Towards a methodology and tooling for Model-Based Probabilistic Risk Assessment (PRA)

Samuel S. Schreiner[1], Matthew L. Rozek[2], Andy Kurum[3],
Chester J. Everline[4], Michel D. Ingham[5], Jeffery A. Nunes[6]
Jet Propulsion Laboratory – California Institute of Technology, Pasadena, CA 91109

A Probabilistic Risk Assessment (PRA) aims to identify and assess potential risks to system technical performance requirements for the purpose of furnishing risk insights into project decisions. PRAs have traditionally been conducted manually using software with an isolated data model. As system complexity rises it becomes difficult to ensure consistency between a PRA, the evolving system design, and other engineering analyses; the techniques for conducting PRAs must evolve to meet this challenge. This work presents progress towards a methodology and tooling for conducting a PRA by leveraging data in the system model, embedded for other purposes and analyses, to conduct a PRA. An approach for identifying the appropriate probabilistic equation for each risk scenario from a standard library is presented, which is a significant step towards the quantification of the likelihood of a risk scenario occurrence. The final calculation of the likelihood of occurrence is left as an item of future work. We also present the development of preliminary tooling to carry out the methodology on a well-formed system model. The information needed to conduct the PRA is embedded in a consistent manner in a system model, so the model-based PRA can be regularly executed as the system model changes. The ability to modify the PRA in concert with lifecycle evolution affords a project the opportunity to track the extent to which system modification impacts compliance with requirements. These aspects of this model-based PRA methodology make it capable of managing risk in increasingly complex technical systems.

Nomenclature

IMCE	=	Integrated Model-Centric Engineering
IMU	=	Inertial Measurement Unit
JPL	=	Jet Propulsion Laboratory
MBSE	=	Model-Based Systems Engineering
NVM	=	Nonvolatile Memory
PRA	=	Probabilistic Risk Assessment
SRU	=	Stellar Reference Unit
SysML	=	System Modeling Language

[1] Systems Engineer, Flight Systems Engineering, Jet Propulsion Laboratory, M/S 321-560.
[2] Systems Engineer, Flight Systems Engineering, Jet Propulsion Laboratory, M/S 321-560, AIAA Member.
[3] Electronics Engineer, Communications, Tracking & Radar Division, Jet Propulsion Laboratory, M/S 238-333.
[4] Principal Systems Engineer/PRA Point of Contact, Mission Assurance, Jet Propulsion Laboratory, M/S 156-206.
[5] Senior Software Systems Engineer, Project Systems Engineering and Formulation, M/S 321-560, AIAA Associate Fellow.
[6] Principal Engineer, Reliability Engineering, M/S 156-206.

I. Introduction

The Probabilistic Risk Assessment Procedures Guide for NASA Managers and Practitioners[1] defines a Probabilistic Risk Assessment (PRA):

"Probabilistic Risk Assessment (PRA) is a comprehensive, structured, and logical analysis method aimed at identifying and assessing risks in complex technological systems for the purpose of cost-effectively improving their safety and performance."

PRAs have traditionally been conducted manually using an isolated data model, requiring an immense amount of knowledge to be collected and stored by reliability engineers, who often may not be integral members of the design and development teams. This transference of information can be a source of error (i.e., game of telephone). The identification of risk and the traversal through causal dependencies has relied upon the engineer's judgement and knowledge of the system design and behavior, and storage of this causal information is often informal. This can lead to discrepancies between the data used to conduct the PRA, the latest system design and performance data, and data used for other engineering analyses. As the complexity of technological systems continues to dramatically rise, the techniques for conducting PRAs must evolve to meet this challenge.

This work presents progress towards a methodology and tooling to conduct a PRA using data from an integrated system model that can also be used for other engineering analyses. JPL has made significant strides towards developing standards for system models (further described in Section II) that allow the system model to rigorously capture the complexity of modern technological systems in such a way that it can be utilized for a variety engineering analyses[2,3]. Previous work has described encoding information related to the requirements, design, behavior, and failure modes of a system[2,4,5,6,7,8,9]. This information can be embedded in the system model by domain experts in a manner that reduces ambiguity. The methodology described herein can be used to conduct a PRA on such a system model, which ensures consistency between the PRA, the latest system design, and other engineering analyses. Due to the fact that the information needed for the PRA is encoded in a consistent manner in a system model the model-based PRA can be regularly executed as the model changes. The ability to modify the PRA in concert with lifecycle evolution affords a project the opportunity to track the extent to which system modification impacts requirements compliance.

The product of a PRA is a set of "risk triplets". Each risk triplet contains a **consequence** (system requirements that may not be satisfied), a **risk scenario** (system behavior which can cause the consequence to occur), and the **likelihood** of the risk scenario occurring[1]. Section II gives an overview of the foundational ontologies that describe the required structure of the data in the system model and its semantic meaning. Section III describes how the model-based PRA methodology elaborates upon the given consequence (a set of requirements) to provide supporting information for the rest of the analysis. Section IV describes how the methodology produces risk scenarios using behavioral and causal information in the system model. Section V describes a draft approach for selecting probabilistic equations from a standard library that can ultimately be used to quantify the likelihood of occurrence of a risk scenario, the third element of a risk triplet. The complete calculation of this likelihood is left as an item of future work, as described in Section VI. Section VII presents preliminary results from tooling designed to carry out the model-based PRA methodology on a well-formed system model.

II. Relevant Ontologies and Semantics

The foundational principle of the model-based PRA methodology presented herein is that all information about the structure, behavior, and causal dependencies of the system under analysis is captured in a semantically-rigorous model. This allows for automated analysis tools to interpret the modeled information in a consistent way to produce meaningful and valuable results. At JPL, the Integrated Model-Centric Engineering (IMCE) effort has been chartered with developing a paradigm for enforcing semantic rigor throughout models[3]. The IMCE effort has settled on an approach where the models conform to a set of centrally-defined ontologies, which at their most fundamental level represent semantics through the use of concepts (classes) and their relations to one another[9]. The ontologies are meant to be embedding-agnostic, meaning that compliant information can be encoded in a multitude of different tools and syntaxes as long as the general logical assertions of the ontologies remain true. The JPL IMCE ontologies have been developed and evolved over a period of the past few years into a rich semantic foundation, now having approximately one-hundred different concepts.

One of the goals of developing the model-based PRA methodology was to make use of the existing IMCE ontologies so PRA analyses could be performed on integrated models that were constructed for other purposes. However, the entire content of the IMCE ontologies are not required to support this methodology – only a subset that captures the relevant aspects of the system under analysis. To support the model-based PRA methodology, the subset of the ontologies used in the model needs to capture the functions the system performs, the requirements for the performance of functions, and how the requirements are associated to the functions. The model must also contain the elements of the system under consideration and their association to the functions that are performed. Within the IMCE ontologies, components are considered to be discrete elements of the engineered system, each with clear boundaries and defined behavior. Each function has a component that is responsible for performing it, so a direct mapping from requirement to function to component can be constructed. This gives the set of components that are responsible for meeting a given requirement. As part of the behavior specification for each component, failure modes need to be identified along with the impaired function(s) that can no longer be performed by the component. For this methodology, we assume that element behaviors are represented as state machines, and therefore failure modes are elements in a state machine. Finally, the model needs to include causal dependencies between the failure modes of a component and other element behavior. This allows for the construction of a causal hierarchy to identify the combinations of basic events that cause certain failure modes, and provides the logic for construction of the risk scenarios.

The subset of the IMCE ontologies that is required to capture all of this needed information is given in Figure 1 below, which is adapted from Ref 6. The conventions used to present the ontology semantics in the figure are as follows:

- » Concepts (classes) are represented using rectangles.
- » The names of abstract concepts are written in italics. All concrete concept names are written in roman type (non-italic). For example, *`mission:PerformingElement`* is an abstract concept and `mission:Function` is a concrete concept.
- » Relations are represented using directed arrows, pointed from the source to the target of the relation, with relation names being written next to the target.
    - o For example, the `mission:performs` relation has a *`mission:PerformingElement`* as a source and a `mission:Function` as a target.
- » Inverse relations are also represented, and their names are written next to the source of the forward relation denoted with prefix of a forward slash "/" (i.e. `/mission:isPerformedBy`).
- » Multiplicities for both forward and inverse relations are shown between brackets next to their respective relation name.
- » Concepts and relations are prefixed with a package name corresponding to the IMCE ontology that they are defined in (i.e. `mission`, `behavior`, `analysis`, or `faultManagement`). This packaging was done to reduce IMCE-internal organization and namespacing concerns. This work may abbreviate or omit the package name where appropriate.

