Opportunities and Challenges in Integrating System Safety Models into SysML:

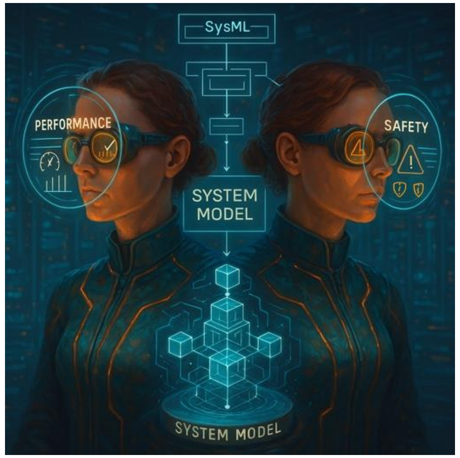
SysML-based Fault Tree Analysis

Lance Sherry, John Shortle, Matthew Amissah, Ali Raz

SERC: AI4SE & SE4AI Workshop 2025

Sept 17,18, 2025

One Model, Multiple Views



Embedded Safety Model Analysis

George Mason University
Systems Engineering and Operations Research

Organization

1. Introduction and Motivation
2. Example: System Safety Analysis and MBSE/SysML Models
3. Advances in Fault Tree Analysis
    1. Representing Fault Trees in SysML
    2. Connected SysML Models (FT, BDD)
    3. Deriving FT from BDD, AD, and IBD
    4. Uncertainty Quantification for Fault Trees
    5. Interval Analysis for Fault Trees
    6. Identifying Common Cause Failures in FT
    7. Calculating Top-Level Probability for FTs with Common Cause
4. Future Work

1 System Safety Analysis (SSA)

- Safety is prevention of fatalities, injuries, property damage, financial losses
- SSA is structured process for identifying, analyzing, and mitigating hazards in complex engineered systems throughout their lifecycle
- SSA supports early identification of design flaws, latent failures, and unsafe interactions across subsystems and human operators

- “Safety must be designed into the system from the beginning, not added as an afterthought.”
  — Leveson (2012), Engineering a Safer World

Leveson, N. G. (2012). *Engineering a Safer World: Systems Thinking Applied to Safety*. MIT Press.

1 Importance of SSA

- Reduces cost and time by catching hazards early in the design phase
- Increases public trust, system reliability, and mission assurance
- Supports certification and compliance with regulatory standards (e.g., MIL-STD-882E, ARP4761)
  - Regulatory standards designed to protect public from deploying and operating unsafe systems

1 SSA Artifacts in the Life-cycle

- Concept Phase: Preliminary Hazard Analysis (PHA)
- Design Phase: FTA, FMEA, STPA, model-based hazard simulations
- Test/Validation: Verification of safety requirements
- Operations: Continuous monitoring and feedback loops

1 Integration of SSA and SysML-based MBSE

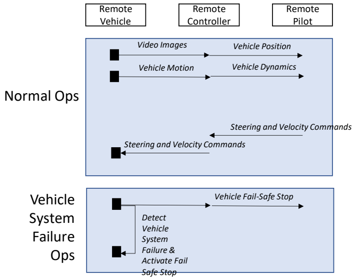
- MBSE (using SysML) is structured framework for modeling requirements, behaviors, structure, and parametrics of complex systems
- SSA can leverage these models to trace hazards, validate safety requirements, and analyze failure propagation early in the lifecycle

- "Integrating safety analysis into SysML models provides visibility into design risks and supports traceable safety assurance." — Friedenthal et al. (2014)

Friedenthal, S., Moore, A., & Steiner, R. (2014). A Practical Guide to SysML: The Systems Modeling Language. Morgan Kaufmann
Thomas, J., Fleming, C. H., & Leveson, N. G. (2021). "STPA Handbook." MIT Partnership for Systems Approaches to Safety and Security.

1 System Engineering and System Safety Artifacts

Con-Ops



Mission Requirements

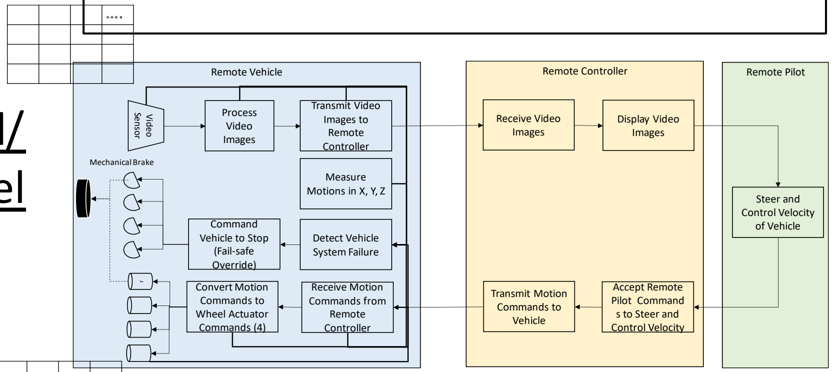
RPR Mission Requirements

….

MR: n The RPR System (vehicle + controller + operator) shall not exceed a fatality, injury, or property damage rate of 10E-3

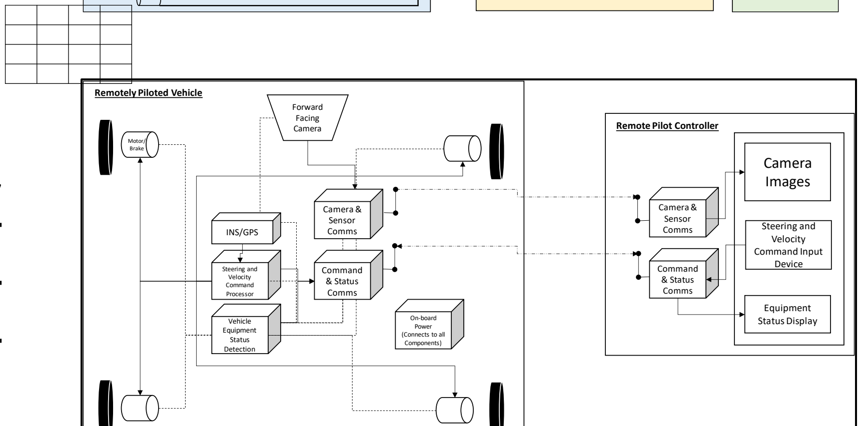
Req to Functions Traceability Matrix

Functional/ Logical Model



Functions to Components Traceability Matrix

Component/ Physical Model

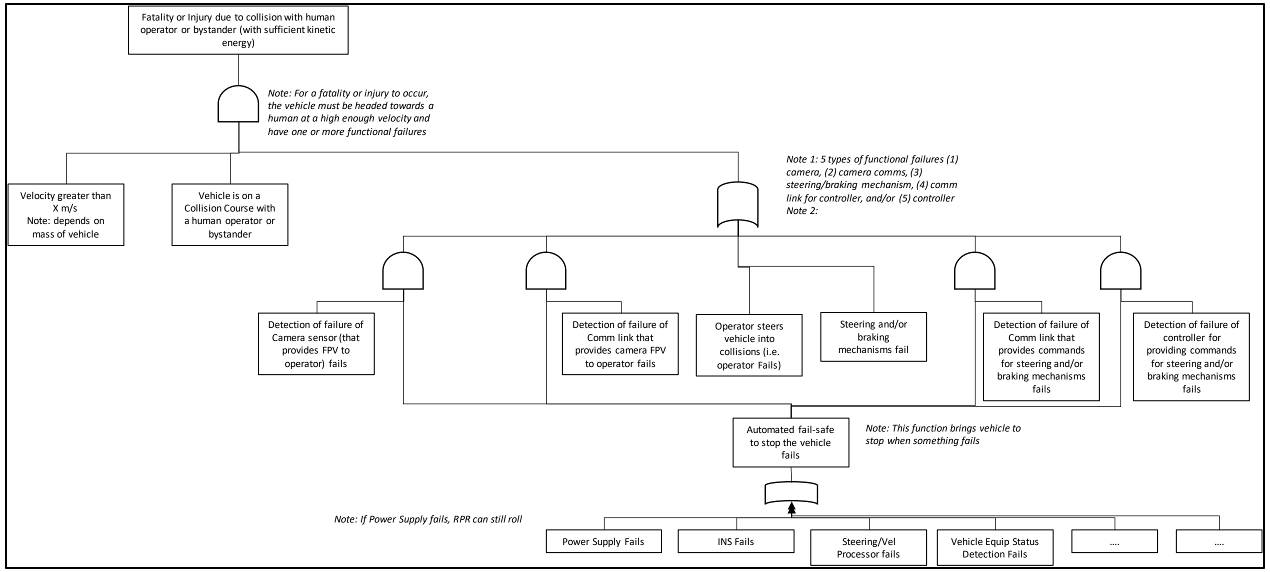


Hazard Analysis

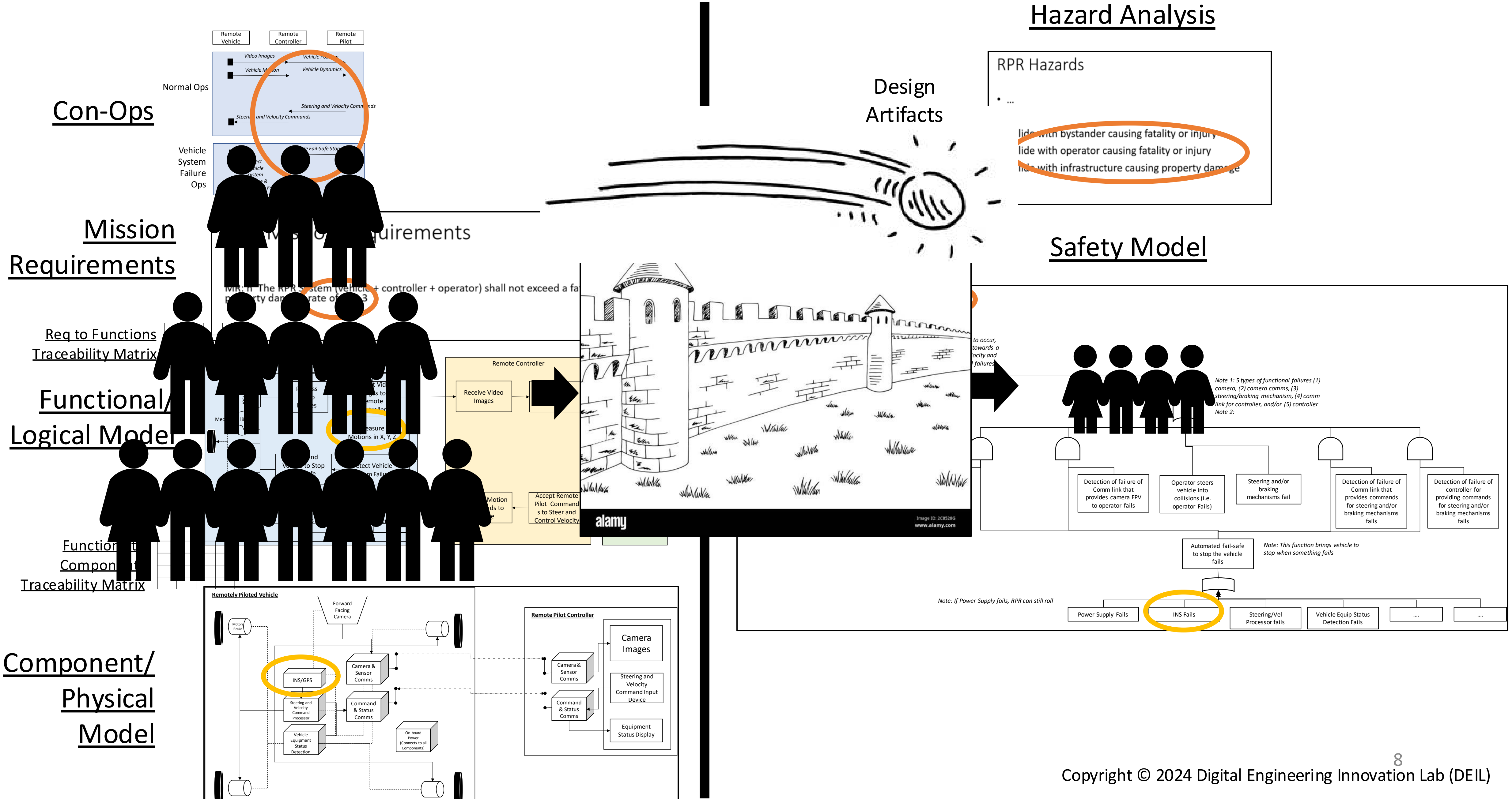
RPR Hazards

- …
- ….
- Collide with bystander causing fatality or injury
- Collide with operator causing fatality or injury
- Collide with infrastructure causing property damage
- …

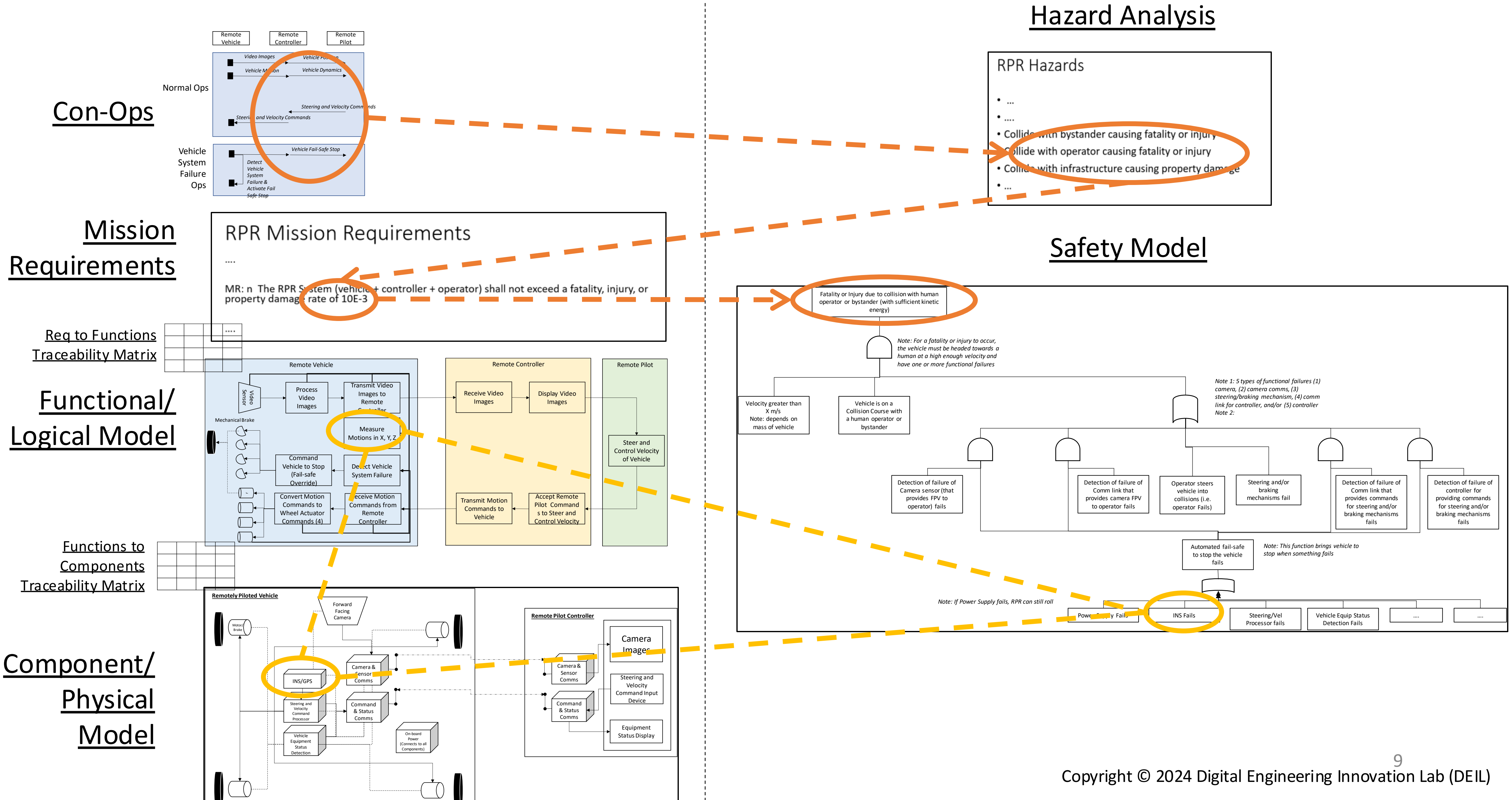
Safety Model



1 Organizational Structure



1 System Engineering and System Safety Artifacts



Inefficiencies in Separating SSA and System Design

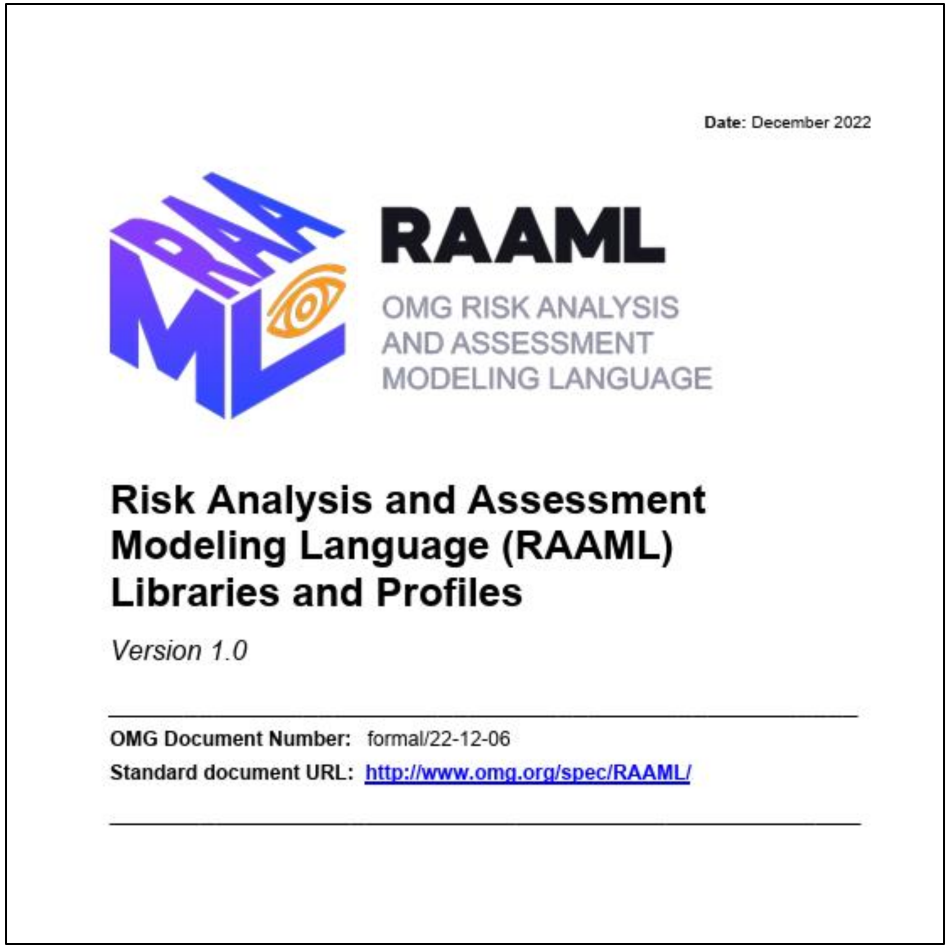
- SSA work on old designs
- Errors introduced in “interpretation” of designs
- Inability to make design tradeoffs

