



# *Toward a Model-Based Approach for Flight System Fault Protection*

John Day  
Peter Meakin  
Alex Murray (presenting)

Jet Propulsion Laboratory,  
California Institute of Technology

IEEE/AIAA Big Sky  
Aerospace Conference  
March 8, 2012



# Fault Protection – Key and Distinct

- Fault Protection (or Fault Management) is a key sub-discipline of Systems Engineering for flight systems
- Has its own distinct techniques for:
  - Analyzing and describing the possible ways in which the system can fail
  - Identifying mitigation or avoidance strategies for those failures
  - Validating its own analyses and verifying the FP architecture
- Produces a distinct set of engineering products, typically:
  - Failure Modes and Effects Analysis (FMEA): an analysis of the failure modes of each component
  - Fault Tree Analysis: a top-down analysis of how the system might be able to fail in a functional sense
    - Sometimes a “Success Tree Analysis”, which describes what has to go right for the system to achieve required functionality, is also done
  - Monitor and Response Dictionary: a definition of the system’s set of operational failure detection and response behaviors
  - A set of trace matrices to ensure that the analyses and verification are complete
  - These products are often maintained in spreadsheets



# Redundancy – a Bad Thing!

- In order to perform the analyses and produce the products, FP engineers must produce a model of the flight system:
  - The Failure Modes & Effects analysis is based on an enumeration and description of the components of the flight system, and interactions among them
  - Fault Tree/Success Tree Analyses are descriptions of the flight system's behavior in achieving (or not achieving) required functionality
- Flight System Systems Engineering in general produces and maintains the same information:
  - They develop and maintain a flight system baseline, in some form
  - This includes a description of FS components and relationships
  - They develop and maintain models of system behavior, in some form (planning and execution scenarios, verification scenarios, resource usage analyses, etc)
- The two groups both create models of system structure and behavior, with a great deal of duplication
- Having two versions of these complex models is inefficient, but worse, strongly tends to inconsistencies and errors



# How a Model-based Approach Can Help

- Modeling is meant to provide an effective way to describe complex systems that consist of many components with relationships among them
- The Unified Modeling Language and System Modeling Language are excellent vehicles for describing such systems
- To be “effective”, the modeling language must allow organization that avoids redundancy and allows re-use
  - Allows elements that are used more than once to exist in only one place
  - Parts of a model must be able to refer to other parts of a model
- This allows things in common to both FP and general flight system systems engineering processes to be easily shared by both
- Why not use the same one? Modeling makes that straight forward!



# How a Model-based Approach Can Help (cont)

- For example, both the FP FMEA, and the FSSE's flight system mass estimate, rely on an underlying model of all of the physical components of the flight system.
- That underlying component model can be well expressed in SysML and UML
- New information can be layered on top of that model for each purpose:
  - FP engineering adds failure mode and criticality information
  - FS systems engineering adds mass information (which may vary with time)
  - Many other uses with associated new layers of information
- If a component or assembly is deleted or changed, the added layers of information are immediately apprised