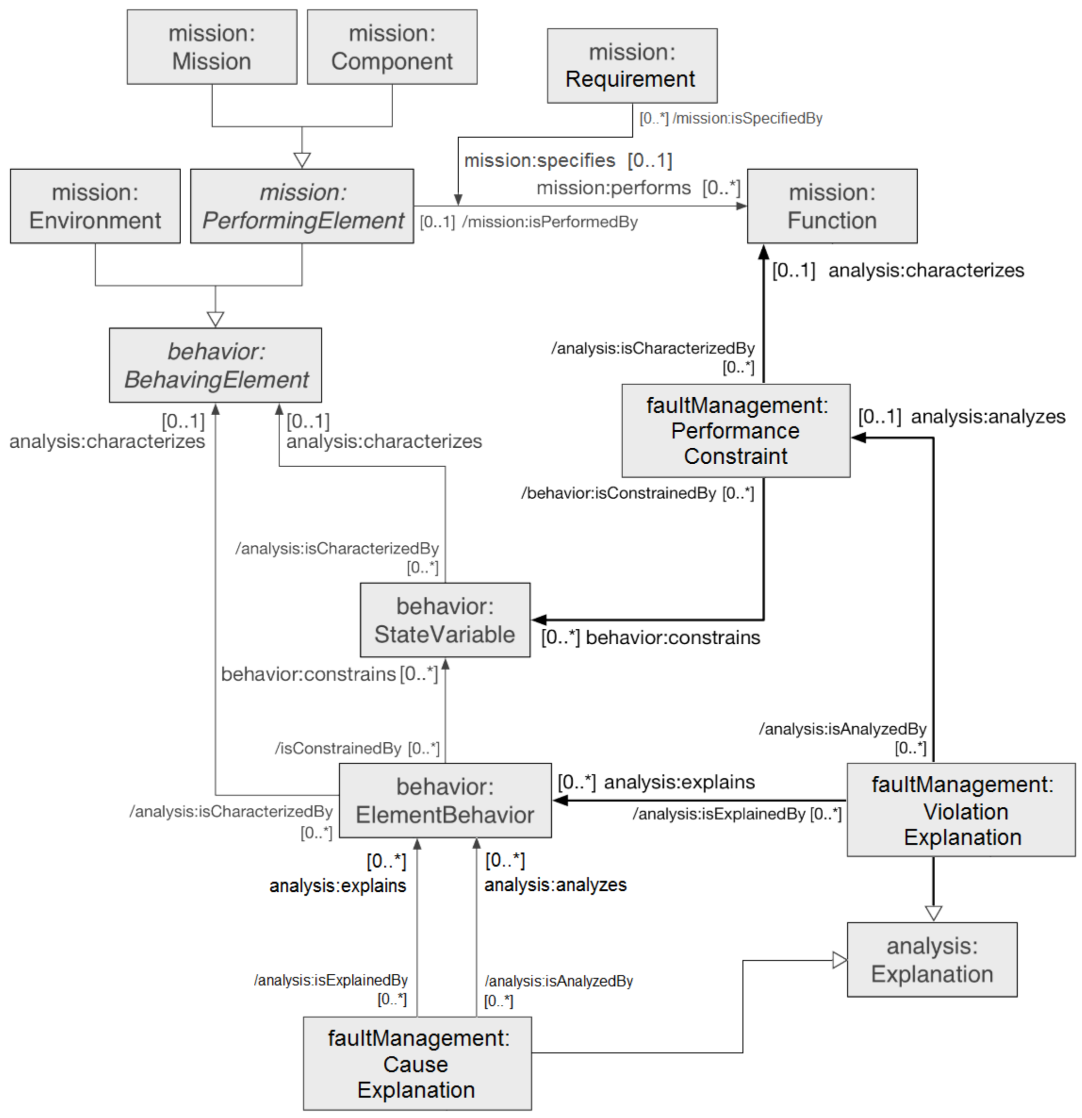

mission: Mission
mission: Component
mission: Requirement
[0..*] /mission:isSpecifiedBy
mission:specifies [0..1]
mission:performs [0..*]
mission: Environment
mission: PerformingElement
[0..1] /mission:isPerformedBy
mission: Function
[0..1] analysis:characterizes
behavior: BehavingElement
/analysis:isCharacterizedBy [0..*]
faultManagement: Performance Constraint
[0..1] analysis:analyzes
[0..1] analysis:characterizes
[0..1] analysis:characterizes
/behavior:isConstrainedBy [0..*]
/analysis:isCharacterizedBy [0..*]
behavior: StateVariable
[0..*] behavior:constrains
behavior:constrains [0..*]
/analysis:isAnalyzedBy [0..*]
/isConstrainedBy [0..*]
behavior: ElementBehavior
[0..*] analysis:explains
/analysis:isExplainedBy [0..*]
faultManagement: Violation Explanation
/analysis:isCharacterizedBy [0..*]
[0..*] analysis:explains
[0..*] analysis:analyzes
analysis: Explanation
/analysis:isExplainedBy [0..*]
/analysis:isAnalyzedBy [0..*]
faultManagement: Cause Explanation

Figure 1. The segment of the IMCE ontologies that are required for the model-based PRA methodology, slightly modified from the ontology diagram presented in Ref 6.

Table 1 summarizes the concepts from the IMCE ontologies used by the model-based PRA methodology, as well as gives brief examples of what each concept could be used to model.

Table 1. Definitions and examples of concepts from the IMCE ontologies used in the model-based PRA methodology. Package names are abbreviated (i.e., `mission:Requirement` is shown as `m:Requirement`)

Concept	Definition *(adapted from internal JPL ontology documentation)*	Example
mission: Requirement	A **`m:Requirement`** specifies an assertion about an element of the system that must be true for every acceptable realization of that element.	"When commanded, the attitude control subsystem (ACS) shall point the camera boresight vector to within 1 deg of the specified target vector."

mission: Function	A **`m:Function`** is an operation or activity performed by a **`m:Component`** in the context of executing a **`m:Mission`**. The performance of **`m:Function`**s can be specified by **`m:Requirement`**s, and therefore, **`m:Function`**s represent intended **`m:Component`** behavior.	"point reference boresight vector to align with specified target vector"
mission: PerformingElement	A ***`m:PerformingElement`*** is an object that performs one or more **`m:Functions`**.	attitude control subsystem (ACS) » note: because *`m:PerformingElement`* is abstract, the ACS would be realized as a `m:Component` in this case
behavior: BehavingElement	A ***`b:BehavingElement`*** is an abstract concept that represents system elements that have some behavioral specification (in the form of **`b:StateVariable`**s, state machines, or other constraint expressions). ***`b:BehavingElement`***s can be **`m:Component`**s, **`m:Mission`**s, or **`m:Environment`**s.	also the ACS » note: all `m:Components` are also *`b:BehavingElements`*
behavior: StateVariable	A **`b:StateVariable`** is a variable that represents a particular quantity (as per ISO-80000 [10]) of a ***`b:BehavingElement`*** that changes with time.	» cameraTargetVector ($\mathbb{R}^3$) » cameraBoresightVector ($\mathbb{R}^3$) » camPointingError (degrees), calculated as the angle between the cameraTargetVector and the cameraBoresightVector
behavior: ElementBehavior	A **`b:ElementBehavior`** specifies how the dynamic state of a ***`b:BehavingElement`*** is allowed to evolve in time. It is expressed by constraining **`b:StateVariable`**s through the use of state machines or other mathematic expressions.	ACS state machine with the following states: » Nominal_Operation » Failed_NoAttitudeKnowldge » Failed_NoThrustCtrl
faultManagement: PerformanceConstraint	Constraint on **`b:StateVariable`**s that represents the acceptable level of performance of an element with respect to a given **`m:Function`**. The violation of a **`fm:PerformanceConstraint`** indicates a degraded level of performance of the element.	» {camPointingError < 1 deg}
analysis: Explanation	An **`a:Explanation`** is a product that captures or summarizes the results of an analysis activity and relates it to one or more other model elements. It may contain narrative prose directly or provide a reference to external products. **`a:Explanation`**s can point to model elements as "inputs" through the **`a:analyzes`** relation and can point to model elements as "outputs" through the **`a:explains`** relation. **`a:Explanation`**s provide rationale for the **`a:explain`**ed elements.	n/a, abstract – see **`fm:ViolationExplanation`** and **`fm:CauseExplanation`** below

faultManagement: ViolationExplanation	This **`a:Explanation`** **`a:analyzes`** a **`fm:PerformanceConstraint`** and **`a:explains`** which **`b:ElementBehavior`**s do not meet it. The **`b:ElementBehavior`**s that violate a **`fm:PerformanceConstraint`** are known as failure modes in the context of that **`fm:PerformanceConstraint`** and its associated **`m:Function`**.	"The {camPointingError < 1 deg} performance constraint is violated when the ACS state machine is in either the Failed_NoAttitudeKnowldge or Failed_NoThrustCtrl states."
faultManagement: CauseExplanation	This **`a:Explanation`** **`a:analyzes`** an **`b:ElementBehavior`** and **`a:explains`** why the **`b:ElementBehavior`**s of other ***`b:BehavingElement`***s cause it. A **`fm:CauseExplanation`** can be used to capture the causal failure propagation between **`b:BehavingElement`**s when the **`fm:CauseExplanation`** indicates the potential causes of a failure mode.	"The Failed_NoThrustCtrl state of the ACS is caused by the ThrusterFailed state of Thruster Bank #1 or the ThrusterFailed state of Thruster Bank #2."

For the model-based PRA methodology, the following usages of the ontologies are particularly relevant:

- `m:Requirements` are used to capture the requirements on the performance of `m:Functions` by certain elements of the system (known abstractly as *`m:PerformingElements`*).
- The performance of functions is quantified through the use of `fm:PerformanceConstraints`, which constrain the range of allowable values for `b:StateVariables` of the *`b:BehavingElement`* that performs the `m:Function`.
- The violation of `fm:PerformanceConstraints` is captured with `fm:ViolationExplanations`, which identify the `b:ElementBehaviors` that are failure modes in the context of a given `m:Function`.
- Causes of failure modes are captured through the use of `fm:CauseExplanations`, which recursively identify `b:ElementBehaviors` of other *`b:BehavingElement`*s that are potential causes, allowing for the construction of a causal hierarchy.

The following sections provide further details on how the model-based PRA utilizes these ontological constructs to conduct a PRA.

III. Part 1: Elaborating on the Consequence/Requirements

The consequence of interest for a PRA is identified beforehand (often as a requirement or set of requirements that may not be met), but additional information is required to support the rest of the PRA. This section describes the additional information gathered from the system model and how it is used to elaborate upon the requirement(s) provided as the consequence of the PRA.

A PRA is typically conducted on performance requirements, which specify the degree to which a component performs a function. The model-based PRA methodology utilizes information in the system model to identify both the function and the component that performs that function. Next, the methodology locates all performance constraints that *characterize* that function. Each performance constraint describes discrete criteria for determining whether or not the system is performing some aspect of the function. Table 2 describes these steps in more detail below, showing the required preconditions and logic to be executed for each step. Figure 2 also provides an illustration of these first steps in the model-based PRA methodology.

Table 2. Steps 0-3 of the model-based PRA methodology, which elaborate upon the consequence given as a starting point for the analysis.

Step	Preconditions	Logic	Notes
0	n/a	Start from a given set of requirements (`RQ_set`)	This can be one or more requirements, identified as the consequence.
1	For each requirement in `RQ_set`, `RQ`:	Locate the *performs* relationship *specified* by `RQ`. Compile these relationships into a set (`Rel_set`).	The set of *performs* relationships are the subjects of the requirements.
2	For each *performs* relationship, `r`, in `Rel_set`:	Locate the function, `F`, that is a target of the relationship and the performing element, `PE`, that is the source of the relationship.	Performing element and function are defined in more detail in Section II.
3	For each `function`, `F`, from Step 2:	Locate all performance constraints, (`PC_set`), that *characterize* `F`.	These performance constraints describe discrete criteria for determining whether or not the system is performing some aspect of the function[7].