Risk Analysis and Assessment Modeling Language (RAAML)

- A SysML-based profile developed by the Object Management Group (OMG)

- Extends SysML with stereotypes for:
  - FMEA (Failure Modes and Effects Analysis)
  - FTA (Fault Tree Analysis)
  - STPA (System-Theoretic Process Analysis)
  - RBD (Reliability Block Diagrams)

- Integrates risk analysis directly into the model-based systems engineering (MBSE) workflow.
  - provides a standardized way to model safety and reliability analysis artifacts within a SysML model


Date: December 2022

RAAML

OMG RISK ANALYSIS AND ASSESSMENT MODELING LANGUAGE

Risk Analysis and Assessment Modeling Language (RAAML) Libraries and Profiles

Version 1.0

OMG Document Number: formal/22-12-06

Standard document URL: http://www.omg.org/spec/RAAML/

Example RAAML

- BrakeControlUnit is a system component modeled with SysML Block
- PressureSensor is annotated with <<FMEAElement>> for risk modeling
- A Failure Mode (Sensor stuck high) is linked directly to the component
- An FTA (Fault Tree Analysis) is constructed using <<FTAEvent>> and <<LogicGate>> stereotypes to model the hazard “Loss of Braking”
- FTA shows how component failures propagate to system-level hazards

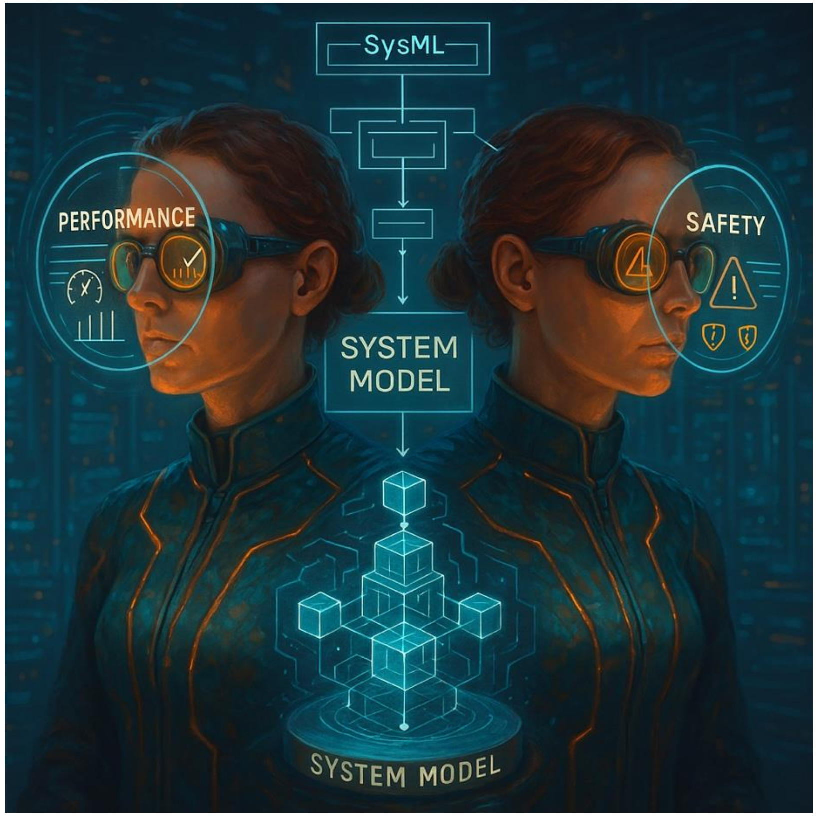
```
+--------------------+
|    <<Block>>          |
|  BrakeControlUnit    |
+--------------------+
| - pressureSensor: Sensor
| - actuator: Actuator   |
+--------------------+
        |
        v
+--------------------+
|  <<FMEAElement>>      |
|  PressureSensor        |
+--------------------+
| <<FailureMode>>:       |
|  "Sensor stuck high"     |
+--------------------+

        |
        v
+--------------------+
|  <<FTAEvent>>       |
|  LossOfBraking      |
+--------------------+
        ^
        |
+--------------------+
|  <<LogicGate>>      |
|      OR              |
+--------------------+
            ^
       /          \
      /            \
     v              v
+-------------+  +---------------+
| <<FTAEvent>>  |  | <<FTAEvent>>    |
| SensorFails   |  | ActuatorFails   |
+-------------+  +---------------+
```

1 Integration of SSA and SysML based MBSE

- Enables **bidirectional traceability** between safety artifacts and system models

- Enables automated analysis of safety models



- Friedenthal, S., Moore, A., & Steiner, R. (2014). A Practical Guide to SysML: The Systems Modeling Language. Morgan Kaufmann
- Thomas, J., Fleming, C. H., & Leveson, N. G. (2021). "STPA Handbook." MIT Partnership for Systems Approaches to Safety and Security.
- Thomas, J., & Leveson, N. (2013). "Performing STPA with SysML." MIT Partnership for Systems Approaches to Safety and Security (PSASS).
- Eames, D. P., & Steiner, R. (2017). "Bringing Safety-Critical Systems into MBSE." INCOSE International Symposium, 27(1), 477–489
- JPL/NASA. (2021). OpenMBEE User Guide https://openmbee.org/
- Lucio, L., et al. (2021). "Collaborative MBSE with OpenMBEE: A NASA Use Case." INCOSE IS 2021 Proceedings

Organization

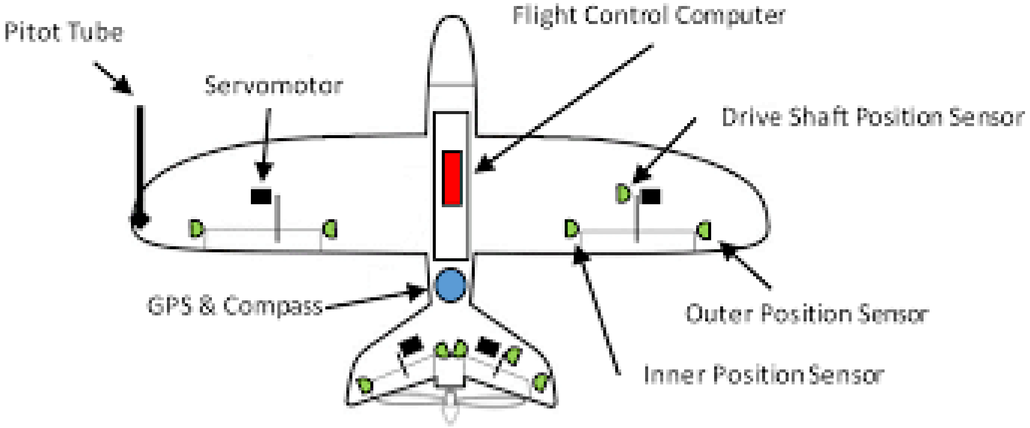
1. Introduction and Motivation
2. Example: System Safety Analysis and MBSE/SysML Models
3. Advances in Fault Tree Analysis
    1. Representing Fault Trees in SysML
    2. Connected SysML Models (FT, BDD)
    3. Deriving FT from BDD, AD, and IBD
    4. Uncertainty Quantification for Fault Trees
    5. Interval Analysis for Fault Trees
    6. Identifying Common Cause Failures in FT
    7. Calculating Top-Level Probability for FTs with Common Cause
4. Future Work

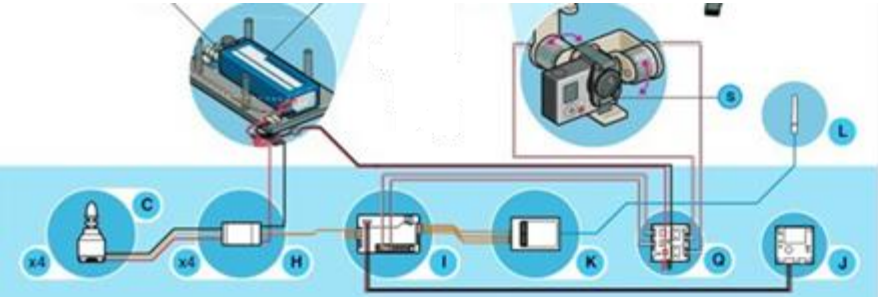
2 Example: SysML and SSA

CON-OPS

Hazard:
Loss of (Flight)
Control (LOC)

DESIGN





COMPONENTS

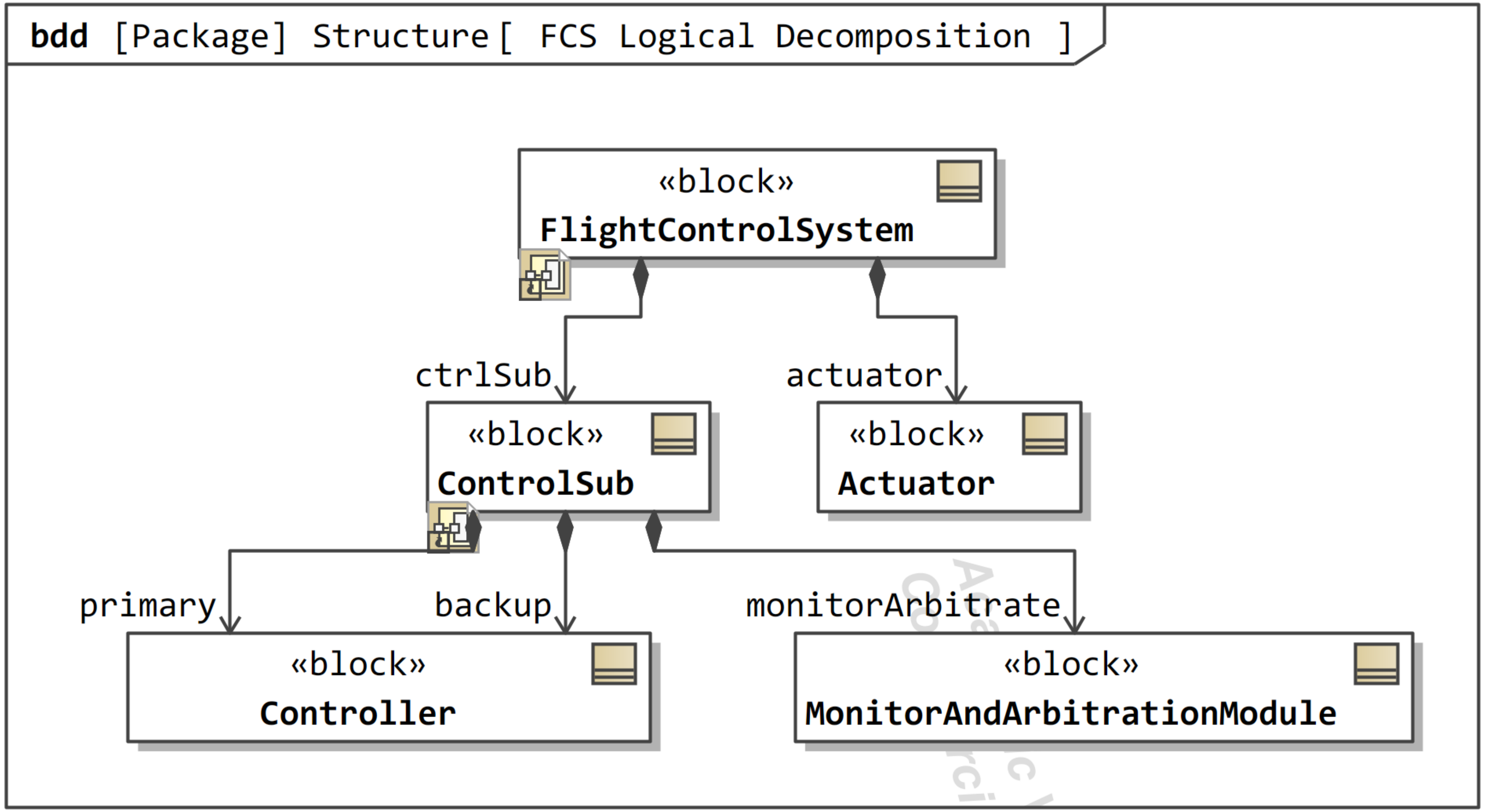
Sensor(s)

Flight Controllers

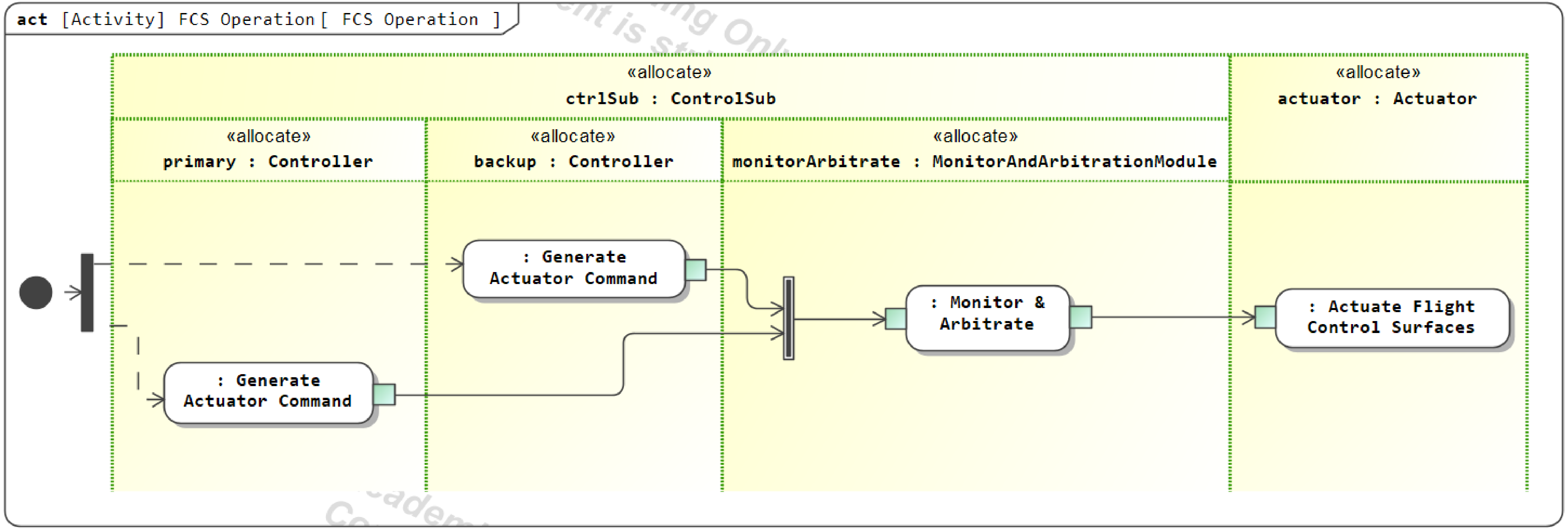
Monitor &
Arbitration

Actuator(s)

