NASA Platform for Autonomous Systems (NPAS)

Fernando Figueroa,[1] Lauren Underwood[2]
NASA Stennis Space Center, Stennis Space Center, MS 39529, USA

Mark G. Walker,[3] Jonathan Morris[4]
D2K Technologies, Ocean Side, CA 92056, USA

NASA Platform for Autonomous Systems (NPAS) is a disruptive software platform and processes being developed by the NASA Stennis Space Center (SSC) Autonomous Systems Laboratory (ASL). Autonomous operations are critical for the success, safety and crew survival of NASA deep space missions beyond low Earth orbit, including the Gateway, and for the future of cost-effective ground mission operations. NPAS represents the embodiment of an innovative paradigm for "thinking" autonomy in contrast to brute-force autonomy. NPAS uniquely addresses the requirements and integrates the primary functionalities for autonomous operations, in one platform that includes: (1) Integrated System Health Management (ISHM); (2) autonomy strategies, guided by system health and concepts of operations; (3) domain objects (system elements) and infrastructure to create complete application domain knowledge models (4) infrastructure to create, schedule, and execute mission plans; (5) infrastructure to develop user interfaces for comprehensive awareness; and (6) infrastructure to integrate distributed autonomous applications across networks. NPAS is a single platform that can be used to make any system operate with any desirable degree of autonomy, as well as provide comprehensive system awareness to operators and users.

I. Nomenclature

NASA	=	National Aeronautics and Space Administration
SSC	=	Stennis Space Center
JSC	=	Johnson Space Center
NPAS	=	NASA Platform for Autonomous Systems
iPAS	=	integrated Power Avionics Software
ISHM	=	Integrated System Health Management
BFA	=	Brute Force Autonomy
TA	=	Thinking Autonomy
DIaK	=	Data, Information, and Knowledge

II. Introduction

NASA needs autonomous operation software capabilities for safety critical applications that can be affordably and timely qualified and deployed. Autonomy has historically been implemented as "brute-force autonomy (BFA)," as opposed to "thinking autonomy (TA)." BFA consists of considering all possible cases for autonomous decisions and applying those specific strategies to generate solutions offline. Cases and solutions are then incorporated in the processor for implementing autonomy, and 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. This method could never truly be comprehensive, since there will inevitably be cases that are not apparent, imagined and/or just missed, thereby limiting the fundamental degree of autonomy. Therefore, brute-force autonomy implementations

[1] Lead Autonomous Systems and Operations, Test Technology Branch, AIAA Associate Fellow.
[2] Project Manager, Autonomous Systems Lab, Test Technology Branch.
[3] V.P. Engineering.
[4] Senior Software Engineer.

are application instance specific (hard-coded), and hence are by their own nature exorbitantly expensive to develop, deploy, maintain, and evolve for anything beyond the specific application for which the code was designed. For these reasons, BFA has limited capability and reusability – critical requirements for developing affordable autonomous systems - and therefore not a viable solution for the sustained evolution of autonomy. “Thinking Autonomy,” in contrast, implies that analysis and reasoning are evoked by the system in real time in order to derive information, knowledge, and optimal solutions. In the case of NPAS, the processor for autonomy is able to reason based on concepts and first principles, and applies these principles in real-time to models that support strategies for Integrated System Health Management (ISHM) and autonomous operations.

The future of “true” autonomous systems requires independent thinking and reasoning so that the need for persistent updates is eliminated, unforeseen human oversight is prevented, passive monitoring of the progress of a task when desirable is enabled, and awareness for rapid comprehension and action by operators is made possible. This represents a paradigm shift in the way NASA must develop autonomous operations software for future space and ground systems. Stennis Space Center (SSC) has developed a software platform that enables implementation of “Thinking” Autonomy - NPAS.

This paper describes autonomy capabilities enabled by using NPAS for a broad range of ground and space systems. It also describes NPAS’ software architecture and initial implementations. Also discussed is a path-to-flight for NPAS as part of current efforts in order to support Gateway and other NASA activities in the near and mid-term, and to achieve long term exploration objectives.

III. NPAS Description

NASA Platform for Autonomous Systems (NPAS), is a software platform developed by NASA Stennis Space Center (SSC) and its evolution is continuously being advanced at the Autonomous Systems Laboratory (ASL). NPAS is used to make systems operate with any desirable degree of autonomy and provides comprehensive health and operational awareness to operators and users, with the embedded capability to evolve systematically. NPAS uniquely extends the paradigm of model-based systems engineering (MBSE) [1] beyond static models, into live models for real-time “thinking” autonomous operations that can be rapidly and affordably implemented and systematically advanced

Autonomous operations are critical technologies required for the success, safety and crew survival of NASA deep space missions beyond low Earth orbit, including Gateway, and for the future of cost-effective ground mission operations. NPAS is an innovative software platform coupled with engineering processes that address NASA’s autonomy needs. NPAS uniquely integrates six primary functionalities for autonomous operations including: (1) Integrated System Health Management (ISHM); (2) autonomy strategies, guided by system health and concepts of operations; (3) domain objects (system elements) and infrastructure to create complete application domain knowledge models (4) infrastructure to create, schedule, and execute mission plans; (5) infrastructure to develop user interfaces for comprehensive awareness; and (6) infrastructure to integrate distributed autonomous applications across networks.

IV. NPAS Functional Architecture

The NPAS Functional Architecture is shown in Figure 1. This consists of 4 reusable modules (white background) that are used to create all the elements required to make any system operate with any desirable degree of autonomy: (1) Tools to create application knowledge model, (2) ISHM Strategies, (3) Autonomy Strategies, and (4) Tools to create mission operations. These modules are used to generate the data, knowledge, and information specific to an application that will operate autonomously: (1) NPAS System Application Knowledge Domain Model, (2) Sensor and Component Health, (3) Autonomous Mission Plans, Schedule, and Execution, (4) Graphical User Interfaces.

A. Tools to Create Application Knowledge Model (AKM)

NPAS provides the infrastructure and tools to create the AKM. The AKM is a detailed model of the application that is being modeled for autonomy, and includes complete data, information, and knowledge (DIaK) that describe a system. The AKM encompasses all the elements that may be needed for an application to operate autonomously, beginning with detailed schematics describing topology and functionality. A good analogy for an AKM are systems models currently used in the practice of Model-Based Systems Engineering (MBSE), whereby models are created using the SysML [2] standard and language. Figure 2 describes how an NPAS model encompass SysML elements.

All categories of SysML models are encompassed by NPAS AKM's; in addition, AKM's include elements for ISHM (i.e. anomaly detection, diagnostics, prognosis, FMEA); and elements for autonomy (i.e. strategies to resolve issues that arise in the execution of a mission - plan creation, scheduling, execution). The AKM is a "live" model that is used in the operation of the system. It is the "Truth" model that encapsulates all data, information, and knowledge that describe the system and all aspects of the application's life cycle (design, operation, maintenance). Some attributes of elements of the application (e.g. health condition, or usage information) are updated functionally by the module "ISHM Strategies," and conversely this module uses the AKM to apply strategies that result in determination of "Sensor and Component Health."

The module "Autonomy Strategies" utilizes the AKM to make decisions about what strategies can be applied when unexpected events occur that risk the mission objectives. The module "Tools to create mission operations" utilizes the AKM for creation, scheduling, and execution of plans.

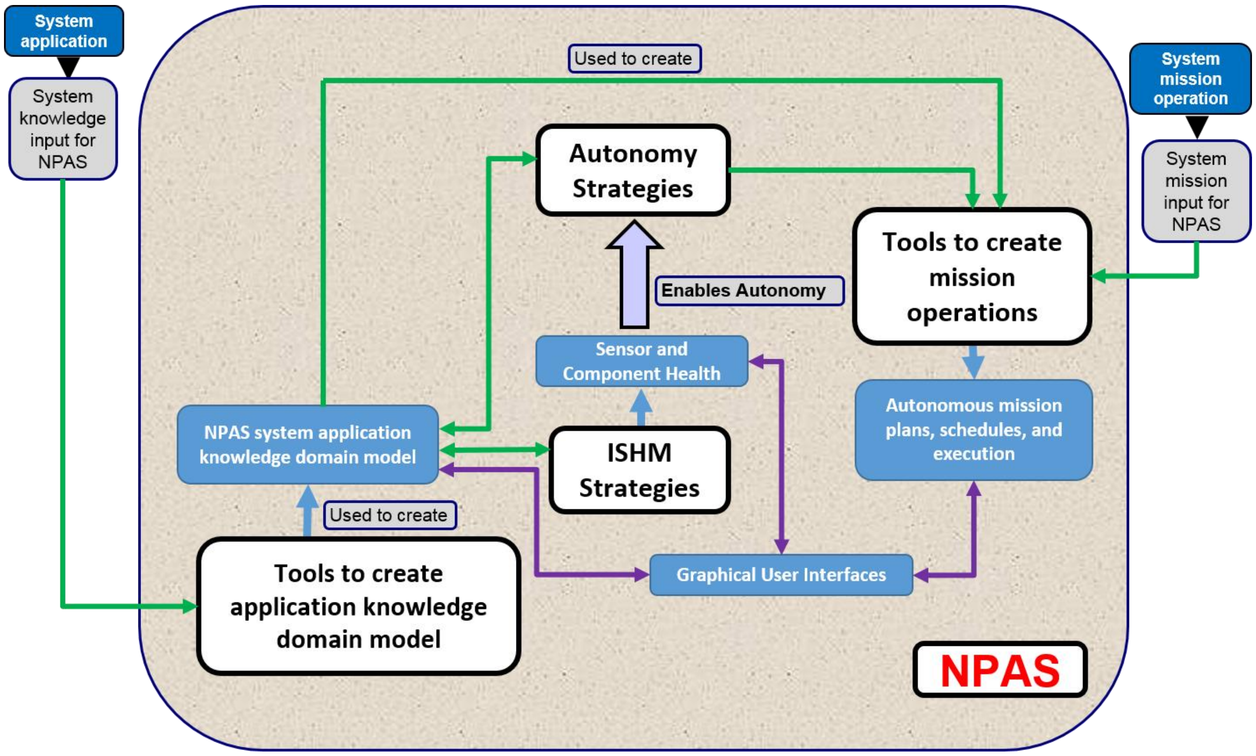

System application
System knowledge input for NPAS
Used to create
System mission operation
System mission input for NPAS
Autonomy Strategies
Tools to create mission operations
Enables Autonomy
Sensor and Component Health
Autonomous mission plans, schedules, and execution
NPAS system application knowledge domain model
ISHM Strategies
Used to create
Graphical User Interfaces
Tools to create application knowledge domain model
NPAS

