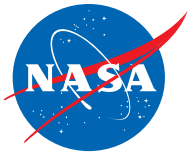
Autonomy Technologies for Systems of a Moon Base

Fernando Figueroa
Autonomous Systems Laboratory
NASA Stennis Space Center

LSIC Autonomy Workshop
August 21-22, 2023

Definitions of Autonomous Systems



Understanding Automated and Autonomous Systems and Thinking Autonomous Systems

Fernando Figueroa and Lauren Underwood

Autonomous Systems Laboratory

NASA Stennis Space Center

- Definition: An ***Automated System*** follows a script, potentially a very sophisticated script (e.g. operational sequences), *in which all allowable courses of action have explicitly been defined and programmed ahead of time*. If the system encounters unanticipated situations, *the outcomes may be unpredictable, or the system may be programmed to stop and wait for human intervention* (e.g. it "phones home"). For automated systems, choices have either already been made, and encoded, or must be externally made.

- Definition: ***Autonomy*** is the ability of a system to achieve goals while operating independently of external control (NASA Technology Roadmaps, TA 4: Robotics and Autonomous Systems, 2015). Autonomy requires self-awareness (to understand its health condition and resources to operate), self-directedness (to achieve goals) and self-sufficiency (to operate independently). An autonomous system applies strategies to mitigate any unforeseen events that may interfere with its current plans to achieve its goals. An autonomous system must encompass knowledge and understanding of the system (itself) that enables analysis, reasoning, and decision-making.

  - The classic approaches and technologies used to achieve autonomy have been denominated ***Brute-Force Autonomy*** (BFA). (BFA) consists of considering all possible cases for autonomous decisions, and then generating solutions offline. In BFA, these cases and solutions are then incorporated in the processor used for implementing autonomy; the processor simply chooses the decisions which correspond to each prescribed case. An interpretation of this methodology suggests that the "thinking" is done offline by experts.

    - BFA has significant shortcomings: (1) cases and solutions are not comprehensive, which results in limited autonomy, which consequently exposes risks for potentials failures; (2) software complexity becomes literally unmanageable for the levels of autonomy required for Gateway and planetary outposts which do not have timely ground control support; and (3) evolve-ability is prohibitive and/or not achievable, due to complexity and cost.

Definitions of Autonomous Systems (cont.)

- The future of "true" autonomous systems requires independent (on-board) thinking in order to eliminate the need for 1- persistent updates and 2- accounting for unforeseen oversight made by humans. A "Thinking System" encompasses the following capabilities: Reason, Intellect, Understanding, Decision, Presentation, Sensory, and Will (actions); and a language founded on a sound ontology. The approach and technologies associated with "Thinking Autonomy" (TA) enable the "thinking" to be done on-board. The code itself transfers to the system the knowledge and "thinking" capabilities of human experts. That is, self-awareness, self-directedness, and self-sufficiency, are not hard-coded off-line; they are exercised on-board, using generic and systemic knowledge of a "true" autonomous system.

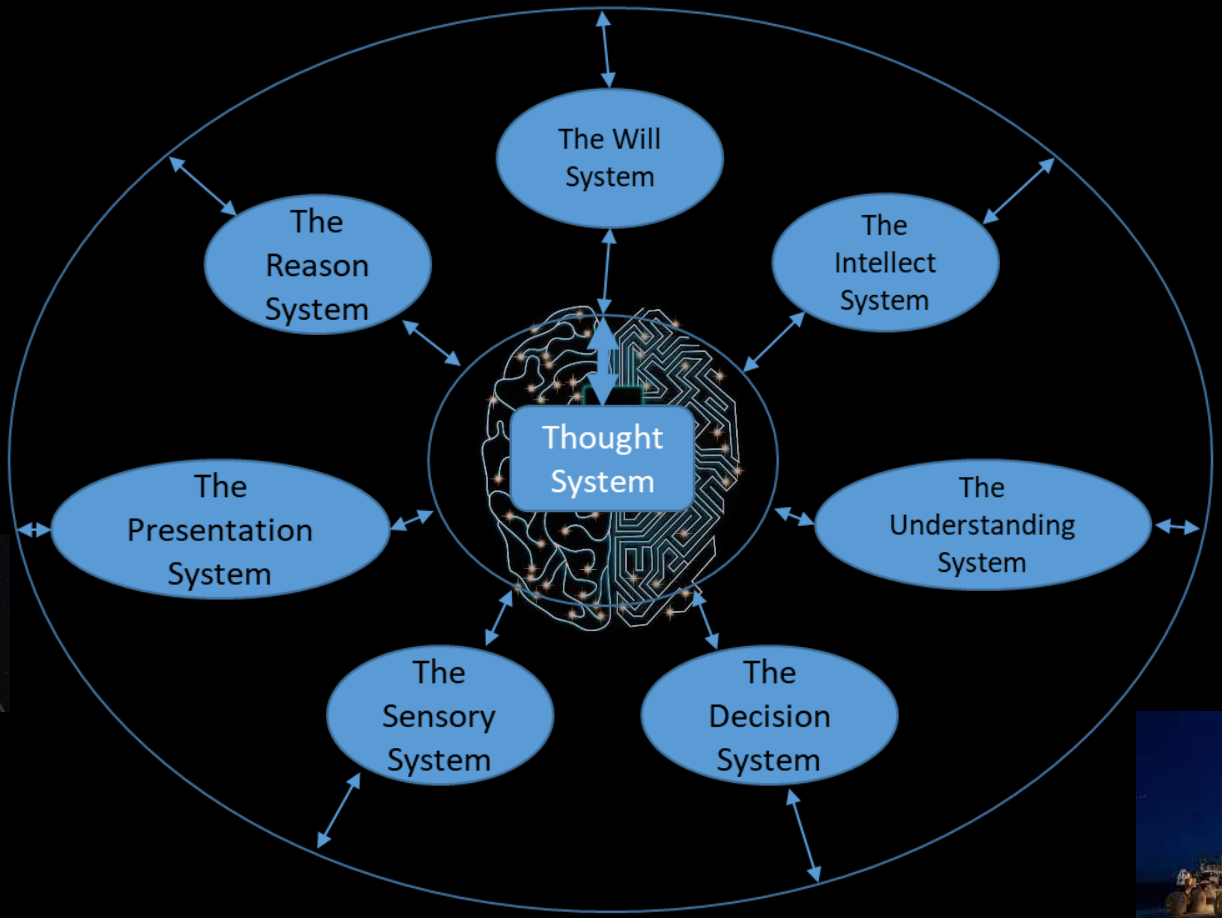
- Definition: ***Hierarchical Distributed Autonomy*** occurs when an overall system is defined as a collection of autonomous systems, where there is a hierarchy that reflects default command authority relationships. For example, the Gateway may be an overall autonomous system that includes modules that are themselves autonomous systems. By default, the Gateway has command authority over its modules, as its function is to execute Gateway activities related to its mission.