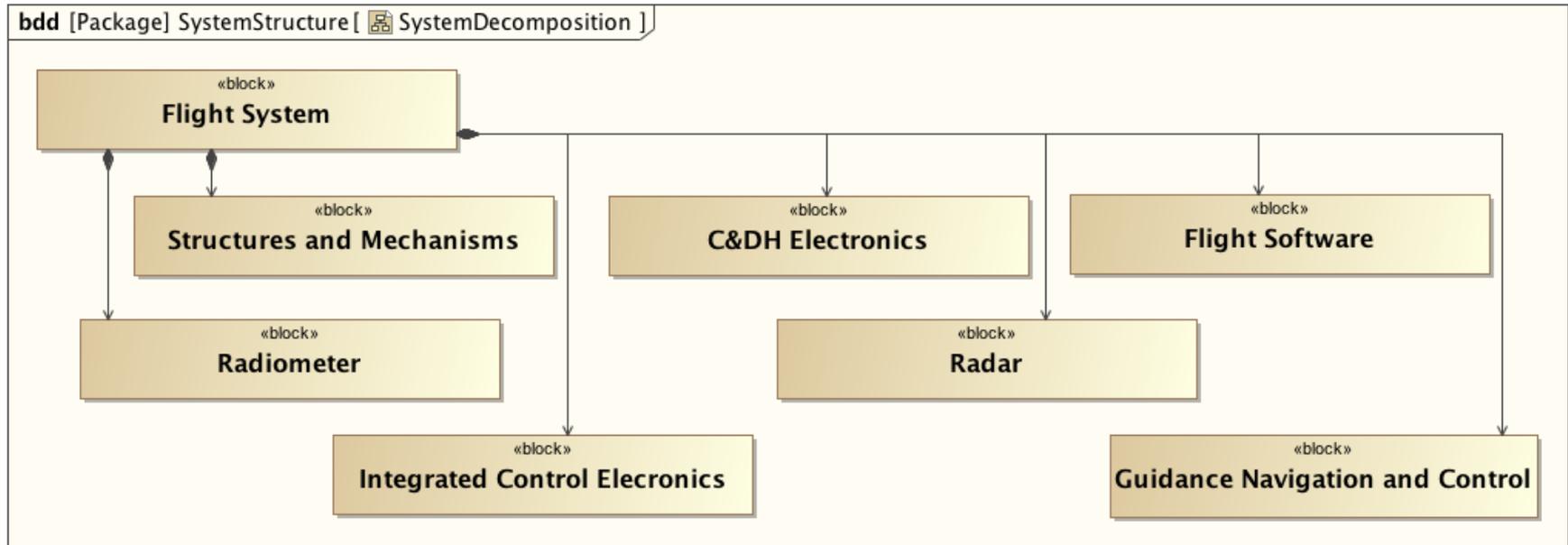
# Summary of our Approach

- Use SysML/UML to describe the physical structure of the system
  - This part of the model would be shared with other teams – FS Systems Engineering, Planning & Execution, V&V, Operations, etc., in an integrated model-based engineering environment
- Use the UML *Profile* mechanism, defining *Stereotypes* to precisely express the concepts of the FP domain
  - This extends the UML/SysML languages to contain our FP concepts
- Use UML/SysML, along with our profile, to capture FP concepts and relationships in the model
- Generate typical FP engineering products (the FMECA, Fault Tree, MRD, V&V Matrices)