Figure 1. NPAS Software Architecture

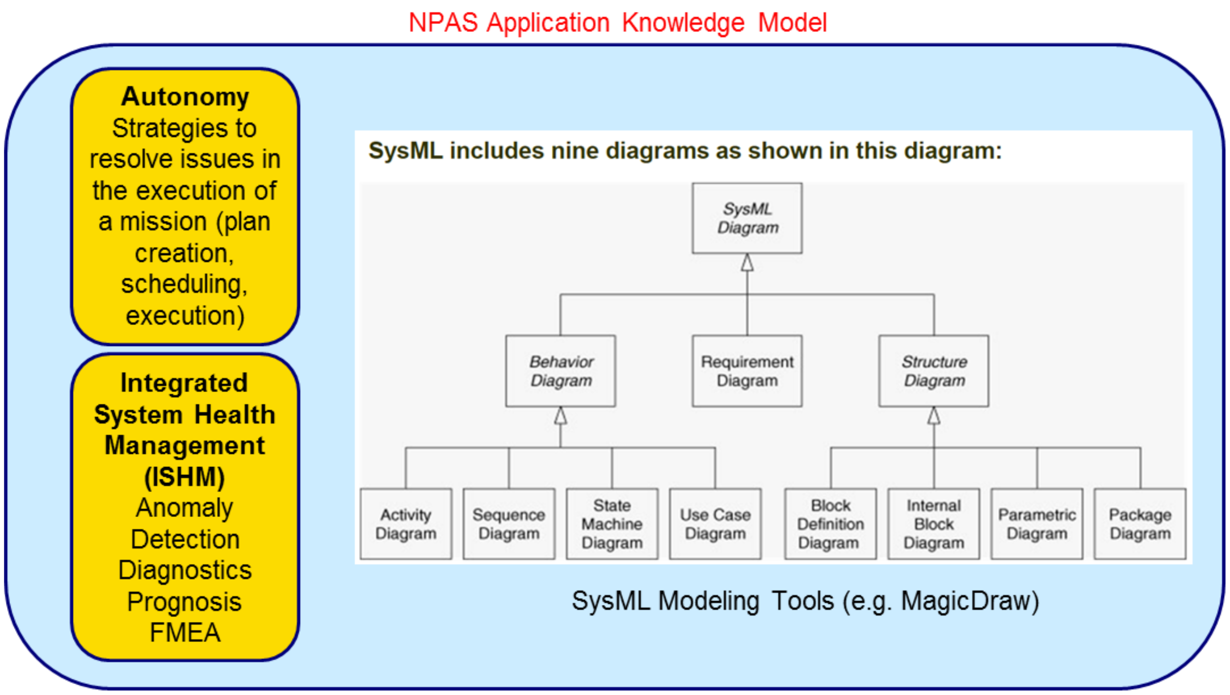

NPAS Application Knowledge Model
Autonomy
Strategies to resolve issues in the execution of a mission (plan creation, scheduling, execution)
Integrated System Health Management (ISHM)
Anomaly Detection
Diagnostics
Prognosis
FMEA
SysML includes nine diagrams as shown in this diagram:
SysML Diagram
Behavior Diagram
Requirement Diagram
Structure Diagram
Activity Diagram
Sequence Diagram
State Machine Diagram
Use Case Diagram
Block Definition Diagram
Internal Block Diagram
Parametric Diagram
Package Diagram
SysML Modeling Tools (e.g. MagicDraw)

Figure 2. NPAS Application Knowledge Model encompasses all elements of SysML models and incorporates elements for Autonomy and ISHM

B. ISHM Strategies

The ISHM module encompasses strategies that are foundational and generic (i.e. reusable). The strategies elicit models that describe how systems function nominally, as well as when anomalies are present. Strategies address anomaly detection, diagnostics, and prognostics. The "ISHM Strategies" module uses the AKM to examine the topology of the application and its configuration in real time; and determines which models should be applied for analysis leading to anomaly detection and/or prediction. At any given time, multiple models analyzing one process from various perspectives may be active. The event occurrences that result from application of the models determine the state (i.e. true, false, or suspect) of nodes on event cause-effect trees. The event cause-effect trees (ECTs) are substantively generic (reusable), since they are created from FMEA analysis of elements and subsystems according to processes established by practice and standards [3]. For example, ECTs embody generic FMEA for valves, or batteries; *while simultaneously* considering that these elements may comprise a commodity distribution flow process, such as fluid flow from a source to a sink. The outcome of ECT resolutions are assessments of sensor *and* component health ("Sensor and Component Health" block in Figure 3). This information, sensor and component health, is in turn used by both, the "ISHM Strategies" and the "Autonomy Strategies" modules.

ISHM is a subject with a substantial body of publications. Generally, anomaly detection is addressed separate from diagnostics or prognostics. An example of a tool to implement anomaly detection using state space models is Livingstone, which was used in NASA's Remote Agent experiment [4]. Livingstone has been extended to address other modeling paradigms resulting in the tool HyDE (Hybrid Diagnostic Engine) [5]. These tools use only partial descriptions of the application system, sufficient to satisfy state transition analysis (Livingstone) or sets of parameters from a perspective of creating simulations from specific physics models (HyDE). In contrast ISHM and autonomy capabilities in NPAS make use of a comprehensive knowledge model of the application that also encompass generic strategies (for analysis, decision making, planning, concepts of operations, policy, functional and topological relationships among members within an application and members of other applications). In addition, all functionalities of ISHM (anomaly detection, diagnostics, prognostics, awareness) are inherently integrated. References of health management implementations include Health and Usage Monitoring Systems – HUMS [6] and the Advanced Health Management System for the Space Shuttle Engine [7]. HUMS is a commercial product, but a significant portion of the analysis and decision making is manual, off line. The Space Shuttle application was active in flights, but it was simple in the sense that it used limits of operation derived from experimental flight data to monitor if parameters were in the acceptable range.

References [8-12] provide additional insight into development and implementations that have evolved into NPAS.

C. Autonomy Strategies

Autonomy strategies are used to address unplanned events that interfere with plans implemented to carry out an activity incorporated in a mission. Note, taxonomy associated with missions and activities will be discussed in section describing the Tools to Create Mission Operations. Autonomy strategies address properties such as self-resilience and self-robustness. Autonomous systems must exploit various attributes of the AKM such as redundancy, and utilize persistence qualities such as "repeat-action"; so that an activity may proceed, or alternatively, be replaced by another activity that enables continuation of the mission.

The reasoning behind a "repeat-action" is the assumption that the first time the action was commanded, it was not successful because a temporary anomaly occurred that may no longer exist. For instance, perhaps a dust particle interfered to impede an electrical contact, and it was dislodged in the first attempt. So, for any command, there is the option of applying the "repeat-action" strategy, however, there are considerations (constraints) that must be met before applying the strategy. For example, some actions may inhibit the use of this strategy, so there is also the concept of inhibiting/de-inhibiting strategies associated with some actions. This "inhibit" assignment is a generic constraint strategy that may be defined by qualifying the actions and corresponding inhibit settings. Planners may inhibit use of the "repeat-action" strategy based on constraint reasoning.

In order to define autonomy strategies, concepts such as redundancy, "repeat-action", inhibit, and others are understood by NPAS, so that the appropriate strategies may be applied to carry out a mission. The ability to extend the glossary of autonomy strategies concepts is also provided.

Autonomy strategies have been generally employed in specific implementations, and most likely in response to a circumstance that was not anticipated. For example, a rover that is stuck with its wheels slipping may repeat a back-and-forth motion in order to swing out from the slippery area. Currently this strategy is planned in detail once the rover is in such situation. NPAS enables a high abstraction level that is based in the "repeat-action" strategy, and applies to a broad range of circumstances that can be resolved with this strategy.

Reference [13] is a detailed report describing the state of autonomous operations capabilities, and it basically indicates that at the time of the report (2002), perhaps only one implementation could qualify as having minimal autonomy, the Remote Agent Experiment [4]. Not much progress in "thinking" autonomy has been made since then. Some of the advances in autonomous systems and operations leading to NPAS are described in references [14-17].

D. Tools to Create Mission Operations

An autonomous system defines missions based on a schedule of activities that are achieved by executing plans. NPAS includes tools to create plans that encompass an activity, schedule activities (encompassing plans), and execute activities. For any application system (e.g. an autonomous power subsystem), sets of activities may be created as a collection of plans. An example activity might be "nominal operation", which in the case of an autonomous power subsystem could refer to a set of plans for autonomous management of power flow to critical spacecraft loads by a spacecraft Vehicle Manager. Accordingly, the power subsystem plans must control the power sources and distribution switches to achieve this mission autonomously, despite any anomalies that the power subsystem may have at that particular point in time. And if the autonomous power subsystem is unable to comply with the "nominal operation" activity commanded by the Vehicle Manager, it must report this, and follow up with an ensuing conversation between the Vehicle Manager and the autonomous power subsystem, and a solution that satisfies the overall mission objective of the Vehicle Manager (i.e. keeping the spacecraft safe) should be provided.

V. NPAS Capabilities

NPAS integrates the following primary functionalities:

1. Integrated System Health Management (ISHM), that includes anomaly detection, diagnostics and effects (FMEA), prognostics, and comprehensive awareness;
2. Autonomy, that encompasses strategies for autonomous behavior constrained by health and system concepts of operations;

3. Infrastructure to rapidly develop and evolve knowledge models of applications.
4. Infrastructure to create, schedule, and execute mission plans.
5. Infrastructure to seamlessly integrate distributed NPAS autonomous applications across networks
6. Infrastructure to develop user interfaces that provide integrated awareness and optimal operator access for a desired degree of autonomy.