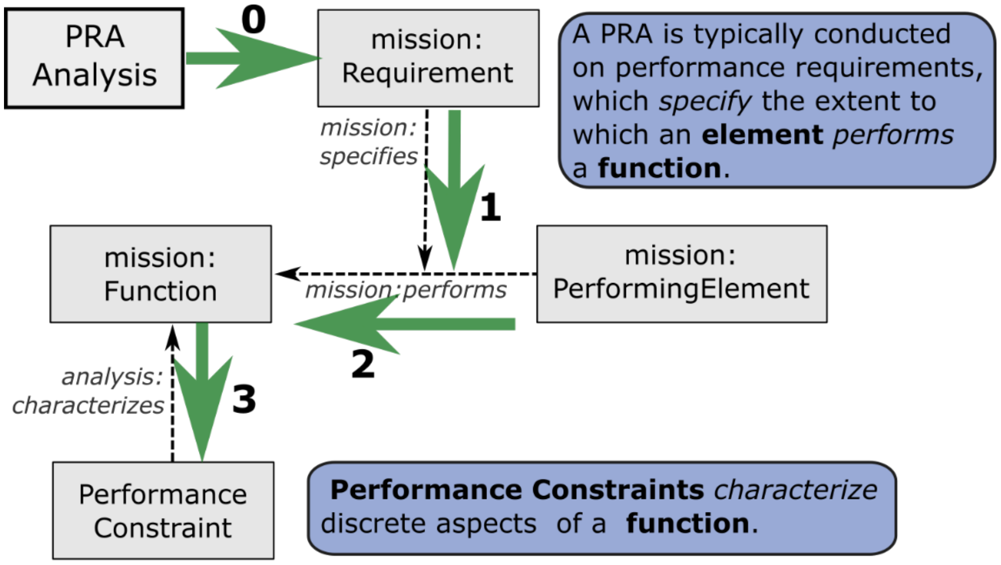

PRA Analysis
0
mission: Requirement
A PRA is typically conducted on performance requirements, which *specify* the extent to which an **element** *performs* a **function**.
mission: specifies
1
mission: Function
mission:performs
mission: PerformingElement
2
analysis: characterizes
3
Performance Constraint
Performance Constraints *characterize* discrete aspects of a **function**.

Figure 2. An illustration of steps 0-3 of the model-based probabilistic risk assessment. The methodology traces from one or more requirements (given as an input) to locate performance constraints that represent discrete acceptable levels of performance of an element with respect to the function specified by the requirement. Numbers next to green arrows indicate references to the steps in Table 2.

The set of performance constraints identified in Step 3 must be satisfied if the performing element is to perform the function that is the subject of the requirement. If any of these performance constraints are violated, the performing element will not be able to perform the function at an acceptable level. The aggregate collection of requirements, performing elements, functions, and performance constraints constitute the full elaboration of the given consequence of the risk triplet.

IV. Part 2: Generating Risk Scenarios

The following section describes how the model-based PRA methodology generates risk scenarios, the second component of a risk triplet.

A. Identifying Failure Modes from Requirements

Starting from the performance constraints identified in Step 3, the next step in the model-based PRA methodology is to identify element behavior that violates those performance constraints. This is done using Violation Explanations[6], which are encoded into the model by domain experts to identify the element behavior that violates each performance constraint. This element behavior is considered a "failure mode" in the context of the performance constraint[6], defined as *"the characteristic manner in which a failure occurs, independent of the reason for failure; the condition or state which is the end result of a particular failure mechanism; the consequence of the failure mechanism though which the failure occurs."*[11]. Table 3 describes these steps in more detail, which are also illustrated in Figure 3.

Table 3. Steps 4 and 5 of the model-based PRA methodology, which locate the highest-level failure modes in the context of the performance constraints identified in Step 3. These failure modes are potential system behavior that could cause the consequence could occur (i.e., performance requirements not being met).

Step	Preconditions	Logic	Notes
4	For each performance constraint, **`PC`**, in **`PC_set`**:	Find all Violation Explanations that *analyze* **`PC`** and compile them into a list of Violation Explanations, **`VE_set`**.	The PRA methodology requires that there be at least one Violation Explanation that *analyzes* a performance constraint in **`PC_set`**.
5	For each Violation Explanation, **`VE`**, in **`VE_set`**	Find the set of element behavior, **`EB_set`**, that **`VE`** *explains* as a violation of **`PC`**. These behaviors are referred to as "failure modes" [6].	These element behaviors violate the given performance constraint, **`PC`**. The PRA methodology assumes that the occurrence of the <u>intersection</u> of this element behavior violates the performance constraint.

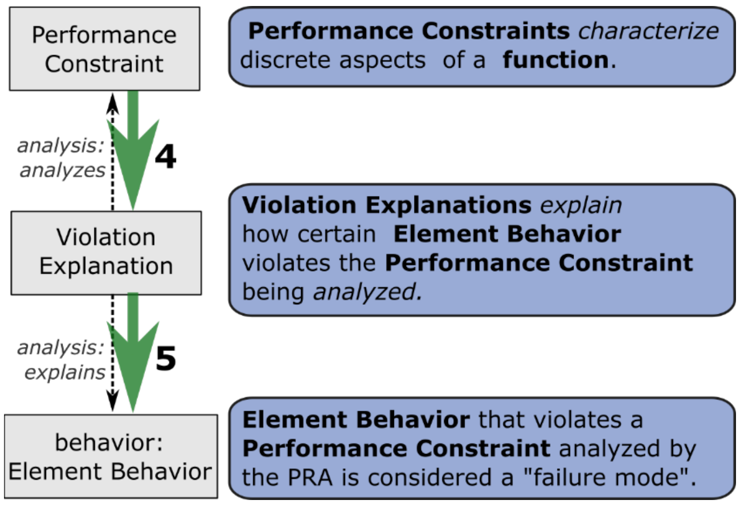

Performance Constraint
Performance Constraints characterize discrete aspects of a function.
analysis: analyzes
4
Violation Explanation
Violation Explanations explain how certain Element Behavior violates the Performance Constraint being analyzed.
analysis: explains
5
behavior: Element Behavior
Element Behavior that violates a Performance Constraint analyzed by the PRA is considered a "failure mode".

Figure 3. An illustration of steps 4 and 5 of the model-based probabilistic risk assessment. The methodology locates any Violation Explanations that analyze the performance constraint(s) identified in step 3 and explain how they could be violated by one or more element behaviors, called "failure modes". Numbers next to green arrows indicate references to the steps in Table 3.

B. Identifying Basic Events/Behavior from Failure Modes

The element behaviors identified in Step 5 can be at any level of abstraction and are not guaranteed to be "basic behavior", defined as element behavior that has no identified causes. The risk scenarios produced by a PRA must be composed of "basic events" [1], defined as the occurrence of basic behavior. To locate basic behavior and basic events, the set of element behavior identified in Step 5 (i.e., Spacecraft unable to perform de-orbit maneuver) must be decomposed to more detailed behavior (i.e., Thruster valve A stuck closed). Traditionally this decomposition is done by the PRA team consulting domain experts at various levels of design abstraction. Using MBSE, domain experts can encode the causal information directly in the model using special links, called Cause Explanations, between different element behaviors. A

design abstraction, may cause them to occur[6]. This approach allows subject matter experts to encode causal relationships in the model in a rigorous, formal manner. This methodology requires that the Cause Explanations and element behaviors compose a directed acyclic graph, which can be leveraged by a host of analyses. With this information encoded in the model, the model-based PRA methodology recursively traverses through the causal graph to locate basic behavior using the steps described in Table 4. The traversal of causal dependencies is illustrated in Figure 4, along with the construction of the logical expression of the basic events, described in the following section.

Table 4. Step 6 of the model-based PRA methodology, which recursively traverses through causal dependencies to locate basic behavior, which has no identified causes. The occurrence of this basic behavior is considered "basic events", which compose the risk scenario.

Step	Preconditions	Logic	Notes
6.1	For each element behavior, **`EB`**, in **`EB_set`**:	Find all Cause Explanations, **`CE_set`**, that *analyze* why **`EB`** might occur.	These Cause Explanations provide information as to why **`EB`**, termed the "consequent" behavior, might occur due to other element behavior.
6.2	**`CE_set`**	If there are no Cause Explanations produced in Step 6.1 (**`CE_set`** is empty): then store the element behavior, **`EB`**, in a logical expression using information from the causal dependency graph (see Section IV.C below).	Any behavior that is not *analyzed* by a Cause Explanation has no identified causes and is therefore considered a "basic behavior." The occurrence of basic behavior is considered a "basic event" from Fault Tree Analysis nomenclature.
6.3	**`CE_set`**	If there are Cause Explanations produced in Step 6.1: for each one, **`CE`**, in **`CE_set`**, find all element behavior, **`EB_set2`**, that are *explained* by **`CE`**.	This element behavior is "antecedent", in that it is the cause of the element behavior **`EB`**.
6.4	**`EB_set2`**	Go back to Step 6.1, using **`EB_set2`** as the initial set of element behavior.	This recursively traverses through causal dependencies.