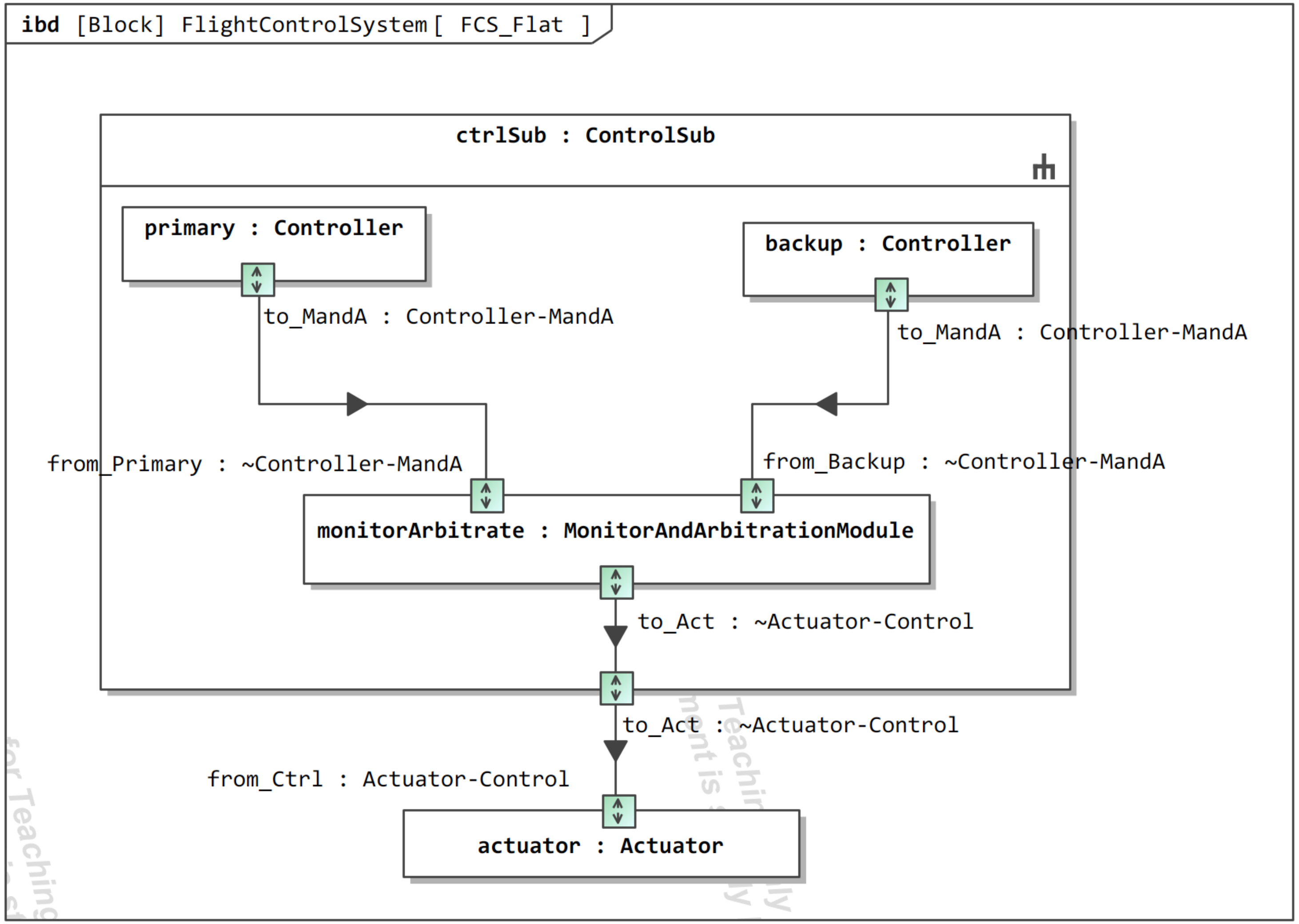
2 Example: SysML and SSA



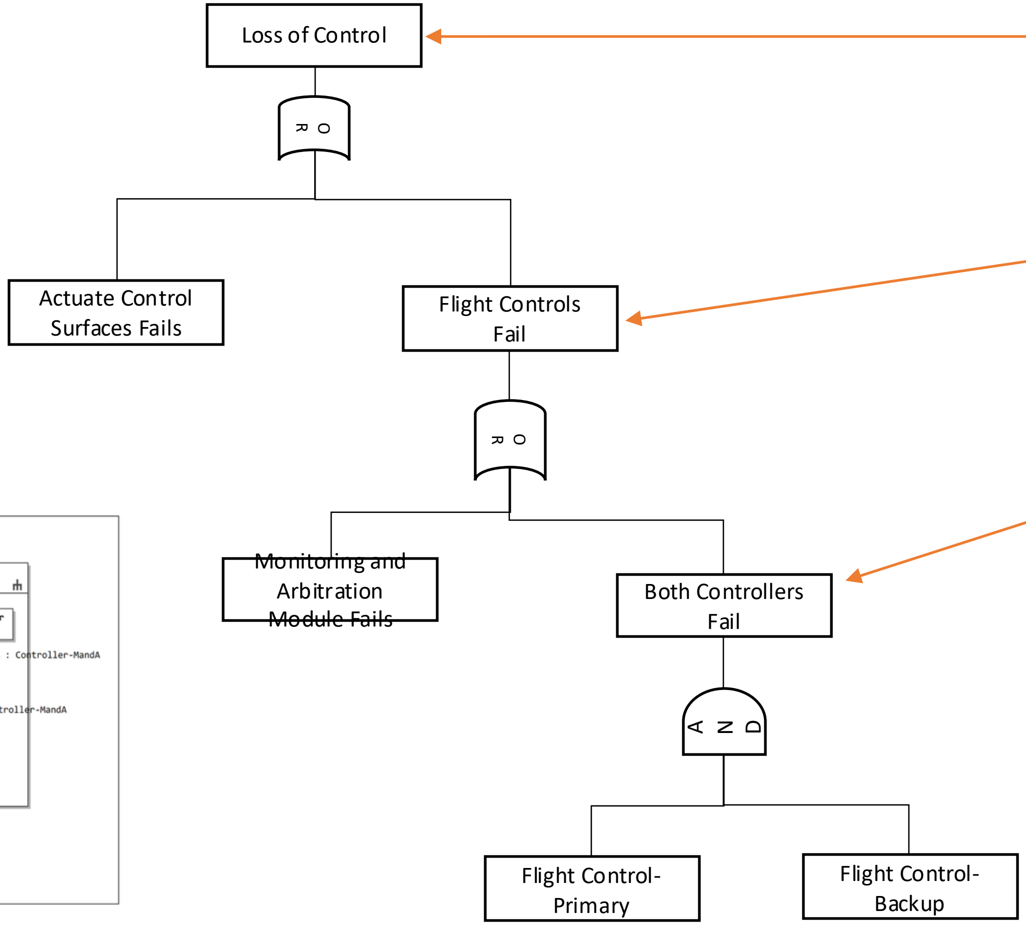
2 Example: SysML and SSA



2 Example: SysML and SSA



2 Example: SysML and SSA

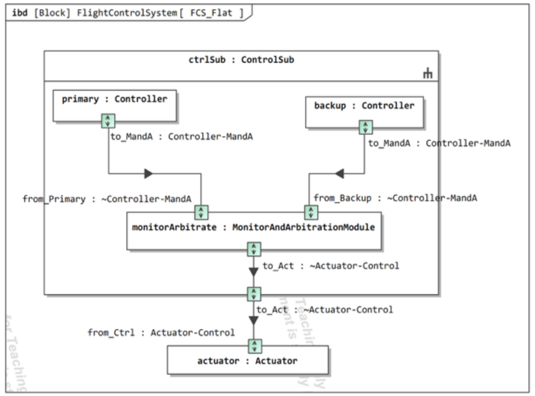


*P_LOC = (P_Actuate_Control_Surfaces + P_Flight_Control) - (P_Actuate_Control_Surfaces * P_Flight_Control)*

*P_Flight_Control = P_Monitoring & Arbitration + P_Flight Control - (P_Monitoring & Arbitration * P_Flight Control)*

*P_Flight Control = P_Primary * P_Backup*

Note: Assumes independence and *no common cause* or shared hardware

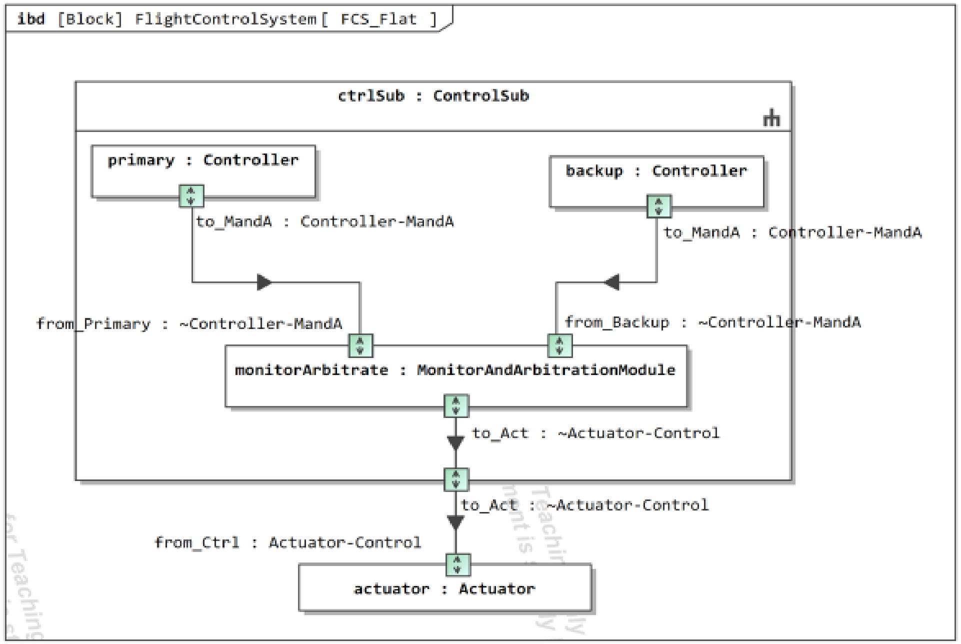


2 Example: SysML and SSA

1

Hazard: Loss of (Flight) Control

2

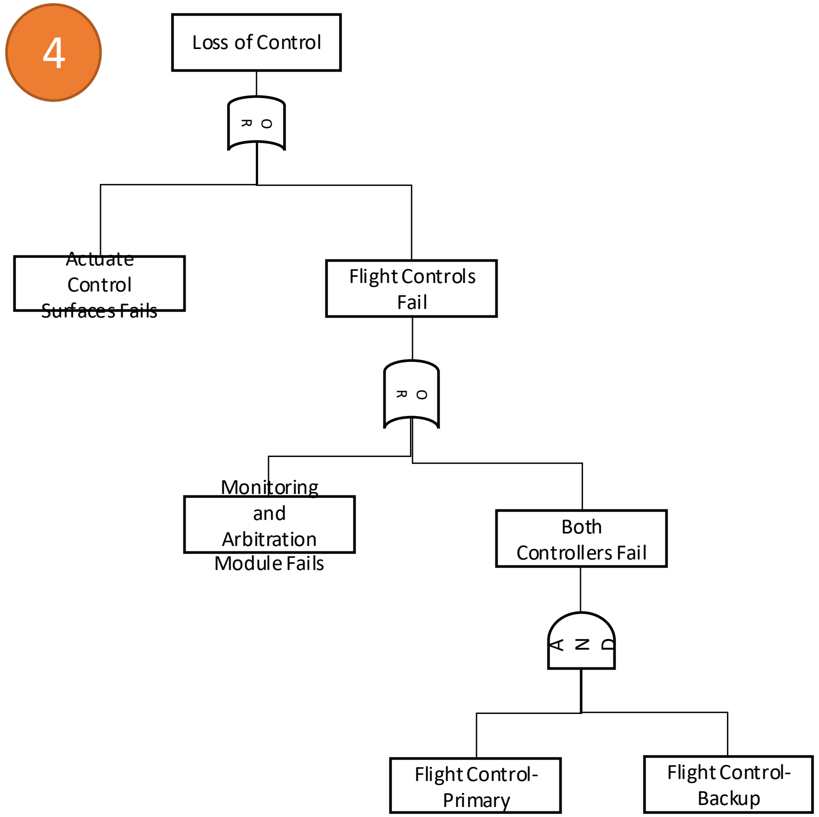


3

Causes of Loss of Control

- Failure: Actuate Control Surfaces
- Failure: Flight Control
  - Failure: Monitor & Arbitrate
  - Failure: Flight Control Primary and Flight Control – Back Up

4



5

*P_LOC = (P_Actuate_Control_Surfaces + P_Flight_Control) - (P_Actuate_Control_Surfaces * P_Flight_Control)*

*P_Flight_Control = P_Monitoring & Arbitration + P_Flight Control - (P_Monitoring & Arbitration * P_Flight Control)*

*P_Flight Control = P_Primary * P_Backup*

Note: Assumes independence and *no common cause* or shared hardware

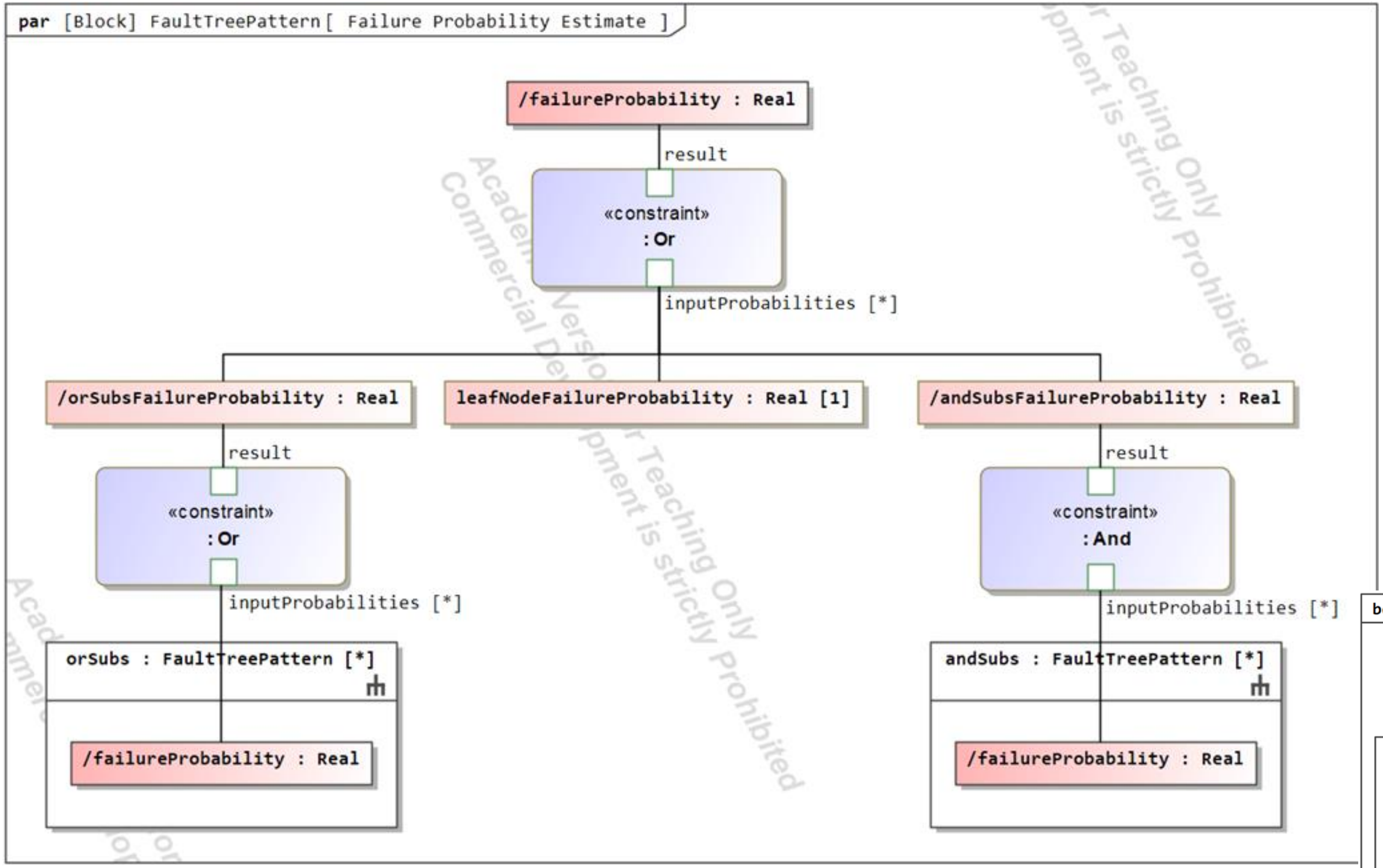
2 Challenges

1. Representing Fault Trees in SysML
2. Deriving FT from BDD, AD, and IBD
3. Connected Models for Bi-directional Traceability (FT, BDD)
4. Uncertainty Quantification for Fault Trees
5. Interval Analysis for Fault Trees
6. Identifying Common Cause Failures in Fault Trees
7. Calculating Top-Level Prob for FTs with Common Cause

Organization

1. Introduction and Motivation
2. Example: System Safety Analysis and MBSE/SysML Models
3. Advances in Fault Tree Analysis
    1. Representing Fault Trees in SysML
    2. Connected SysML Models (FT, BDD)
    3. Deriving FT from BDD, AD, and IBD
    4. Uncertainty Quantification for Fault Trees
    5. Interval Analysis for Fault Trees
    6. Identifying Common Cause Failures in FT
    7. Calculating Top-Level Probability for FTs with Common Cause
4. Future Work

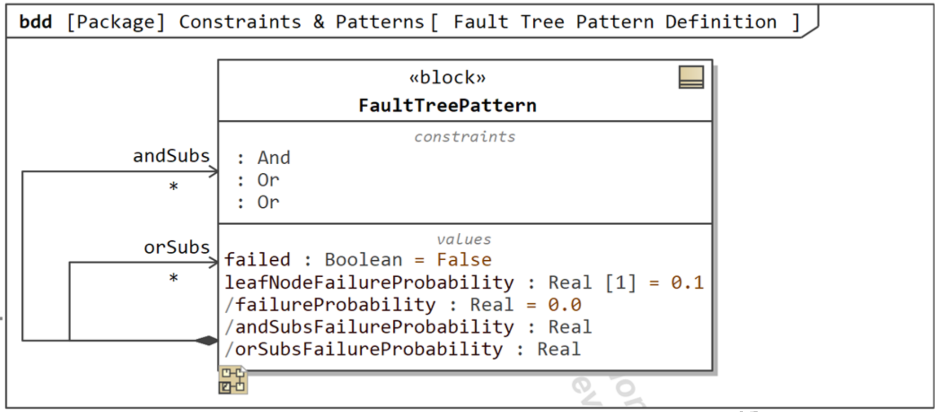
3-1 Representing FT in SysML – FT Templale



