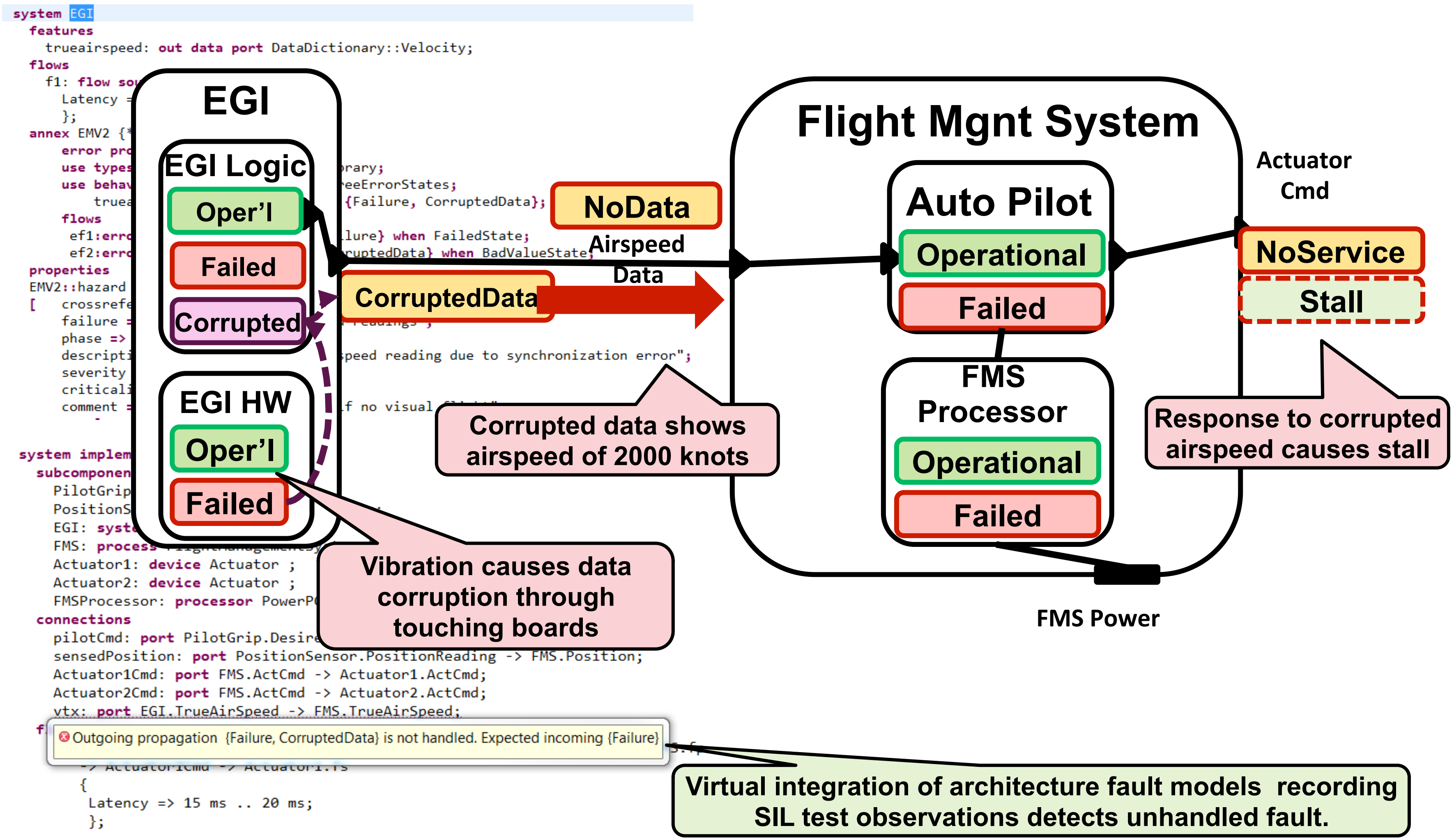


Software Induced Flight Safety Issue



Original Preliminary System Safety Analysis (PSSA)
System engineering activity with focus on failing components.

Unhandled Hazard Discovery through Virtual Integration



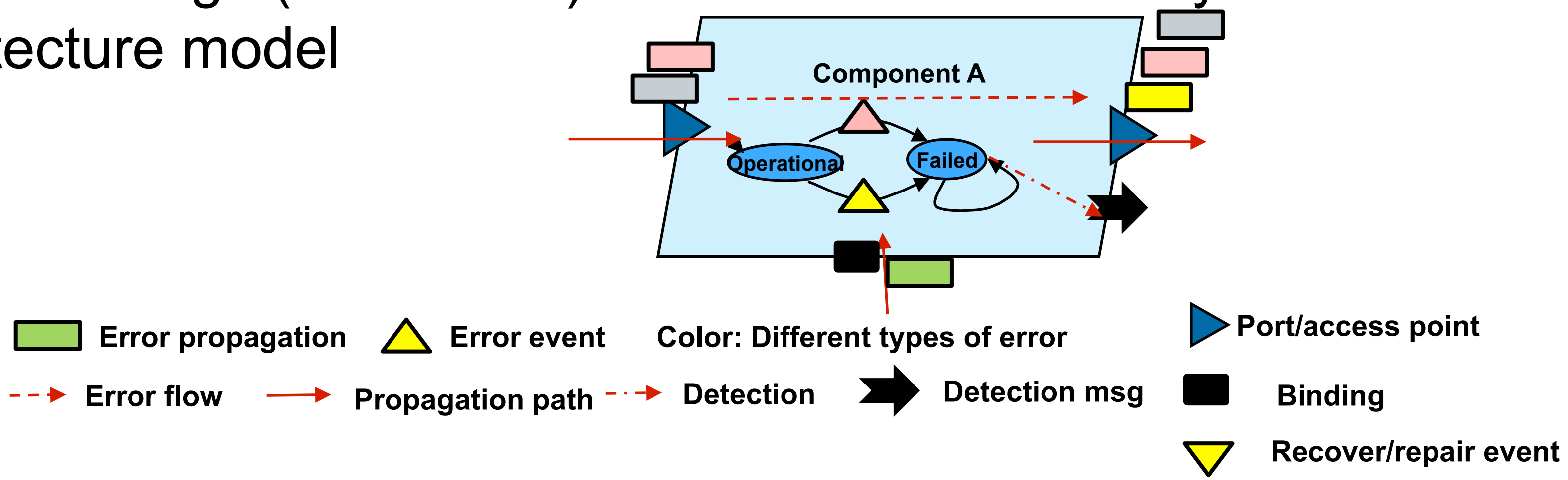
Component Error Behavior

Components have error, mitigation, and recovery behavior specified by an error behavior state machine

Transitions between *states* triggered by *error events* and *incoming propagations*.