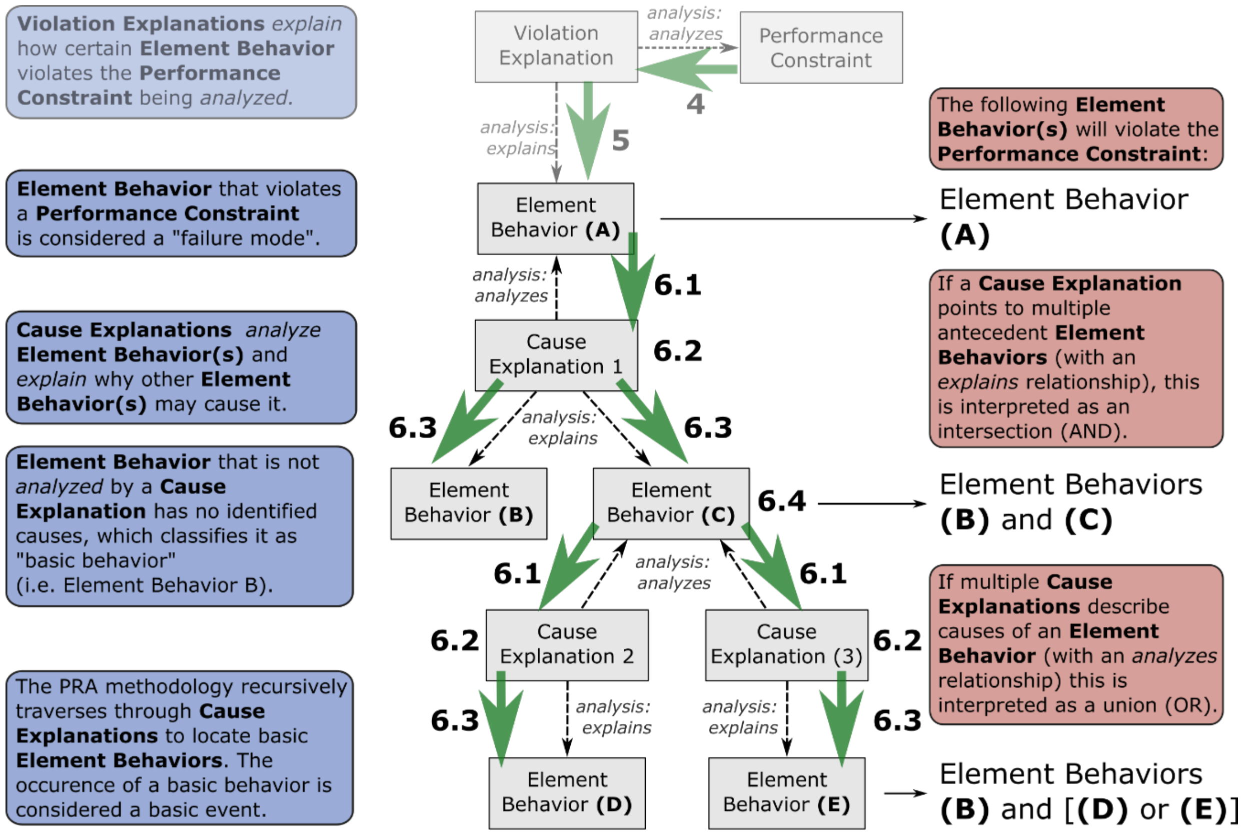

Violation Explanations *explain* how certain **Element Behavior** violates the **Performance Constraint** being *analyzed*.
Element Behavior that violates a **Performance Constraint** is considered a "failure mode".
Cause Explanations *analyze* **Element Behavior(s)** and *explain* why other **Element Behavior(s)** may cause it.
Element Behavior that is not *analyzed* by a **Cause Explanation** has no identified causes, which classifies it as "basic behavior" (i.e. Element Behavior B).
The PRA methodology recursively traverses through **Cause Explanations** to locate basic **Element Behaviors**. The occurence of a basic behavior is considered a basic event.
Violation Explanation
analysis: analyzes
Performance Constraint
4
analysis: explains
5
Element Behavior (A)
analysis: analyzes
6.1
Cause Explanation 1
6.2
6.3
analysis: explains
6.3
Element Behavior (B)
Element Behavior (C)
6.4
6.1
analysis: analyzes
6.1
Cause Explanation 2
Cause Explanation (3)
6.2
6.2
6.3
analysis: explains
analysis: explains
6.3
Element Behavior (D)
Element Behavior (E)
The following **Element Behavior(s)** will violate the **Performance Constraint**:
Element Behavior **(A)**
If a **Cause Explanation** points to multiple antecedent **Element Behaviors** (with an *explains* relationship), this is interpreted as an intersection (AND).
Element Behaviors **(B)** and **(C)**
If multiple **Cause Explanations** describe causes of an **Element Behavior** (with an *analyzes* relationship) this is interpreted as a union (OR).
Element Behaviors **(B)** and [**(D)** or **(E)**]

Figure 4. An illustration of Step 6 of the PRA methodology, which is executed recursively until it locates a set of basic element behavior that has no identified causes. The methodology for determining the logical expression is presented in red text boxes on the right. Numbers next to green arrows indicate references to the steps in Table 4.

C. Boolean Expression Construction, Logical Reduction, and Risk Scenario Generation

As shown in Figure 4, the model-based PRA methodology traverses through the Cause Explanation linkages to locate a set of basic events. As it does so, it uses the following methodology to aggregate the basic events within a logical expression of intersections and unions. If a Cause Explanation points to multiple antecedent element behaviors (with an *explains* relationship, these are the "causes"), this is interpreted as an intersection of the set of element behaviors. In Figure 4, this equates to the Cause Explanation 1 asserting that the occurrence of the intersection (AND) of element behaviors B and C will cause element behavior A to occur. If multiple Cause Explanations describe a consequent element behavior (with an *analyzes* relationship, these are the "effects") this is interpreted as the occurrence of the union (OR) of the sets of antecedent element behavior will cause the consequent element behavior to occur. In Figure 4, this equates to Cause Explanations 2 and 3 asserting that element behaviors D or E will cause element behavior C to occur. We note that the latest work on the Fault Management Ontology[6] provides a more rigorous method for describing more complex logical expressions within a single Cause Explanation, but that ontological update has not yet been incorporated into this methodology.

This logical expression of basic events and operators describes the system behavior that results in the performance constraints associated with the requirement (from which the PRA began) not being met. However, the set of basic events and logical operators is not necessarily in disjunctive normal form, as can be seen in the example in Figure 4. The risk scenario in each risk triplet produced by a PRA must be composed only of intersections of events[1]. To achieve this, the set of basic events and logical operators must be logically reduced to disjunctive normal form *(i.e., [B and (D or E)] → [(B and D) or (B and E)])* via logical equivalences, De Morgan's laws, and the distributive law[1].

The element behavior identified in Step 6.2 is not sufficient to fully define a risk scenario. Each element behavior must be associated with at least one behaving element (behavior interactions will have multiple behaving elements). Because the behavior information is encoded in the model according to the Behavior Ontology[7], it can be leveraged to determine which behaving element is associated with each element behavior. This allows the same fundamental behavior model to be used for the PRA as well as any other behavior modeling that may be needed, such as a power profile analysis or science data collection modeling[4].

After the logically reduced expression of element behavior is coupled with the associated behaving elements, minimal cut sets are produced for each intersection of element behavior that is separated by a union operator ("OR"). For each minimal cut set, a risk scenario is produced by aggregating the set of element behavior ("Stuck Closed"), their associated behaving elements ("Valve A"), and the Violation Explanation ("Fuel cannot flow to the main thruster, which inhibits the thruster's ability to execute the deorbit maneuver").

V. Part 3: Identifying Appropriate Probabilistic Equations

This section describes a draft methodology for identifying the appropriate probabilistic equation for a given risk scenario, which describes the probability of the risk scenario occurring as a function of time. Automatic identification of appropriate probabilistic equations is a key first step in quantifying the likelihood of each risk scenario occurring, though the complete calculate of the likelihood is left as an item of future work. To identify the appropriate probabilistic equations, the model-based PRA methodology leverages information in the system model concerning redundancy, behavior, and operational scenarios. Behavior and scenario information is used to determine how the components involved in a risk scenario are operated throughout a given mission scenario. For example, a component that is continuously operating will be characterized by a different equation than a component that operates in a cyclical manner. To make this determination, this methodology requires the use of an additional stereotype, *«PRA:operating»*, on top of the behavior ontology[7]. This stereotype is used to identify element behavior that is considered "operating" from a probabilistic equation perspective, which can be referred to as an operating mode. Other model embeddings may be used to identify operational modes in the future, but the use of this stereotype was chosen as an initial method for capturing that information.

The sections below describe a preliminary library of four probabilistic equations, their applicable situations, and what information is needed in the system model to enable probabilistic model identification. It is important to note that this part of the methodology is specifically suited to element behavior that is captured as elements in a state machine (states, transitions, and state machine regions). Other types of element behavior can be included in the future, but for this work state machine elements were chosen for two reasons. First, they are the most common form of element behavior currently used in model-based systems engineering at JPL, and second, they provided a mechanism to capture the majority of the element behavior seen in risk scenarios generated via previous methods.

A. Equation 1: Single String, Continuous Operation

This equation describes a single component that has no redundant backups ("single string") and is continuously operational. One example would be a single-string Stellar Reference Unit (SRU), which is solely responsible for absolute spacecraft attitude determination and is operational across the entire mission. If the SRU fails, there are no redundant components that can provide the same functionality. The equation that governs the failure rate, *R(t)*, of such a component is as follows[1]:

$$R(t) = e^{-\lambda t} \tag{1}$$

where λ is the failure rate of the component (Note: for Equation 1 this is technically equal to the transition probability from the initial operating state to the problematic basic behavior identified by the Cause Explanation) and t is the time of continuous operation since the component began operating (at $t=0$). The conditions required in the system model to positively identify risk scenarios that can be characterized by Equation 1 are enumerated in Table 5 and illustrated in Figure 5. This equation assumes that 1) λ is time-independent, 2) if the component is in an operational state, it is able to perform its function (i.e., all supporting functionality is available), and 3) that at $t=0$ the component is able to operate with a probability of unity.

Table 5. The model-based PRA methodology requires the following conditions to be met in order to positively identify risk scenarios that can be characterized by Equation 1.

Condition	Description	Notes/Justification
1-A	The risk scenario must include only one element behavior: the parent Cause Explanation (from Step 6.3) or Violation Explanation (from Step 5) must point to only one antecedent element behavior (using the *explains* relationship).	If more than one element behavior is involved in the risk scenario, Equation 1 does not hold.
1-B	The element must not be a part of a redundant set (either block or functional): the behaving element must not be *explained* by a Redundancy Explanation[6].	This is assured by the satisfaction of condition A with a properly modeled Cause Explanation.
1-C	The behaving element must have at least one operating mode, captured as a state with the «PRA:operating» stereotype.	At least one operating mode is required to determine how the component is operated during a given mission scenario.
1-D	In the mission scenario analyzed by the PRA, the behaving element must be in a single operating mode continuously from the beginning of the scenario until the end of intended use.	If the element switches between operational and non-operating modes (or even between different operating modes), a different equation must be used.