We' ll show just a few limited examples in this presentation



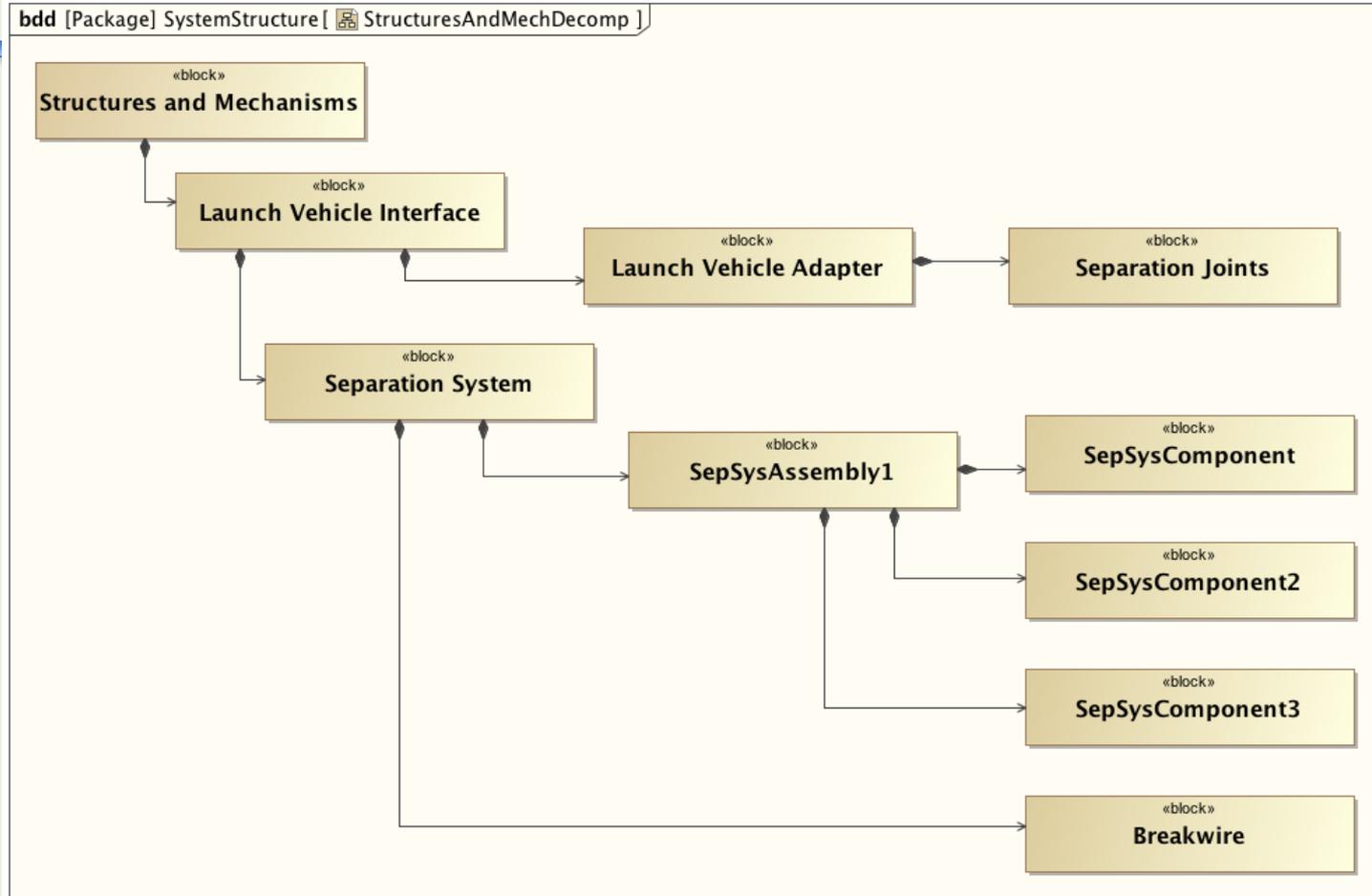
# The Basic Structural Model



- This diagram shows the top-level decomposition of the hypothetical flight system we use for our examples.
- In the paper, some blocks decomposed further, down to the level of individual hardware and, for GN&C, software, components, for use in examples