- Definition: ***Crewed Autonomy*** occurs when an autonomous system operates in collaboration with crew. This is considered a case of distributed autonomy where crew members are autonomous systems themselves. In this case, the expectation is that crew will have command authority over the autonomous system it occupies, or it will decide which member of the autonomous system has command authority at any given time.

- Definition: **Un-Crewed Autonomy** occurs when an autonomous system functions without crew, in accordance with the definition of ***Autonomy,*** as provided above; and also operates according to the definition of ***Hierarchical Distributed Autonomy,*** as un-crewed.

An autonomous system includes on-board knowledge (about systems topology, functions, behaviors; about strategies to detect and mitigate anomalies; about concepts of operations and policies; about mission planning and execution) and knows how to use that knowledge to achieve its missions.

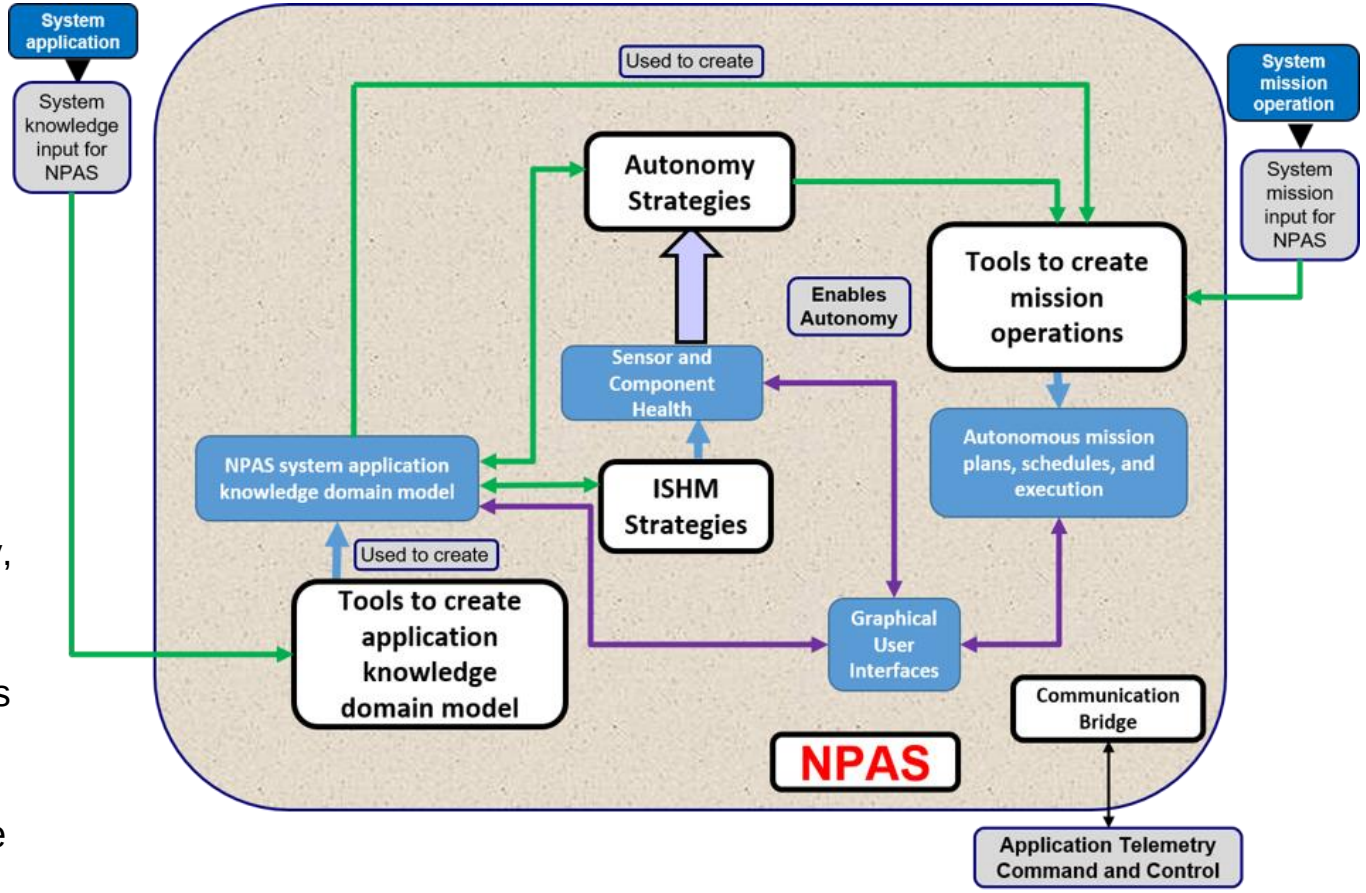
Architecture for Thinking Autonomy



Paradigm and Functional NPAS Architecture

NPAS provides tools for creating autonomous systems:

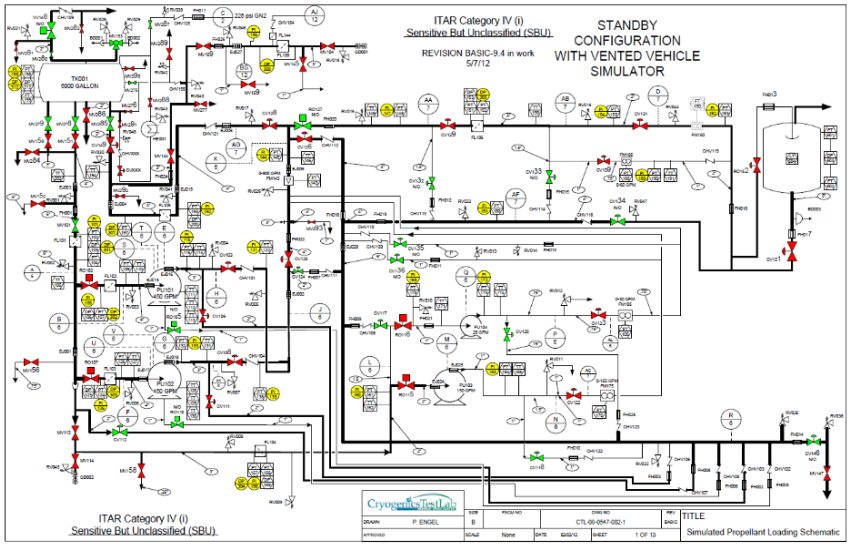
- ISHM framework and strategies for integrated system health management, including diagnosis, prognosis, and FMEA
- Autonomy framework and strategies based on concepts of operation, redundancy, nominal and off-nominal operation
- Domain object libraries –reusable software, thus cost effective
- Infrastructure to create application domain models (digital twins) and mission operations that are scalable