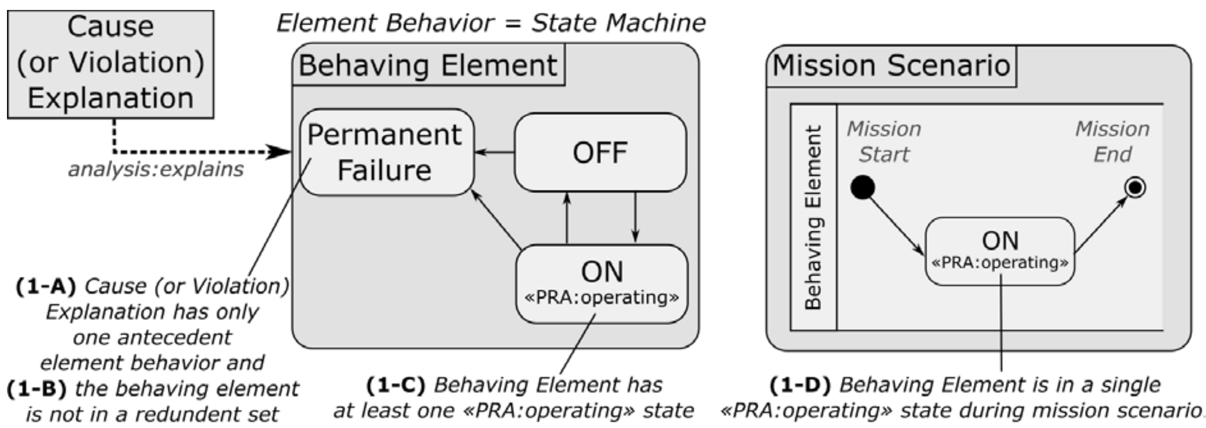

Cause (or Violation) Explanation
analysis:explains
Element Behavior = State Machine
Behaving Element
Permanent Failure
OFF
ON «PRA:operating»
Mission Scenario
Behaving Element
Mission Start
Mission End
ON «PRA:operating»
(1-A) Cause (or Violation) Explanation has only one antecedent element behavior and (1-B) the behaving element is not in a redundent set
(1-C) Behaving Element has at least one «PRA:operating» state
(1-D) Behaving Element is in a single «PRA:operating» state during mission scenario

Figure 5. An illustration of the conditions required to positively identify Equation 1 as the correct probabilistic model: (1-A) There must be only one element behavior in the risk scenario, the behaving element must (1-B) not be redundant, (1-C) have at least one «PRA:operating» state, and (1-D) be in a single operating state continuously from the beginning of the scenario.

B. Equation 2: Single-String, Cyclic Operation

This equation describes a single component that has no redundant backups (a.k.a "single string") and operates cyclically over the mission. An example is a standalone heater with no redundant backup that operates cyclically through the mission. The equation that governs the failure rate, *R(t)*, of such a component is as follows (derived in Appendix B):

$$R(t) = (1-p)^M (1-q)^N exp(-\lambda \textstyle\sum_{n=1}^{N} t_n) \tag{2}$$

where λ is the failure rate of the component while operating, t_n is the time of the n^{th} operation of the component, p is the probability that the component fails to stop operating at the end of a cycle (i.e., heater stuck on), q is the probability that the component fails to start operating at the beginning of a cycle (i.e., heater stuck off), N is the number of transitions from non-operating to operating (heater turning on) encountered from $0<time<t$, and M is the number of transitions from operating to non-operating (heater turning off) encountered from $0<time<t$. The conditions in the model required to positively identify a risk scenario that can be characterized by Equation 2 are enumerated in Table 6 and illustrated in Figure 6. This equation assumes that if the component is in an operational state, it is able to perform its function (i.e., all supporting functionality is available), λ, p, and q are time-independent, and that at $t=0$ the component is able to operate with a probability of unity.

Table 6. The model-based PRA methodology requires the following conditions to be met in order to positively identify risk scenarios that can be characterized by probabilistic Equation 2.

Condition	Description	Notes/Justification
2-A	The risk scenario must include only one element behavior: the parent Cause Explanation (from Step 6.3) or Violation Explanation (from Step 5) must point to only one antecedent element behavior (using the *explains* relationship).	If more than one element behavior is involved in the risk scenario, Equation 1 does not hold.
2-B	The element must not be a part of a redundant set (either block or functional): the behaving element must not be *explained* by a Redundancy Explanation[6]	This is assured by the satisfaction of condition A with a properly modeled Cause Explanation.
2-C	The behaving element must have at least one operating mode, captured as a state with the «PRA:operating» stereotype.	At least one operating mode is required to determine how the component is operated during a given mission scenario.
2-D	In the mission scenario analyzed by the PRA, the behaving element must transition from a non-operational state to an operational state (or vice versa) at least once.	If no transitions occur (and all other conditions hold), Equation 1 should be used.

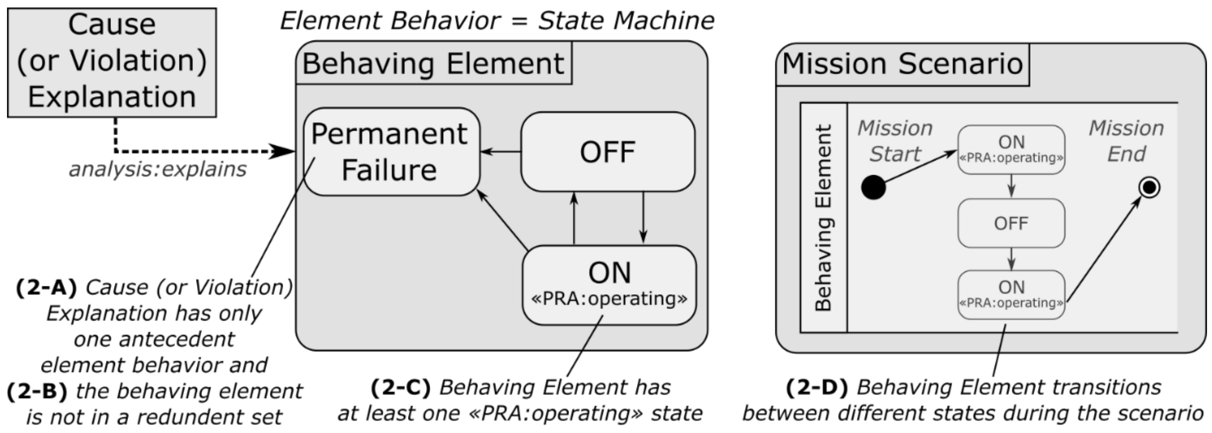

Cause (or Violation) Explanation
analysis:explains
Element Behavior = State Machine
Behaving Element
Permanent Failure
OFF
ON «PRA:operating»
Mission Scenario
Behaving Element
Mission Start
ON «PRA:operating»
OFF
ON «PRA:operating»
Mission End
(2-A) Cause (or Violation) Explanation has only one antecedent element behavior and (2-B) the behaving element is not in a redundent set
(2-C) Behaving Element has at least one «PRA:operating» state
(2-D) Behaving Element transitions between different states during the scenario

Figure 6. An illustration of the conditions required to positively identify Equation 2 as the appropriate probabilistic model: (2-A) There must be only one element behavior in the risk scenario, the behaving element must (2-B) not be redundant, (2-C) have at least one «PRA:operating» state, and (2-D) transition from a non-operating state to an operating state (or vice versa) at least once.

C. Equation 3: Redundant, Continuous Operation ("Hot Spare"), Common Cause Failure

This equation describes a grouping of two components that are block redundant, operate continuously over the mission, and can potentially fail due to a common cause failure. An example would be two identical thermostats that measure the temperature of the same battery to ensure it remains within appropriate limits. If one thermostat fails, the second thermostat can still provide the required functionality. The equation that governs the failure rate, *R(t)*, of a grouping of such components is as follows[1]:

$$R(t) = 2\, e^{-(\lambda_I+\lambda_C)t} - e^{-(2\lambda_I+\lambda_C)t} \qquad (3)$$

where λ_I is the independent failure rate of each component, λ_C is the common-cause failure rate of each component, and *t* is the time of continuous operation. Table 7 enumerates the conditions required in the system model to positively identify risk scenarios that can be characterized by Equation 3. This equation assumes that the components be block redundant, which ensures that they will have the same independent failure rate, λ_I. This equation captures "acute" common-cause failures, where the failure of both components essentially happens simultaneously, rather than long-term common-cause failures (i.e., two components failing 6 months apart due to common radiation degradation). Furthermore, Equation 3 assumes that if the component is in an operational state, it is able to perform its function (i.e., all supporting functionality is available), λ_I and λ_C are time-independent, and that at *t=0* each component is able to operate with a probability of unity.

Table 7. The model-based PRA methodology requires the following information in the system model to positively identify risk scenarios that can be characterized by probabilistic Equation 3.

Condition	Description	Notes/Justification
3-A	The risk scenario must include two, and only two, element behaviors: the parent Cause Explanation (from Step 6.3) must *explain* two, and only two, element behaviors	Equation 3 describes the reliability of a two-element redundant set
3-B	The two components must be block redundant with, and only with, each other: a Block Redundancy Explanation[6] must *explain* both components in the risk scenario and no other components.	The block redundancy ensures that the independent failure rate of both elements is the same. This equation does not capture redundancy with more than 2 components.
3-C	The behaving element must have at least one operating mode, captured as a state with the «PRA:operating» stereotype.	At least one operating mode is required to determine how the components are operated during a given mission scenario.
3-D	In the mission scenario analyzed by the PRA, both of behaving elements must be in an operating mode from the beginning of time.	This satisfied the "Hot-Spare" condition of Equation 3.
3-E	Both element behaviors must be susceptible to occurring from a common cause (i.e., manufacturing defect): a Common Cause Explanation must *analyze* both of the element behaviors.	Equation 3 describes the combined probability of both independent and common-cause occurrences.