- This structural model of the system could be used as a basis for many general flight system systems engineering purposes.
- In our paper, we use the components for FP examples, especially the FMEA.



# Further Decomposition



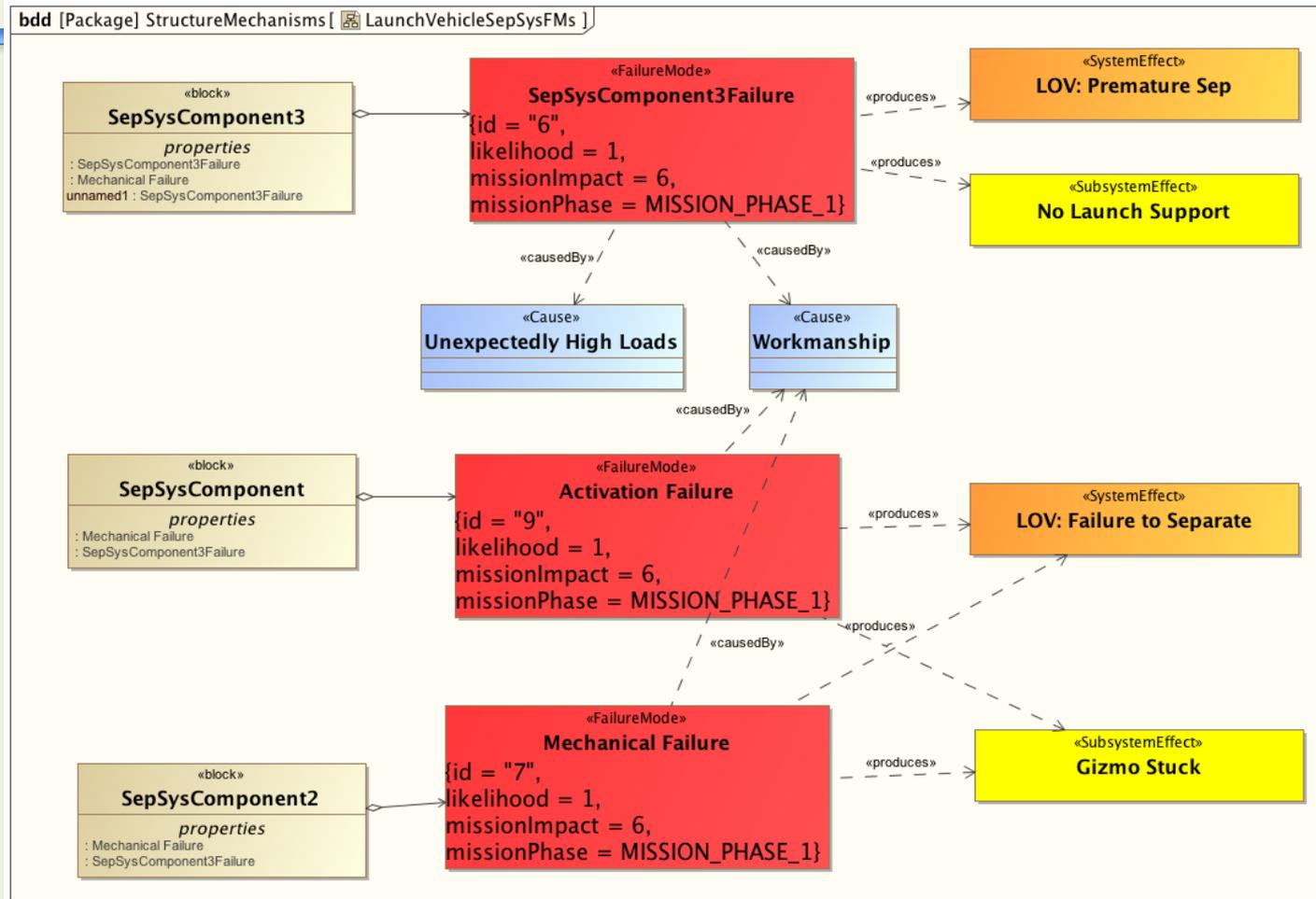
- Here we show a further decomposition of the Structures and Mechanisms subsystem.
- A key part of the subsystem is the interface to the launch vehicle, in turn containing a separation subsystem.