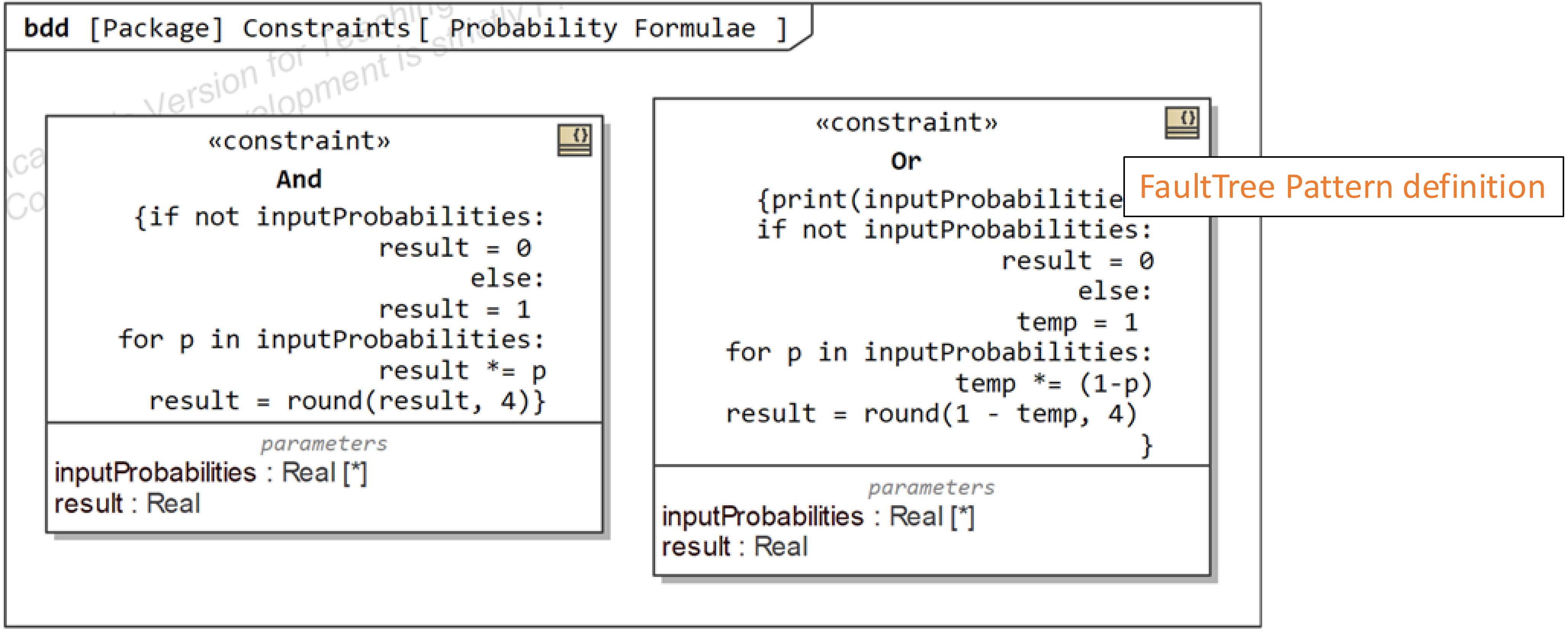
Parametric Diagram

FaultTree Pattern

Provides SysML template for capturing a Fault Tree

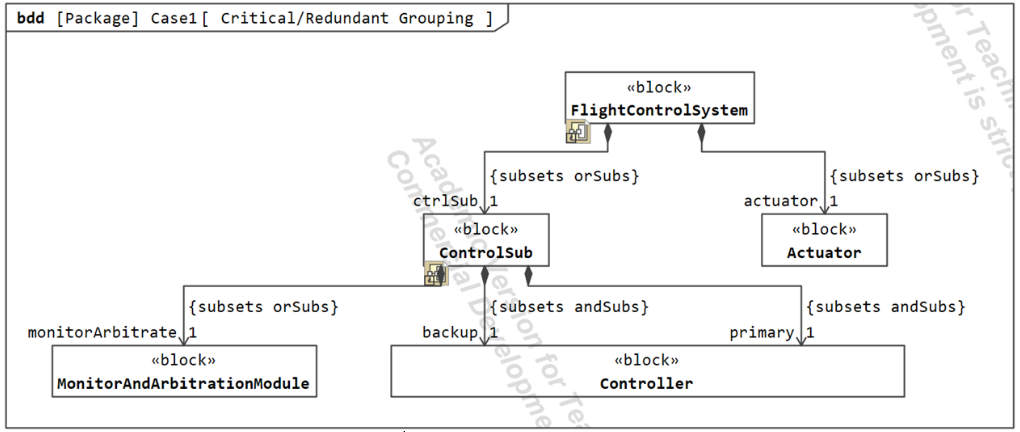


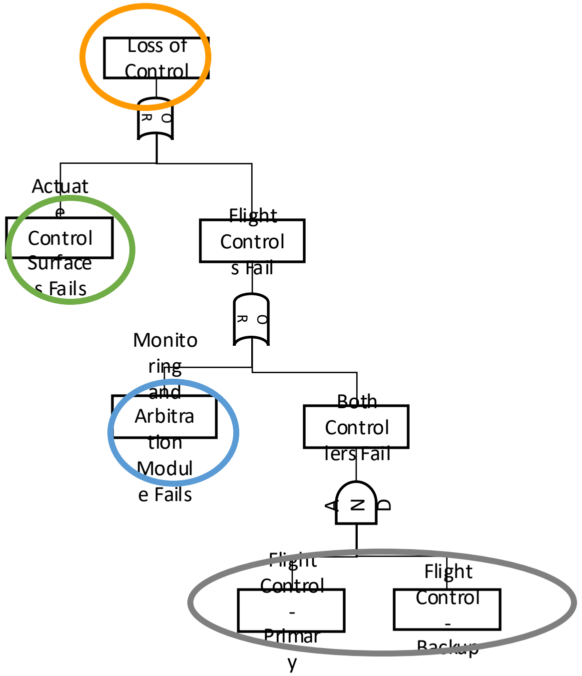
3-1 Representing FT in SysML



3-1 Representing FT in SysML

- Instance Table is an instantiation of the BDD
- Instance Table is the Fault Tree





#	Name	leafNodeFailureProba...	failureProbability	orSubs	andSubs
1	⊟ FCS	0	0.424	ctrlSub : Structure::Case1: actuator : Structure::Case	
2	actuator	0	0.1		
3	⊟ ctrlSub	0	0.36	monitor&Arbitrate : Struc	primaryCtrl : Structure::C backupCtrl : Structure::C
4	primaryCtrl	0.4	0.4		
5	backupCtrl	0.5	0.5		
6	monitor&Arbitrate	0.2	0.2		

Top-level Node

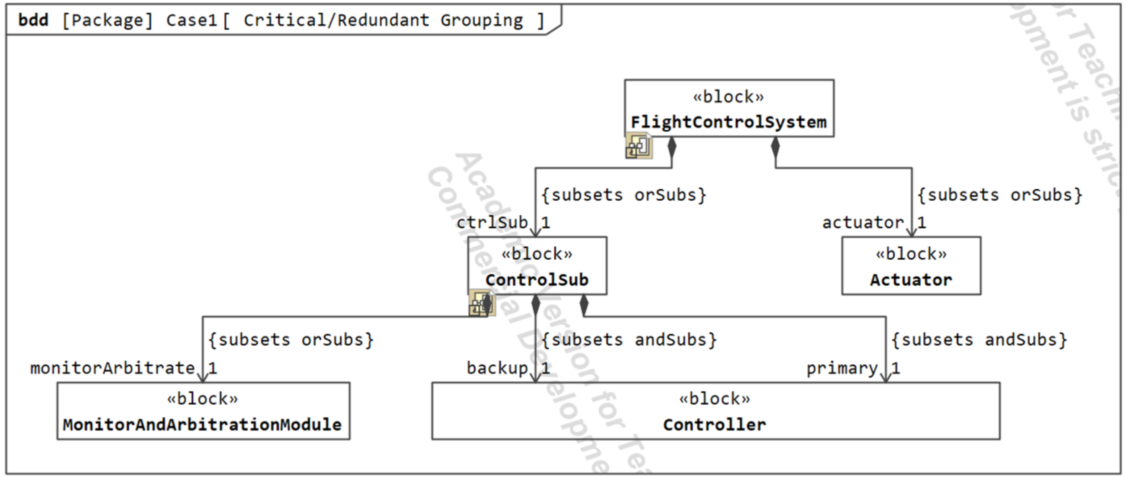
Intermediate Node

Leaf Nodes

3.2 Connected Models

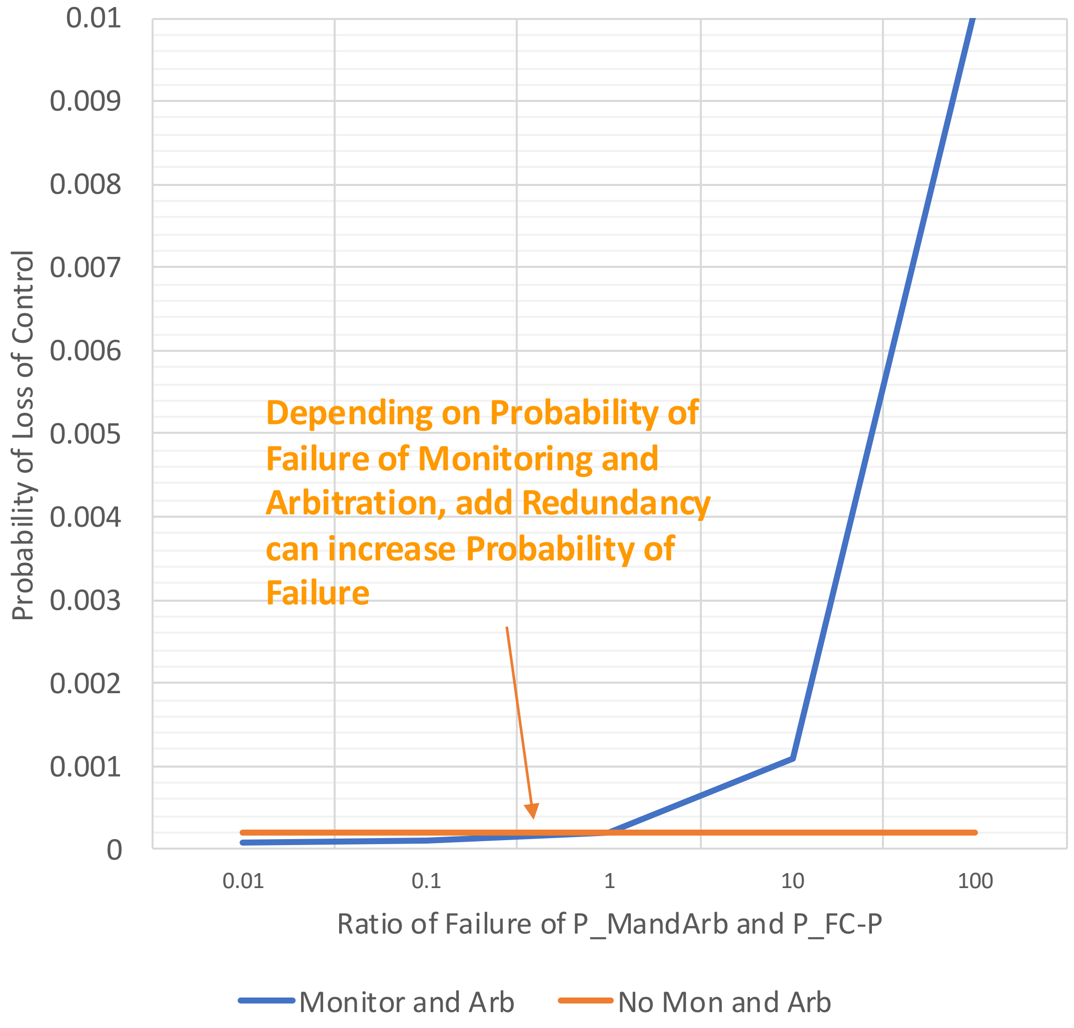
A change in the BDD is reflected in the Instance Table

A change in the Instance Table is reflected in the BDD



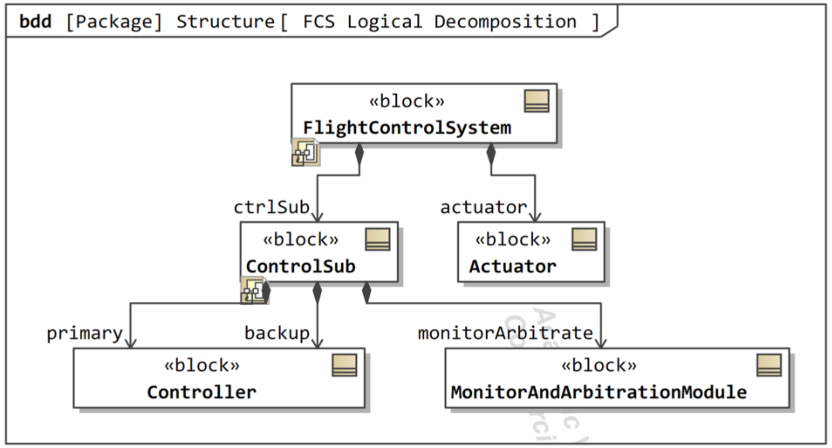
#	Name	leafNodeFailureProba...	failureProbability	orSubs	andSubs
1	FCS	1.0E-12	0.424	ctrlSub : Structure::Case1: actuator : Structure::Case	
2	actuator	0.1	0.1		
3	ctrlSub	1.0E-12	0.36	monitor&Arbitrate : Struc	primaryCtrl : Structure: backupCtrl : Structure::
4	primaryCtrl	0.4	0.4		
5	backupCtrl	0.5	0.5		
6	monitor&Arbitrate	0.2	0.2		

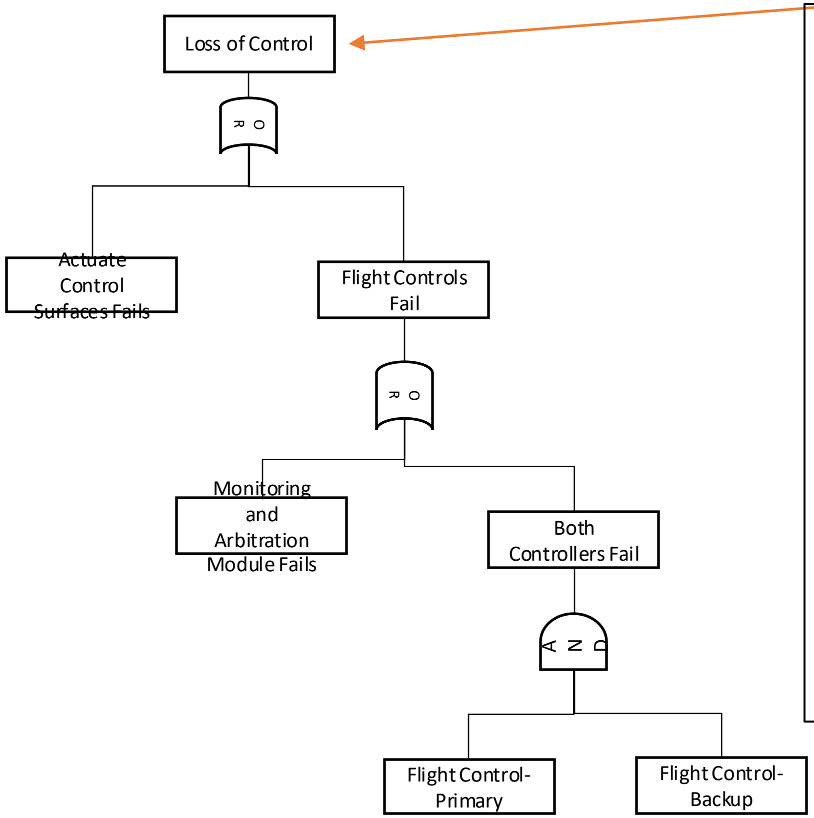
3-2 Connected Models



Probability of Failure of "Monitor and Arbitrate Module" cannot exceed a threshold before it becomes a liability

3-3 Deriving FTs





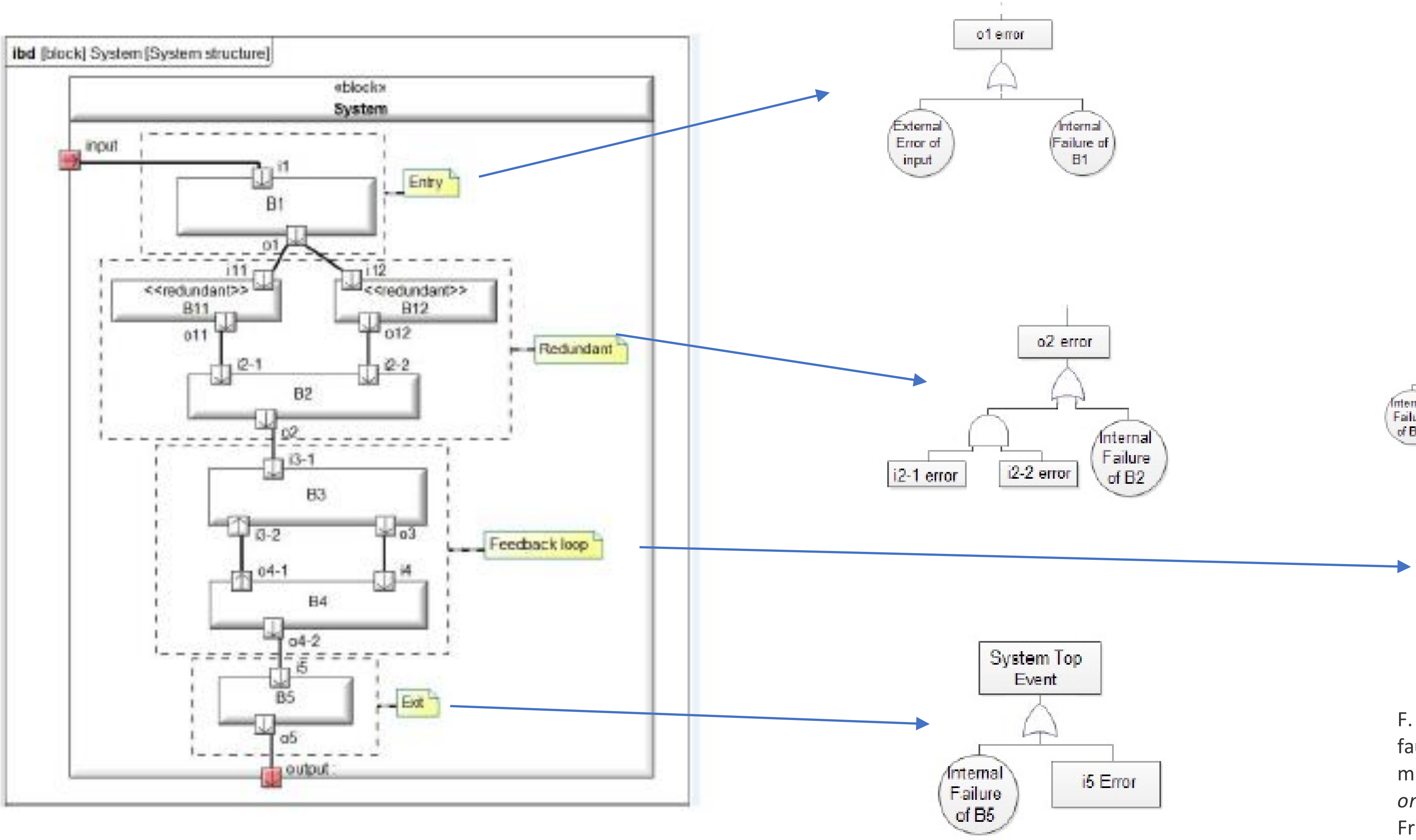
*P_LOC = (P_Actuate_Control_Surfaces + P_Flight_Control) - (P_Actuate_Control_Surfaces * P_Flight_Control)*