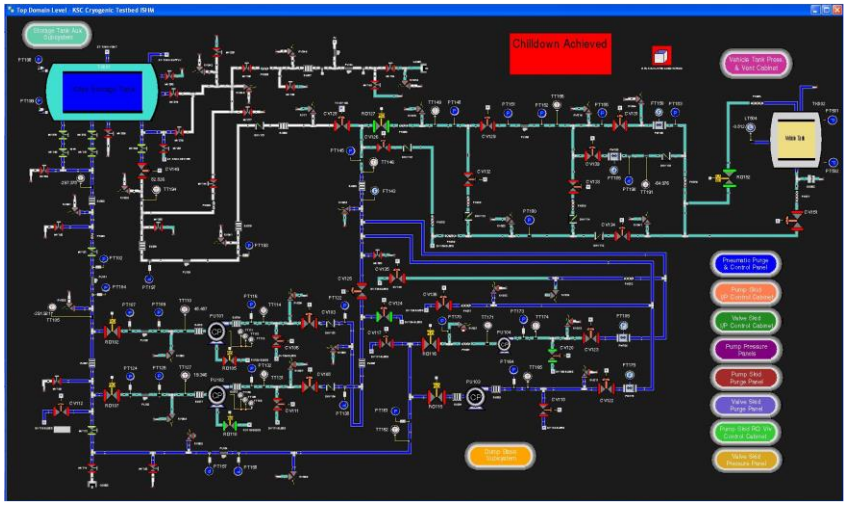
Digital Twins (live): NASA Platform for Autonomous Systems (NPAS)

Stennis Space Center

- NASA Stennis Space Center has developed and continues to evolve the NASA Platform for Autonomous Systems (NPAS) (https://techport.nasa.gov/view/94884)
- NPAS-based autonomous operations utilize a software domain reference model—a ”digital twin” (DT) for mission management as well as for detection, diagnosis/prognosis, and resolution of anomalies
- NPAS supports the creation of such domain specific DTs using built-in and extensible component model libraries. These model libraries are typically designed to match the level of representation included in provided schematics and Interface Control Documents (ICDs)
- One benefit of the digital twin model is the provision of a dynamic system state model that the autonomous software can reference in real-time. This enables the software to make use of both topological and operational context when employing its reasoning logic
- NPAS Digital Twin domain representations are
  - ***ontological*** (e.g., they are composed of model constructs that map well to the ontologies present within the systems, processes, and equipment)
  - ***topological*** (e.g., they include the components, the interconnections, and the associations between the components in an operational context), and
  - ***dynamic*** (e.g., they are populated with live telemetry data and real-time state assessment regarding health and operational availability).
  - The representations are incorporated into the autonomous system manager software architecture seamlessly so that regardless of domain, all applicable measurement data can be properly integrated into a state model.



Schematic of the KSC Cryogenic Testbed Laboratory

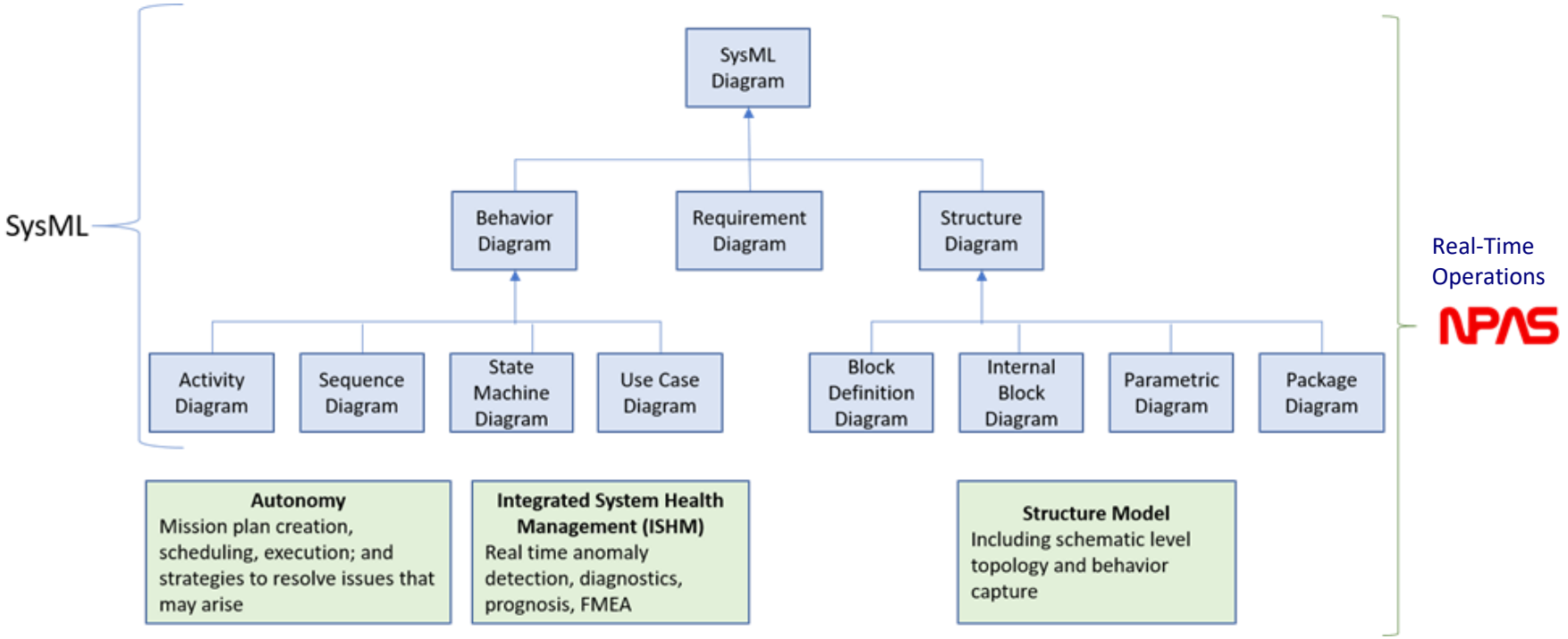


NPAS Digital Twin (live) of KSC Cryogenic Testbed Laboratory

Model-Based Systems Engineering, Real-Time Operations and Autonomy

G2/NPAS Model Development (Life Digital Twin)

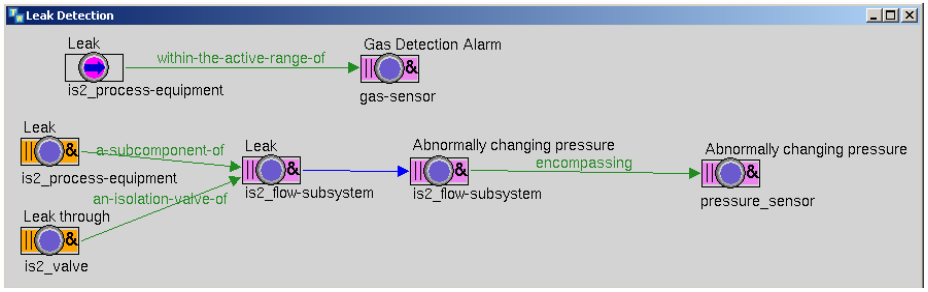
- Includes content associated with all SysML diagrams
- Augments behavior content for ISHM and Autonomy
- Augments detail of structure diagrams by including schematic details
- NPAS uses model in real-time autonomous operations