- The ‘SepSysComponent-x’ blocks represent specific hardware components of the separation subsystem assembly, but the names have generic out of ITAR concerns
- All of these blocks that represent hardware can be in a separate model, used as a library by both the FP and FSSE teams, as well as others.



# Failure Mode and Effects Analysis Example

- The failure modes of our generic separation subsystem components.
- The failure modes (in red) are attached to the component – this is one example of adding a layer of information.
- Each FM has several attributes: *likelihood*, *missionImpact*, etc.



- The FMEA includes an analysis of the effects of a failure, distinguishing local, subsystem effects from system effects.
- The FMEA also analyzes and documents the causes of each failure mode



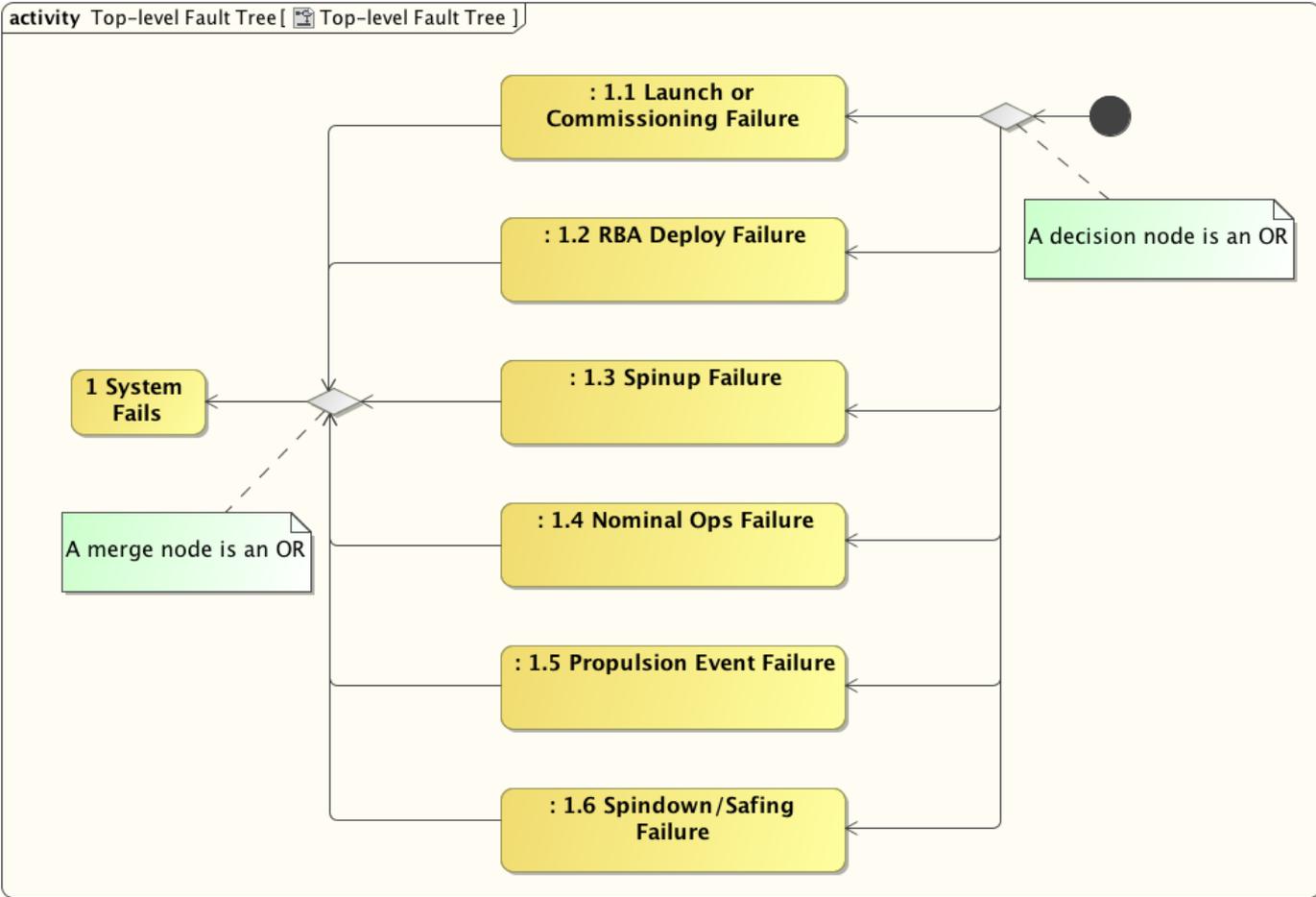
# Fault Tree Analysis

- Fault tree analysis begins with one or more high-level failure scenarios, e.g. “Mission Failure” or “Loss of Vehicle”, and then ask the question “what are all the possible ways we could get here?”
- Each scenario is decomposed into necessary preconditions, then for each precondition, that question is asked.
  - In this recursive way, the preconditions are further decomposed
- Sometimes just one precondition out of multiple alternatives is sufficient to cause a scenario (or state); sometimes all in a list of preconditions are needed
  - This shows in FTAs by having some way of representing logical ‘AND’ and ‘OR’ relations between a group of preconditions
  - This is often done in spreadsheets, with the tree growing to the right as scenarios are further decomposed



# Fault Tree Analysis Example

- In this fault tree, the top-level failure scenarios is the generic “System Fails”.
- The ways in which the system could fail are enumerated as other nodes in the activity, any of which would suffice to cause the system to fail.