Conditions for *outgoing propagations* are specified in terms of the *current state* and *incoming propagations*.

Detection of error states and incoming propagations is mapped into a message (event data) with error code in the system architecture model



Reusable Error Behavior State Machine

```
annex EMV2 {**
    error behavior ExampleBehavior
    events
        Fault: error event;
        SelfRepair: recover event;
        Fix: repair event;
   states
    Operational: initial state ;
    FailStopped: state;
    FailTransient: state;
transitions
     SelfFail: Operational -[Fault]-> (FailStopped with 0.7, FailTransient with 0.3);
     Recover: FailTransient -[SelfRepair]-> Operational;
end behavior;
Properties
      EMV2::OccurrenceDistribution => [ ProbabilityValue => 0.00004 ; Distribution => Poisson;]
               applies to Fault;

**};
```

State machine with branching transition

Component Error Behavior Specification

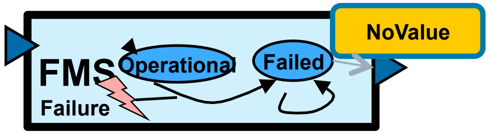
Component-specific behavior specification

- Identifies an error behavior state machine
- Optionally defines component specific error events
- Specifies transition trigger conditions in terms of incoming propagated errors or working condition of connected component
- Specifies propagation conditions for outgoing propagated errors in terms of states & incoming propagated errors
- Specifies detection conditions under which becomes an event with error code in the core AADL model

```
use types ErrorLibrary ;
use behavior MyErrorLibrary::ExampleBehavior ;
component error behavior
transitions  -- additional transitions that are component specific
    Operational-[Port1{NoData} and Port2{NoError}]->FailTransient;
    FailStopped-[port1{BadData}];
 propagations
    all –[2 ormore (Port1{BadData}, Port2{BadData},Port3{BadData})]-> Outport3(BadData);
 detections
     FailedState –[]-> Self.Failed ( FailCode ) ;       -- Could also report on an outgoing error port
 properties
     EMV2::OccurrenceDistribution => [ ProbabilityValue => 0.00005 ; Distribution => Poisson;]
        applies to Fault; -- component specific occurrence value
 end behavior;
```

Error Model at Each Architecture Level

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



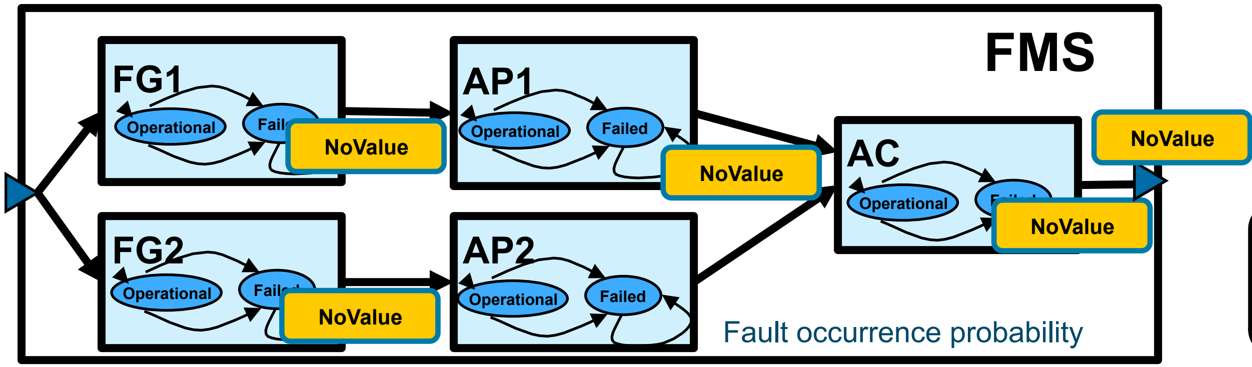
Fault occurrence probability

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**

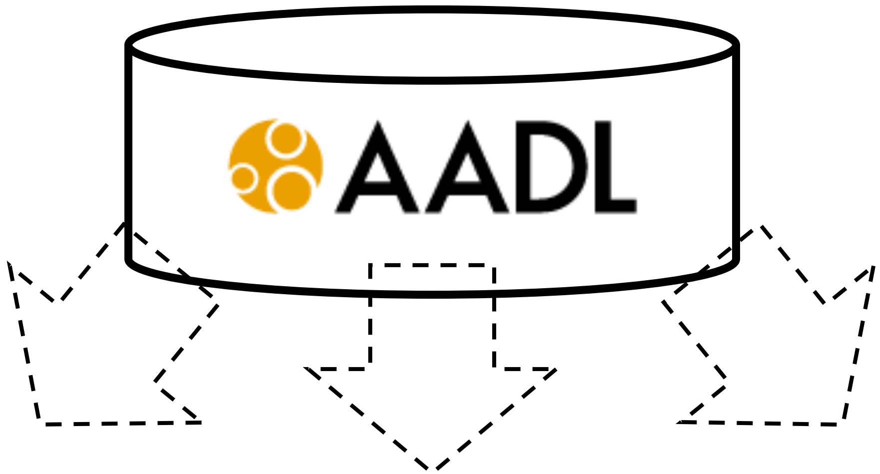
1 **ormore**(FG2.Failed **or** AP2.Failed) **or** AC.Failed]->Failed