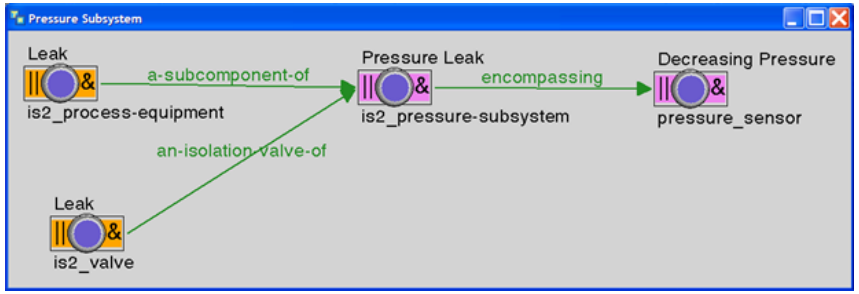


NPAS Failure Modes and Effects Analysis (FMEA)

MIL-STD-1629A (NOTICE 2), Military Standard: Procedures for performing a Failure Mode, Effects, and criticality analysis (28 NOV 1984)

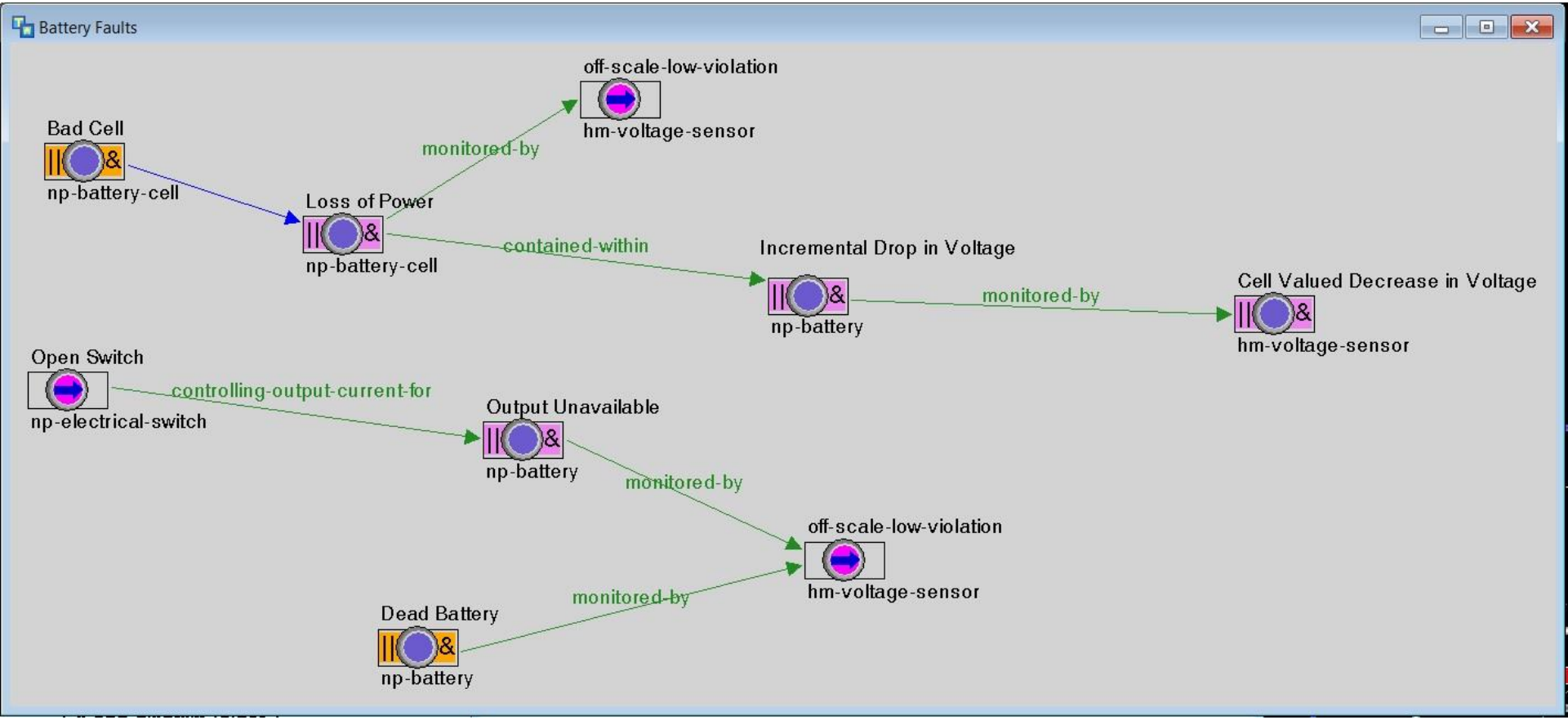
ID #	Item-Functional Identification	Function	Failure Modes and Causes	Mission Phase-Operational Mode	Failure Effects		Failure Detection Method
					Local End Effects	Next Higher Level	
	Process Equipment	Fluid feed subsystem	Leak	Sealed subsystem maintaining pressure		Pressure leak	Decreasing pressure measurement	Identify sealed subsystem, and check pressure sensors for decreasing pressure.





Two rich examples of implementation of leak detection root cause trees and encompasses Mil Standard and how FMEAs are defined. Right: leak event and consequences of leak; left: same leak event used for another diagnosis.
Root cause trees re-usable/generic, events can be used in multiple root cause trees

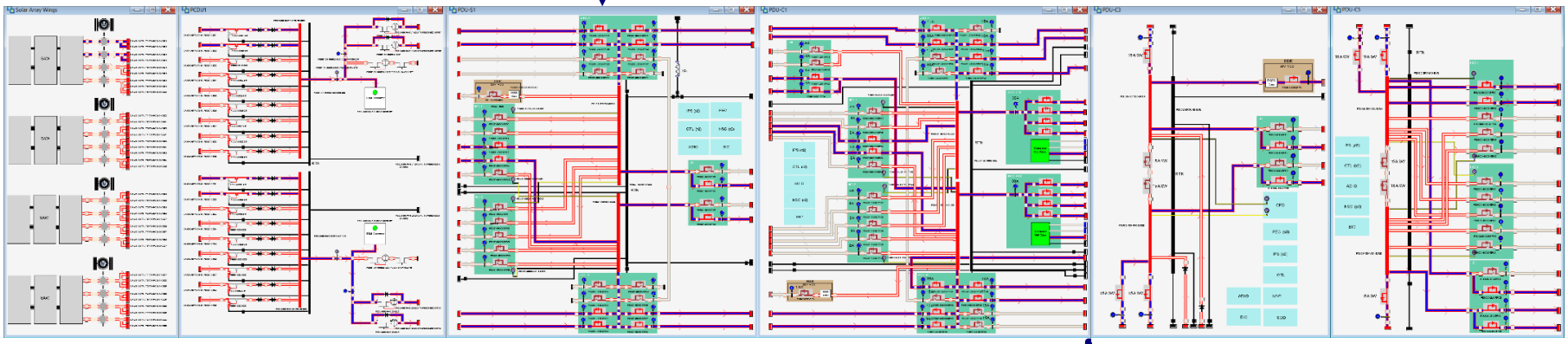
NPAS: Failure Modes and Effects Analysis
FMEA