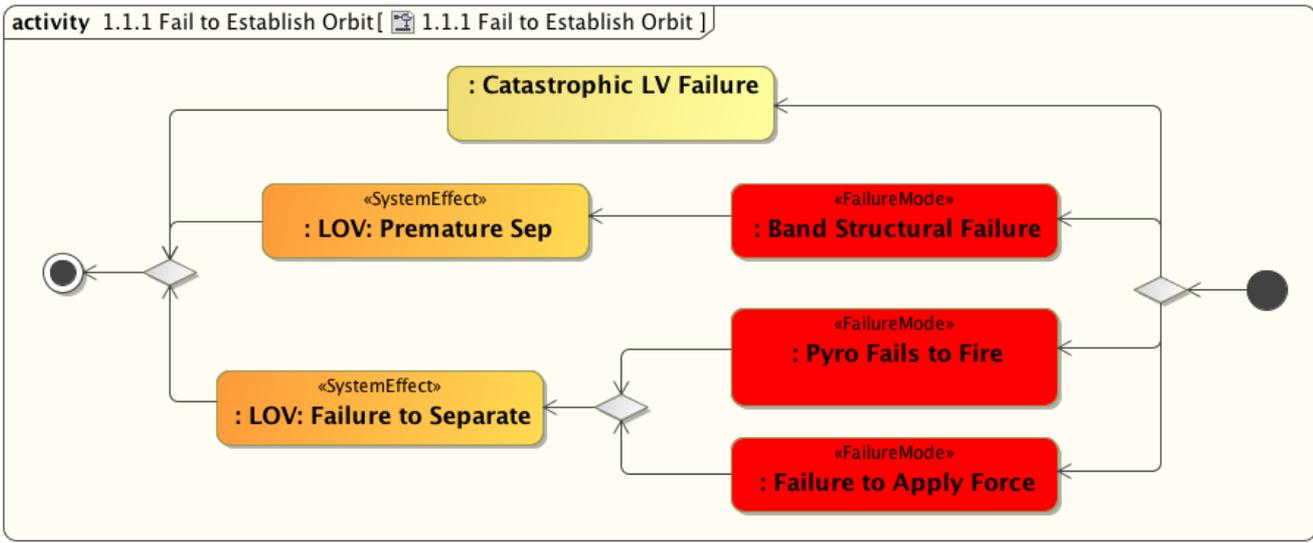
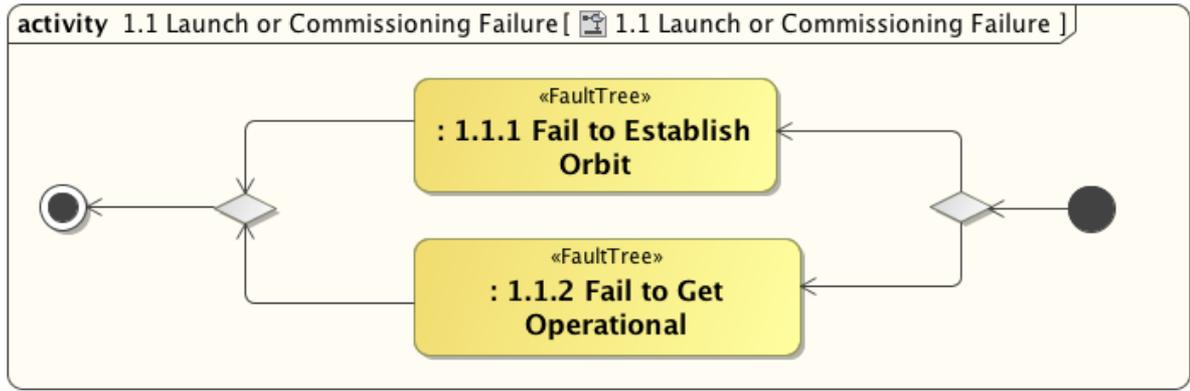


- Each of the precondition scenarios (e.g. 1.1 Launch or Commissioning Failure) are themselves UML Activities, and so are further decomposed.



# Fault Tree Analysis Example (cont)

- These two diagrams show another two levels of decomposition of scenario “1.1 Launch or Commissioning Failure” from the previous diagram
- Note that at the lowest level, Failure Modes and System Effects, identified in the FMECA, appear as precondition scenarios