Consistency Checking Across Levels of the Hierarchy

Error Model Annex v2 Safety Analysis tools

AADL & Safety Evaluation – Tool Overview



FHA
- Spreadsheet
- Use error propagations

FTA
- CAFTA OpenFTA
- Use composite behavior
- Error flows

Markov Chain
- PRISM
- Use error flow
- Error behavior

FMEA
- Spreadsheet
- Error behavior
- Propagations

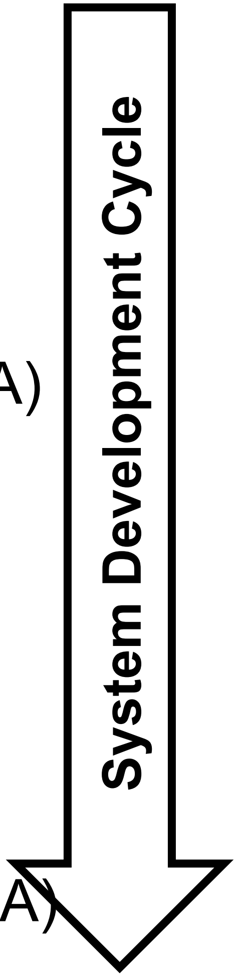
Safety Analysis & AADL

Preliminary System Safety Assessment (PSSA) support

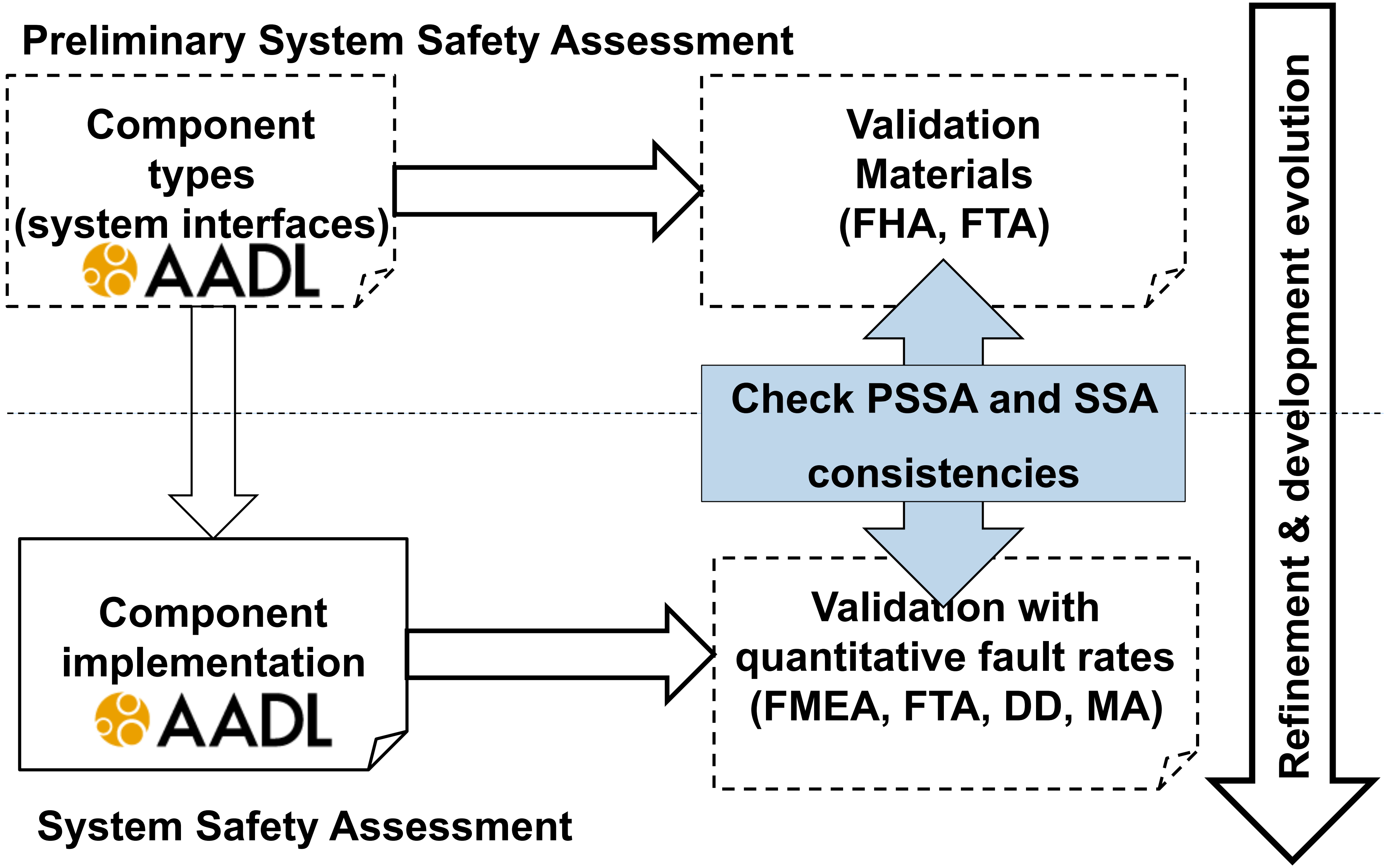
- High-level component, interfaces from the OEM
- Automatic generation of validation materials (FHA, FTA)

System Safety Assessment (SSA) support

- Use refined models from suppliers
- Enhancement of error specifications
- Support of quantitative safety analysis (FTA, FMEA, MA)