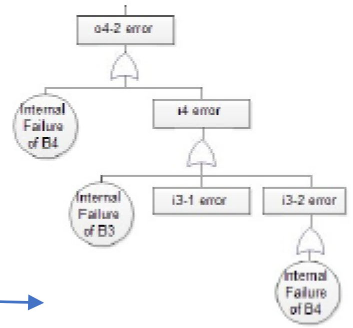
*P_Flight_Control = P_Monitoring & Arbitration + P_Flight Control - (P_Monitoring & Arbitration * P_Flight Control)*

*P_Flight Control = P_Primary * P_Backup*

Note: Assumes independence and *no common cause* or shared hardware

3-3 Deriving Fault Trees from SysML

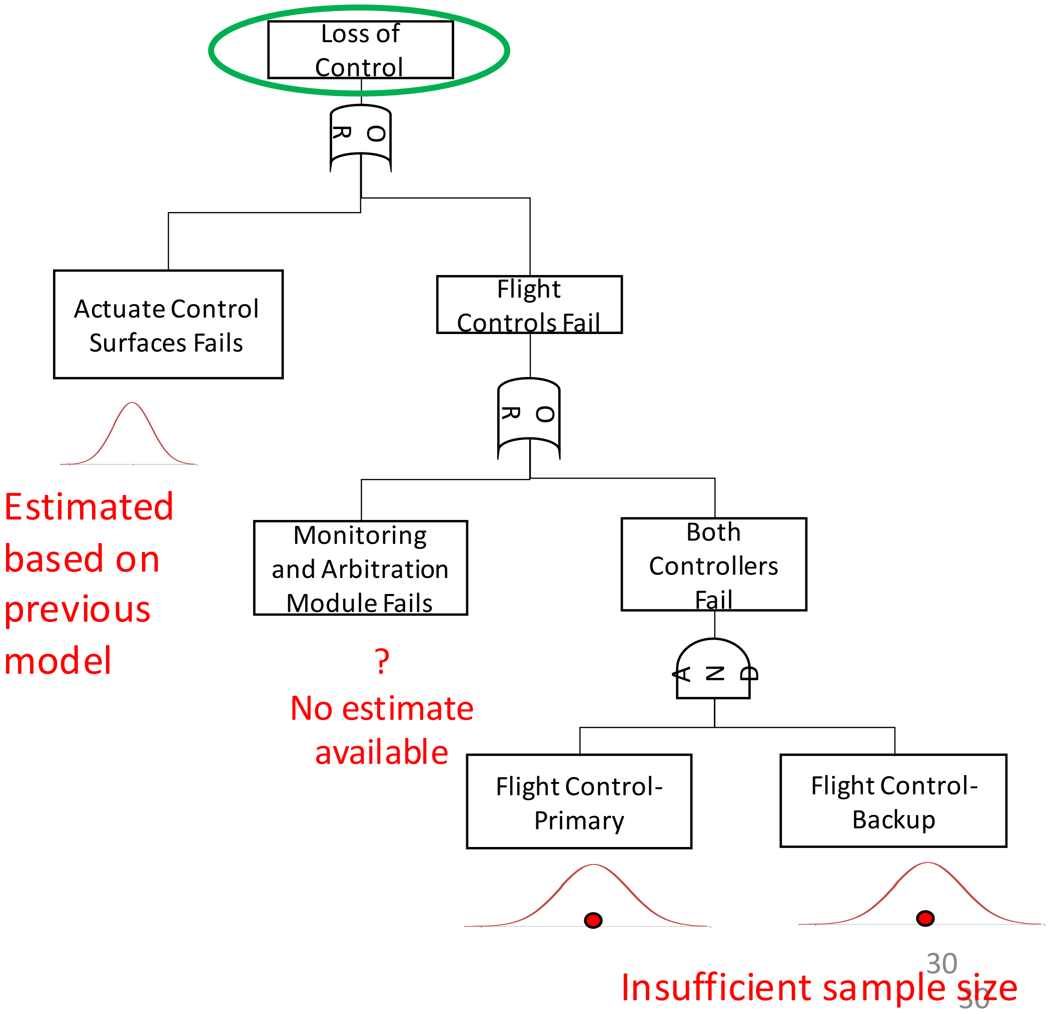




F. Mhenni, N. Nguyen and J. -Y. Choley, "Automatic fault tree generation from SysML system models," *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Besacon, France, 2014, pp. 715-720, doi: 10.1109/AIM.2014.6878163.

Challenges in Fault Tree Analysis

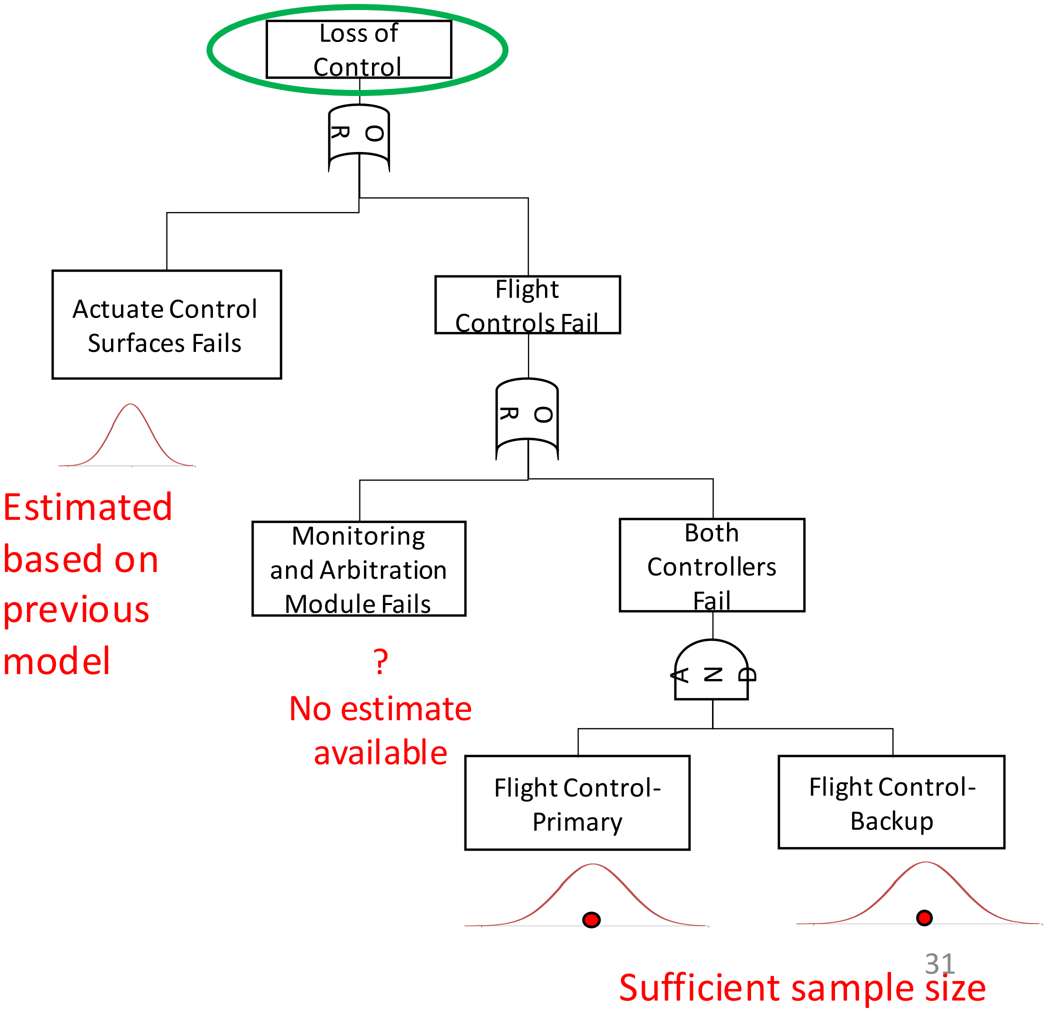
1. Rare event probabilities with estimates or small sample size from testing
2. Missing probabilities
   1. Estimated based on previous model
   2. No estimate available
   3. Insufficient Sample Size
3. Common cause failures



3.4 FT Uncertainty Analysis – SSA/Design Challenges

Design Challenge

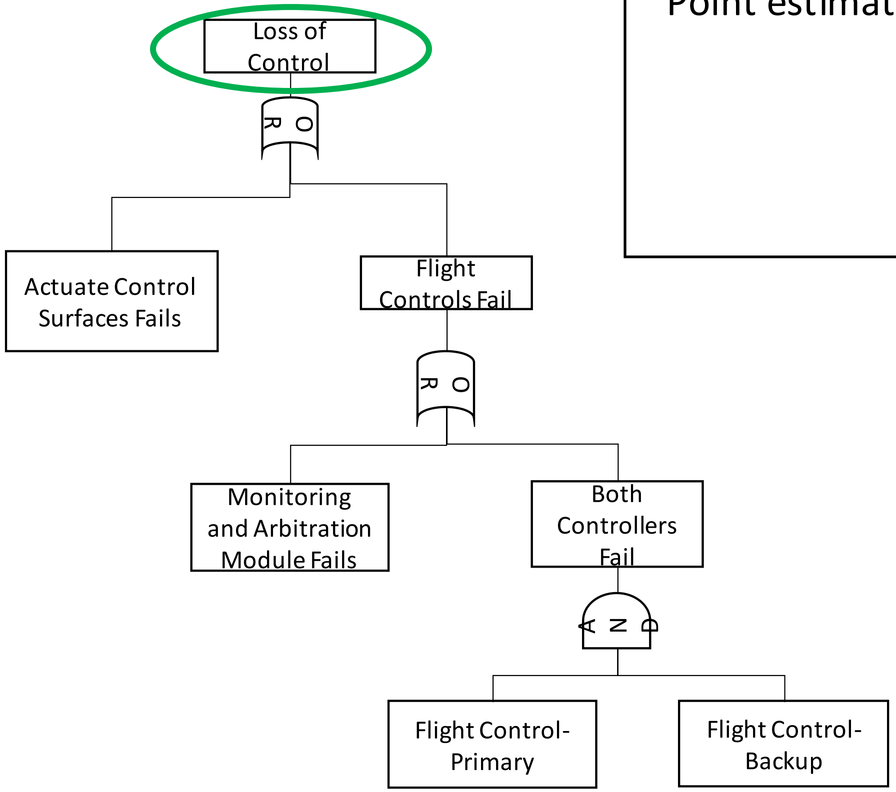
- FT parameters may have no supporting data for quantification
- FT parameters may be extreme events that are typically rare
  - arise from a combination of events that may have never been previously observed
- Rare-event nature of data makes point estimates inherently noisy

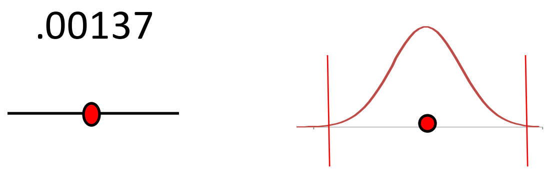


3.4 Fault Tree Uncertainty Analysis – Insufficient Sample Size

Background

- Probability of Top-Level Event is calculated as a Point-Estimate
    1. Point-Estimate is the "mode" of a distribution representing a Confidence Interval
    2. Point-Estimate is dependent on Point-Estimate probabilities in FT nodes with their own Uncertainty Distribution





Point estimate => Uncertainty distribution with Confidence Interval

Make decisions with appropriate to level of confidence in model

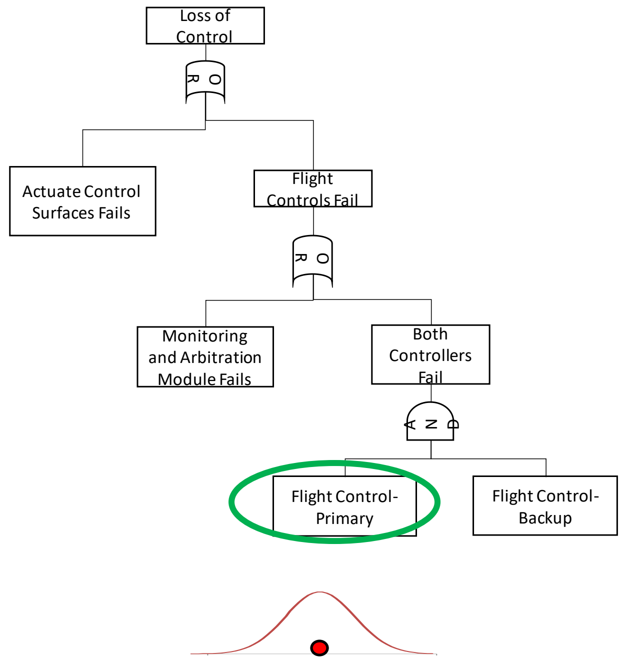
3.4 Fault Tree Uncertainty Analysis – Insufficient Sample Size

- Primary Flight Control Unit (PFCU):
  - tested for 1,000 hours
  - 1 failure is observed
  - ---> Estimated failure rate is 1E-3 per hour

- How accurate is this?

- Repeat the testing 10 times (i.e., 10 tests of 1,000 hours each)
  - Resulting failures in each test: 2, 1, 0, 1, 3, 1, 0, 0, 2, 1

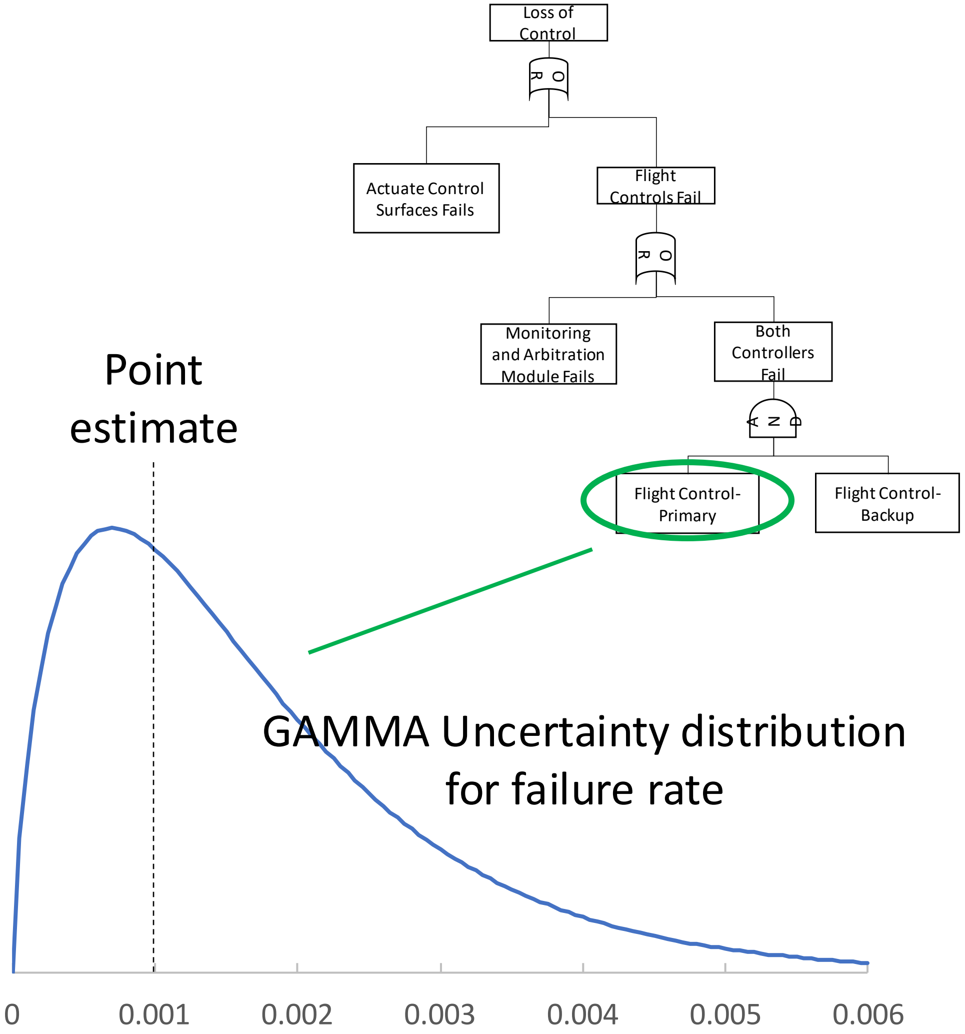
- If the true (unknown) failure probability is 1E-3, and the component is tested for 1,000 hours, for 63% of the tests) the number of failures observed will **not be 1** (i.e., 0 or 2 or 3…).