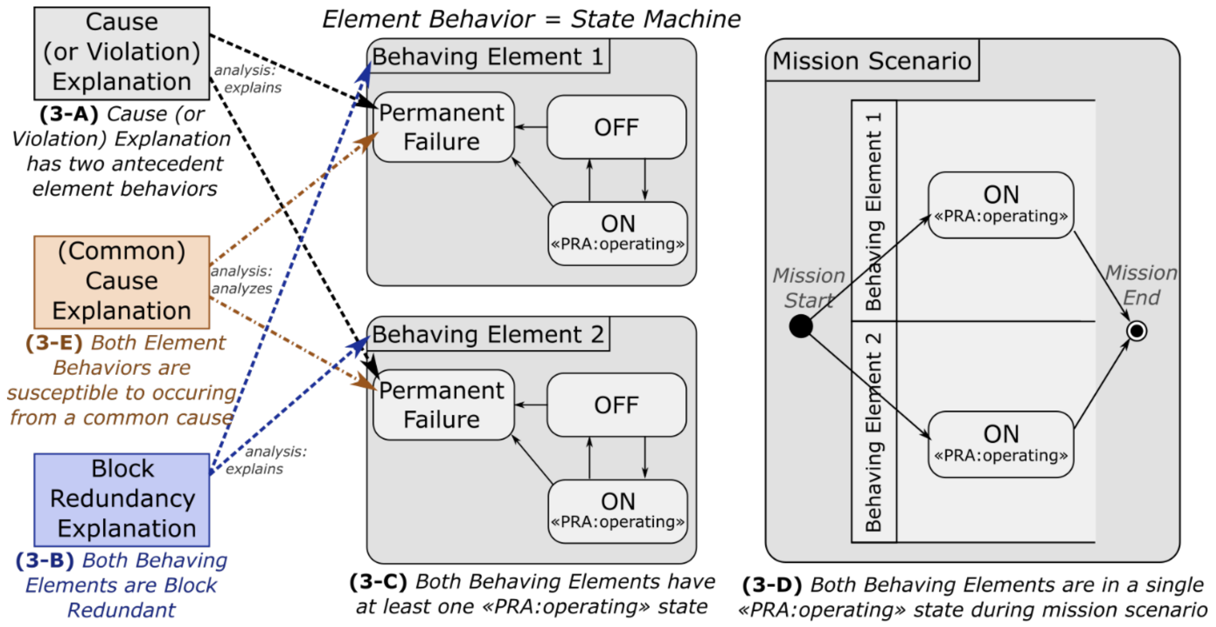

Element Behavior = State Machine
Cause (or Violation) Explanation
analysis: explains
(3-A) Cause (or Violation) Explanation has two antecedent element behaviors
(Common) Cause Explanation
analysis: analyzes
(3-E) Both Element Behaviors are susceptible to occuring from a common cause
Block Redundancy Explanation
analysis: explains
(3-B) Both Behaving Elements are Block Redundant
Behaving Element 1
Permanent Failure
OFF
ON «PRA:operating»
Behaving Element 2
Permanent Failure
OFF
ON «PRA:operating»
(3-C) Both Behaving Elements have at least one «PRA:operating» state
Mission Scenario
Mission Start
Mission End
Behaving Element 1
Behaving Element 2
ON «PRA:operating»
ON «PRA:operating»
(3-D) Both Behaving Elements are in a single «PRA:operating» state during mission scenario

Figure 7. An illustration of the conditions required to positively identify Equation 3 as the appropriate probabilistic model: (3-A) There must be two element behaviors in the risk scenario, the behaving elements must (3-B) be block redundant, (3-C) have at least one «PRA:operating» state, (3-D) continuously be in an «operating» state during the mission scenario, and (3-E) be susceptible to a common cause effect.

D. Equation 4: Redundant, Single Component Operational ("Cold Spare")

This equation describes a grouping of three components. Two components are block redundant, one component, called the "primary", operates continuously over the mission while the other, called the "backup", is not operating and acts as a "cold spare". The third component, termed the "controller", is responsible for transferring operation to the cold spare after the failure of the primary component. An example would be a set of redundant IMUs that measure rotation rates of the spacecraft, only one of which is operating at the start of the mission. If the primary IMU fails, the controller (i.e., fault protection system) sends a command for a transition to the backup IMU, which may or may not succeed. If the transition successfully occurs, the possibility of a failure of the backup IMU must also be considered. The equation that governs the failure rate, *R(t)*, of a grouping of such components is as follows (derived in Appendix A):

$$R(t) = e^{-\lambda t}(1 + (1 - p)\lambda t) \tag{4}$$

where λ is the independent failure rate of both redundant components, p is the probability that the controller is unable to transfer functionality to the backup, and t is the time since the primary began operating. This equation assumes that the primary and backup components are block redundant, which ensures that they will have the same failure rate, λ, which is also assumed to be time-independent. Equation 4 also assumes that the probability of a failure of the cold-spare component is zero while it is non-operational, if a component is in an operational state it is able to perform its function (i.e., all supporting functionality is available), and that at $t=0$ the initially operating component is able to operate with a probability of unity. Another critical assumption in Equation 4 is that p, the probability of a failure to swap to the cold-spare, is time-independent. Historical fault data from JPL has shown that the majority of swap failures (i.e. captured by p) are due to software faults, the failure rate of which does not depend on time.

Table 8. The model-based PRA methodology requires the following information in the system model to positively identify risk scenarios that can be characterized by probabilistic Equation 4.

Condition	Description	Notes/Justification
4-A	Two risk scenarios must exist, the union of which is characterized by Equation 4: • One must be derived from a Cause Explanation that *explains* the failure using a single element behavior (i.e., failure of the primary component) and a state machine self-loop transition (i.e., failure of controller to transition to the cold-spare) • The other must be derived from a Cause Explanation *explains* the failure using same element behavior (i.e., failure of the primary component), a state machine transition between two different states (i.e., a successful transition to the cold-spare), and a second element behavior (i.e., failure of the cold-spare).	Equation 4 characterizes the probability of the union of either risk scenario occurring. An additional nuance of these conditions is that the transition from the controller's state machine must reflect a transfer of operation from the primary to the backup, though the model embedding for this is not made explicit.
4-B	The two components must be block redundant with, and only with, each other: a Block Redundancy Explanation[6] must *explain* both components in the risk scenario and no other components.	The block redundancy ensures that the independent failure rate of both elements is the same. This equation does not capture redundancy with more than 2 components.
4-C	The behaving element that is initially operating must have at least one operating mode, captured as a state with the «operating» stereotype. The transition in the state machine of the 3rd behaving element must reflect an attempt to transfer control between the two other behaving elements in the risk scenario (i.e., a fault protection system)	At least one operating mode is required to determine how the components are operated during a given mission scenario. Equation 4 captures the probability of failure including a failed or successfully attempt to swap control to a cold-spare.
4-D	In the mission scenario analyzed by the PRA, one of behaving elements must be in an operating mode while the other is in a non-operating mode continuously from the beginning of the scenario.	This satisfied the "Cold-Spare" condition of Equation 4.

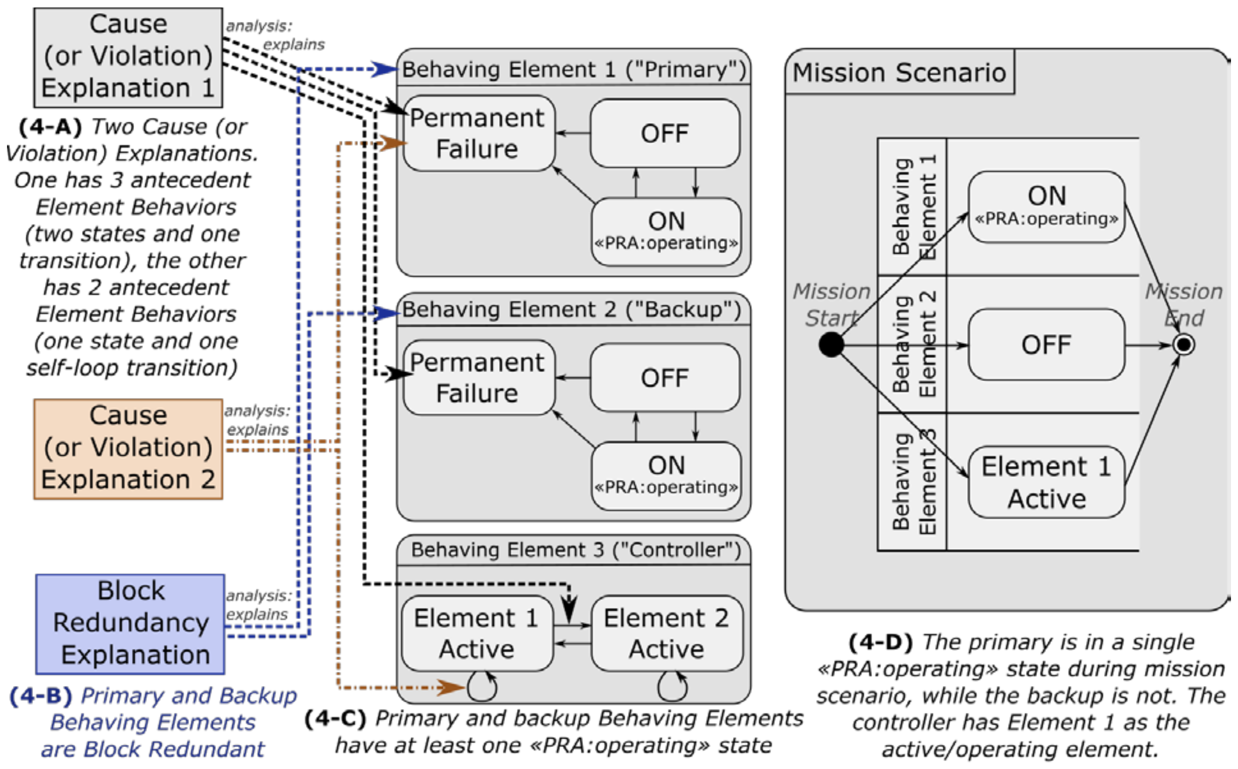

Cause (or Violation) Explanation 1
analysis: explains
(4-A) Two Cause (or Violation) Explanations. One has 3 antecedent Element Behaviors (two states and one transition), the other has 2 antecedent Element Behaviors (one state and one self-loop transition)
Cause (or Violation) Explanation 2
analysis: explains
Block Redundancy Explanation
analysis: explains
(4-B) Primary and Backup Behaving Elements are Block Redundant
Behaving Element 1 ("Primary")
Permanent Failure
OFF
ON «PRA:operating»
Behaving Element 2 ("Backup")
Permanent Failure
OFF
ON «PRA:operating»
Behaving Element 3 ("Controller")
Element 1 Active
Element 2 Active
(4-C) Primary and backup Behaving Elements have at least one «PRA:operating» state
Mission Scenario
Mission Start
Mission End
Behaving Element 1
Behaving Element 2
Behaving Element 3
ON «PRA:operating»
OFF
Element 1 Active
(4-D) The primary is in a single «PRA:operating» state during mission scenario, while the backup is not. The controller has Element 1 as the active/operating element.

Figure 8. An illustration of the conditions required to positively identify Equation 4 as the appropriate probabilistic model: (4-A) There must exist two risk scenarios, one of which has one state and one transition for element behaviors, and the other which has two states and a transition for element behaviors, in the risk scenario, the behaving elements must (4-B) be block redundant, (4-C) have at least one «operating» state, and (4-D) the primary must continuously be in an «PRA:operating» state during the mission scenario, while the backup is not.

VI. Future Work: Calculating the Probability of Occurrence

The third element of a risk triplet, the probability of occurrence, requires two steps. The appropriate probabilistic equations must first be identified according to the methodology presented in Section V. Next, appropriate values must be selected for each parameter in the probabilistic equations and the quantitative calculation carried out. This second step is left as an item of future work. The current methodology does not include the calculation of this element as a choice of scope. The methodology described in the previous steps will identify an equation that captures the probability of a given risk scenario (or a union of risk scenarios, in the case of Equation 4) occurring, but the identification of the appropriate values for each parameter in the equation is not a straightforward task. These values depend on the component described by the equation and how it will be operated during the mission. Furthermore, there often exist multiple values for a single parameter (e.g., the time-independent failure rate) for a given component, and these various values may be orders of magnitude apart.