Evolution of Safety Analysis process with AADL



Safety Analyses on Refined Architecture

Aircraft-Level Safety Analysis

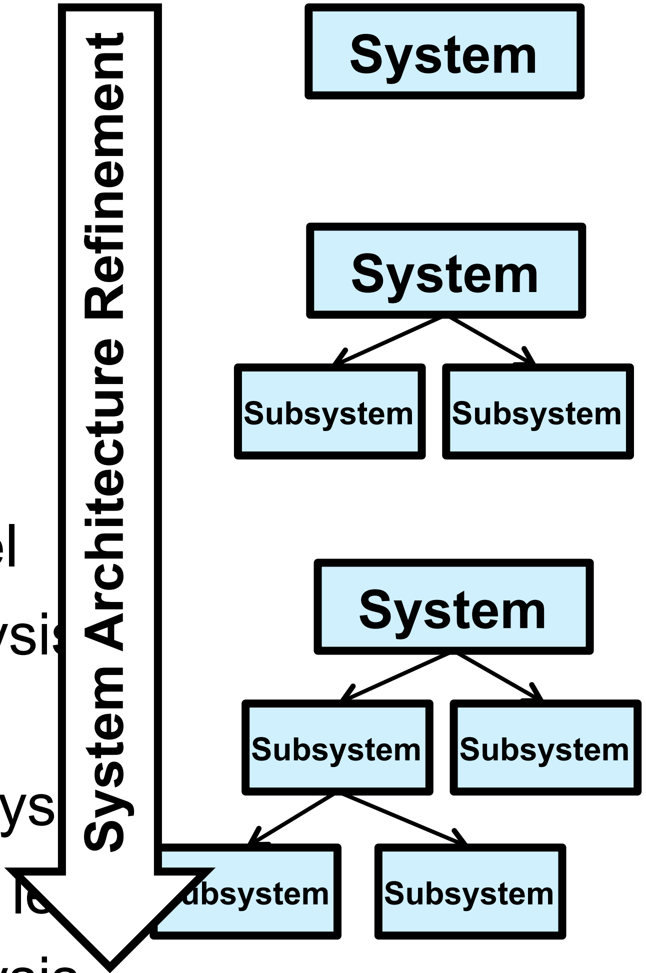
- Define aircraft failure conditions
- Allocate failure to system functions
- Perform PSSA and SSA

Avionics Subsystem Level Safety Analysis

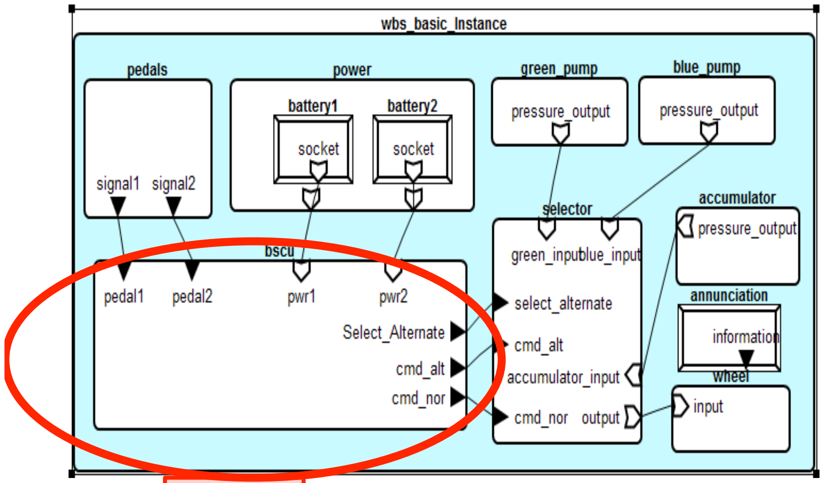
- Perform PSSA and SSA at subsystem level
- Ensure consistency with aircraft level analysis

Navigation Sub-Subsystem Level Safety Analys

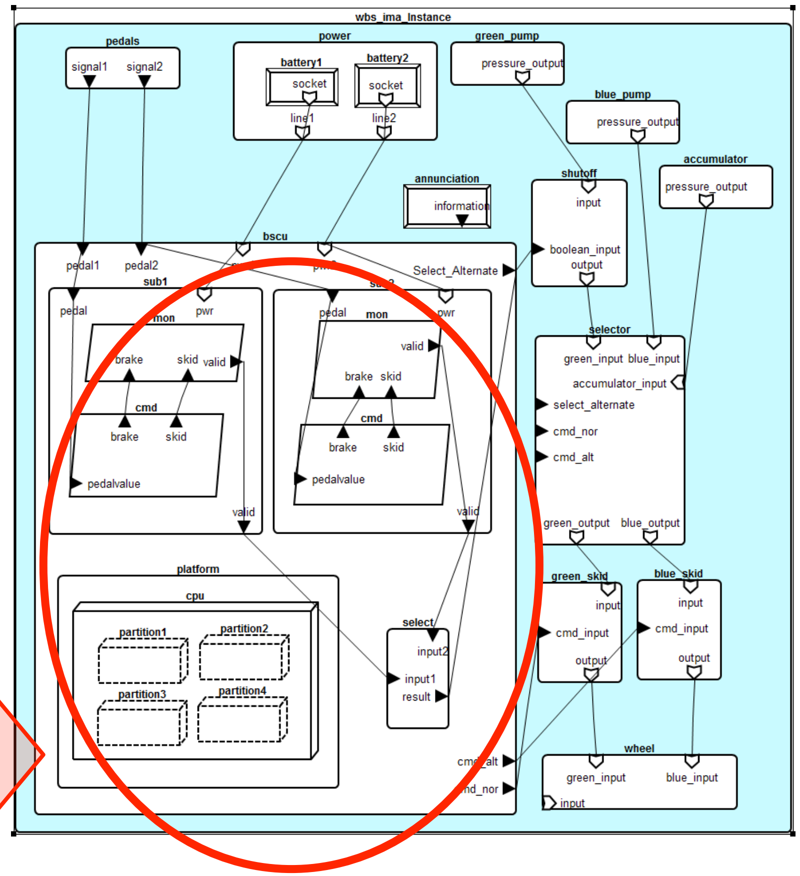
- Perform PSSA and SSA at sub-subsystem le
- Ensure consistency with aircraft level analysis