Sufficient sample size

3-4 FT Uncertainty - Replace Point Estimates with Distributions

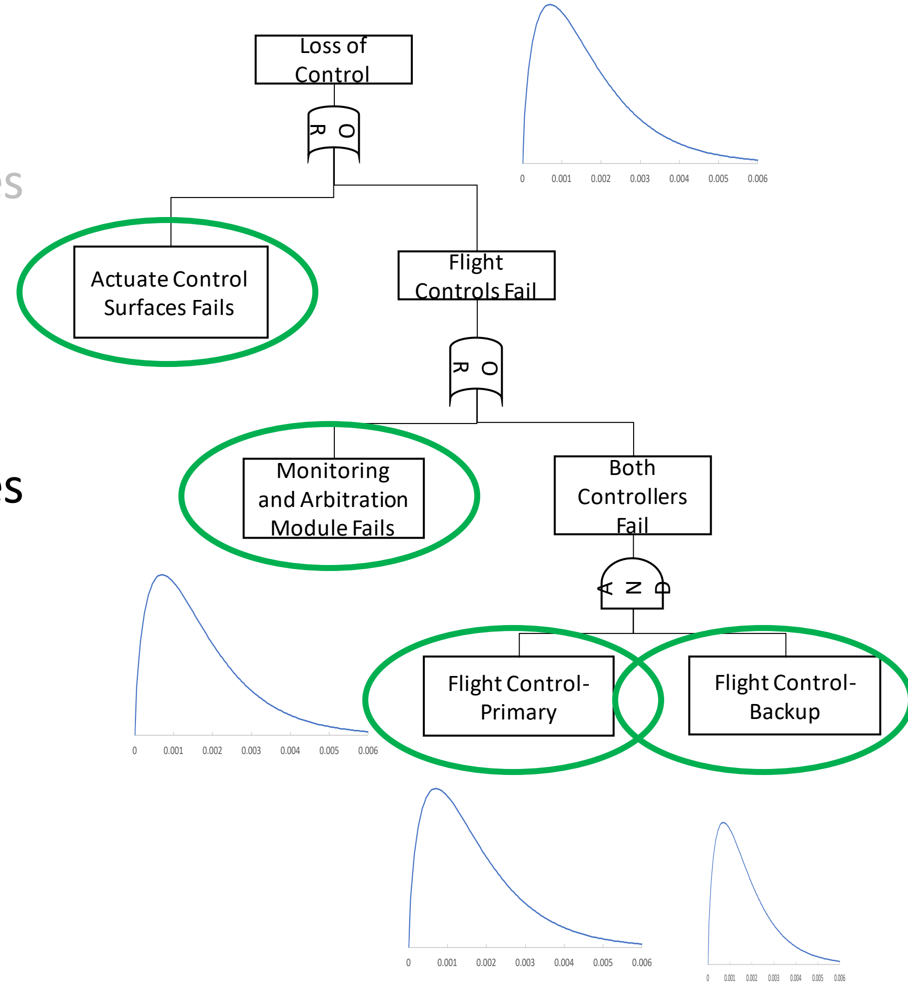
- Use GAMMA distribution with parameters α,□β
- Suppose 1 failure is observed in 1,000 hours of testing for primary flight controller
  - The point estimate for the failure rate is 1E-3
  - Based on the point estimate, the <u>uncertainty distribution</u> for the failure rate is a gamma distribution with parameters $\alpha = 1.5$, $\beta = 1{,}000$
- General method: If *k* failures observed from *n* trials, assign uncertainty distribution for failure probability as a gamma distribution with parameters $\alpha = 0.5 + k$, $\beta = n$
- Note: Approach based on Bayesian updating of Poisson distribution



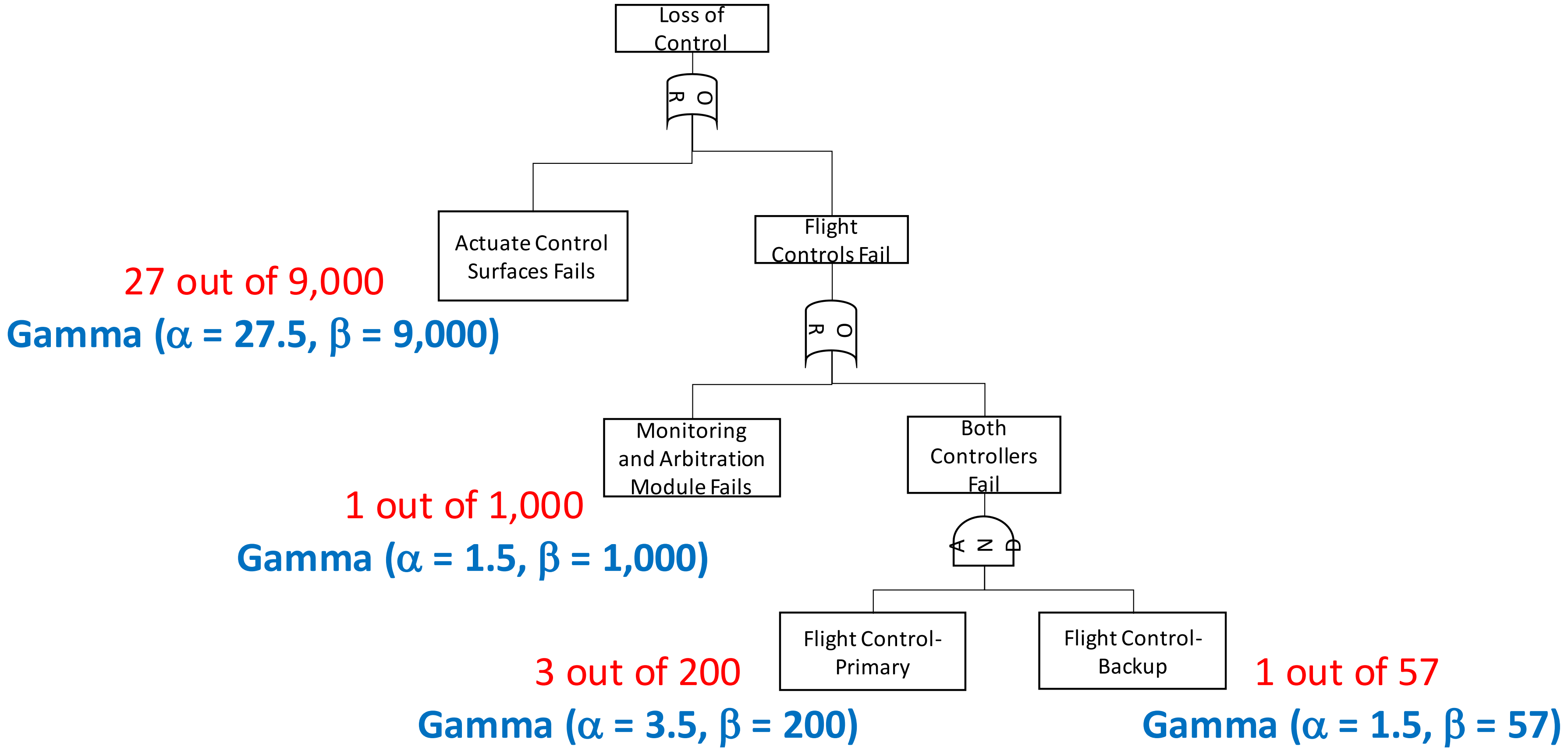
Nikdel, S., J. Shortle. 2023. A framework for uncertainty assessment in event tree safety models. Integrated Communications, Navigation, and Surveillance Conference, Herndon, VA.

3.4 FT Uncertainty – Calculating Top-Level Uncertainty

- Standard approach
  - Measure failure probability of each base event (e.g., *k* failures out of *n* trials)
  - Quantify each base event with its point estimate (e.g., *k*/*n*)
  - Quantify fault tree from bottom-up using AND/OR gate logic
- Uncertainty approach
  - Measure failure probability of each base event (e.g., *k* failures out of *n* trials)
  - Assign each base event an uncertainty distribution (gamma distribution with $\alpha = 0.5 + k$, $\beta = n$)
- Monte Carlo simulation loop:
  - For each base event, take a random draw its uncertainty distribution
  - Quantify the fault tree from bottom-up with AND/OR logic
- Assemble distribution of top-level event and any other events of interest



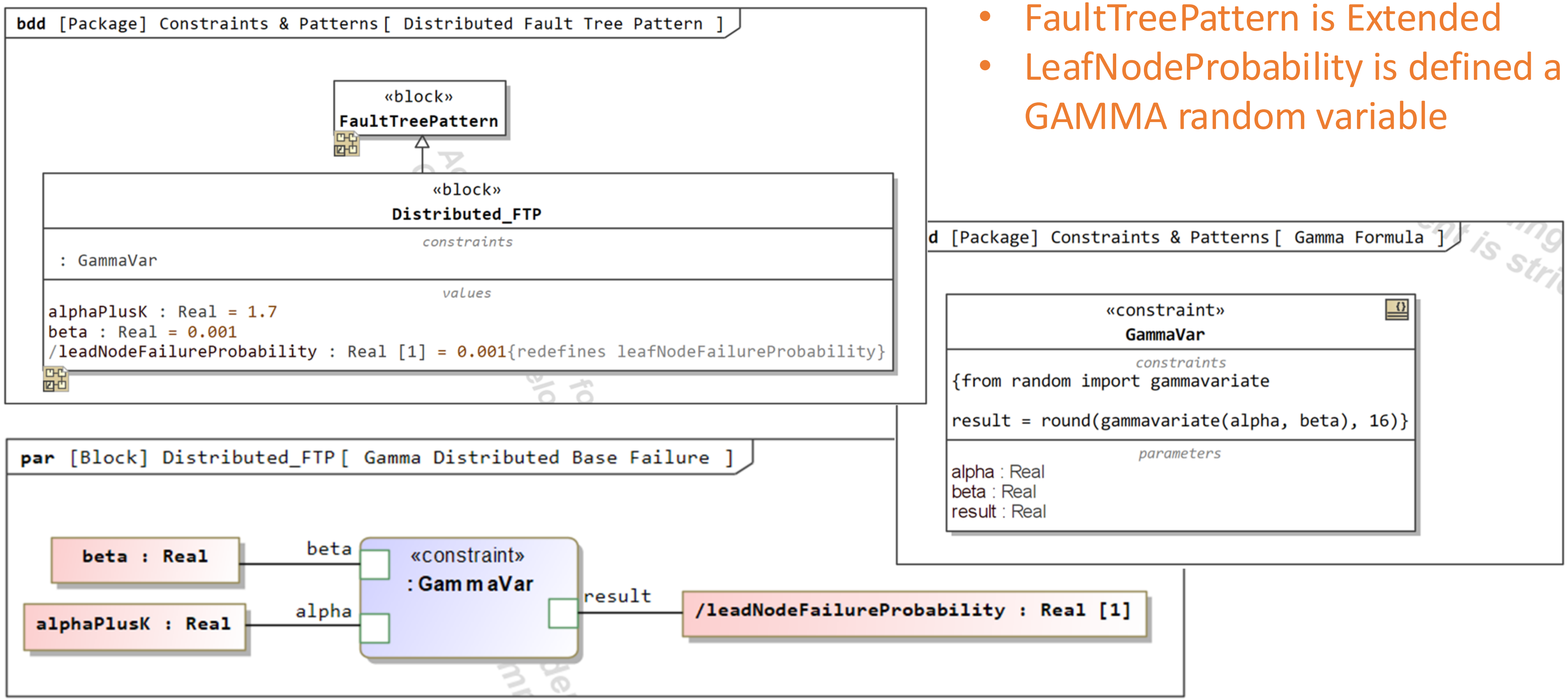
3.4 FT Uncertainty – Example Implementation in SysML



Test data used to quantified failure probability

Uncertainty distribution for failure probability

3.4 FT Uncertainty – Example Implementation inSysML



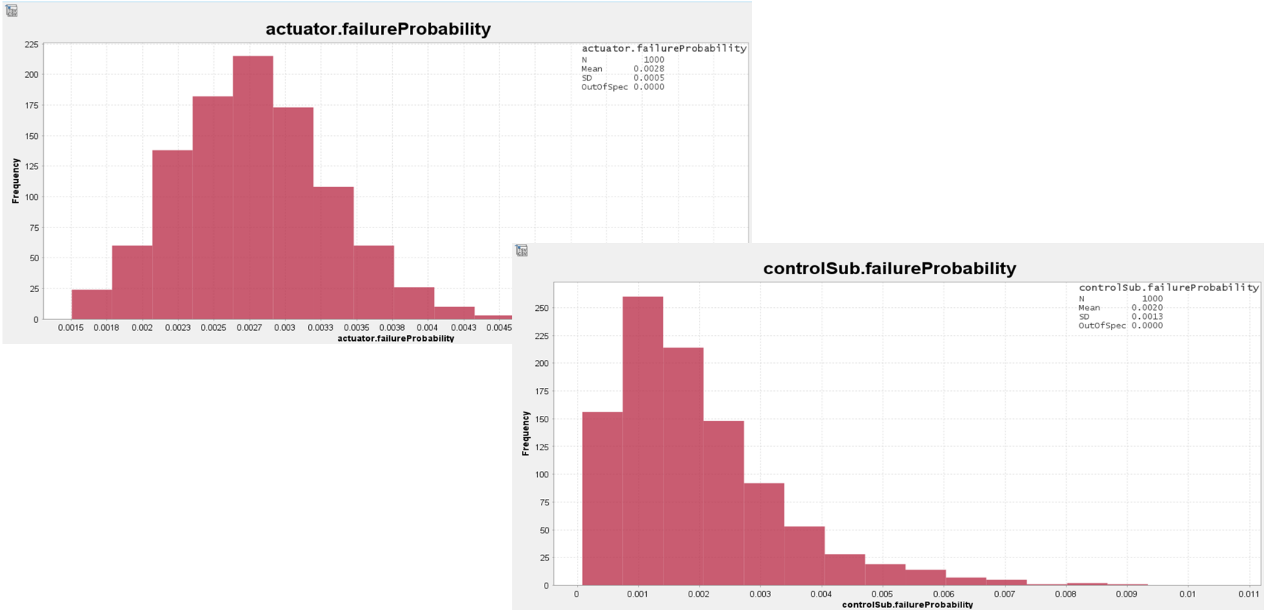
- FaultTreePattern is Extended
- LeafNodeProbability is defined a GAMMA random variable

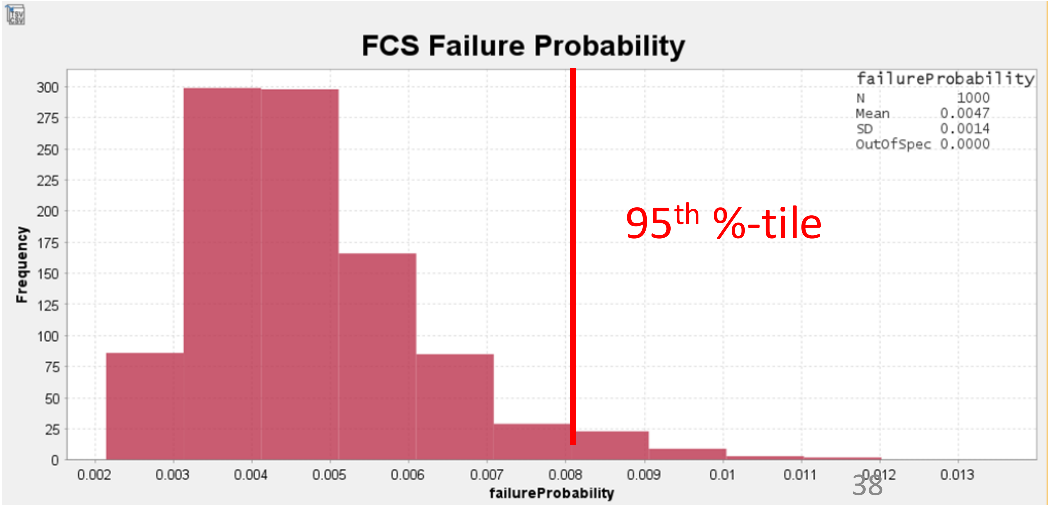
Amissah, 2025

3.4 FT Uncertainty – Example Implementation in SysML

Monte-Carlo Simulation Initial Values

#	Name	beta	alphaPlusK	leadNodeFailur...	failureProbability	orSubs	andSubs
1	flightControlSystem	1.0E-12	1	4.0E-12	0.0035	controlSub : Struc actuator : Structur	
2	actuator	0.0001	27.5	0.0017	0.0017		
3	controlSub	1.0E-12	1	1.0E-12	0.0018	mAndA : Structure	primaryCtrl backupCtrl :
4	primaryCtrl	0.005	3.5	0.0118	0.0118		
5	backupCtrl	0.0175	1.5	0.0372	0.0372		
6	mAndA	0.001	1.5	0.0014	0.0014		



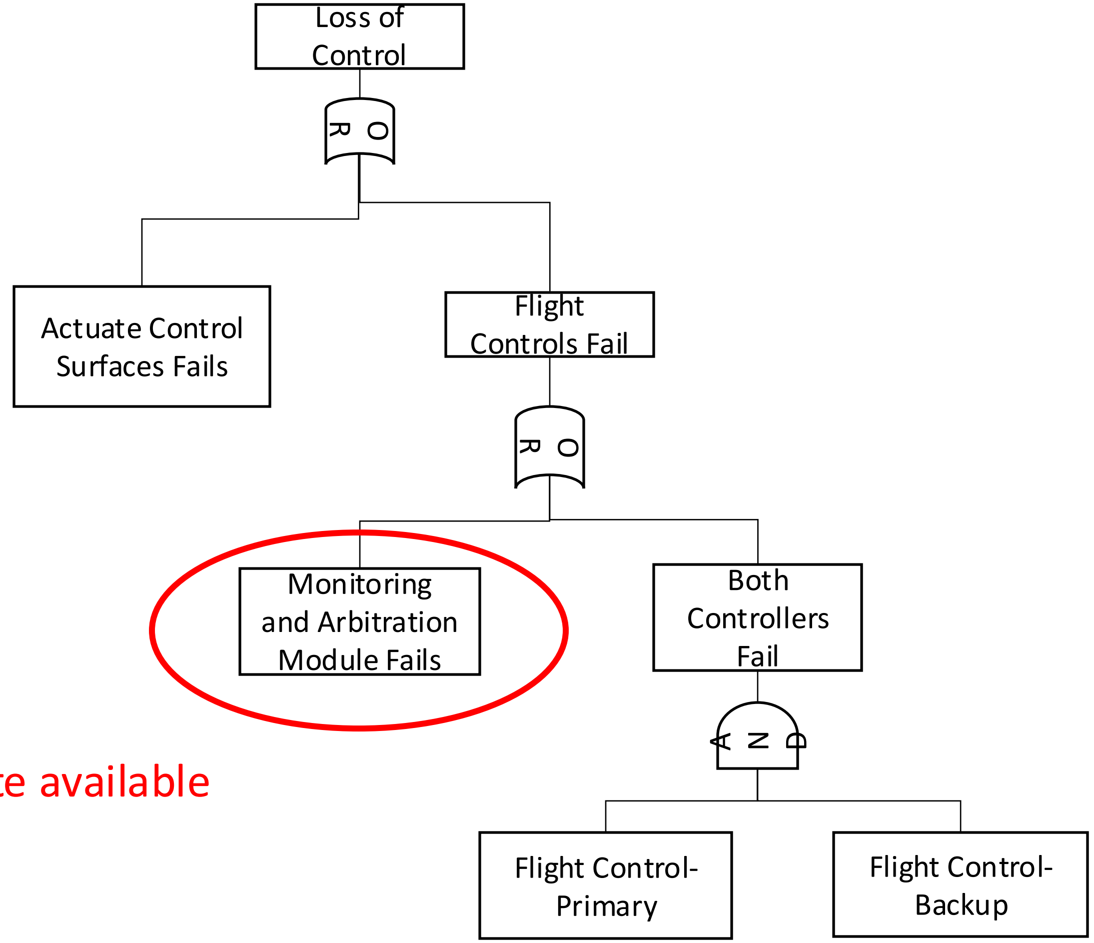


3.5 Fault Tree Uncertainty – Interval Analysis

<u>Background:</u>

- Frequently, especially in early design phases, one or more nodes in a fault tree are unknown
- SME quantify as a range

- What is Risk budget assigned to (new) Function to meet Top-level Risk
  - What are implications for ranges of other nodes in the tree?
  - How do constraints on top level probabilities flow down to requirements on base events?