NPAS Autonomous Power System – Full Domain (Digital Twin)

NPAS Power System Manager controls theh Power System

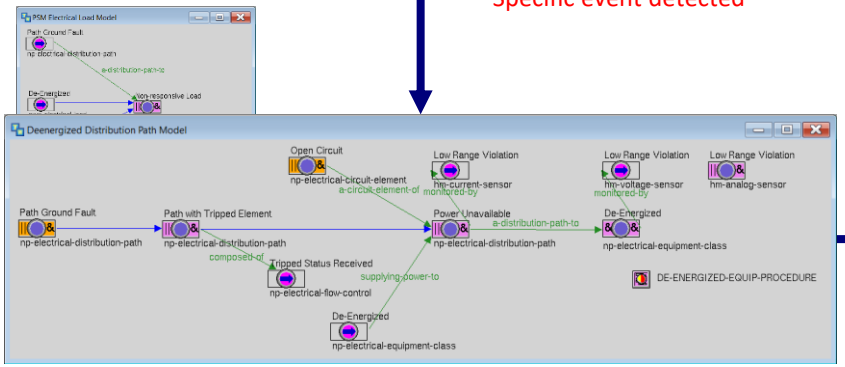
Data and signals from the power system populate the digital twin

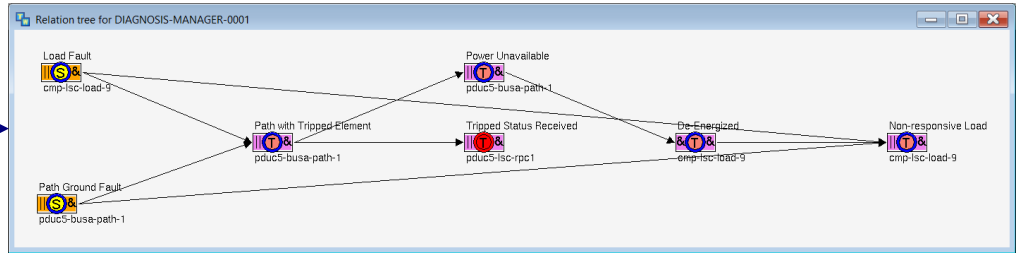


A detailed model (digital twin) of the electrical power system

NPAS Event Detection

Specific event detected

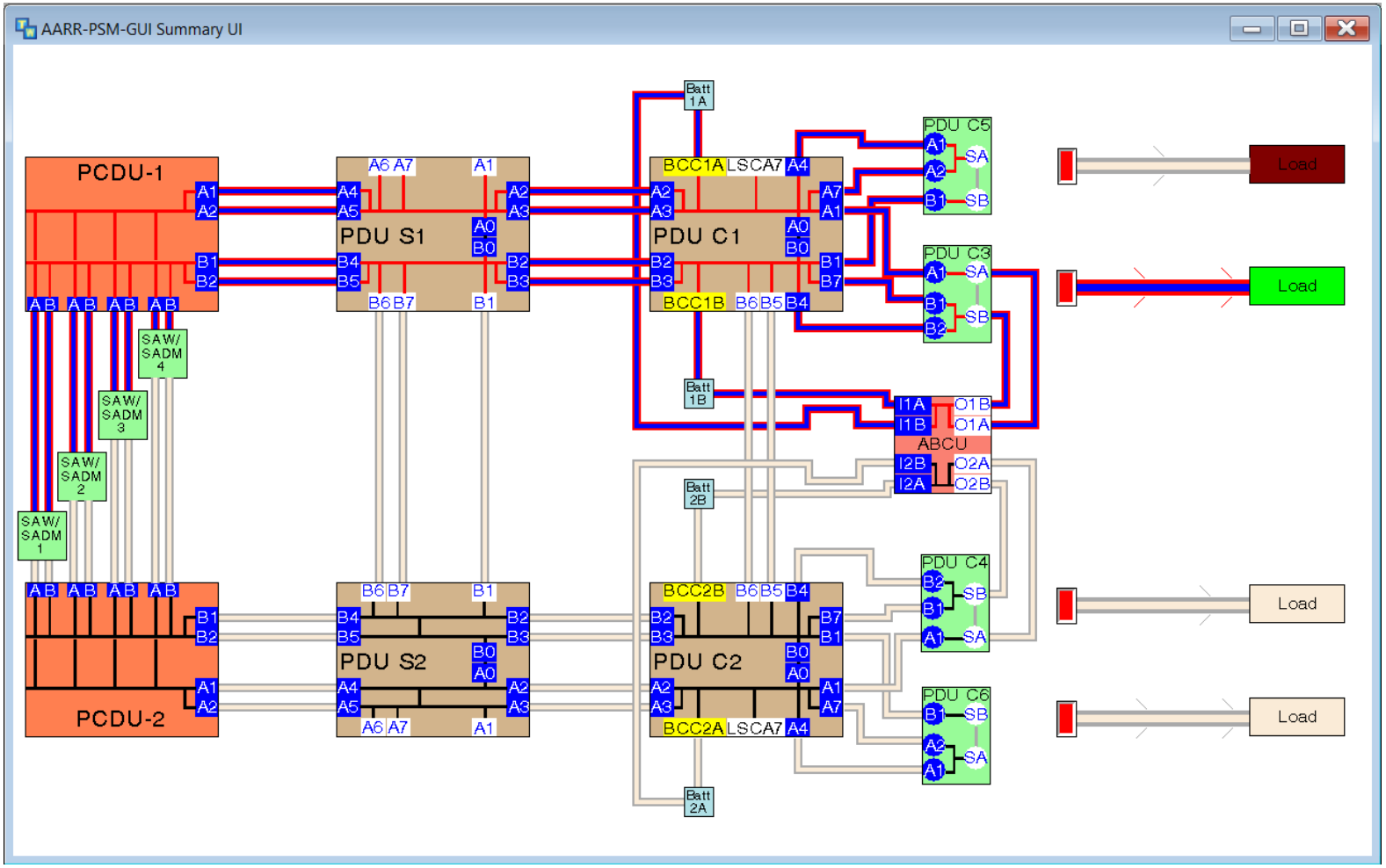




Generic Fault Models for Distribution Paths and Electrical Loads

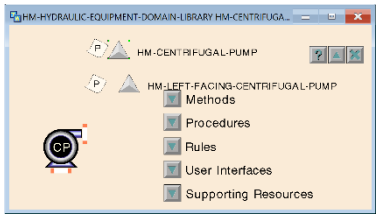
Specific Fault Model with two suspect root causes that could explain the observations in the context of the of the electrical power system's current state and operational phase

NPAS Autonomous Power System – High Level Schematic



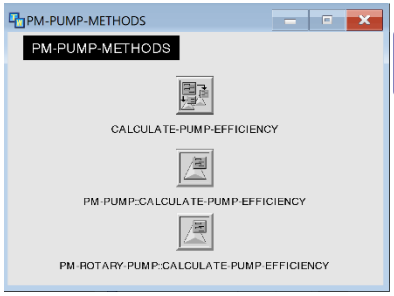
High level schematic of Logistics Module's Power System with representative loads that are Faulted (brown), Powered (green), And Off (beige)