Evolution of the AADL model



Component extension, refinement & implementation

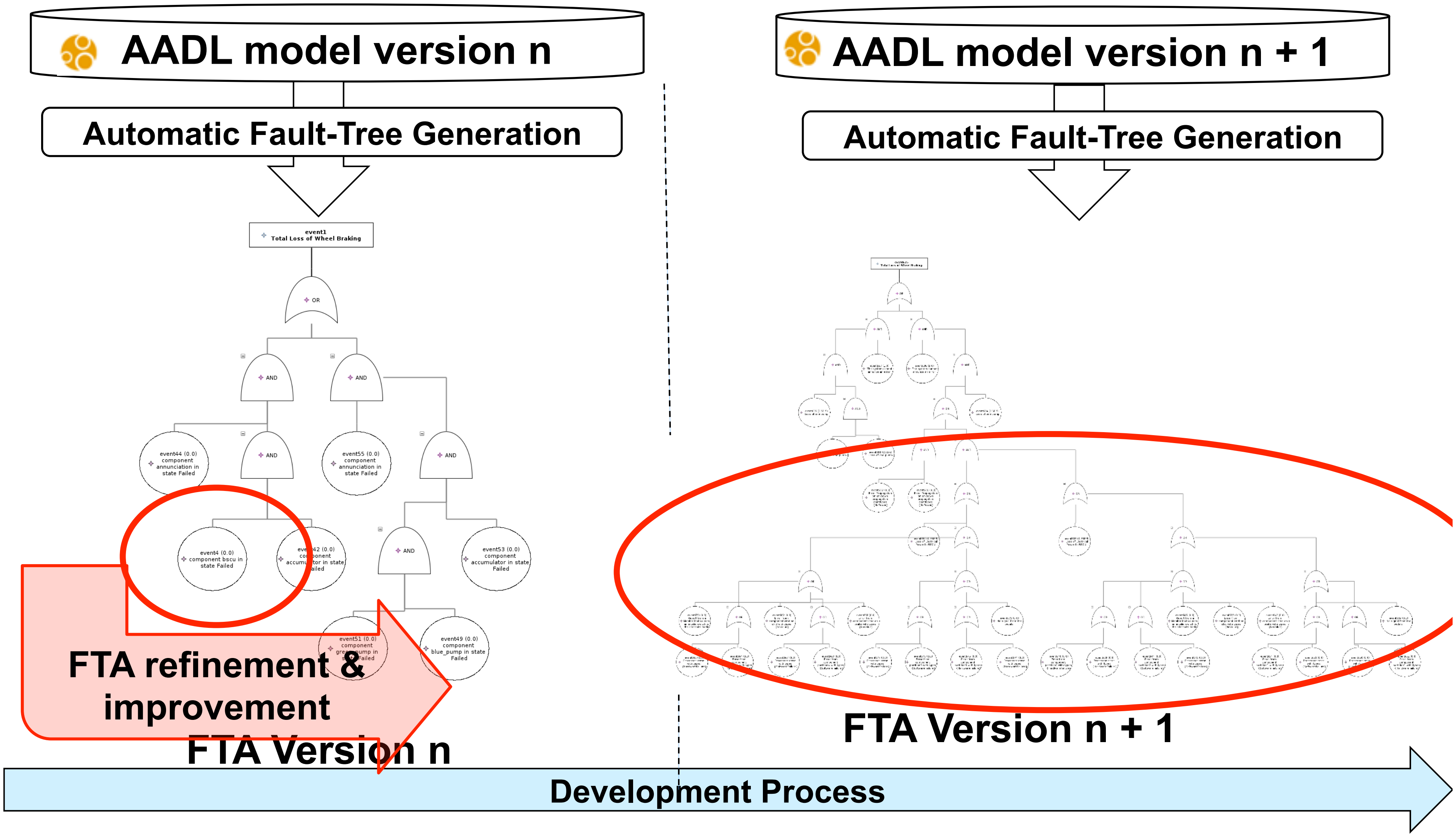


AADL model Version n

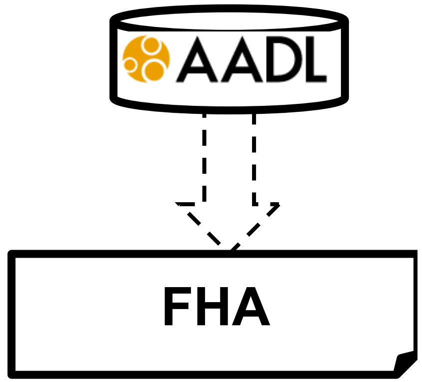
AADL model Version n + 1

Development Process

Evolution of Safety Assessment with AADL



Functional Hazard Analysis Support



Use of **component error behavior**

- Error propagations rules
- Internal error events

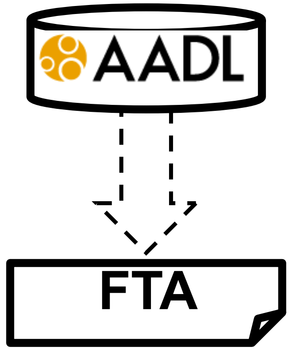
Specify initial failure mode

Define error description and related information

Create spreadsheet containing FHA elements

- To be reused by commercial or open-source tools

Fault-Tree Analysis Support



Use **of composite error behavior**

- FTA nodes

Use of **component error behavior**

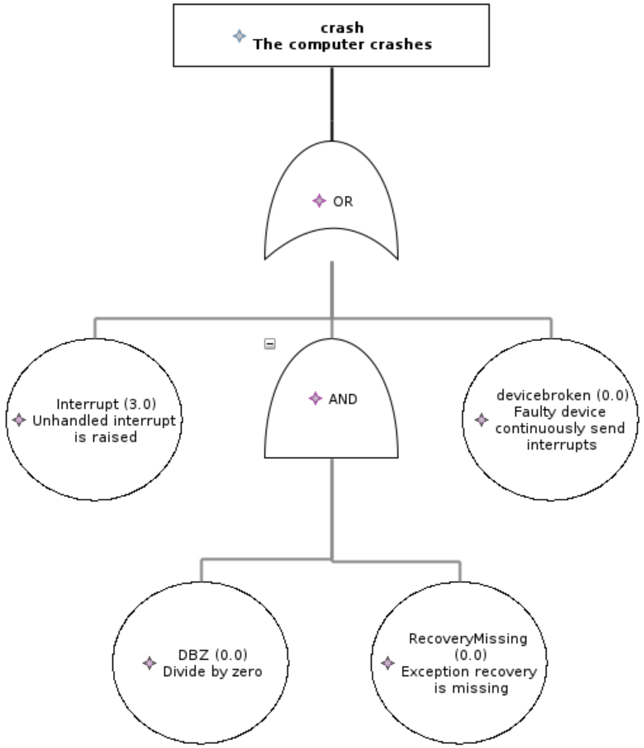
- Incoming error events

Walk through the components hierarchy

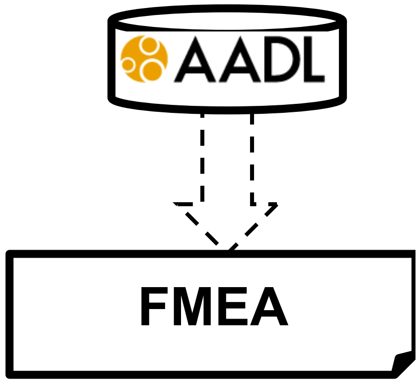
- Generate the complete fault-tree