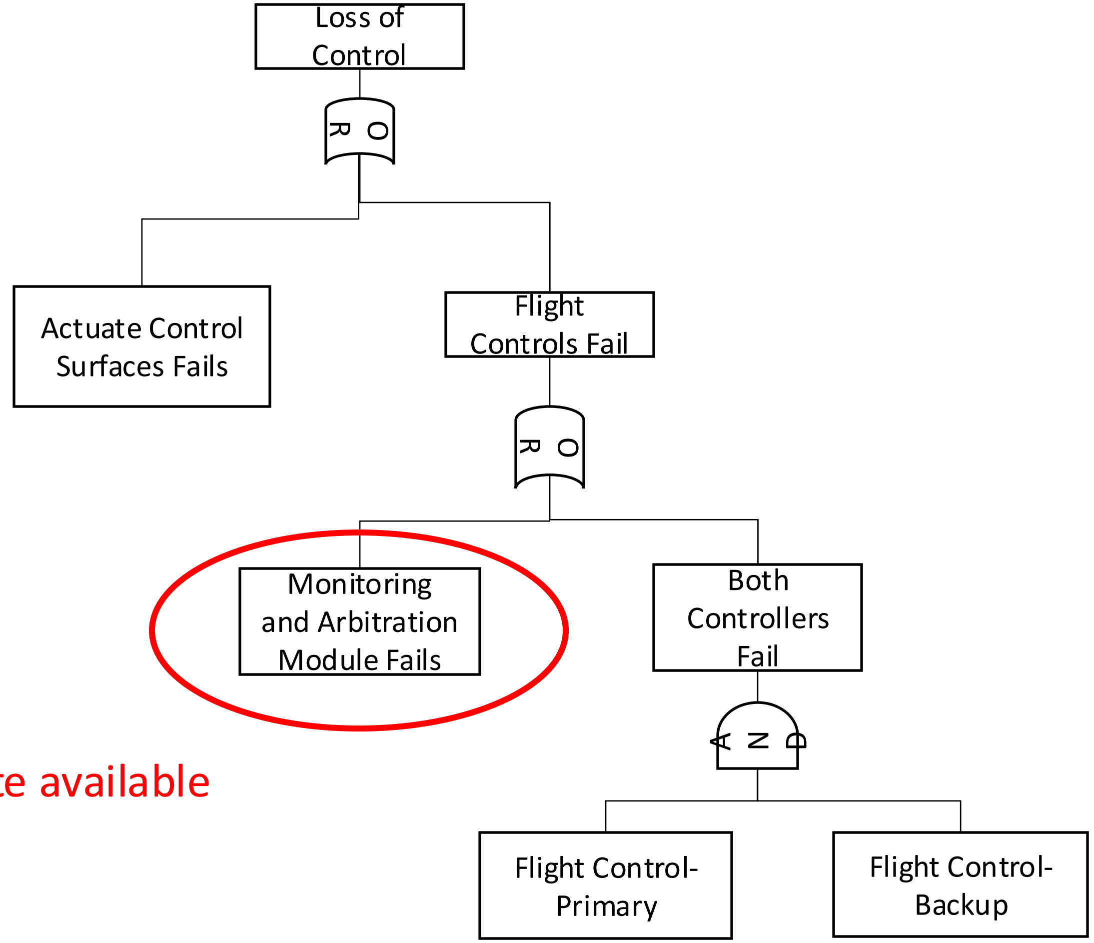


No estimate available

3.5 Fault Tree Uncertainty – Interval Analysis

Design Challenge:

- What is Risk budget assigned to (new) Function to meet Top-level Risk
  - What are implications for ranges of other nodes in the tree?
  - How do constraints on top level probabilities flow down to requirements on base events?

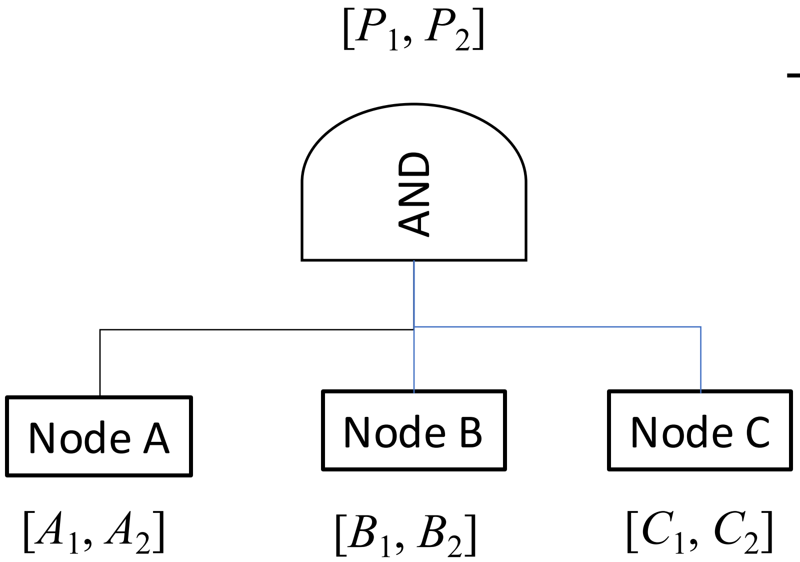


No estimate available

3.5 Fault Tree Uncertainty – Interval Analysis

Theory

- Each node is quantified by a lower bound and upper bound on the event probability
  - If the probability of an event is completely unknown it's probability interval is [0, 1]
  - If lower bound = upper bound, the event probability is known exactly



Constraints on interval bounds

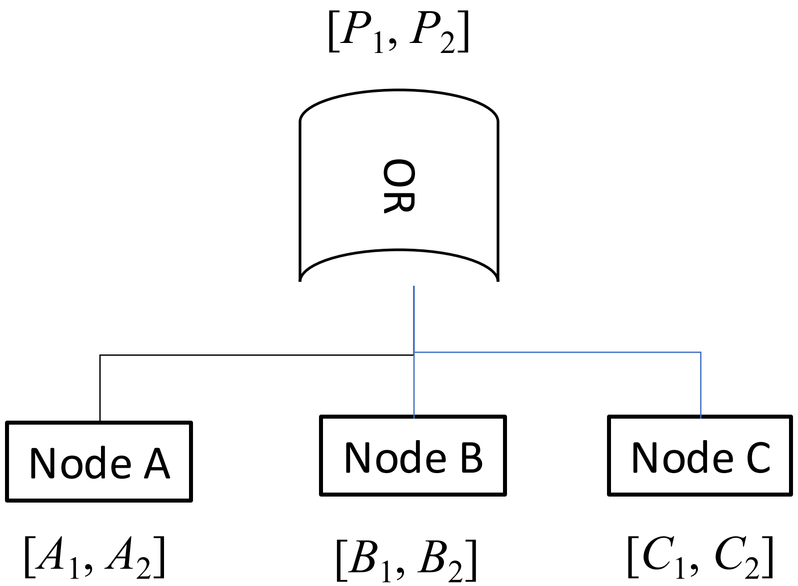
$$P_1 \geq A_1 B_1 C_1$$

$$P_2 \leq A_2 B_2 C_2$$

$$A_1 \geq \frac{P_1}{B_2 C_2}$$

$$A_2 \leq \frac{P_2}{A_1 C_1}$$

$$B_1 \geq \mathrm{L}$$



Constraints on interval bounds

$$P_1 \geq 1-(1-A_1)(1-B_1)(1-C_1)$$

$$P_2 \leq 1-(1-A_2)(1-B_2)(1-C_2)$$

$$A_1 \geq 1-\frac{1-P_1}{(1-B_2)(1-C_2)}$$

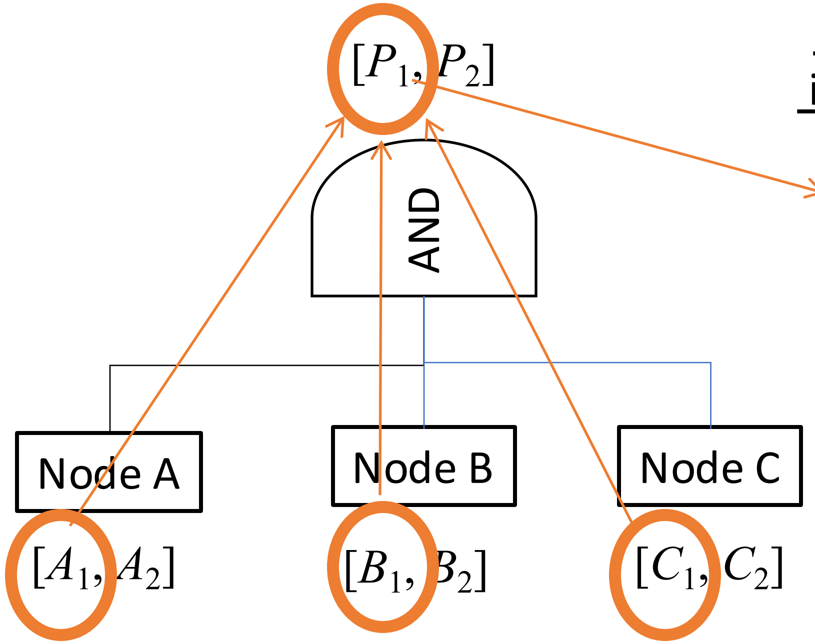
$$A_2 \leq 1-\frac{1-P_2}{(1-B_1)(1-C_1)}$$

$$B_1 \geq \mathrm{L}$$

3.5 Fault Tree Uncertainty – Interval Analysis

Theory

- Each node is quantified by a lower bound and upper bound on the event probability
  - If the probability of an event is completely unknown it's probability interval is [0, 1]
  - If lower bound = upper bound, the event probability is known exactly



Constraints on interval bounds

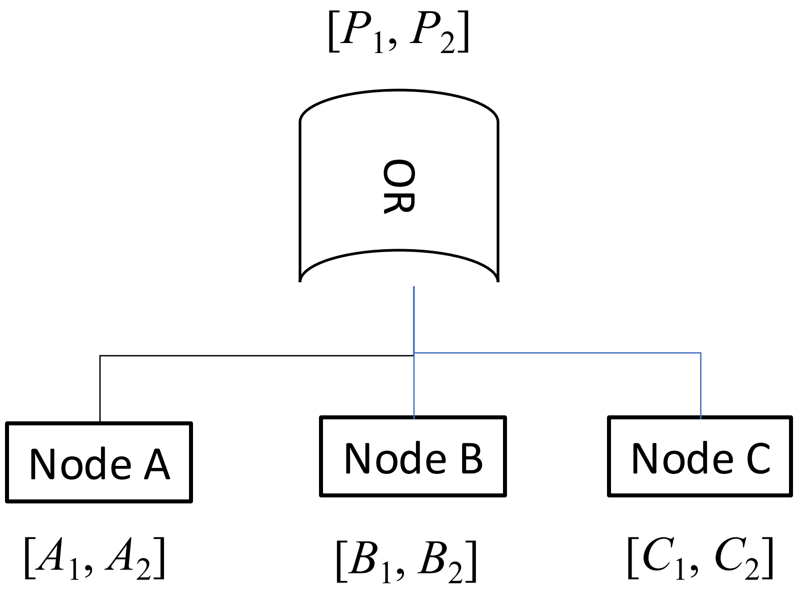
$$P_1 \geq A_1 B_1 C_1$$

$$P_2 \leq A_2 B_2 C_2$$

$$A_1 \geq \frac{P_1}{B_2 C_2}$$

$$A_2 \leq \frac{P_2}{A_1 C_1}$$

$$B_1 \geq \text{L}$$



Constraints on interval bounds

$$P_1 \geq 1-(1-A_1)(1-B_1)(1-C_1)$$

$$P_2 \leq 1-(1-A_2)(1-B_2)(1-C_2)$$

$$A_1 \geq 1-\frac{1-P_1}{(1-B_2)(1-C_2)}$$

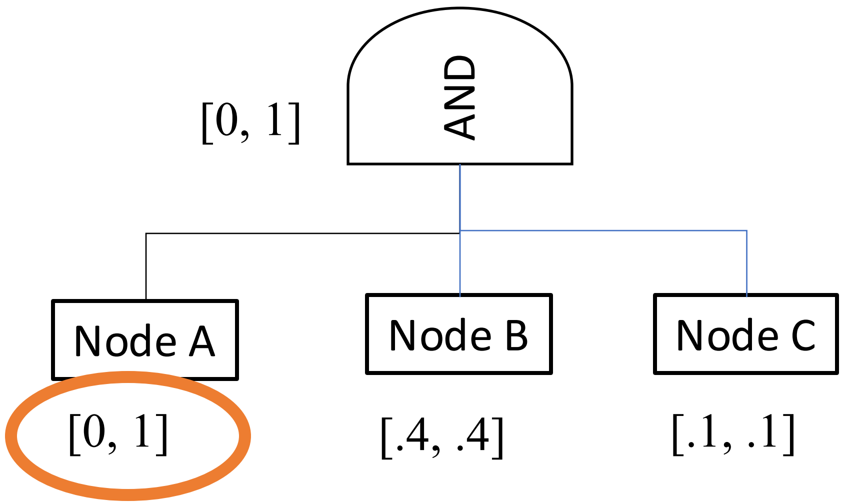
$$A_2 \leq 1-\frac{1-P_2}{(1-B_1)(1-C_1)}$$

$$B_1 \geq \text{L}$$

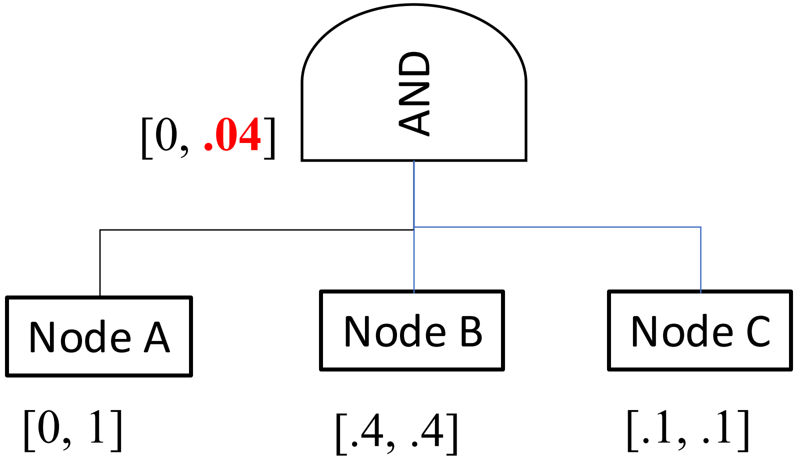
3.5 Fault Tree Uncertainty – Interval Analysis

- Node A unknown (assign it [0,1]
- Nodes B and C known exactly

<u>Initially known intervals</u>



<u>Derived intervals</u>



$$P_1 \geq A_1 B_1 C_1$$

$$P_2 \leq A_2 B_2 C_2$$

$$A_1 \geq \frac{P_1}{B_2 C_2}$$

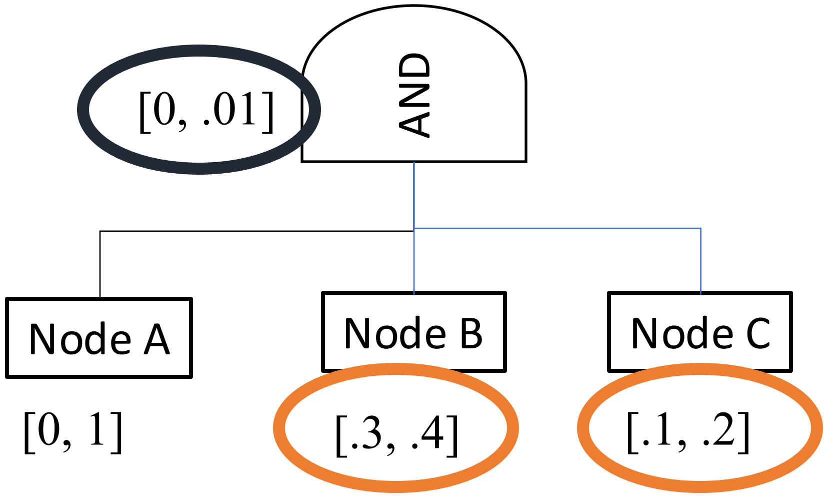
$$A_2 \leq \frac{P_2}{A_1 C_1}$$

- Calculate Interval for [P1, P2]

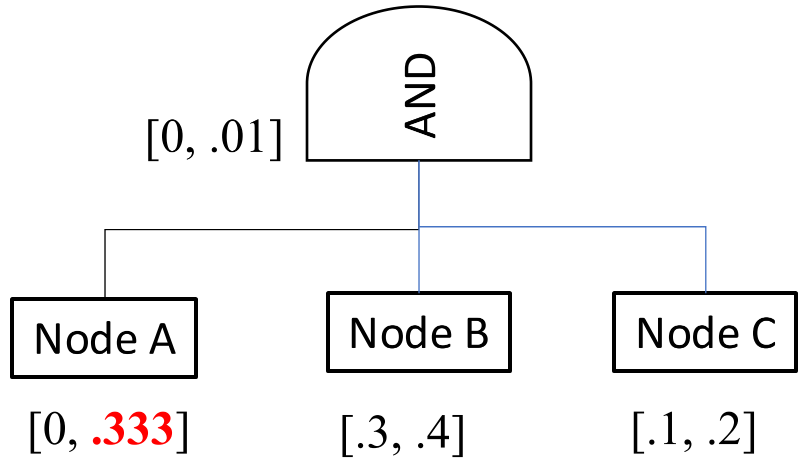
3.5 Fault Tree Uncertainty – Interval Analysis

- Top probability required <= .01
- Nodes B and C quantified as intervals
- Requirements on A?