NPAS includes physics based models

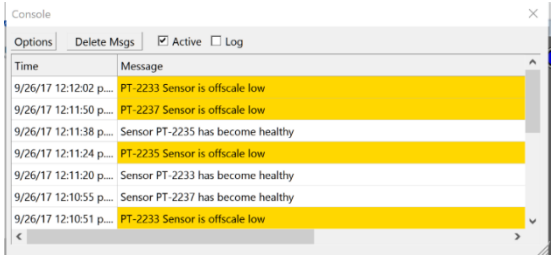


Definition of a class for a rotary pump

For every rotary pump instance

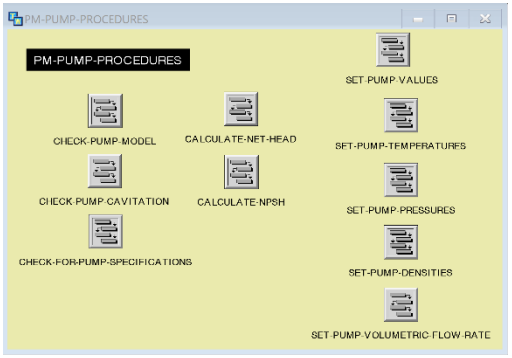


Calculating pump efficiency

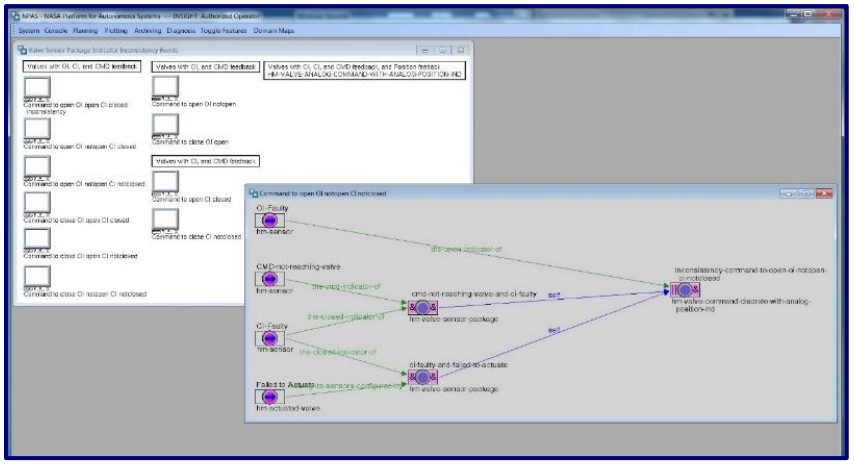


Example, of an inconsistency of a sensor, and this inconsistency is triggered by NPAS models being applied; physics, or other models in NPAS are detecting events; these events are used in root cause trees for diagnosis (as shown in example below)

Checking consistency of models against real-time data



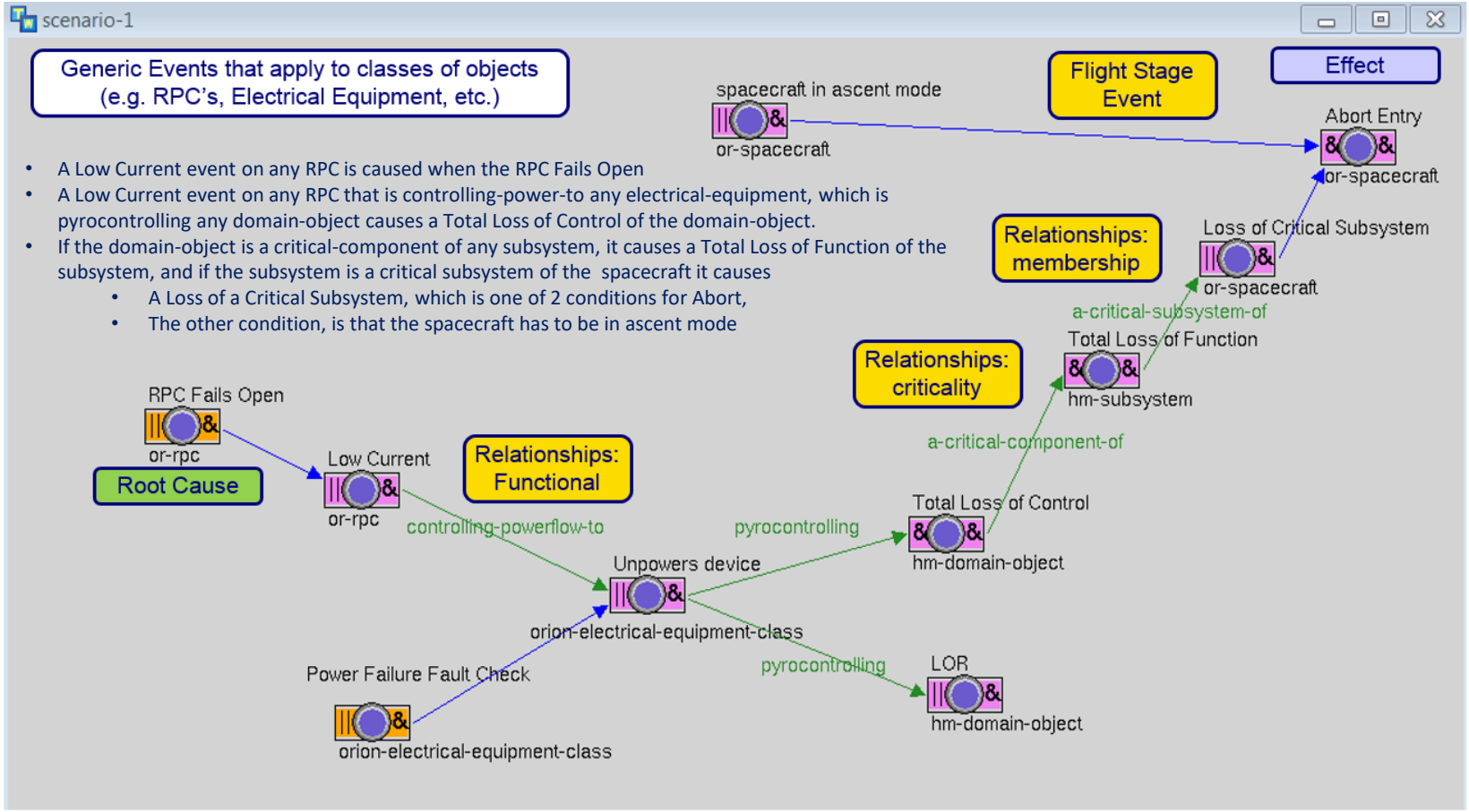
NPAS is applying models comprehensively throughout the system periodically or based on triggers