NPAS encompasses an infrastructure to implement intelligent autonomy for a broad range of systems (Figure 3). To date, this includes tools to create the AKM's such as domain object libraries and knowledge for hydraulic, mechanical, electrical, and computing systems; process and event management, health management, autonomy, functional relationships, constraints, and concepts of operations. Specific capabilities are described below.

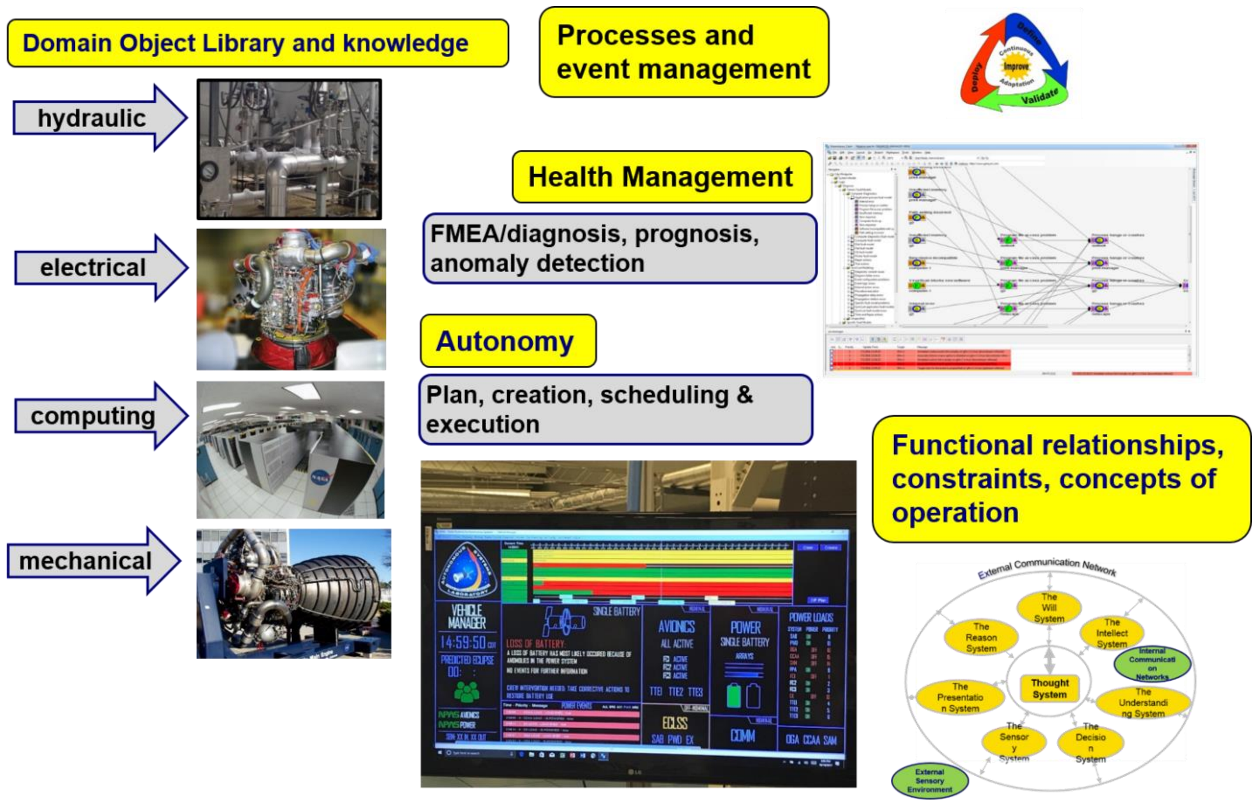

Domain Object Library and knowledge
hydraulic
electrical
computing
mechanical
Processes and event management
Define
Continuous Improve Adaptation
Validate
Deploy
Health Management
FMEA/diagnosis, prognosis, anomaly detection
Autonomy
Plan, creation, scheduling & execution
VEHICLE MANAGER
LOSS OF BATTERY
AVIONICS
POWER
POWER LOADS
ECLSS
COMM
Functional relationships, constraints, concepts of operation
External Communication Network
The Will System
The Reason System
The Intellect System
Internal Communication on Networks
Thought System
The Presentation System
The Understanding System
The Sensory System
The Decision System
External Sensory Environment

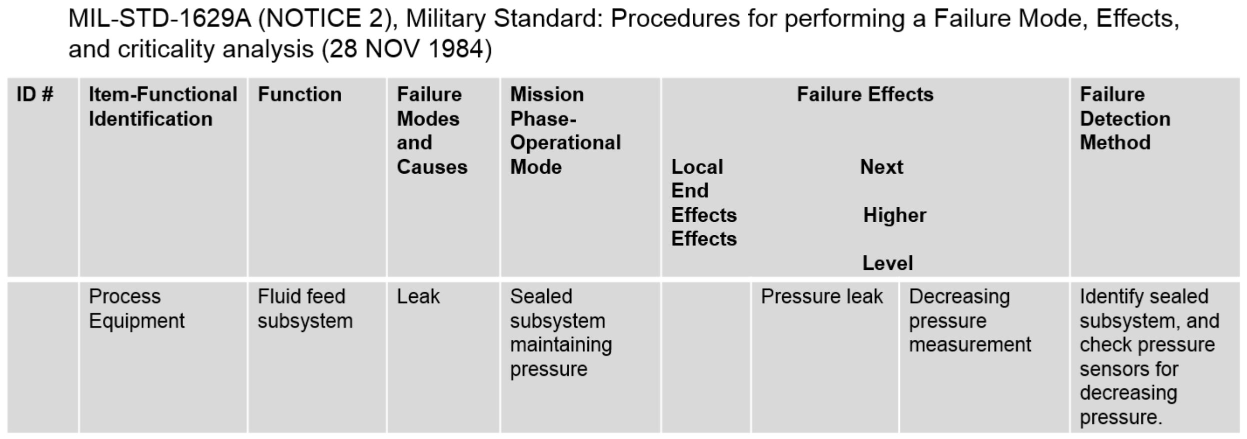
Figure 3. NPAS Capabilities

A. NPAS Failure Modes and Effects Analysis

FMEA is part of the NPAS ISHM Strategies, as it is used to implement diagnostics and model observable effects of failure events. NPAS includes tools for modeling cause-effect trees based on available FMEA, capturing all constraints and details needed for the analysis represented by each FMEA entry.

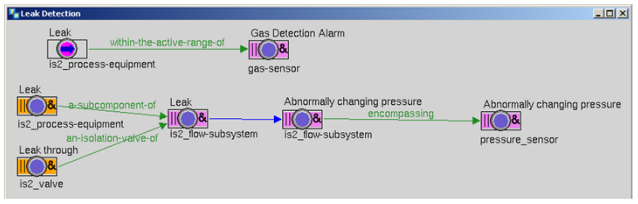
Figure 4 shows how FMEA is implemented in NPAS. The table shown corresponds to MIL-STD-1629A (Notice 2) Military Standard: Procedures: Procedures for performing a Failure Mode, Effects, and criticality Analysis. In NPAS FMEA is implemented graphically as cause-effect trees with nodes representing events (failure or anomaly). Any number of trees can be created, each addressing a particular type of analysis related to anomalies on classes of sensors, components, or subsystems. The cause-effect trees on the left (Figure 4) depict a "Leak" event occurring on any "is2-flow-subsystem" (a "class" of subsystem that is isolated by closed valves, stops, etc., where commodity in the subsystem is not expected to leave). Root Causes include (1) any of the elements encompassing the subsystem leak ("Leak" on any "is2-process-equipment" that is a subcomponent of an "is2-flow-subsystem" where a Leak has occurred), or (2) the valves isolating the subsystem leak ("Leak" through any "is2-valve" that is "an-isolation-valve-of" an "is2-flow-subsystem" where a "Leak" has occurred). Note the relationship assignment to the arrow indicating that the cause-effect relationship involves encompassing member elements of a subsystem (the isolated subsystem).

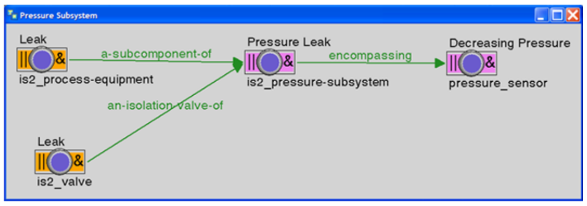
As NPAS navigates and understands the AKM (learns its physical and functional topology), relationships like this (membership in a subsystem) are automatically discovered. An immediate consequence of a “Leak” event is that an “Abnormally Changing Pressure” event occurs in the isolated subsystem, and a consequence of that event is that an “Abnormally Changing Pressure” event occurs on any pressure-sensor that the isolated system encompasses. Also, note that sets of trees may be connected virtually by re-use of an event node from one tree on another. That is the case of the event Leak that occurs on an “is2-flow-subsystem” (isolated subsystem) shown on the top tree of the left of the figure (figure xx). The icon shows a blue arrow that indicates that it is an event that was defined previously in another tree. Using this capability one can link any number of trees to create comprehensive cause-effect analysis for complex systems. Note, that events correspond to classes of objects that encompass the AKM. This enables inherent re-usability within an implementation and in new implementations.



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 Effects	Next Higher Level	End Effects	
	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 tree and encompasses Mil Standard and how FMEAs are defined; right: leak event and consequences of leak; same leak event used for another diagnosis; root cause trees re-usable/generic, events can be used in multiple root cause trees

Figure 4. NPAS cause-effect trees for FMEA implementation.