- Focus on specific AADL subcomponents

Export to several tools

- Commercial: CAFTA
- Open-Source: EMFTA, OpenFTA



Failure Mode and Effects Support



Use of **component error behavior**

- Error propagations rules (source, sink, etc.)
- Internal error events

Traverse all error paths

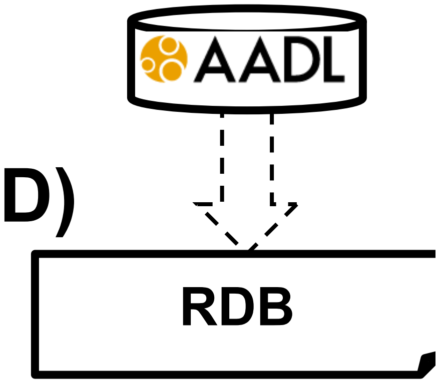
- Record impact over the components hierarchy

Use error description and related information

Create spreadsheet containing FHA elements

- To be reused by commercial or open-source tools

Reliability Block Diagram
aka ARP4761 Dependence Diagram (DD)



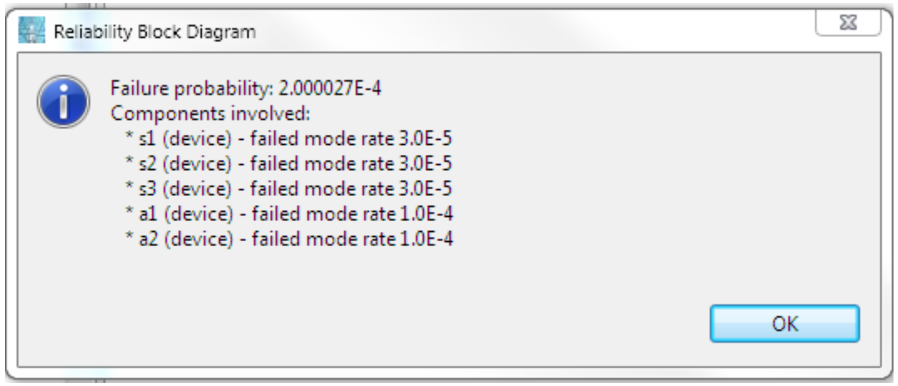
Use of **composite error behavior**

- Error propagations rules (source, sink, etc.)
- Internal error events

Compute reliability of the Dependence Diagram

- Use of recover and failure events
- Overall probability of system failure

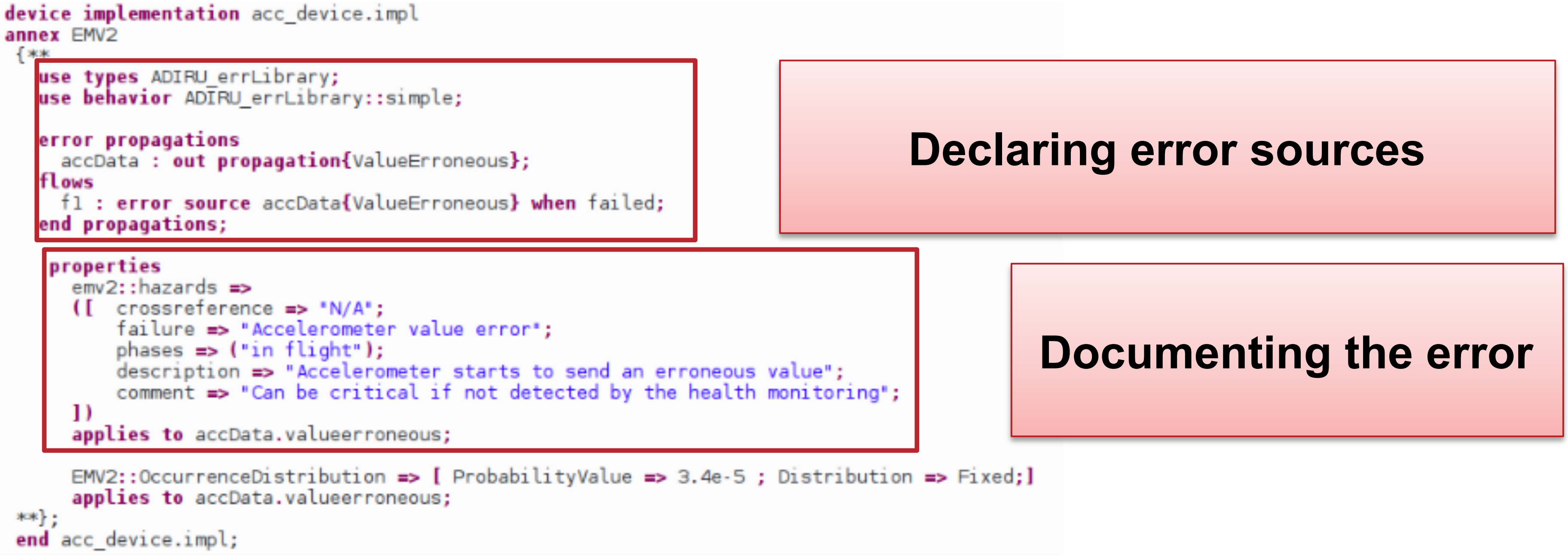
Support in OSATE (built-in)