Root cause tree associated with a valve assessment model

Execution of models (blocks above) generate truth values on events that form part of cause-effect trees (on the right), corresponding to FMEA and/or used for other reasoning as needed by autonomous operations activities

Root Cause Leading to Abort

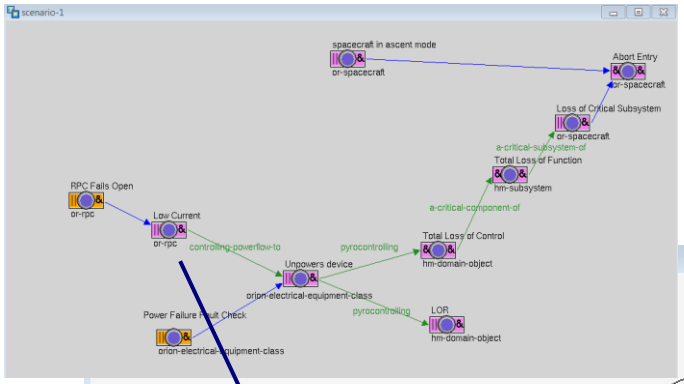


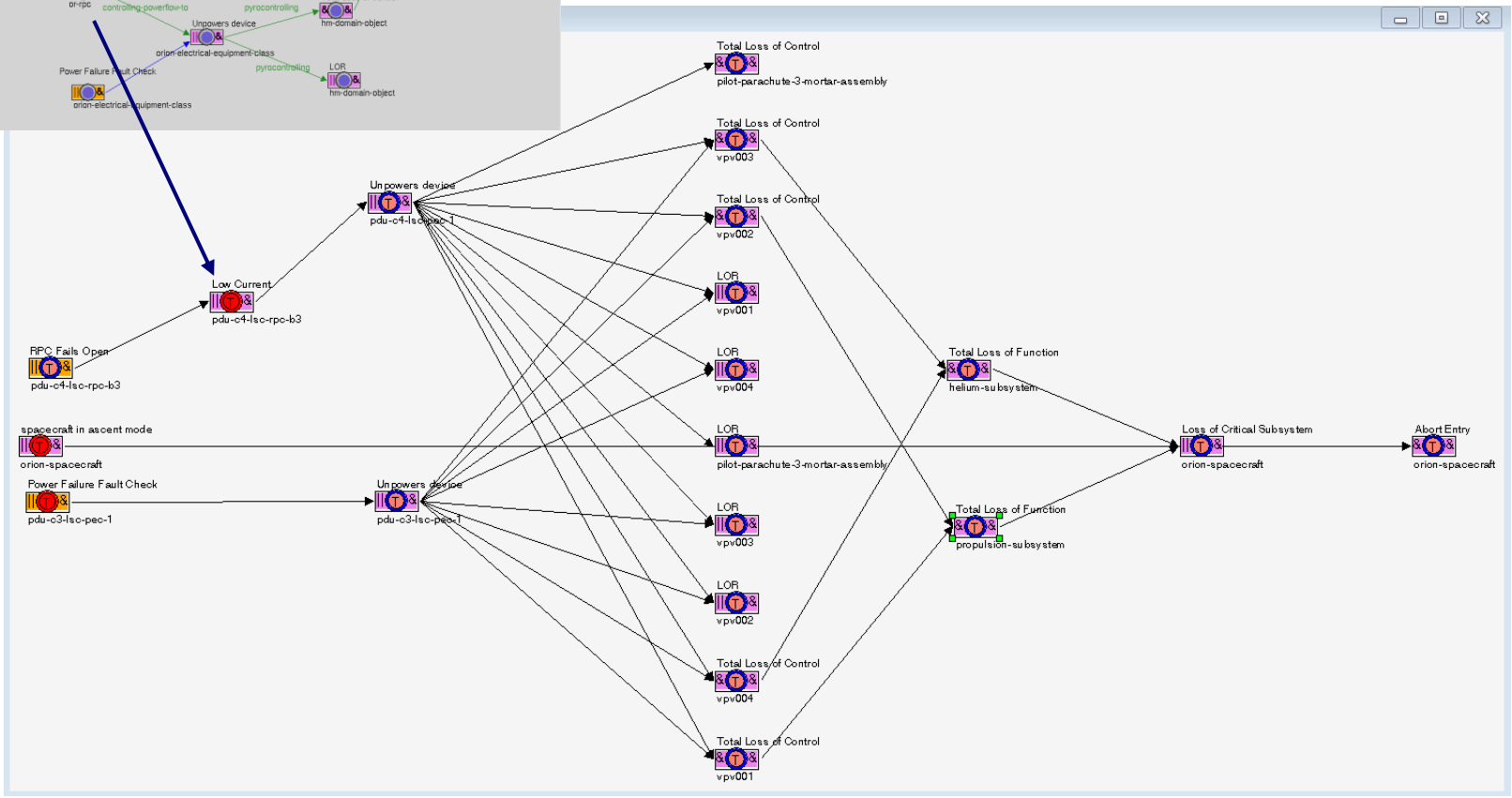
The system can send an event to any instance represented by a node. Causes and consequences will propagate upstream and downstream to assert status of all events (true, false, suspect).

Sample Specific Fault Model automatically generated when a particular RPC fails open