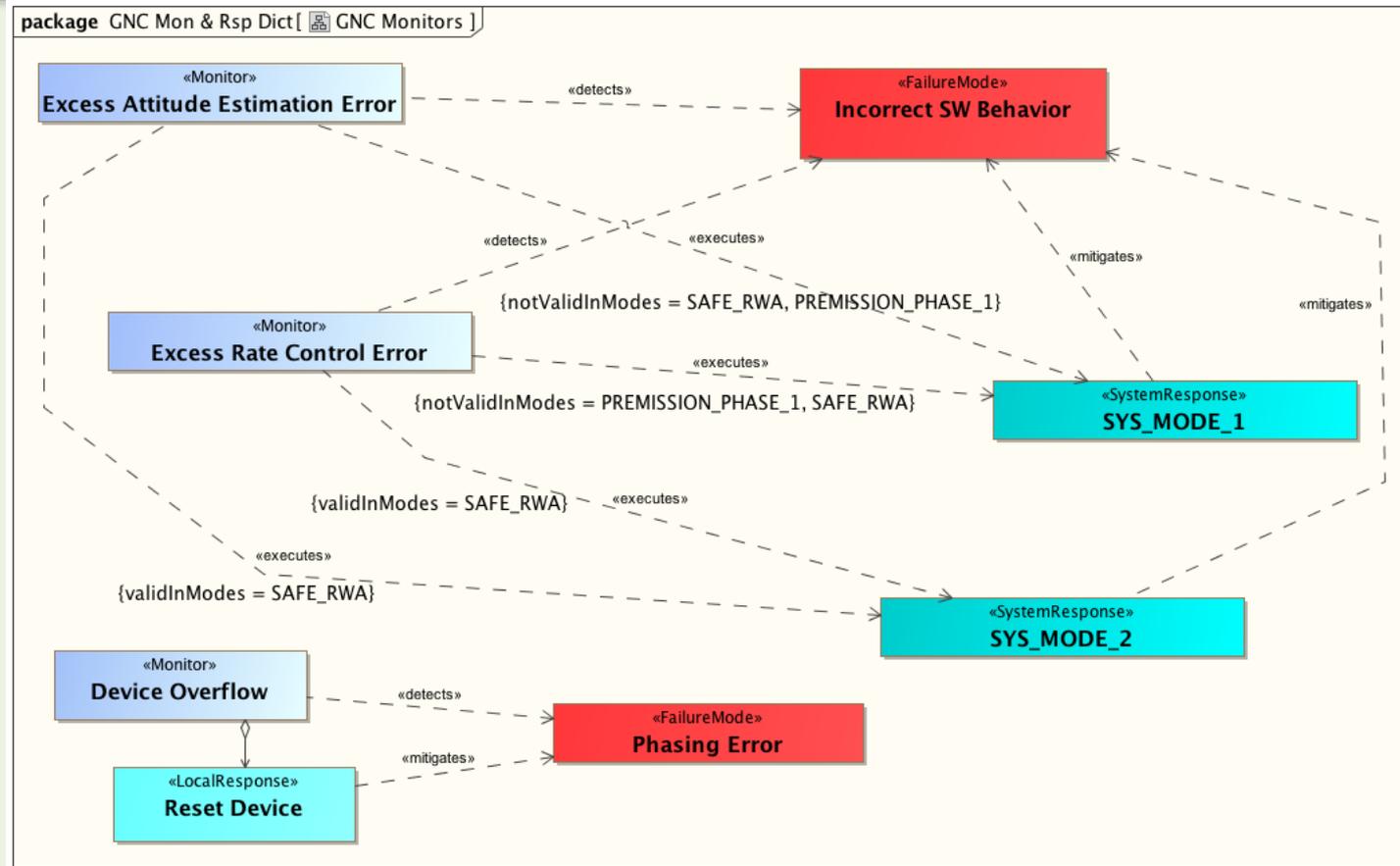


- Not every leaf node in the Fault Tree is a specific Failure Mode.



# Monitor & Response Dictionary Example

- This diagram shows a part of the MRD of our example.
- Shown are three Fault Monitors, two Failure Modes, two System Responses, and a Local Response.
- The relations between the elements shown are explained by the application of stereotypes (e.g. `<<detects>>`).
- The “validInModes” and “notValidInModes” expressions constraint the relation: e.g. The Excess Rate Control Error monitor may not cause execution of system response SYS\_MODE\_1 if the system is in PREMISSION\_PHASE\_1 or SAFE\_RWA modes.