Initially known intervals



Derived intervals



$$P_1 \geq A_1 B_1 C_1$$

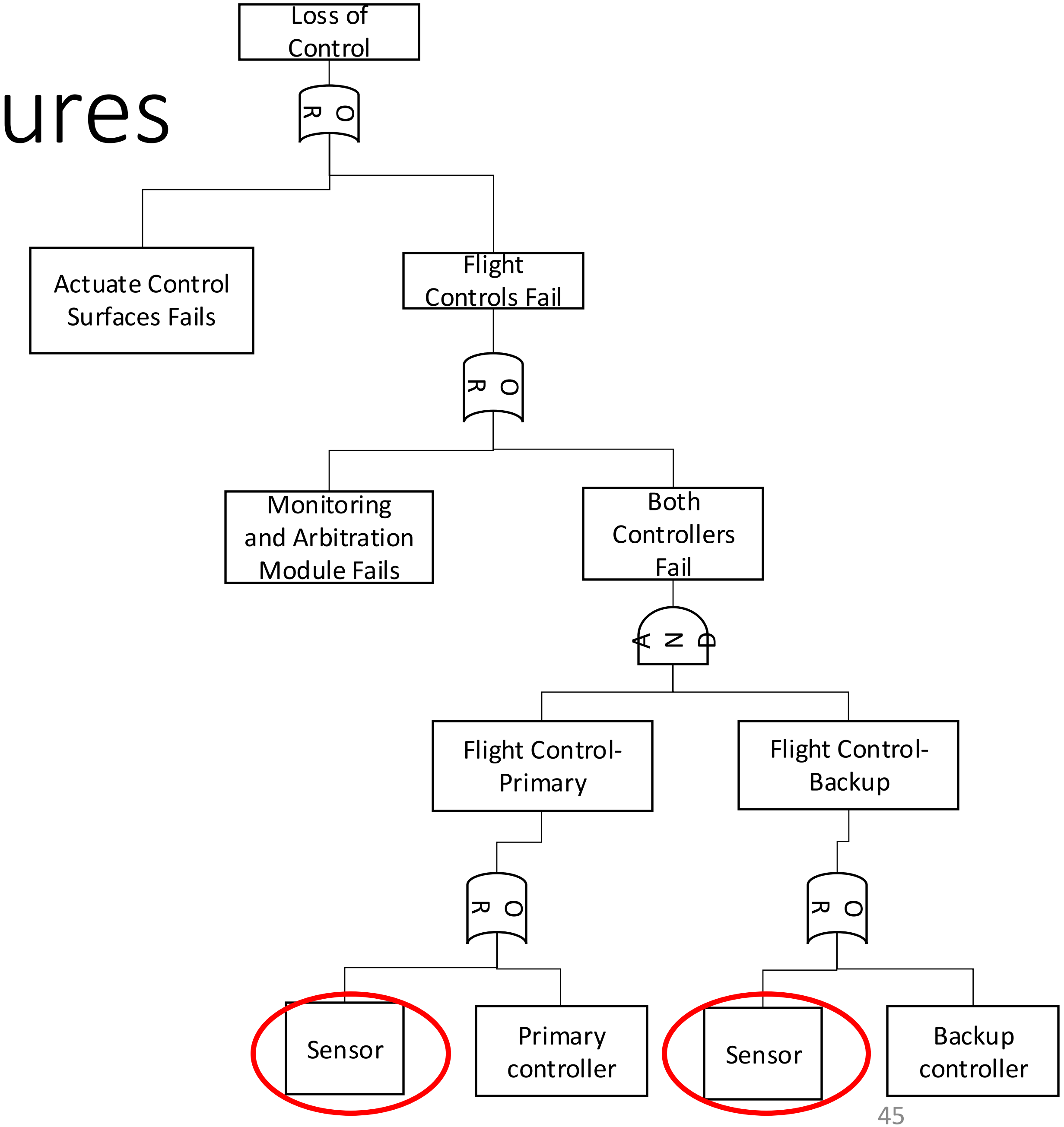
$$P_2 \leq A_2 B_2 C_2$$

$$A_1 \geq \frac{P_1}{B_2 C_2}$$

$$A_2 \leq \frac{P_2}{A_1 C_1}$$

- Calculate Interval for [A1, A2]

3.6 Common Cause Failures



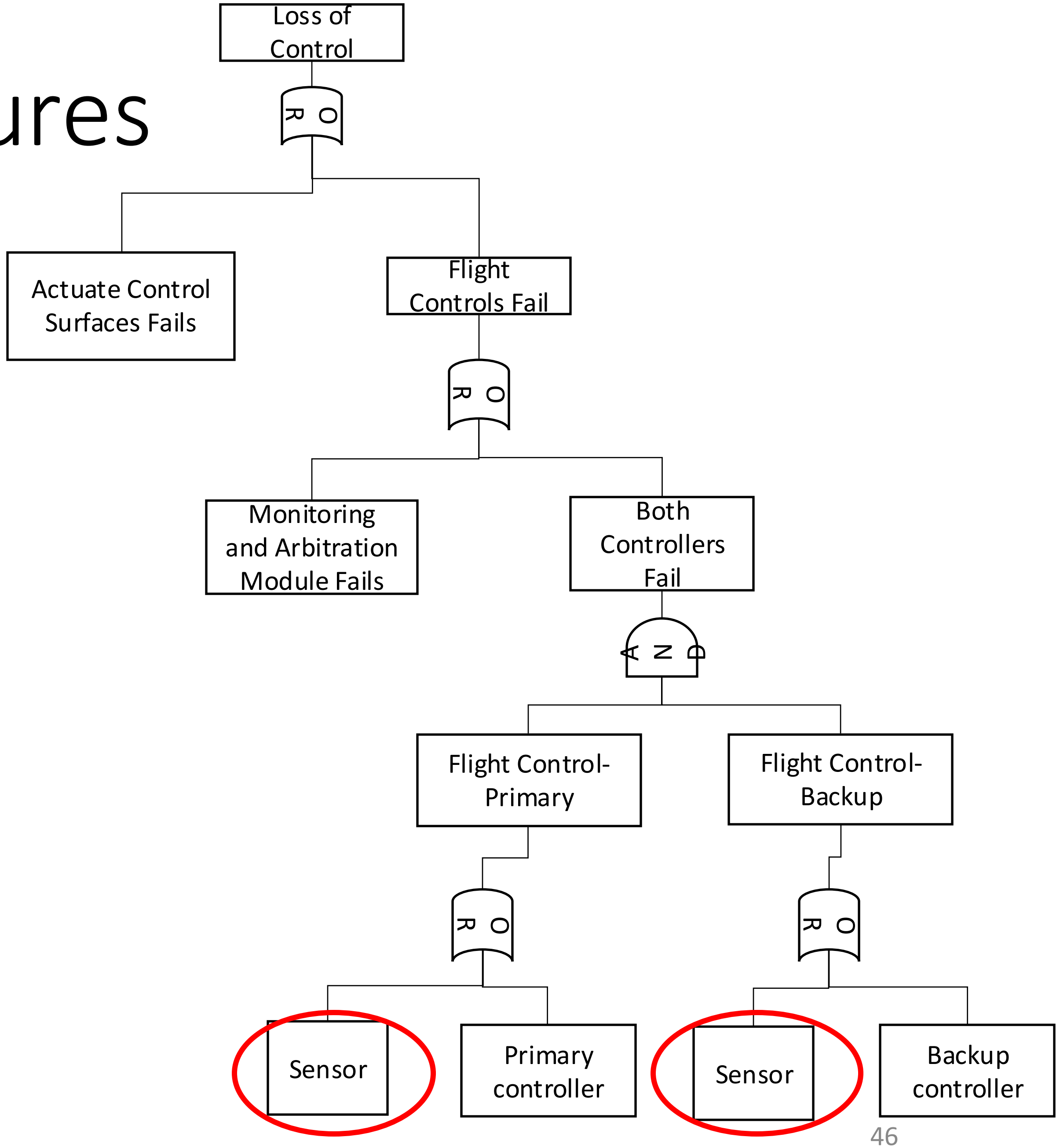
<u>Problem</u>

- Large, complex systems -> Large complex FTs
- Where are the Common Cause Failures?
- How to Calculate Top-level Risk with common cause failures

- Note:
  - When base events are not independent, the bottom-up approach of calculating each parent from its children does not yield the correct result
  - An alternate algorithm is required

3.6 Common Cause Failures

- Example: Primary and backup flight control use the same (or same type of) sensor

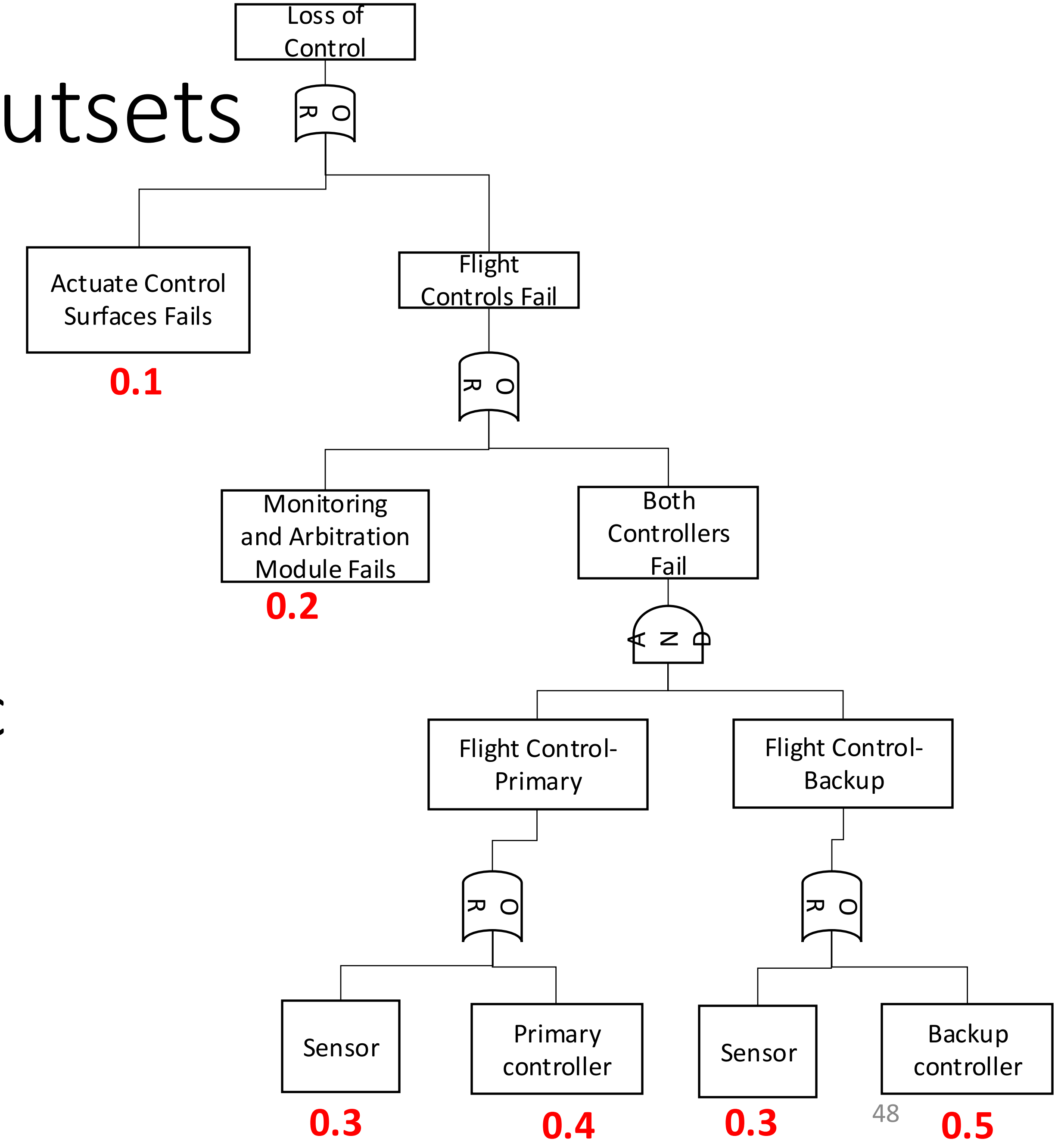


3.7 Algorithm for Evaluating Fault Tree with Common Cause Failures

- Identify all minimal cutsets in the tree
  - A cutset is a set of base events such that if each event in the set occurs/fails, then the top event occurs/fails
  - A minimal cutset is a cutset such that if any event is removed from the set, it is no longer a cutset
- Failure of top event is Pr{any minimal cutset occurs}

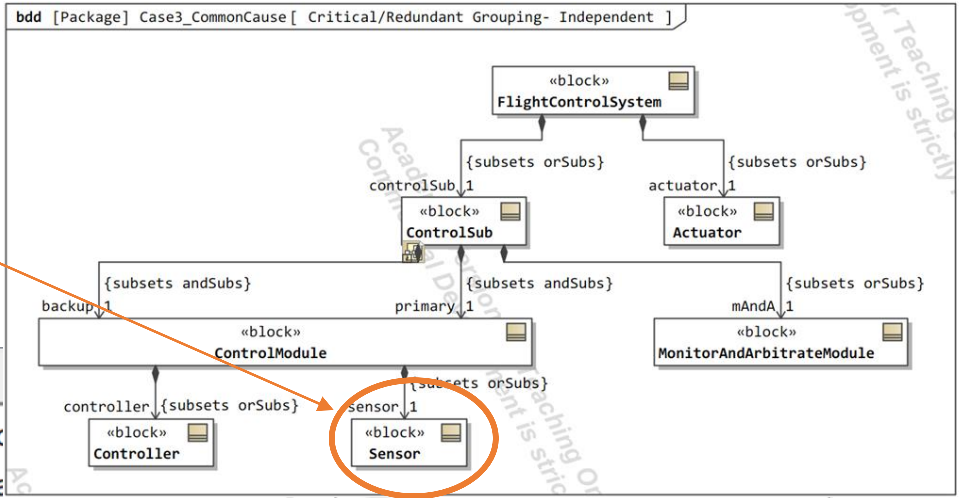
3.7 Example: Minimal Cutsets

- FT for Flight Control minimal cutsets:
  - {A: actuator control surface fails}
  - {B: monitoring / arbitration fails}
  - {C: sensor fails}
  - {D: primary controller fails, backup controller fails}
- Probability of top event = Pr{any minimal cutset occurs} = Pr{A or B or C or D}
  - = 1 - (1-Pr{A}) (1-Pr{B}) (1-Pr{C}) (1-Pr{D})
  - = 1 - (1-0.1)(1-0.2)(1-0.3)(1-0.4*0.5)
  - = .5968



3.7 CCF: SysML Implementation

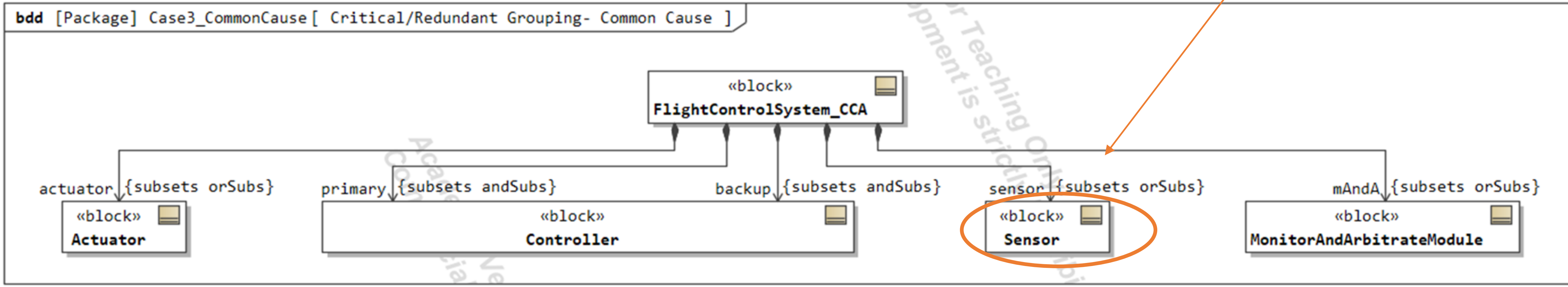
Assuming *independent* sensors, Top-level Probability is lower than it should be



#	Name	leafNodeFailureP...	failureProbability		
1	FCS	1.0E-12	0.5514	c a	
2	actuator	0.1	0.1		
3	controlSub	1.0E-12	0.5016	mAndA : Structure::Cas	primaryCtrlModule : Structur backupCtrlModule : Structure
4	primaryCtrlModule	1.0E-12	0.58	sensor : Structure::Case controller : Structure::C	
5	sensor	0.3	0.3		
6	controller	0.4	0.4		
7	backupCtrlModule	1.0E-12	0.65	backup.sensor : Structu backup.controller : Stru	
8	backup.sensor	0.3	0.3		
9	backup.controller	0.5	0.5		
10	mAndA	0.2	0.2		

3.7 CCF: SysML Implementation

Assuming *dependent* sensors (i.e. Common Cause), Top-level Probability is correct (and higher)



#	Name	leafNodeFailure... : Real	failureProbabili... : Real	orSubs : FaultTreePattern	andSubs : FaultTreePattern
1	FCS_CCA	1.0E-12	0.5968	actuator : Structure::Case mAndA : Structure::Case sensor : Structure::Case3	primary : Structure::Ca backup : Structure::Cas
2	actuator	0.1	0.1		
3	mAndA	0.2	0.2		
4	sensor	0.3	0.3		
5	primary	0.4	0.4		
6	backup	0.5	0.5		

Organization

1. Introduction and Motivation
2. Example: System Safety Analysis and MBSE/SysML Models
3. Advances in Fault Tree Analysis
    1. Representing Fault Trees in SysML
    2. Connected SysML Models (FT, BDD)
    3. Deriving FT from BDD, AD, and IBD
    4. Uncertainty Quantification for Fault Trees
    5. Interval Analysis for Fault Trees
    6. Identifying Common Cause Failures in FT
    7. Calculating Top-Level Probability for FTs with Common Cause
4. Future Work

4 Future Work

- Training Module
- Case Studies
- Pilot Projects

Lance Sherry – lsherry@gmu.edy
John Shortle – jshortle@gmu.edu
Matthew Amissah – mamissah@gmu.edu
Ali Raz – araz@gmu.edu