B. NPAS Creation of Operational Plans for Activities

Systems customarily have operational modes that provide context to activities that must be carried out. For example, an operational mode (OpMod) of a space habitat may be “Un-crewed.” In this mode, the habitat will carry out multiple activities according to a concept of operations (ConOps) for “Un-crewed” space habitats. NPAS has been used to demonstrate a hierarchical distributed architecture for an autonomous systems operation integration activity at the Integrated Power Avionics Software (iPAS) laboratory, located at NASA Johnson Space Center. Figure 5 represents the hierarchy, with the Vehicle Manager at the highest level, denoting the Vehicle Manager’s responsibility for managing the entire system (for example a space habitat).

Hierarchical distributed autonomy architecture means that each subsystem of the habitat will autonomously carry out its own activities, the whole of which encompass an OpMod of the habitat.

An activity may be considered as a collection of plans defined by operational sequences. Plans may be directly executed and/or be triggered by other plans that are running (e.g. an activity of the Vehicle Manager that is running a plan for “Un-crewed” nominal power operations, may trigger a plan for “Un-crewed” off-nominal operations whenever the power system reports a failure and corresponding diminished power availability).

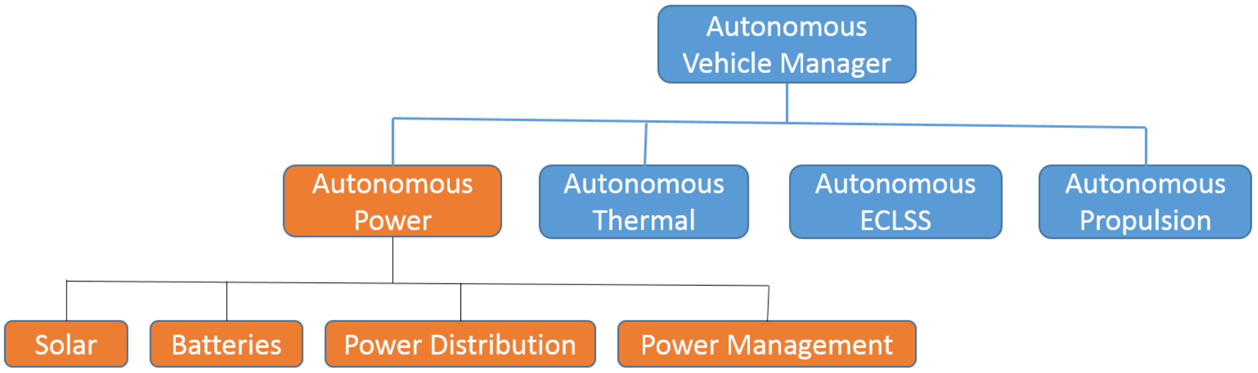
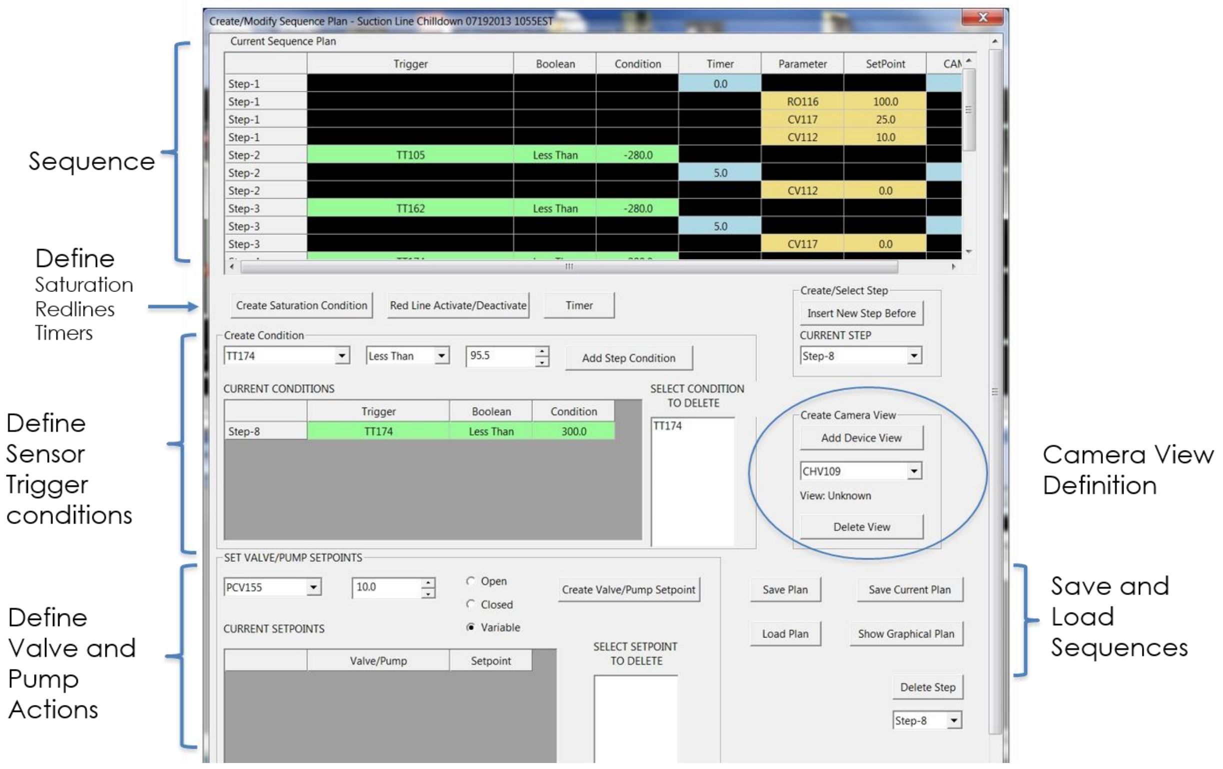


Figure 5. Hierarchical Distributed Autonomy Architecture

NPAS also provides separate user interfaces for creating and running sequences. Figure 6 shows an example of the sequence creation user interface. It has selection menus to create steps that carry out tasks (e.g. opening of a valve), as well as menus for each step for defining conditions (constraints), time delays, and actions (open a valve, start a pump, start another plan or sequence). An important quality of this user interface is that the menus update automatically whenever changes occur in the application knowledge model (AKM). For example, if a new sensor is included, this will appear as an option to define conditions related to measurements by that sensor. This is because the AKM incorporates, in real time, everything that relates to the new sensor in the knowledge model.

NPAS also has a user interface that provides full situational awareness associated with sequence execution. Figure 7 shows the sequence steps that have been executed on the top portion of the window, the step currently being executed in the middle, and the upcoming steps at the bottom.


Sequence
Define Saturation Redlines Timers
Define Sensor Trigger conditions
Define Valve and Pump Actions
Camera View Definition
Save and Load Sequences
Create/Modify Sequence Plan - Suction Line Chilldown 07192013 1055EST
Current Sequence Plan
Create Saturation Condition
Red Line Activate/Deactivate
Timer
Create/Select Step
Insert New Step Before
Add Step Condition
Add Device View
Delete View
Create Valve/Pump Setpoint
Save Plan
Save Current Plan
Load Plan
Show Graphical Plan
Delete Step

	Trigger	Boolean	Condition	Timer	Parameter	SetPoint	CAN
Step-1				0.0			
Step-1					RO116	100.0	
Step-1					CV117	25.0	
Step-1					CV112	10.0	
Step-2	TT105	Less Than	-280.0				
Step-2				5.0			
Step-2					CV112	0.0	
Step-3	TT162	Less Than	-280.0				
Step-3				5.0			
Step-3					CV117	0.0	

Create Condition: TT174 | Less Than | 95.5

CURRENT STEP: Step-8

CURRENT CONDITIONS

	Trigger	Boolean	Condition
Step-8	TT174	Less Than	300.0

SELECT CONDITION TO DELETE: TT174

Create Camera View: CHV109; View: Unknown

SET VALVE/PUMP SETPOINTS: PCV155 | 10.0 | Open [✓ Variable] Closed

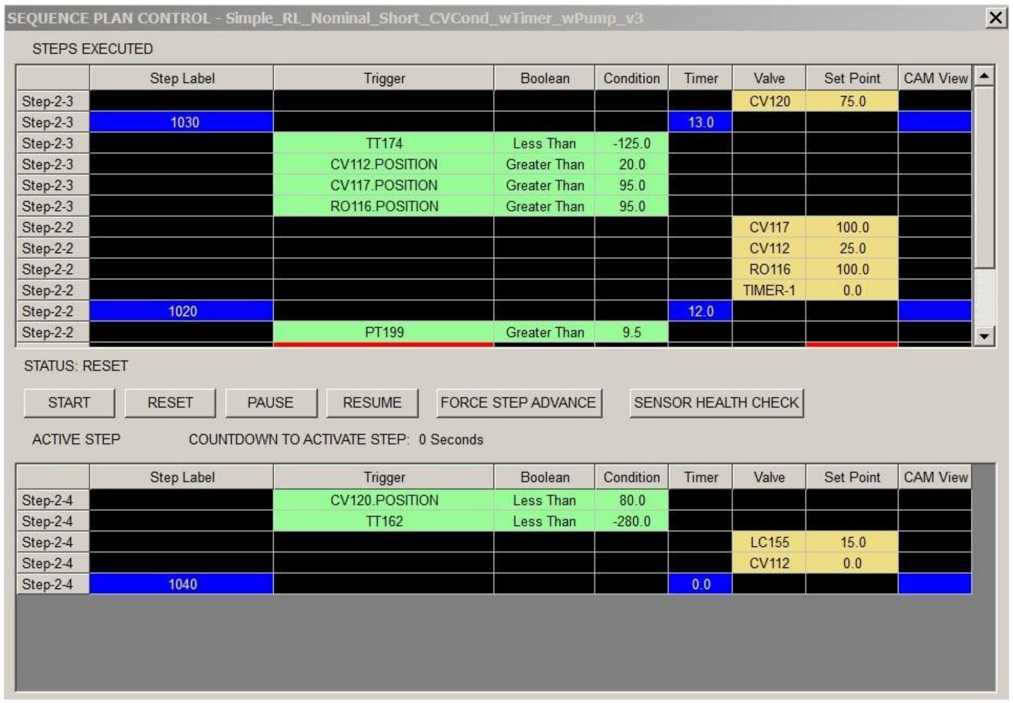
CURRENT SETPOINTS

	Valve/Pump	Setpoint

SELECT SETPOINT TO DELETE

Step-8

Figure 6. NPAS sequence creation user interface.