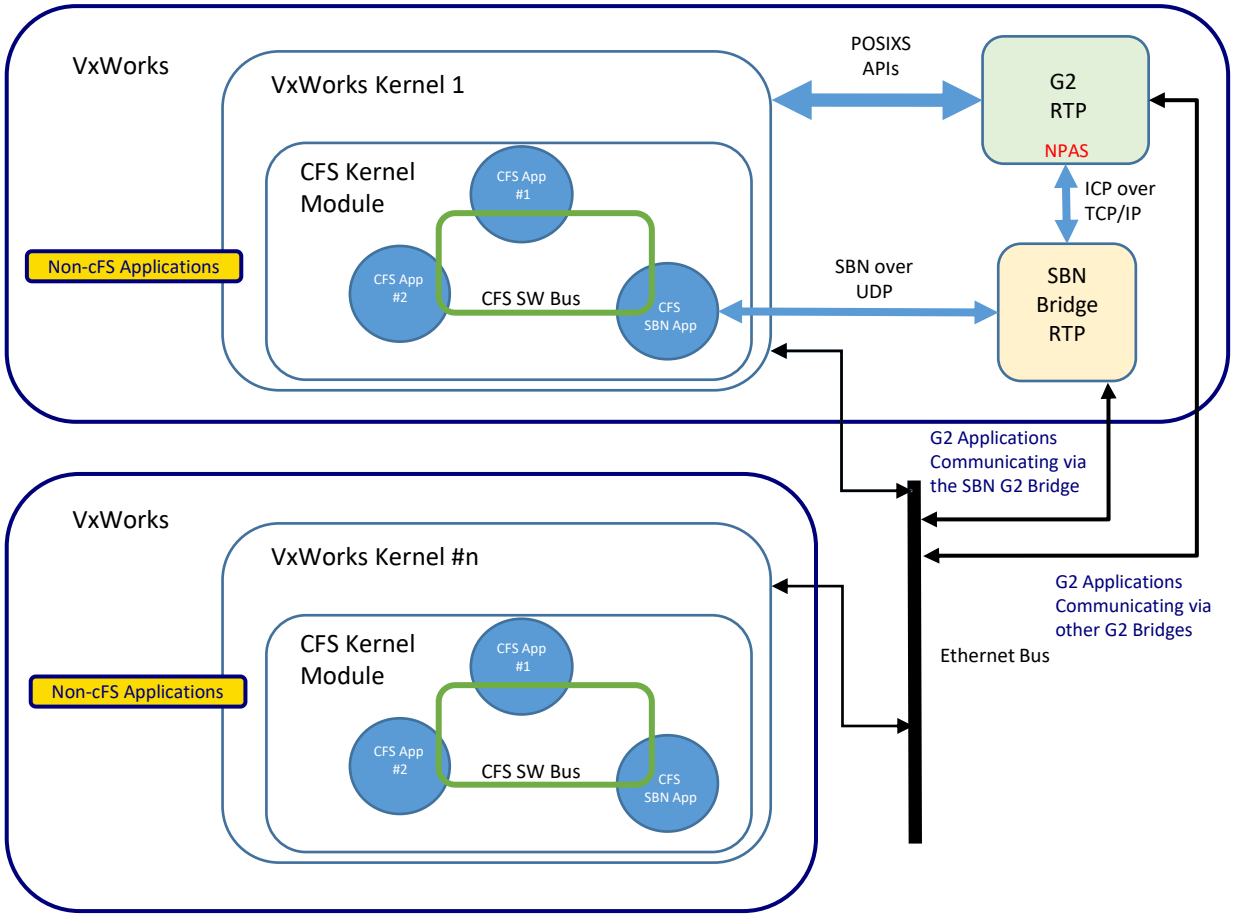
When RPC (PDU-c4-lsc-rpc-b3) experiences a low current event

- **the full blown specific root cause tree is generated**



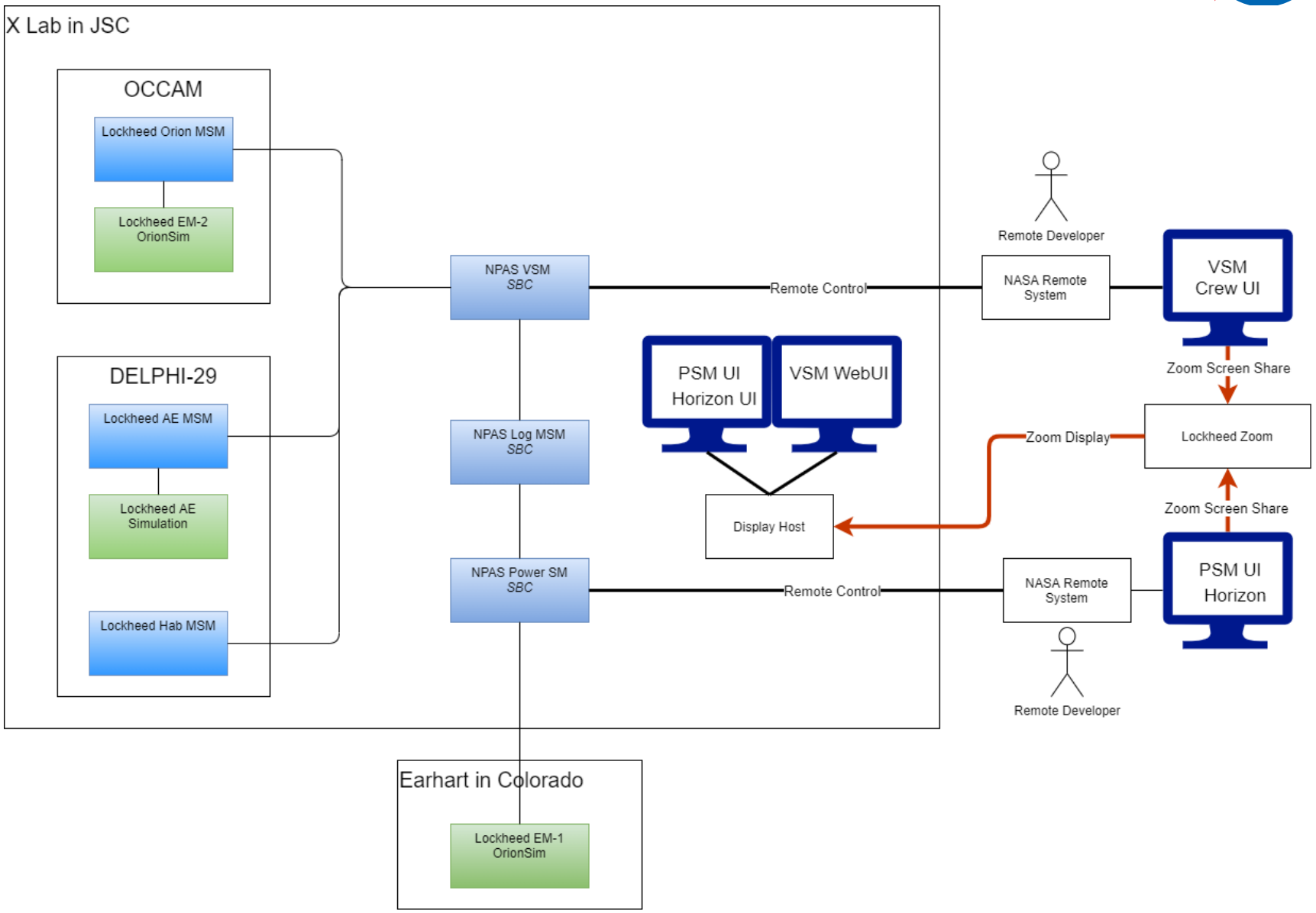


G2 SBN Bridge to Integrate with CFS Apps



- An integration solution for G2 and CFS over the **Software Bus Network (SBN)** application. This solution involves a custom program (SBN Bridge), which **works as a middleware between G2 and CFS applications** running on the same or any other hardware connected via network.
- The SBN Bridge (product), built as a VxWorks RTP application, allows the NPAS applications (G2 RTP), also built as VxWorks RTPs, to **exchange CCSDS messages with CFS system** (Flight Software Architecture consisting of an OS Abstraction Layer (OSAL), Platform Support Package (PSP), cFE Core, cFSLibraries, and cFS Applications), running in VxWorks kernel space.
- CFS SBN app can route messages that are on the CFS Software Bus (SB) to the SBN Bridge via UDP protocol.
- The SBN Bridge (product) can then process, re-format & route the messages to NPAS applications (G2 RTP) via G2 ICP protocol (TCP/IP).
- This setup/configuration **also allows/enables NPAS applications (G2 RTP) to exchange messages with other CFS systems running in a distributed network**, by using CFS SBN app to route messages from other CFS systems on the distributed network to the SBN Bridge (product).

Avionics Architecture



VSM Crew Interface