When significant disparity in reported values exists, it becomes incumbent on the risk analysts to either understand the technical basis for the variation and select a value representative of the system being analyzed, or perform sensitivity studies on how uncertainty in which values to apply translates into changes in system risk. The ability to defensibly decide which values are applicable to the various parameters in a reliability model is not something that can be easily automated, although the information needed for risk assessors to make such determinations should be available within the system model. The challenge is often that the parameter values reported in the literature have insufficient provenance to indicate their applicability to a specific hardware item within a particular system. This is where application of engineering judgement and performing additional literature searches or expert interviews becomes vital.

VII. Results

Preliminary tooling was created to carry out the model-based PRA methodology on a sample system model. A traditional PRA was conducted on an actual spacecraft and several hundred risk scenarios were generated, ranging from an SRU failure to thruster latch valves becoming stuck. A sample SysML[12] model of the same spacecraft was created using MagicDraw™[13,*], including a subset of the failure modes and causal dependencies analyzed in the traditional PRA. Preliminary tooling was created as a plugin for MagicDraw™[13], which can carry out Steps 1-6 of the model-based PRA methodology on a well-formed system model. Figure 9 shows a subset of the results from a traditional PRA (top) compared to those generated by the model-based PRA methodology (bottom).

The results from the model-based PRA provide additional information from each step in the methodology in the purple section at the top, including the requirements, performing elements (components), performance constraints, violation explanations and the top-level failure modes. Each blue section below depicts a single risk scenario, including the plain text documentation of the Cause Explanation in bold and its associated formal element behavior and behaving element (in the form "{*behaving element = element behavior}*"). The number of the corresponding risk scenario from the traditional PRA is shown on the left side of the figure. From Figure 9 it is clear that each risk scenario selected from the traditional PRA was able to be captured in a SysML model[12] and identified using the prototype software. We note that although a draft methodology has been presented for identifying appropriate probabilistic equations, that section of the methodology has not yet been implemented in the software prototype so the results are not shown in Figure 9.

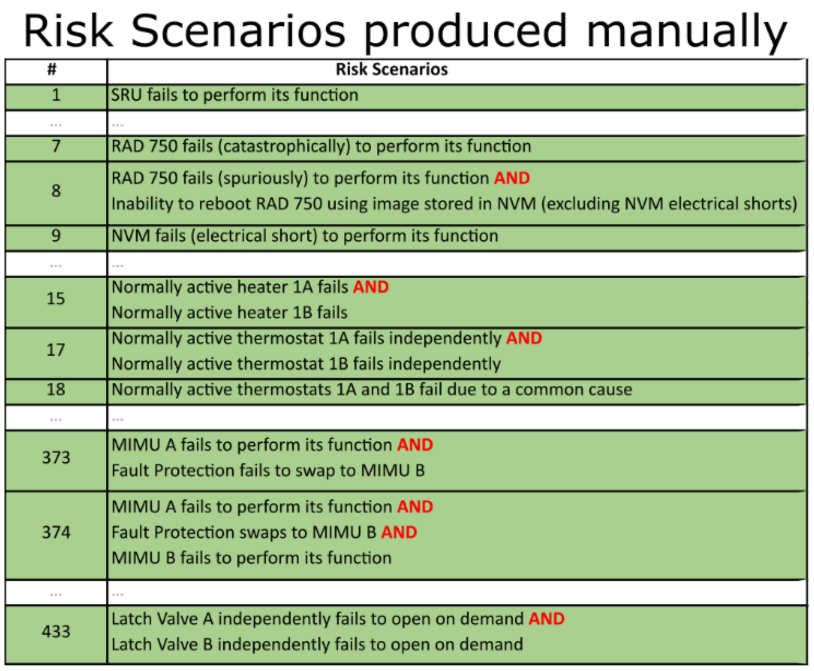
VIII. Conclusion

A PRA aims to identify and assess potential risks to system technical performance requirements for the purpose of furnishing risk insights into project decisions. PRAs have traditionally been conducted manually using an isolated data model, but as the complexity of technological systems continues to rise the techniques for conducting PRAs must evolve to meet this challenge.

In this work we present progress towards a model-based PRA methodology that can address some of the issues of rising system complexity. As described in Section I, the product of a model-based PRA is a set of risk triplets, each of which contains a consequence (addresses the question *"what are the consequences?"*), a risk scenario (addresses the question *"what are the causes?"*), and a probability of occurrence (addresses the question *"how likely are the causes?"*). Section III describes how the model-based PRA methodology takes the requirement(s), given as an input to the PRA, and gathers information from the system model to fully define the consequence as the first element of a risk triplet. Section IV covers the section of the methodology that identifies the top level failure modes and then traverses through causal dependencies in the system model to produce a set of risk scenarios, each of which describes system behavior that can prevent requirements from being met. Section V presents a draft methodology for determining the appropriate probabilistic equation for each risk scenario (i.e., continuous operation, cyclic operation, block redundancy, etc.), the critical first step towards quantifying the likelihood of a particular risk scenario occurring. The second step, selecting values for parameters in the equations and carrying out the quantitative calculation, is left as an item of future work.

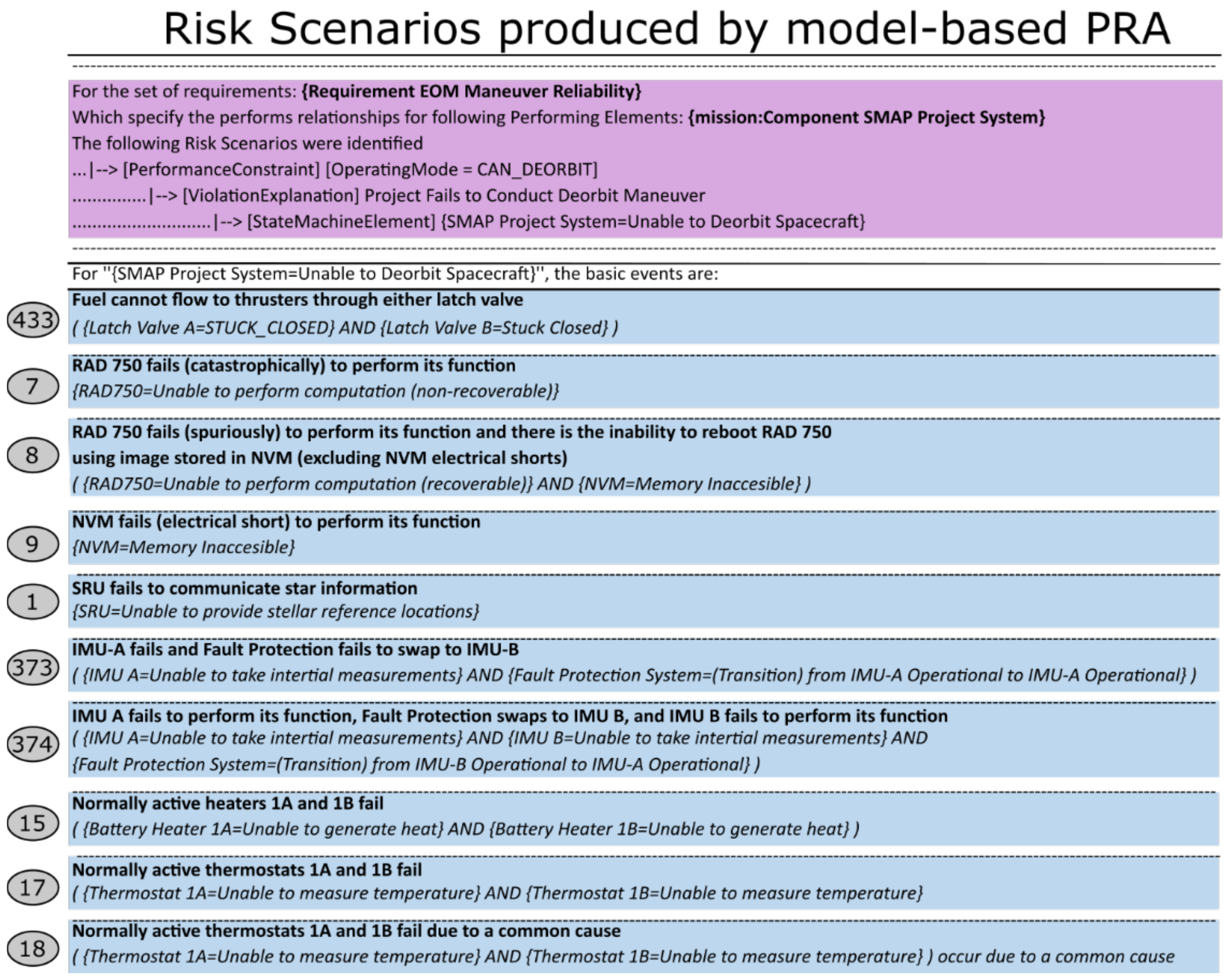
The information required for this methodology is relatively easy to embed in the system model and does not require a dramatic departure from existing modeling patterns and ontologies. The task of entering information in the system model could even be further streamlined with software that ingests spreadsheets and automatically creates the appropriate Cause and Violation Explanations. Because the information needed for the PRA is encoded in a consistent manner in a system model, the model-based PRA can be regularly executed as the design, behavior, and operational usage of the system evolves. This allows a project to track the extent to which system modification impacts compliance with requirements. A review of JPL's PRA planning guidelines indicates that 50% to 90% of PRA costs relate to model development (i.e., identifying risk scenarios). The model-based PRA methodology automates a portion of this effort in an attempt to reduce cost. These aspects of model-based PRA make it capable of managing risk in increasingly complex technical systems. Additionally, because the fault information in the model is encoded according to standard ontologies, it can also be leveraged to ensure consistency not only across different risk analyses (i.e., Fault Tree Analysis (FTA), Failure Modes, Effects, and Criticality Analysis (FMECA), and Single Event Effect Analysis (SEEA)), but also across various other engineering analyses. This integration of engineering analyses with a single-source-of-truth data model will prove invaluable as the complexity of technical systems continues to rise.