SEQUENCE PLAN CONTROL - Simple_RL_Nominal_Short_CVCond_wTimer_wPump_v3

STEPS EXECUTED

	Step Label	Trigger	Boolean	Condition	Timer	Valve	Set Point	CAM View
Step-2-3						CV120	75.0	
Step-2-3	1030				13.0			
Step-2-3		TT174	Less Than	-125.0				
Step-2-3		CV112.POSITION	Greater Than	20.0				
Step-2-3		CV117.POSITION	Greater Than	95.0				
Step-2-3		RO116.POSITION	Greater Than	95.0				
Step-2-2						CV117	100.0	
Step-2-2						CV112	25.0	
Step-2-2						RO116	100.0	
Step-2-2						TIMER-1	0.0	
Step-2-2	1020				12.0			
Step-2-2		PT199	Greater Than	9.5				

STATUS: RESET

ACTIVE STEP COUNTDOWN TO ACTIVATE STEP: 0 Seconds

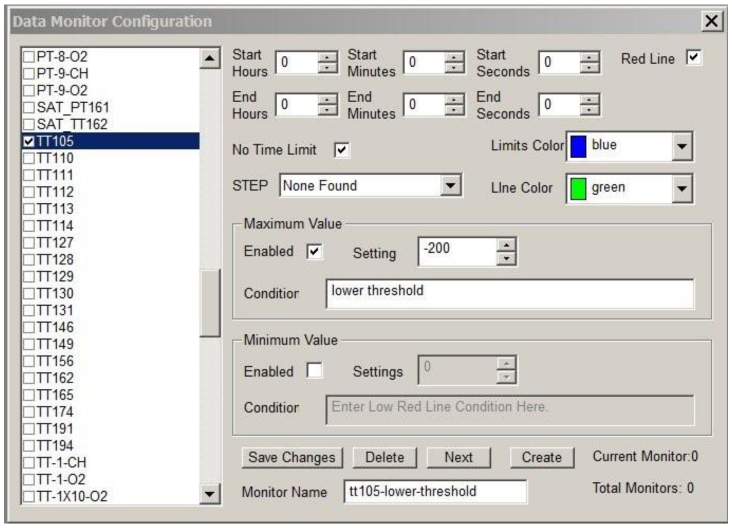
	Step Label	Trigger	Boolean	Condition	Timer	Valve	Set Point	CAM View
Step-2-4		CV120.POSITION	Less Than	80.0				
Step-2-4		TT162	Less Than	-280.0				
Step-2-4						LC155	15.0	
Step-2-4						CV112	0.0	
Step-2-4	1040				0.0			

Figure 7. NPAS sequence execution user interface.

C. NPAS Redline Implementation

Redlines are a standard approach for providing awareness regarding sensors measurements or events that are deemed critical; and when evoked, trigger actions to be carried out in order to contain adverse consequences. In NPAS, redlines may be activated, de-activated, inhibited, and de-inhibited programmatically (e.g. within a sequence step of a plan). Multiple redline definitions may be associated with a sensor measurement, or a virtual sensor that may represent occurrence of an event. Figure 8 shows the redline creation user interface.

Similar to the user interface used to create operational sequences, the menu used to create redlines is automatically updated when sensors or events (events may be represented by virtual sensors) are updated (added or deleted) in the AKM.


Data Monitor Configuration

☐PT-8-O2
☐PT-9-CH
☐PT-9-O2
☐SAT_PT161
☐SAT_TT162
☑TT105
☐TT110
☐TT111
☐TT112
☐TT113
☐TT114
☐TT127
☐TT128
☐TT129
☐TT130
☐TT131
☐TT146
☐TT149
☐TT156
☐TT162
☐TT165
☐TT174
☐TT191
☐TT194
☐TT-1-CH
☐TT-1-O2
☐TT-1X10-O2

Start Hours 0 Start Minutes 0 Start Seconds 0 Red Line ☑

End Hours 0 End Minutes 0 End Seconds 0

No Time Limit ☑ Limits Color 🟦 blue

STEP None Found Line Color 🟩 green

Maximum Value

Enabled ☑ Setting -200

Conditior lower threshold

Minimum Value

Enabled ☐ Settings 0

Conditior Enter Low Red Line Condition Here.

Current Monitor:0

Monitor Name tt105-lower-threshold

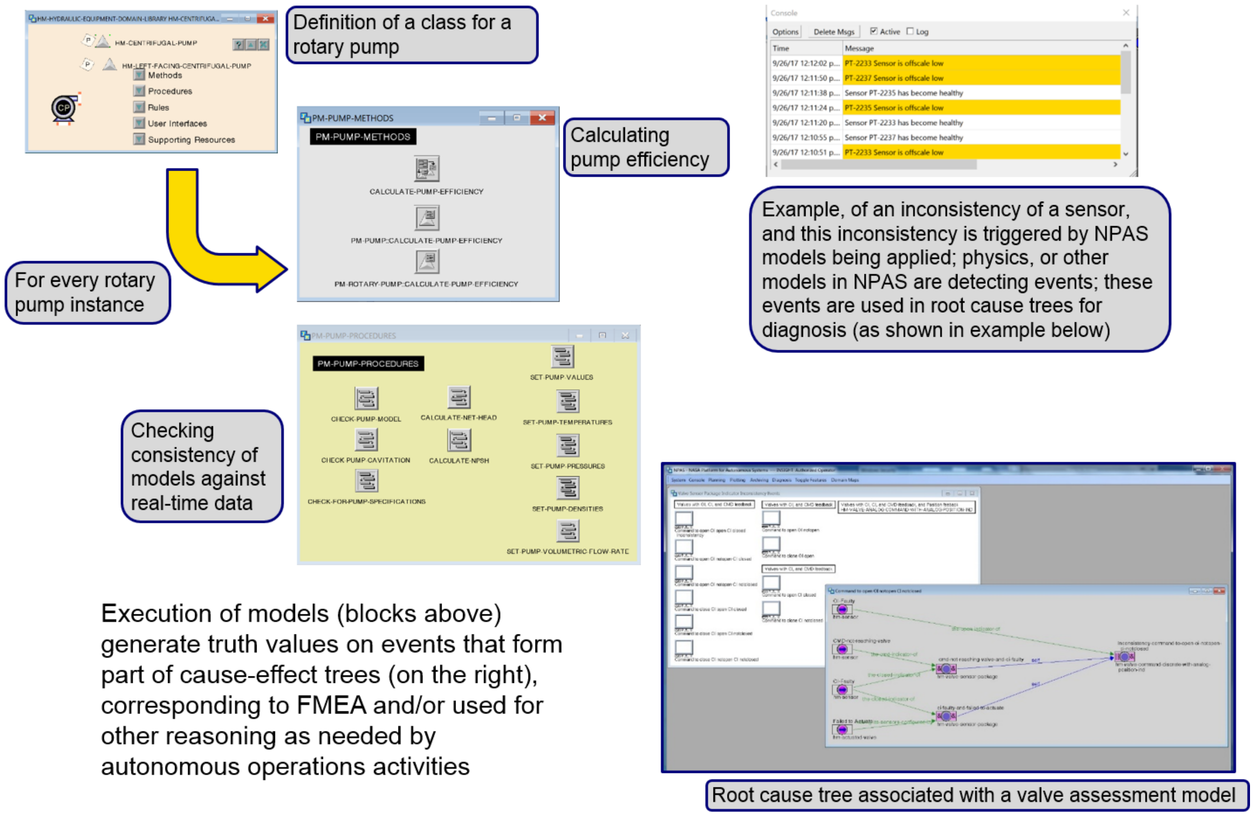
Total Monitors: 0

Figure 8. NPAS Redline creation user interface.

D. NPAS Physics Modeling for real-time Reasoning and Decision Making

The NPAS paradigm for employing physics models for reasoning and decision making is to use any and all available models of sensors, components, and subsystems whenever enough information is available to execute the models. NPAS is able to navigate through any system and understand its composition, and its static and dynamic topology. With this understanding, concepts such as connectivity and readiness for flow of a commodity; and the application of process models for that context (e.g. analysis of flow in a flow subsystem) are enabled with NPAS. Therefore, in contrast with the classic approach of developing physics (or other) models that encompass entire systems (application), NPAS enables creation of piece-wise models and "model packages" that correspond to individual elements (e.g. sensors, valves, pumps, pipes, batteries) and subsystems (collection of elements that comprise an isolated subsystem or a flow subsystem). A detailed description of how NPAS enables incorporation of physics and other models is provided in [17]; the uniqueness is that NPAS is able to understand concepts, define/recognize the boundaries of a subsystem/component/sensor for analysis, assert in real-time a process that is taking place, and apply model packages that are suitable. In this capacity, NPAS is able to "think", to reason, and make decisions.

As an example, Figure 9 shows how pump model packages (or more generically, model packages for components that provide a "pumping" effect as part of a commodity management system) are defined and used. For every instance of a rotary pump, models that calculate operational quantities for reasoning such as efficiency are executed according to a schedule and/or triggered by events. A fundamental model relating pressure across the pump, flow rate, pump speed, pump specifications, and commodity is also applied for consistency checking, or to predict the value of a parameter that may not be available. A "Failed" consistency check results in a generic event for rotary pumps "model-consistency-failure" becoming "true." This type of event is part of one or more cause effect trees that link failure modes of pumps and effects across the systems encompassing the pumps.