Error Model Annex v2
Application to the ADIRU

Software Engineering Institute | Carnegie Mellon University

Annotating the model with Error Information (1)



Annotating the model with Error Information (2)

```
process implementation acc_process_emv2.impl extends acc_process.impl
subcomponents
  -- We extend the initial implementation, and add error modeling elements.

  acc1: refined to thread threads::acc1_dataOutput_emv2.impl
    { Classifier_Substitution_Rule => Type_Extension; };
  acc2: refined to thread threads::acc2_dataOutput_emv2.impl
    { Classifier_Substitution_Rule => Type_Extension; };
  acc3: refined to thread threads::acc3_dataOutput_emv2.impl
    { Classifier_Substitution_Rule => Type_Extension; };
  acc4: refined to thread threads::acc4_dataOutput_emv2.impl
    { Classifier_Substitution_Rule => Type_Extension; };
  acc5: refined to thread threads::acc5_dataOutput_emv2.impl
    { Classifier_Substitution_Rule => Type_Extension; };
  acc6: refined to thread threads::acc6_dataOutput_emv2.impl
    { Classifier_Substitution_Rule => Type_Extension; };

connections
  C21 : port acc1_input -> acc1.acc1_input;
  C22 : port acc2_input -> acc2.acc2_input;
  C23 : port acc3_input -> acc3.acc3_input;
  C24 : port acc4_input -> acc4.acc4_input;
  C25 : port acc5_input -> acc5.acc5_input;
  C26 : port acc6_input -> acc6.acc6_input;

annex EMV2{**
  use types ADIRU_errLibrary;
  use behavior ADIRU_errLibrary::simple;

  error propagations
    acc1_input  : in  propagation{ValueErroneous};
    acc1_output : out propagation{ValueErroneous};
    acc2_input  : in  propagation{ValueErroneous};
    acc2_output : out propagation{ValueErroneous};
    acc3_input  : in  propagation{ValueErroneous};
    acc3_output : out propagation{ValueErroneous};
    acc4_input  : in  propagation{ValueErroneous};
    acc4_output : out propagation{ValueErroneous};
    acc5_input  : in  propagation{ValueErroneous};
    acc5_output : out propagation{ValueErroneous};
    acc6_input  : in  propagation{ValueErroneous};
    acc6_output : out propagation{ValueErroneous};
  flows
    f1 : error path acc1_input{ValueErroneous} -> acc1_output{ValueErroneous};
    f2 : error path acc2_input{ValueErroneous} -> acc2_output{ValueErroneous};
    f3 : error path acc3_input{ValueErroneous} -> acc3_output{ValueErroneous};
    f4 : error path acc4_input{ValueErroneous} -> acc4_output{ValueErroneous};
    f5 : error path acc5_input{ValueErroneous} -> acc5_output{ValueErroneous};
    f6 : error path acc6_input{ValueErroneous} -> acc6_output{ValueErroneous};
  end propagations; **};
end acc_process_emv2.impl;
```

Passing the error directly through components features

Annotating the model with Error Information (3)

```
annex EMV2{**
 use types ADIRU_errLibrary;
 use behavior ADIRU_errLibrary::simple;

 error propagations
   acc1_input : in propagation{ValueErroneous};
   acc2_input : in propagation{ValueErroneous};
   acc3_input : in propagation{ValueErroneous};
   acc4_input : in propagation{ValueErroneous};
   acc5_input : in propagation{ValueErroneous};
   acc6_input : in propagation{ValueErroneous};
 flows
   f1 : error sink acc1_input{ValueErroneous};
   f2 : error sink acc2_input{ValueErroneous};
   f3 : error sink acc3_input{ValueErroneous};
   f4 : error sink acc4_input{ValueErroneous};
   f5 : error sink acc5_input{ValueErroneous};
   f6 : error sink acc6_input{ValueErroneous};
 end propagations;

 component error behavior
 transitions
     t1 : operational -[acc1_input{ValueErroneous}]-> failed;
     t2 : operational -[acc2_input{ValueErroneous}]-> failed;
     t3 : operational -[acc3_input{ValueErroneous}]-> failed;
     t4 : operational -[acc4_input{ValueErroneous}]-> failed;
     t5 : operational -[acc5_input{ValueErroneous}]-> failed;
     t6 : operational -[acc6_input{ValueErroneous}]-> failed;
 detections
    operational -[1 ormore(acc1_input{ValueErroneous})]-> acc_error_out!;
    operational -[1 ormore(acc2_input{ValueErroneous})]-> acc_error_out!;
    operational -[1 ormore(acc3_input{ValueErroneous})]-> acc_error_out!;
    operational -[1 ormore(acc4_input{ValueErroneous})]-> acc_error_out!;
    operational -[1 ormore(acc5_input{ValueErroneous})]-> acc_error_out!;
    operational -[1 ormore(acc6_input{ValueErroneous})]-> acc_error_out!;
 end component;
**};
```

Receiving a erroneous value makes the component to fail

Functional Hazard Assessment

Component	Error	Hazard Description	ossreferer	Functional Failure	Operational Phases	Comment
acc1	"ValueErroneous on accData"	"Accelerometer starts to send an erroneous value"	"N/A"	"Accelerometer value error"	"in flight"	"Can be critical if not detected by the health monitoring"
acc2	"ValueErroneous on accData"	"Accelerometer starts to send an erroneous value"	"N/A"	"Accelerometer value error"	"in flight"	"Can be critical if not detected by the health monitoring"
acc3	"ValueErroneous on accData"	"Accelerometer starts to send an erroneous value"	"N/A"	"Accelerometer value error"	"in flight"	"Can be critical if not detected by the health monitoring"
acc4	"ValueErroneous on accData"	"Accelerometer starts to send an erroneous value"	"N/A"	"Accelerometer value error"	"in flight"	"Can be critical if not detected by the health monitoring"
acc5	"ValueErroneous on accData"	"Accelerometer starts to send an erroneous value"	"N/A"	"Accelerometer value error"	"in flight"	"Can be critical if not detected by the health monitoring"
acc6	"ValueErroneous on accData"	"Accelerometer starts to send an erroneous value"	"N/A"	"Accelerometer value error"	"in flight"	"Can be critical if not detected by the health monitoring"