[*] *Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.*



Risk Scenarios produced manually

#	Risk Scenarios
1	SRU fails to perform its function
...	...
7	RAD 750 fails (catastrophically) to perform its function
8	RAD 750 fails (spuriously) to perform its function AND Inability to reboot RAD 750 using image stored in NVM (excluding NVM electrical shorts)
9	NVM fails (electrical short) to perform its function
...	...
15	Normally active heater 1A fails AND Normally active heater 1B fails
17	Normally active thermostat 1A fails independently AND Normally active thermostat 1B fails independently
18	Normally active thermostats 1A and 1B fail due to a common cause
...	...
373	MIMU A fails to perform its function AND Fault Protection fails to swap to MIMU B
374	MIMU A fails to perform its function AND Fault Protection swaps to MIMU B AND MIMU B fails to perform its function
...	...
433	Latch Valve A independently fails to open on demand AND Latch Valve B independently fails to open on demand



Risk Scenarios produced by model-based PRA

For the set of requirements: **{Requirement EOM Maneuver Reliability}**
Which specify the performs relationships for following Performing Elements: **{mission:Component SMAP Project System}**
The following Risk Scenarios were identified
...|--> [PerformanceConstraint] [OperatingMode = CAN_DEORBIT]
...............|--> [ViolationExplanation] Project Fails to Conduct Deorbit Maneuver
............................|--> [StateMachineElement] {SMAP Project System=Unable to Deorbit Spacecraft}

For "{SMAP Project System=Unable to Deorbit Spacecraft}", the basic events are:

433	**Fuel cannot flow to thrusters through either latch valve** *({Latch Valve A=STUCK_CLOSED} AND {Latch Valve B=Stuck Closed})*
7	**RAD 750 fails (catastrophically) to perform its function** *{RAD750=Unable to perform computation (non-recoverable)}*
8	**RAD 750 fails (spuriously) to perform its function and there is the inability to reboot RAD 750 using image stored in NVM (excluding NVM electrical shorts)** *({RAD750=Unable to perform computation (recoverable)} AND {NVM=Memory Inaccesible})*
9	**NVM fails (electrical short) to perform its function** *{NVM=Memory Inaccesible}*
1	**SRU fails to communicate star information** *{SRU=Unable to provide stellar reference locations}*
373	**IMU-A fails and Fault Protection fails to swap to IMU-B** *({IMU A=Unable to take intertial measurements} AND {Fault Protection System=(Transition) from IMU-A Operational to IMU-A Operational})*
374	**IMU A fails to perform its function, Fault Protection swaps to IMU B, and IMU B fails to perform its function** *({IMU A=Unable to take intertial measurements} AND {IMU B=Unable to take intertial measurements} AND {Fault Protection System=(Transition) from IMU-B Operational to IMU-A Operational})*
15	**Normally active heaters 1A and 1B fail** *({Battery Heater 1A=Unable to generate heat} AND {Battery Heater 1B=Unable to generate heat})*
17	**Normally active thermostats 1A and 1B fail** *({Thermostat 1A=Unable to measure temperature} AND {Thermostat 1B=Unable to measure temperature}*
18	**Normally active thermostats 1A and 1B fail due to a common cause** *({Thermostat 1A=Unable to measure temperature} AND {Thermostat 1B=Unable to measure temperature}) occur due to a common cause*

Figure 9. The results from the model-based PRA (bottom) compared to the results from a manual PRA (top).

Appendix A: Normally Operating, Cold Spare Configurations

A normally operating, cold spare configuration is exhibited in Figure A-1. Successful operation will result if:

- side A operates over the entire interval; or
- side A fails but side B awakens and operates over the remainder of the interval.

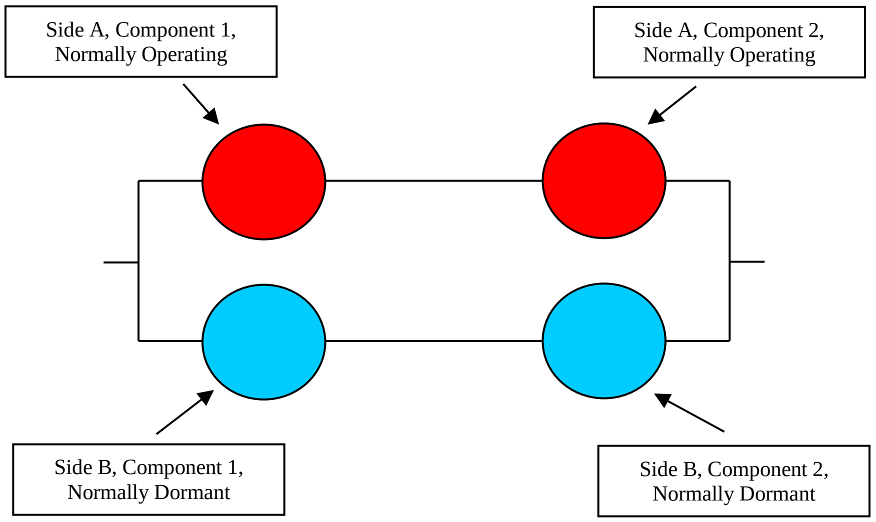

Side A, Component 1, Normally Operating
Side A, Component 2, Normally Operating
Side B, Component 1, Normally Dormant
Side B, Component 2, Normally Dormant

Figure A-1. Normally Operating, Cold Spare Configuration

The interval of operation is $[0, t)$ and it is assumed that side A is operational at the beginning of the interval (i.e., time, zero). Mathematically, this results in the following reliability equation:

$$R(t) = \lambda\int_t^\infty e^{-\lambda x}\, dx + (1-p)\lambda\int_0^t e^{-\lambda x} e^{-\lambda(t-x)}\, dx = e^{-\lambda t}\left[1+\lambda t\,(1-p)\right] \qquad \text{(A-1)}$$

The probability that side A operates over [0,t) is:

$$\lambda\int_t^\infty e^{-\lambda x}\, dx = e^{-\lambda t} \qquad \text{(A-2)}$$

since successful operation over [0,t) necessitates that side fails after time, t. The probability that side A fails in some infinitesimal interval, dx, about time, x, is $\lambda e^{-\lambda x} dx$. Given that side A fails at time, x, the probability the B side is awakened and operates over the remainder of the interval is $(1-p)e^{-\lambda(t-x)}$.

The second integral in Eq. A-1 is simply the aggregate probability that side A fails but side B awakens and operates over the remainder of the interval.

Appendix B: Multiple Operating Cycles

The probability a hardware item starts on demand, operates for a period of time, and then shuts-down on demand is:

$$R(t) = (1-p)(1-q)\,e^{-\lambda t} \qquad \text{(B-1)}$$

The probabilities the hardware item fails to start or stop on demand are independent of both the operating time and previous number of duty cycles, λ is the time-independent failure rate, and t symbolizes the operating time. Since the values of p, q, and λ are independent of time and the number of duty cycles, the probability the hardware item successfully completes N duty cycles, but may not be expected to shut-down after starting the last cycle:

$$R(t) = (1-p)^M (1-q)^N \exp\left(-\lambda \sum_{n=1}^{N} t_n\right) \tag{B-2}$$

Here t_n is the operating time for the nth cycle, and M equals N if the last cycle terminates with hardware shutdown. If the hardware item is not expected to shut down after starting the last cycle, M is one less than

Acknowledgments

The work described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration, as well as funding from JPL burden funds. The authors would like to thank R. Rasmussen, J. Day, S. Jenkins, D. Wagner, and T. Paulos for helping review this methodology.

References

[1]Stamatelatos, Michael, Homayoon Dezfuli, George Apostolakis, Chester Everline, Sergio Guarro, Donovan Mathias, Ali Mosleh et al. "Probabilistic risk assessment procedures guide for NASA managers and practitioners." NASA Scientific and Technical Program. NASA/SP-2011-3421. December 2011.

[2]Branscomb, J. M., Paredis, C. J., Che, J., & Jennings, M. J. "Supporting multidisciplinary vehicle analysis using a vehicle reference architecture model in SysML." *Procedia Computer Science*, Vol. 16, 2013, pp. 79-88.

[3]Bayer, T. J., Bennett, M., Delp, C. L., Dvorak, D., Jenkins, J. S., & Mandutianu, S. (2011, March). "Update-concept of operations for Integrated Model-Centric Engineering at JPL." *IEEE Aerospace Conference,* 2011 (pp. 1-15).

[4]Ingham, D., Donahue, K., Kennedy, K. and Post, S., 2012, May. "A Model-Based Approach to Engineering Behavior of Complex Aerospace Systems." *Infotech @ Aerospace, AIAA SciTech*, Santa Ana, CA, 2012.

[5]Day, J., Donahue, K., Ingham, M., Kadesch, A., Kennedy, A., and Post, E., "Modeling Off-Nominal Behavior in SysML", *Proceedings of the AIAA Infotech@Aerospace 2012 Conference,* Garden Grove, CA, 2012.

[6]Castet, JF et al, "Fault Management Ontology and Modeling Patterns". *AIAA Space 2016*. Long Beach, CA, 2016.

[7]Castet, J. F., Rozek, M. L., Ingham, M. D., Rouquette, N. F., Chung, S. H., Jenkins, J. S., et al (2015). "Ontology and Modeling Patterns for State-Based Behavior Representation," *Infotech @ Aerospace, AIAA SciTech*, Kissimmee, Florida, 2015. http://dx.doi.org/10.2514/6.2015-1115

[8]Donahue, K., and Chung, S. H., "Timeline and the Timeline eXchange Infrastructure: A framework for exchanging temporal information," *IEEE Aerospace Conference 2013*, Big Sky, MT, 2-9 March 2013, doi: 10.1109/AERO.2013.6496945.

[9]Jenkins, J. S., and Rouquette, N. F., "Semantically-Rigorous Systems Engineering Modeling using SysML and OWL," *5th International Workshop on Systems & Concurrent Engineering for Space Applications,* Lisbon, Portugal, October 17–19, 2012.

[10]International Organization for Standardization. "ISO 80000-1:2009: Quantities and units -- Part 1: General," 2009. URL: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=30669

[11]NASA Technical Standard. "Planning, Developing and Managing an Effective Reliability and Maintainability (R&M) Program," NASA-STD-8729.1 December, 1998. https://standards.nasa.gov/standard/nasa/nasa-std-87291

[12]Friedenthal, S., Moore, A., and Steiner, R., *A Practical Guide to SysML: The Systems Modeling Language,* Morgan Kaufmann Publishers / OMG Press, 2008.

[13]No Magic, Inc., MagicDraw User Manual, Version 17.0.1, 2011, URL: http://www.nomagic.com/files/manuals/MagicDraw%20UserManual.pdf