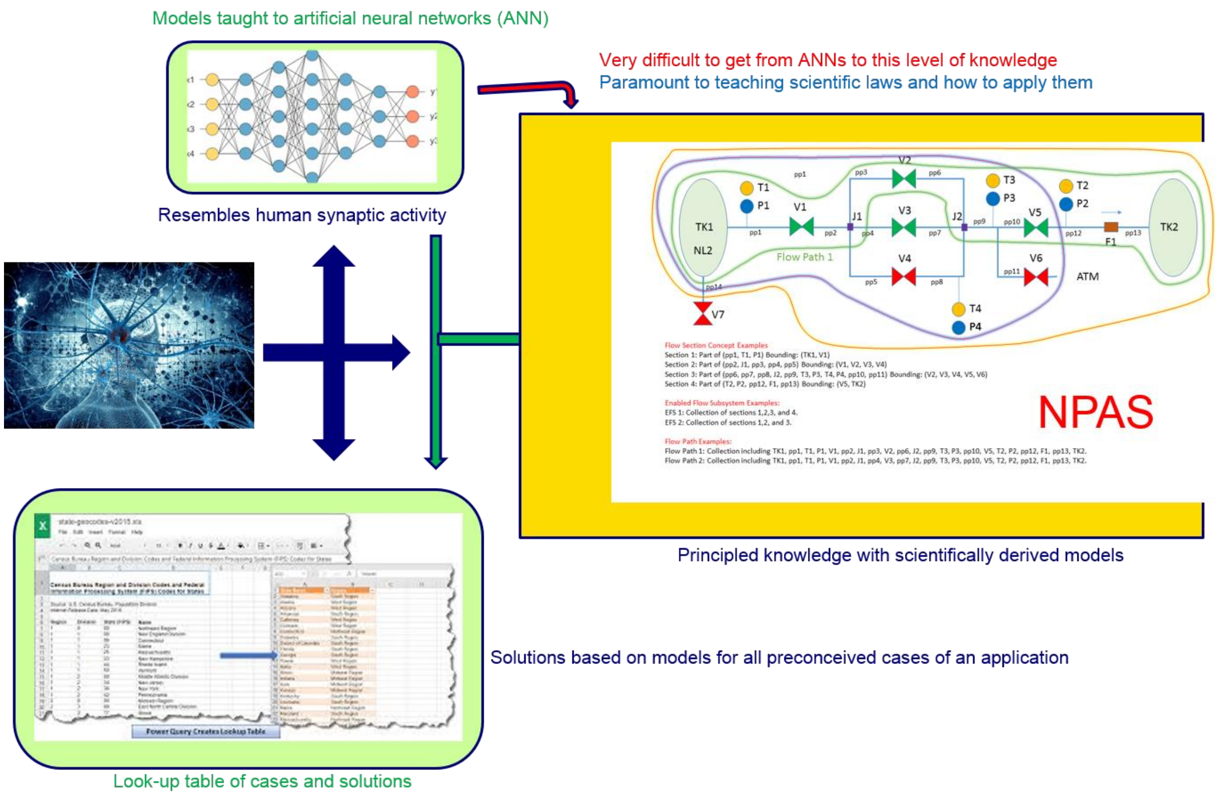

HM-HYDRAULIC-EQUIPMENT-DOMAIN-LIBRARY HM-CENTRIFUGA...
HM-CENTRIFUGAL-PUMP
HM-LEFT-FACING-CENTRIFUGAL-PUMP
Methods
Procedures
Rules
User Interfaces
Supporting Resources
Definition of a class for a rotary pump
PM-PUMP-METHODS
CALCULATE-PUMP-EFFICIENCY
PM-PUMP::CALCULATE-PUMP-EFFICIENCY
PM-ROTARY-PUMP::CALCULATE-PUMP-EFFICIENCY
Calculating pump efficiency
For every rotary pump instance
PM-PUMP-PROCEDURES
SET-PUMP-VALUES
CHECK-PUMP-MODEL
CALCULATE-NET-HEAD
SET-PUMP-TEMPERATURES
CHECK-PUMP-CAVITATION
CALCULATE-NPSH
SET-PUMP-PRESSURES
CHECK-FOR-PUMP-SPECIFICATIONS
SET-PUMP-DENSITIES
SET-PUMP-VOLUMETRIC-FLOW-RATE
Checking consistency of models against real-time data
Console
Options
Delete Msgs
Active
Log
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)
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 tree associated with a valve assessment model

Time	Message
9/26/17 12:12:02 p....	PT-2233 Sensor is offscale low
9/26/17 12:11:50 p....	PT-2237 Sensor is offscale low
9/26/17 12:11:38 p....	Sensor PT-2235 has become healthy
9/26/17 12:11:24 p....	PT-2235 Sensor is offscale low
9/26/17 12:11:20 p....	Sensor PT-2233 has become healthy
9/26/17 12:10:55 p....	Sensor PT-2237 has become healthy
9/26/17 12:10:51 p....	PT-2233 Sensor is offscale low

Figure 9. NPAS physics/engineering modeling

E. NPAS and AI

NPAS includes model-based reasoning that is founded by physics models (or other kinds of models) and processes that resemble how experts (e.g. engineers, scientists) apply models and strategies to reason and make decisions. This approach is inspired by work on qualitative modeling done by the AI community over the past several decades [insert reference here]. Figure 10 represents the contrast among AI technologies/approaches including neural networks, brute-force, and the NPAS methodology.


Models taught to artificial neural networks (ANN)
Resembles human synaptic activity
Very difficult to get from ANNs to this level of knowledge
Paramount to teaching scientific laws and how to apply them
TK1
NL2
TK2
T1
P1
T2
P2
T3
P3
T4
P4
V1
V2
V3
V4
V5
V6
V7
J1
J2
F1
ATM
Flow Path 1
pp1
pp2
pp3
pp4
pp5
pp6
pp7
pp8
pp9
pp10
pp11
pp12
pp13
pp14
NPAS
Principled knowledge with scientifically derived models
Solutions based on models for all preconceived cases of an application
Power Query Creates Lookup Table
Look-up table of cases and solutions

Flow Section Concept Examples
Section 1: Part of (pp1, T1, P1) Bounding: (TK1, V1)
Section 2: Part of (pp2, J1, pp3, pp4, pp5) Bounding: (V1, V2, V3, V4)
Section 3: Part of (pp6, pp7, pp8, J2, pp9, T3, P3, T4, P4, pp10, pp11) Bounding: (V2, V3, V4, V5, V6)
Section 4: Part of (T2, P2, pp12, F1, pp13) Bounding: (V5, TK2)

Enabled Flow Subsystem Examples:
EFS 1: Collection of sections 1,2,3, and 4.
EFS 2: Collection of sections 1,2, and 3.

Flow Path Examples:
Flow Path 1: Collection including TK1, pp1, T1, P1, V1, pp2, J1, pp3, V2, pp6, J2, pp9, T3, P3, pp10, V5, T2, P2, pp12, F1, pp13, TK2.
Flow Path 2: Collection including TK1, pp1, T1, P1, V1, pp2, J1, pp4, V3, pp7, J2, pp9, T3, P3, pp10, V5, T2, P2, pp12, F1, pp13, TK2.

Figure 10. NPAS physics/engineering models and reasoning.

F. Electronic Data Sheets

NPAS also supports inclusion of IEEE 1451 Standard Transducer Electronic Data Sheets (TEDS). In addition, extensions to the standard have been developed to define Component Electronic Data Sheets (CEDS) and Health Electronic Data Sheets (HEDS). This competency enables interoperability and the creation of domain languages and capability languages that are fundamental to implement "thinking" autonomy that is evolutionary and sustainable. Many industrial providers of transducers currently support TEDS.

G. Network capabilities

NPAS includes infrastructure for network communications such as an OPC Bridge, to communicate with industrial systems, and in turn, enables them for autonomous operations. Also included is a JAVA Bridge to enable sophisticated interactions with other applications across a network, an Oracle Bridge to interact with Oracle databases, and an ODBC Bridge to interact with standard databases. At NASA, bridges that support communications protocols such as Space Packet Protocol (SPP) and Software Bus Network (SBN) protocol have been developed [insert reference here]. Using the SBN Bridge, NPAS applications can interact with other applications in a network that support the SBN protocol using the subscribe and broadcast modalities. NPAS also includes infrastructure to develop custom protocols, for interaction with a broad range of other applications; for example, bridges to run Matlab applications, or executable files. This enables incorporation of external applications, and doesn't require re-coding in the G2 language (NPAS is a platform built in the G2 software environment) [18]. For example, Figure 11 shows an architecture that has been used to implement autonomous operations in industrial systems that use Programmable Logic Controllers (PLC's).

H. Distributed Autonomy Capabilities

NPAS is built using the G2 software environment, and multiple NPAS applications can run in one processor, or can be distributed among processor cores, or among processors physically distributed across a network. Integration of multiple NPAS applications that may encompass a distributed autonomous system of systems can be accomplished by using the network capabilities described above (in section G.). In addition, G2 includes a native G2-to-G2 communication capability that enables NPAS applications to access and make reference to objects and object attributes defined in another NPAS. Integration is handled using the subscribe and broadcast modalities, whereby a procedure can be alerted by an attribute of an object and execute when the attribute experiences a change in value. In this manner, procedural code doesn't need to have while-loops waiting for a change to occur. Figure 5 shows a Hierarchical Distributed Autonomy Architecture where the vehicle manager and each subsystem are expected to be independent NPAS autonomous applications. This architecture application was implemented as a pilot validation demonstration project that included development of an NPAS Vehicle Manager, and an NPAS Power, and an NPAS Avionics.