List all potential error sources

Include documentation from the model

Required by ARP4761 safety standard

Fault Impact Analysis

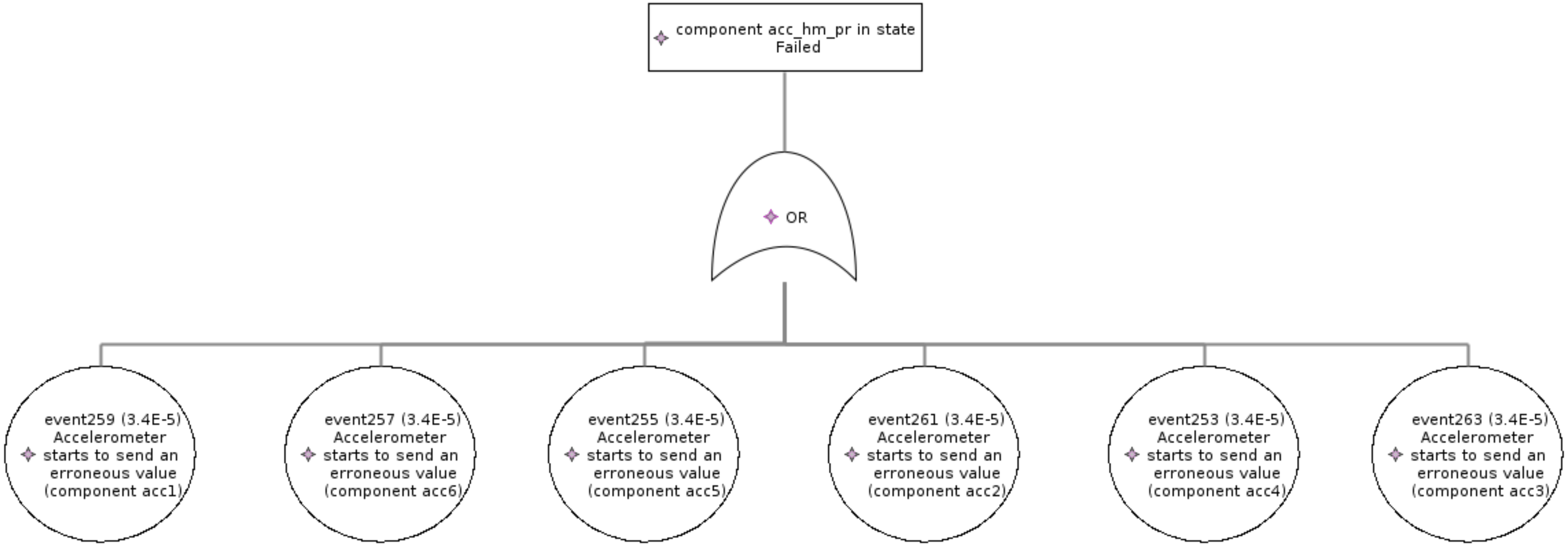
Component	Initial Failure Mode	1st Level Effect	Failure Mode	second Level Effect	Failure Mode
acc1	Failed	{ValueErroneous} accData -> acc_pr:acc1_input	acc_pr {ValueErroneous}	{ValueErroneous} acc1_output -> acc_hm_pr:acc1_input	acc_hm_pr {ValueErroneous} [Masked]
acc2	Failed	{ValueErroneous} accData -> acc_pr:acc2_input	acc_pr {ValueErroneous}	{ValueErroneous} acc2_output -> acc_hm_pr:acc2_input	acc_hm_pr {ValueErroneous} [Masked]
acc3	Failed	{ValueErroneous} accData -> acc_pr:acc3_input	acc_pr {ValueErroneous}	{ValueErroneous} acc3_output -> acc_hm_pr:acc3_input	acc_hm_pr {ValueErroneous} [Masked]
acc4	Failed	{ValueErroneous} accData -> acc_pr:acc4_input	acc_pr {ValueErroneous}	{ValueErroneous} acc4_output -> acc_hm_pr:acc4_input	acc_hm_pr {ValueErroneous} [Masked]
acc5	Failed	{ValueErroneous} accData -> acc_pr:acc5_input	acc_pr {ValueErroneous}	{ValueErroneous} acc5_output -> acc_hm_pr:acc5_input	acc_hm_pr {ValueErroneous} [Masked]
acc6	Failed	{ValueErroneous} accData -> acc_pr:acc6_input	acc_pr {ValueErroneous}	{ValueErroneous} acc6_output -> acc_hm_pr:acc6_input	acc_hm_pr {ValueErroneous} [Masked]

Bottom-up approach

Trace the error flow defined in the architecture

Required by ARP4761 safety standard

Fault Tree Analysis



Error Model Annex v2 Conclusion

Architecture Fault Modeling Summary

Architecture Fault Modeling with AADL

- Error Model Annex was originally published in 2006
  - Supported in AADL V1 and AADL V2
- Standardized Error Model Annex (V2) based on user experiences
- Error Model V2 concepts and ontology can be applied to other modeling notations

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, WW Technology)
- FHA, FMEA, fault tree, Markov models, stochastic Petri net generation from AADL/Error Model
  - Open source implementation as part of Error Model V2 publication

References

Website www.aadl.info

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

EMFTA https://github.com/juli1/emfta

Dependability Modeling with AADL (EMV1), SEI Technical Report, 2006.

Draft Error Model V2 Annex Standard, in ballot. Available on request.

AADL Fault Modeling and Analysis Within an ARP4761 Safety Assessment, SEI Technical Report, 2014.

Architecture Fault Modeling and Analysis with the Error Model Annex V2, SEI Technical Report, 2014 (awaiting completion of EMV2 ballot).