# An Example Matrix: Failure Mode Detection

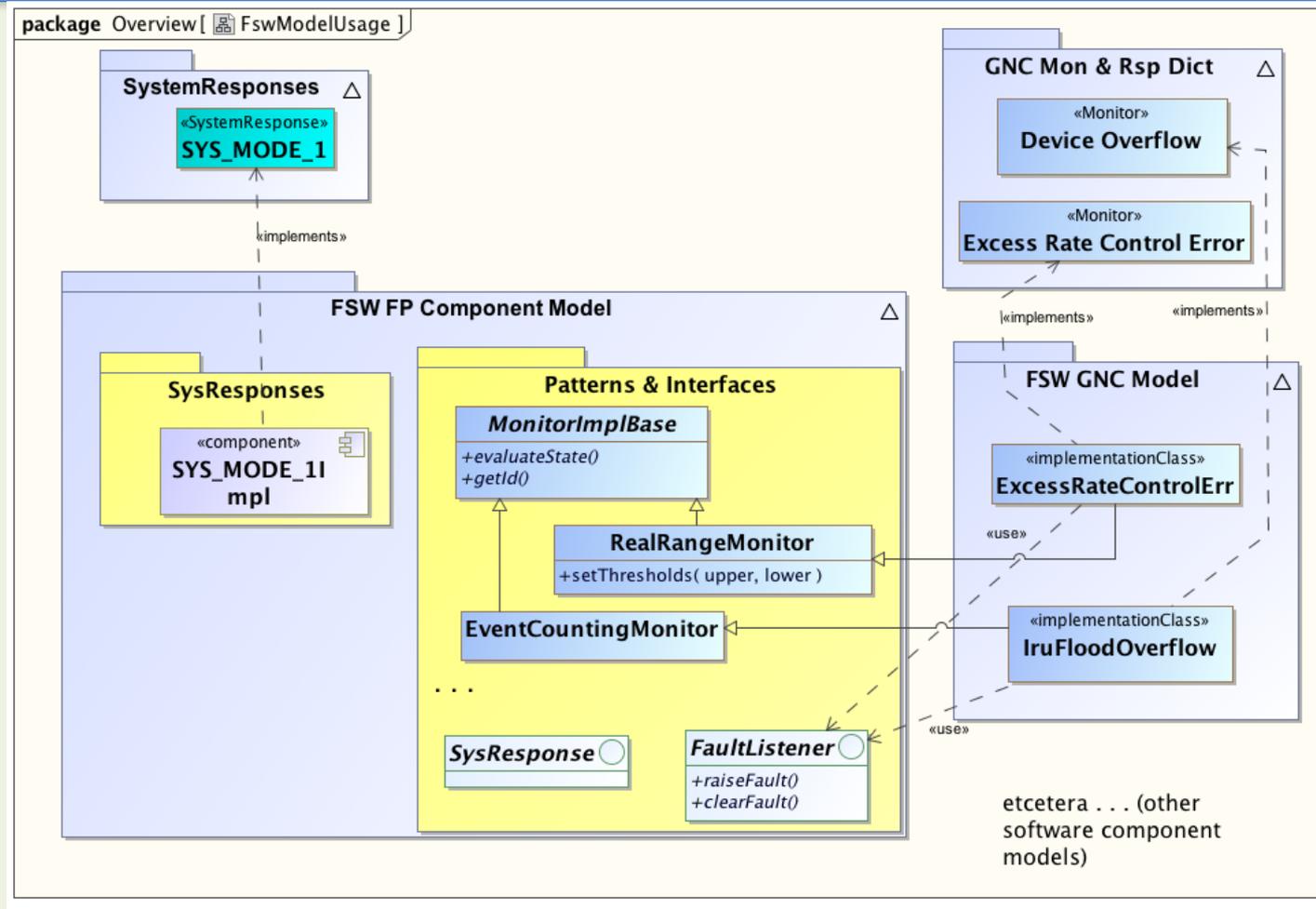
- The previous diagram shows only one part of the MRD of our example: the MRD is shown piecewise on many diagrams, but the relationships can be queried to report the entire MRD.
- This example show a matrix with Failure Modes as rows, and Fault Monitors as columns.
- If there' s an arrow in the intersection, it means that the column' s monitor detects the row' s failure mode.
- These matrices are generated automatically by MagicDraw (after being manually set up).
- We can show the set of mitigation relations in the entire MRD, or we can subset them and show only portions, e.g. by subsystem.
- We use the same technique for several relations:
  - A monitor mitigating a failure mode
  - A verification scenario precluding a failure mode
  - A monitor causing a response to execute
  - ...

	Excess Attitude Estimation Error [GNC Mon & Rsp Dict]	Excess Rate Control Error [GNC Mon & Rsp Dict]
[-] GNC	3	3
[-] Erroneous SW Behavior (major)		
[-] Incorrect Phasing - non-Key HW	↙	↙
[-] Incorrect SW Behavior	↙	↙
[-] Phasing Error	↙	↙
[-] Software Hangs		
[-] Some Error		
[-] StructureMechanisms		
[-] Activation Failure		
[-] Mechanical Failure		
[-] Mechanical Structure Failure		
[-] SepSysComponent3Failure		
[-] Warp or Crack		



# The Software Connection

- FSW is responsible for implementing the monitors and responses
- The FP MRD being used directly by FSW models is another example of re-use and elimination of redundancy
- Part of FSW verification is ensuring comprehensive implementation of the MRD



- The *implements* relationship can be used to build a matrix



# Conclusions & Future Work

- We think these techniques holds a great deal of promise for making the practice of Fault Protection
  - More efficient
  - More accurate
  - Less error-prone
- We think there are more areas to explore for further advances, such as:
  - Finding failure modes that are not in a fault tree.
  - Reporting propagation paths: where one effect causes another one or another failure mode.
  - Validating propagation paths by mapping them to the physical model of the system
  - Likelihood of a failure mode could be determined/calculated from the likelihood of the set of associated causes.
- We hope to try applying these techniques on a more life-sized example in the near future