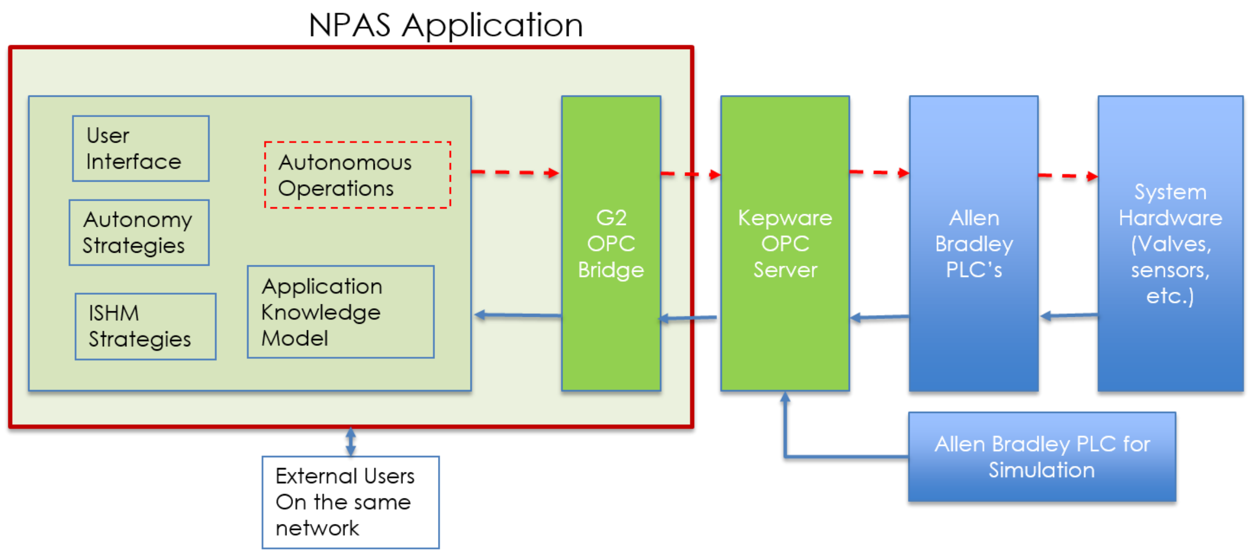
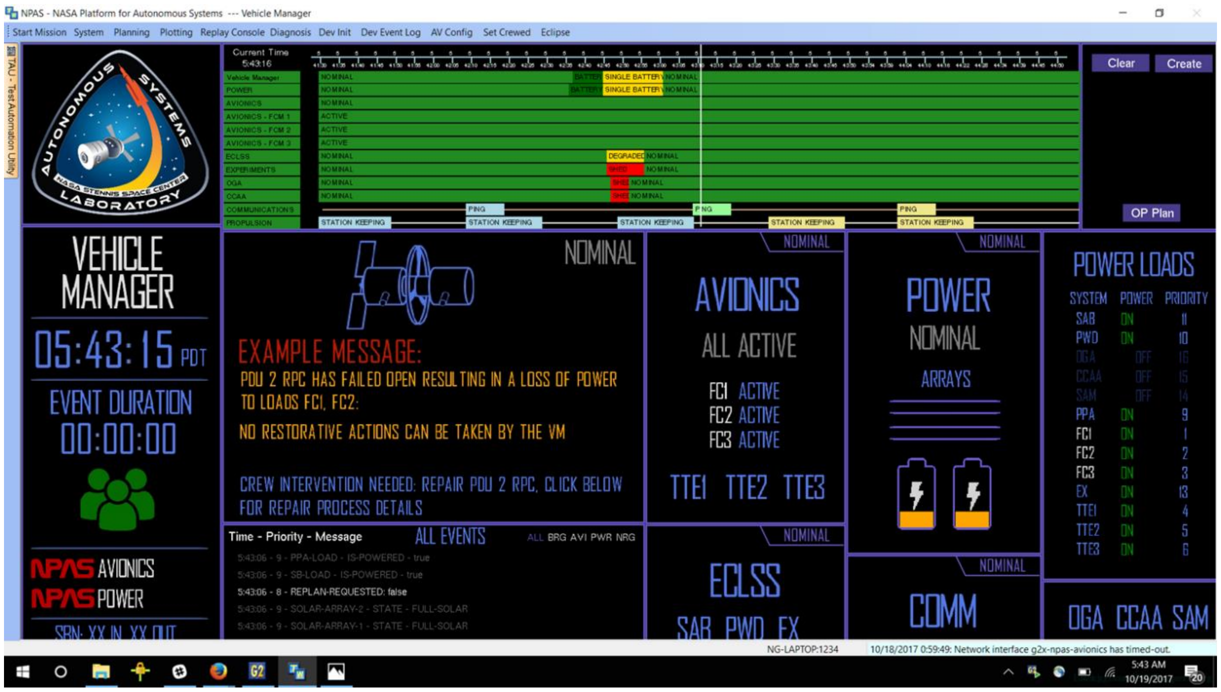


Figure 11. Architecture for NPAS autonomy implementation in industrial systems.

VI. Implementations of Autonomous Systems using NPAS

Autonomous operational capability implemented using NPAS has been demonstrated successfully for 1) Autonomous Propellant Loading from a storage tank to a flight tank at the Cryogenic Test Bed Laboratory (KSC); 2) a simulated (physics) implementation/demonstration of simultaneous multi-system autonomous propellant loading operations at KSC's UPSS (Universal Propellant Servicing System); 3) an demonstration that used EFT-1 telemetry data from Orion to implement an integrated system monitoring, anomaly detection, and FMEA capability for a subset of Orion's EFT-1 systems; 4) a demonstration for implementing a deep space autonomous habitat exhibiting hierarchical distributed autonomy encompassing 3 autonomous systems (vehicle manager, power, and avionics), in JSC's A & S iPAS Lab [Figure 12]; and 5) is currently being infused for autonomous operations at the SSC High Pressure Gas Facility [Figure 13] - a critical piece of test infrastructure used to support Space Launch System (SLS), Department of Defense, DoD, and commercial propulsion testing for SLS. This infusion includes certification of NPAS as Class C Safety Critical for autonomous operations. All implementations exemplify the benefit of using NPAS for rapid autonomy development and have demonstrated associated significant cost savings.



Current Time 5:43:16

Vehicle Manager	NOMINAL / BATTERY / SINGLE BATTERY / NOMINAL
POWER	NOMINAL / BATTERY / SINGLE BATTERY / NOMINAL
AVIONICS	NOMINAL
AVIONICS - FCM 1	ACTIVE
AVIONICS - FCM 2	ACTIVE
AVIONICS - FCM 3	ACTIVE
ECLSS	NOMINAL / DEGRADED / NOMINAL
EXPERIMENTS	NOMINAL / SHED / NOMINAL
OGA	NOMINAL / SHED / NOMINAL
CCAA	NOMINAL / SHED / NOMINAL
COMMUNICATIONS	PING / PING / PING
PROPULSION	STATION KEEPING / STATION KEEPING / STATION KEEPING / STATION KEEPING / STATION KEEPING

EXAMPLE MESSAGE:
PDU 2 RPC HAS FAILED OPEN RESULTING IN A LOSS OF POWER TO LOADS FC1, FC2:
NO RESTORATIVE ACTIONS CAN BE TAKEN BY THE VM
CREW INTERVENTION NEEDED: REPAIR PDU 2 RPC, CLICK BELOW FOR REPAIR PROCESS DETAILS

Time - Priority - Message ALL EVENTS ALL BRG AVI PWR NRG

```
5:43:06 - 9 - PPA-LOAD - IS-POWERED - true
5:43:06 - 9 - SB-LOAD - IS-POWERED - true
5:43:06 - 8 - REPLAN-REQUESTED: false
5:43:06 - 9 - SOLAR-ARRAY-2 - STATE - FULL-SOLAR
5:43:06 - 9 - SOLAR-ARRAY-1 - STATE - FULL-SOLAR
```

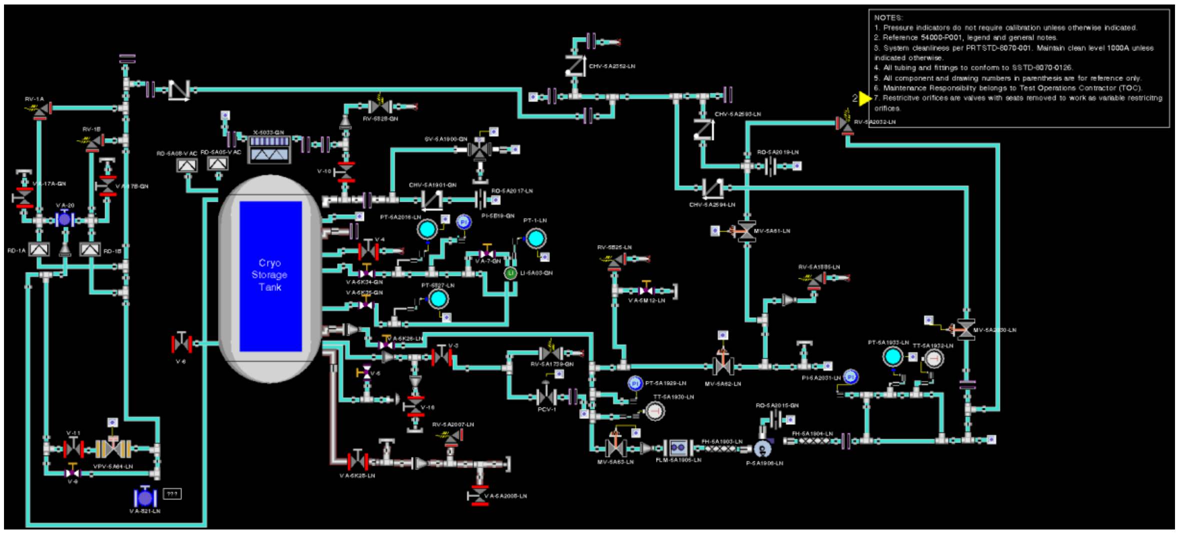
POWER LOADS

SYSTEM	POWER	PRIORITY
SAB	ON	11
PWD	ON	10
OGA	OFF	16
CCAA	OFF	15
SAM	OFF	14
PPA	ON	9
FC1	ON	1
FC2	ON	2
FC3	ON	3
EX	ON	13
TTE1	ON	4
TTE2	ON	5
TTE3	ON	6

NG-LAPTOP:1234 10/18/2017 0:59:49: Network interface g2x-npas-avionics has timed-out.

5:43 AM 10/19/2017

Figure 12. NPAS Vehicle Manager user interface for a pilot autonomous space habitat implementation.



NOTES:
1. Pressure indicators do not require calibration unless otherwise indicated.
2. Reference 54000-P001, legend and general notes.
3. System cleanliness per PRTSTD-8070-001. Maintain clean level 1000A unless indicated otherwise.
4. All tubing and fittings to conform to SSTD-8070-0126.
5. All component and drawing numbers in parenthesis are for reference only.
6. Maintenance Responsibility belongs to Test Operations Contractor (TOC).
7. Restricitve orifices are valves with seats removed to work as variable restriciting orifices.

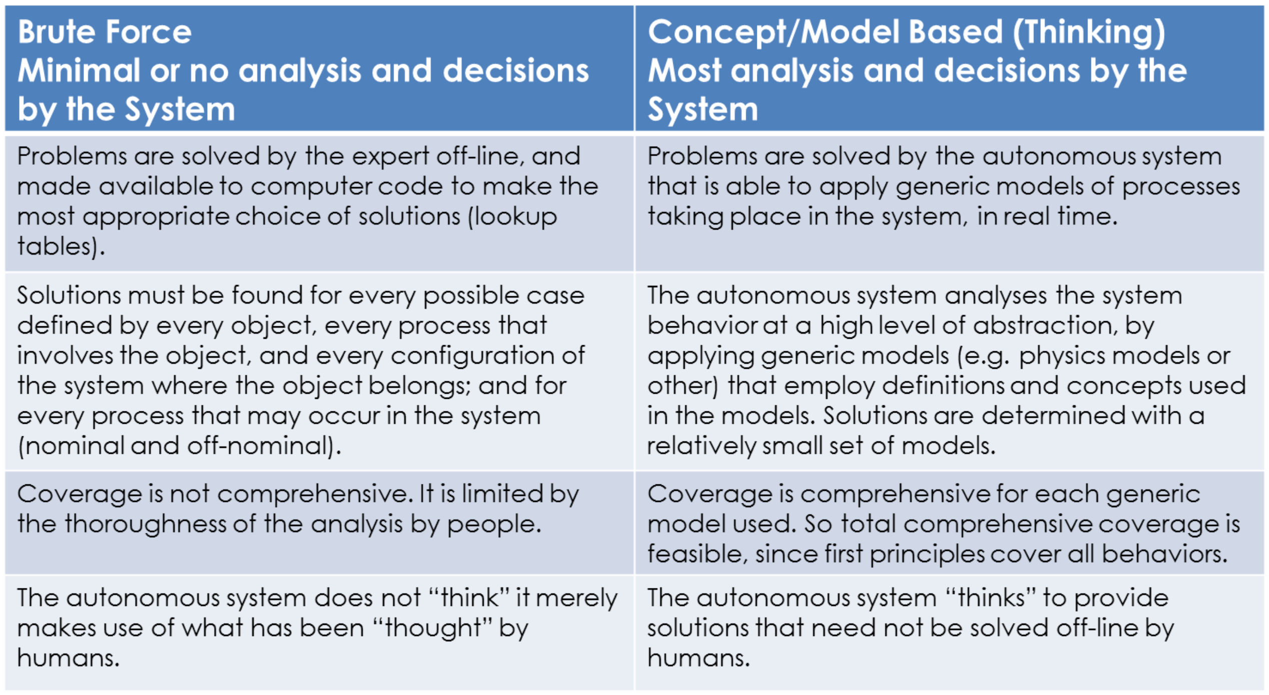
Figure 13. Life schematics as part of the domain knowledge model of SSC's High Pressure Gas Facility.

NPAS Path-to-Flight

Current NPAS efforts are focused on benchmarking performance and advancing NPAS operation on flight-qualified hardware/software platform PowerPC (SP0-S) running in VxWorks environment, and on ARM-A53 running in Linux environment to create a design guideline that supports other NPAS-based architectures for future missions (e.g., Gateway, rovers, In-Situ Resource Utilization ISRU, and habitat). These NPAS capabilities will enable the advance of a disruptive autonomous operations software capability for use in safety critical applications that can be affordably and timely qualified and deployed.

VII. Conclusions and Recommendations

NPAS is a platform that provides “thinking” autonomous operations capability and leverages a COTS product to address NASA’s autonomous capability needs. A comparison of the differences between brute-force autonomy and “thinking” autonomy is exemplified in Figure 14. Without the ability to implement “thinking” autonomy, the technical and cost challenges to achieve the degrees of autonomy required by NASA for deep space exploration will be insurmountable.



Brute Force Minimal or no analysis and decisions by the System	Concept/Model Based (Thinking) Most analysis and decisions by the System
Problems are solved by the expert off-line, and made available to computer code to make the most appropriate choice of solutions (lookup tables).	Problems are solved by the autonomous system that is able to apply generic models of processes taking place in the system, in real time.
Solutions must be found for every possible case defined by every object, every process that involves the object, and every configuration of the system where the object belongs; and for every process that may occur in the system (nominal and off-nominal).	The autonomous system analyses the system behavior at a high level of abstraction, by applying generic models (e.g. physics models or other) that employ definitions and concepts used in the models. Solutions are determined with a relatively small set of models.
Coverage is not comprehensive. It is limited by the thoroughness of the analysis by people.	Coverage is comprehensive for each generic model used. So total comprehensive coverage is feasible, since first principles cover all behaviors.
The autonomous system does not “think” it merely makes use of what has been “thought” by humans.	The autonomous system “thinks” to provide solutions that need not be solved off-line by humans.

Figure 13. Brute-Force Autonomy VS "Thinking Autonomy."

NPAS is a single software platform that encompasses all functionality required to implement intelligent autonomous operation, which are natively integrated, and includes generic re-usable strategies for ISHM and autonomy applications (systems) infrastructure. NPAS provides the foundation to achieve autonomy that is anchored on a comprehensive knowledge model of an application (system) that can evolve systematically as information and knowledge is augmented, added, refined, as well as when other external applications that increase autonomy functionality are incorporated.

VIII. Acknowledgements

This work was made possible by funds from NASA’s Advanced Exploration Systems (AES). The authors are thankful to Duane Armstrong from NASA SSC, Test Technology Branch Chief, for his sustained support. The personnel from D2K Technologies and Kim Wilkins from General Atomics have been involved in the development of NPAS, and in a significant portion of the implementations described.

IX. References

[1] International Council on Systems Engineering, INCOSE (https://www.incose.org)
[2] SysML Open Source Project (https://sysml.org/)
[3] Mil Standard 1626A
[4] D.E. Bernard et al, “Design of the Remote Agent experiment for spacecraft autonomy,” 1988 IEEE Aerospace Conference, March 28-28, Snowmass at Aspen, CO, USA.

[5] https://www.nasa.gov/ames-partnerships/technology/technology-opportunity-hybrid-diagnostic-engine-hyde-for-fault-analysis.

[6] https://aerospace.honeywell.com/en/product-listing/health-and-usage-monitoring

[7] Davidson, M., and Stephens, J., "Advanced Health Management System for the Space Shuttle Main Engine," 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference, 11-14 July 2004

[8] Fernando Figueroa and Kevin Melcher, "Integrated System Health Management for Intelligent Systems," chapter in the book Advances in intelligent and Autonomous Aerospace Systems, AIAA Progress in Astronautics and Aeronautics, Vol. 241, Editor: John Valasek, Professor and Director, Vehicle Systems & Control Laboratory Aerospace Engineering Department Texas A&M University. 2012, pp. 173-200.

[9] F. Figueroa and J. Schmalzel, "Rocket Testing and Integrated System Health Management", Chapter in the book Condition Monitoring and Control for Intelligent Manufacturing (Eds. L. Wang and R. Gao), pp. 373-392, Springer Series in Advanced Manufacturing, Springer Verlag, UK, 2006.

[10] Fernando Figueroa, Jon Morris, Mark Turowski, Richard Franzl, Mark Walker, Ravi Kapadia, and Meera Venkatesh, "Monitoring System for Storm Readiness and Recovery of Test Facilities: Integrated System Health Management (ISHM) Approach," MFPT: The Applied Systems Health Management Conference 2011, 10-12 May 2011, Virginia Beach, VA, USA.

[11] Fernando Figueroa and Kevin Melcher, "Integrated Systems Health Management for Intelligent Systems," AIAA Infotech@Aerospace, 29 - 31 Mar 2011, Hyatt Regency St. Louis at the Arch, St. Louis, Missouri.

[12] Fernando Figueroa, Randy Holland, John Schmalzel, Dan Duncavage, Rick Alena, Alan Crocker, "ISHM Implementation for Constellation Systems," 42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, July 9-12, 2006, Sacramento Convention Center, Sacramento, CA.

[13] Ashley W. Stroupe et al, "Technology for Autonomous Space Systems," CMU-RI-TR-00-02, The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, September 2002.

[14] Fernando Figueroa, Mark Walker, Kim Wilkins, Jaime Toro-Medina, Gerald Stahl, Robert Johnson, Jared Sass, Justin Youney, "Ground Operations Autonomous Control and Integrated Health Management," Commercial and Government Response Access to Space Technology Exchange, June 22-25, 2015, Westfields Marriott Washington Dulles, chantilly, Virginia.

[15] Fernando Figueroa, Mark Walker, Kim Wilkins, Jaime Toro-Medina, Gerald Stahl, Robert Johnson, Jared Sass, Justin Youney, "Ground Operations Autonomous Control and Integrated Health Management," Commercial and Government Response Access to Space Technology Exchange, June 22-25, 2015, Westfields Marriott Washington Dulles, chantilly, Virginia

[16] Fernando Figueroa, "Implementations Using NASA's Autonomous Operation Mission Development Suite (AO-MDS)," G2 Users Group Conference, September 27-29, 2016, Gaylord Palms Resort and Convention Center, Orlando, FL, USA.

[17] Fernando Figueroa and Mark Walker, "Integrated System Health Management (ISHM) and Autonomy," AIAA SciTech 2018, Gaylord Palms, Kissimmee, Florida, January 8 – 12, 2018.

[18] http://www.gensym.com/