https://ntrs.nasa.gov/search.jsp?R=20170012201 2019-05-01T03:18:29+00:00Z

Integrated System Health Management (ISHM) and Autonomy

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

and
Mark G. Walker[2]
D2K Technologies, Ocean Side, CA 92056, USA

Systems capabilities on ISHM and autonomy have traditionally been addressed separately. This means that ISHM functions, such as anomaly detection, diagnostics, prognostics, and comprehensive system awareness have not been considered traditionally in the context of autonomy functions such as planning, scheduling, and mission execution. One key reason is that although they address systems capabilities, both ISHM and autonomy have traditionally individually been approached as independent strategies and models for analysis. Additionally, to some degree, a unified paradigm for ISHM and autonomy has been difficult to implement due to limitations of hardware and software. This paper explores a unified treatment of ISHM and autonomy in the context of distributed hierarchical autonomous operations.

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. I. Introduction

Throughout its evolution, ISHM has been migrating toward a more fitting role in the engineering of systems. Initially, even the development of new sensor technologies have been done aligned with ISHM. The ISHM community now addresses ISHM as a capability that integrates data, information, and knowledge which enables the implementation of ISHM functions that considers interactions among systems' components and subsystems; as well as interactions among systems. These functions are primarily (1) anomaly detection (2) diagnostics, (3) prognostics, and (4) integrated awareness for the operator. However, it is difficult to identify a successful operational implementations of real-time on-board ISHM systems. Today, the closest example of an operational ISHM system could be seen in the Fault-Check capability of the Orion capsule, which is very good demonstration, however, it is merely the result of local FMEA rather than system wide analysis. Also, in this case, FMEA is implemented as a lookup table, where the "thinking" is done offline and the system simply applies results of analysis previously established off-line.

Autonomy is often mentioned in the context of planetary robots and planetary spacecraft [1]. The main challenge in autonomy is to have a system that can select strategies to deal with unplanned events so that the system can accomplish the mission or exchange plans based on alternate results. It is true that there are systems that can apply autonomy strategies, but they do it by brute-force. This means that all events that the designers are able to consider

---

[1] Lead Autonomous Systems and Operations, Advanced Technology and Technology Transfer, AIAA Associate Fellow.
[2] V.P. Engineering.

and corresponding strategies to deal with them have been specified. Therefore, in this approach, autonomy strategies have been applied offline and all the system (computer) does is match events with corresponding strategies. Brute-force autonomy does not require a thinking system, but has profound limitations in coverage (misses cases), in adherence to foundational models, in re-use, and in the ability to evolve.

This paper will address a theory and process to implement ISHM and autonomy by enabling a system to "think" in real-time, based on models that derive from foundational principles. It will also address the interaction between ISHM and autonomy.

III. ISHM

ISHM encompasses the following functionalities: (1) anomaly detection, (2) diagnostics, (3) prognostics, and (4) integrated awareness for the operator. Health management is not a new concept. The question is HOW it is done. It relies on people who operate the systems and those who support the systems. As technology advanced, the individuals doing ISHM were assisted by tools that made them more effective in achieving the functionalities at higher capability levels. What must be highlighted is that analyses based on models is the fundamental approach to implementing ISHM. The question is, if the analysis is integrated (as opposed to localized), and how it is achieved (the combination of operators and technologies).

A capability that has been in use for a long time is the Health and Usage Monitoring System (HUMS) for helicopters. The HUMS was designed to monitor data from helicopter subsystems and processes it using a set of specialized algorithms. The resulting anomaly indicators and original data are used by experts to infer if critical elements might be trending toward failure. In this system, knowledge and its integrated interpretation is primarily done by people. HUMS is commercially available for aircraft applications from Honeywell [2]. Another ISHM implementation is the Advanced Health Management System for the Space Shuttle Main Engine (SSME) [3]. The effort incorporated automated analysis of trending to predict if critical data streams might be approaching out-of-norm values.

One way to describe how ISHM is done currently is by considering layers where activities to achieve ISHM is performed. Figure 1 shows how ISHM may be done in a rocket engine test stand or a spacecraft such as the International Space Station (ISS). At each layer; data, information, and knowledge (DIaK) is applied to achieve a degree of capability (functional capability level - FCL) to help manage the health of the system. The layers represent the resources of DIaK available for ISHM. Because knowledge is so crucial to achieve high FCL, and because knowledge normally resides in individuals, the lower layers include the most people and the most knowledge. If the spacecraft could accommodate a large number of people to operate it, people could do the job (analysis, conclusions, and operational decisions). But that is not the case, and most operators and support personnel are on Earth (on the ground) while the ISS is on orbit (in space). At the top layer (Layer 1) is the system itself with some automated capability to manage its health; generally detection of signal range/limit violations that activate alarms. At the next layer down (Layer 2) are the astronauts who can directly operate the station. They represent the local knowledge and have local data and information to manage the Station's health. At the next layer down (Layer 3) are the individuals in the control room. Additional DIaK is accrued with the control room personnel, and issues can be resolved faster and better in support of the crew. Here, diverse knowledge is employed regarding each subsystem and their interactions.

In this layered implementation of ISHM, strategies for analysis and use of models is done by expert personnel. In order to make the system capable of ISHM, these strategies and analyses must be embedded into the system (spacecraft/test stand) so that they may be accomplished without human intervention, or intervention from operators (astronauts/test engineers) as per desired concepts of operations (Con-Ops). The analyses and conclusions conducted by experts must move to the two upper layers, and preferably to the top layer (Layer 1), the system. When this happens, Layer 1 becomes a fully autonomous system.

How on-board ISHM is implemented becomes the key issue. Typically, implementation is done by having a collection of algorithms that detect specific anomalies that are possible during specific regimes of operation. That is, the problem-space has been analyzed off-line by experts, and resulted in pre-defined solutions. The "thinking" has been done and defined off-line, and not by the system. The health management system simply applies the "thinking" done by experts off-line. The authors prescribe the "thinking" that must be done by the system. For this to happen, analysis and thought processes must be embedded rather than just by pre-conceived cases and their respective solutions. However, a fundamental element to apply "thinking" is to have a comprehensive knowledge model of the system. This is the approach that is used for development of a capability for "thinking" ISHM, as well as for "thinking" autonomy. NASA Platform for Autonomous Systems (NPAS), software platform, which encompasses integrated

technologies to achieve hierarchical distributed autonomy, is the result of this innovative approach to ISHM. Evolution of concepts and technology development leading to NPAS is documented in references [4-8].

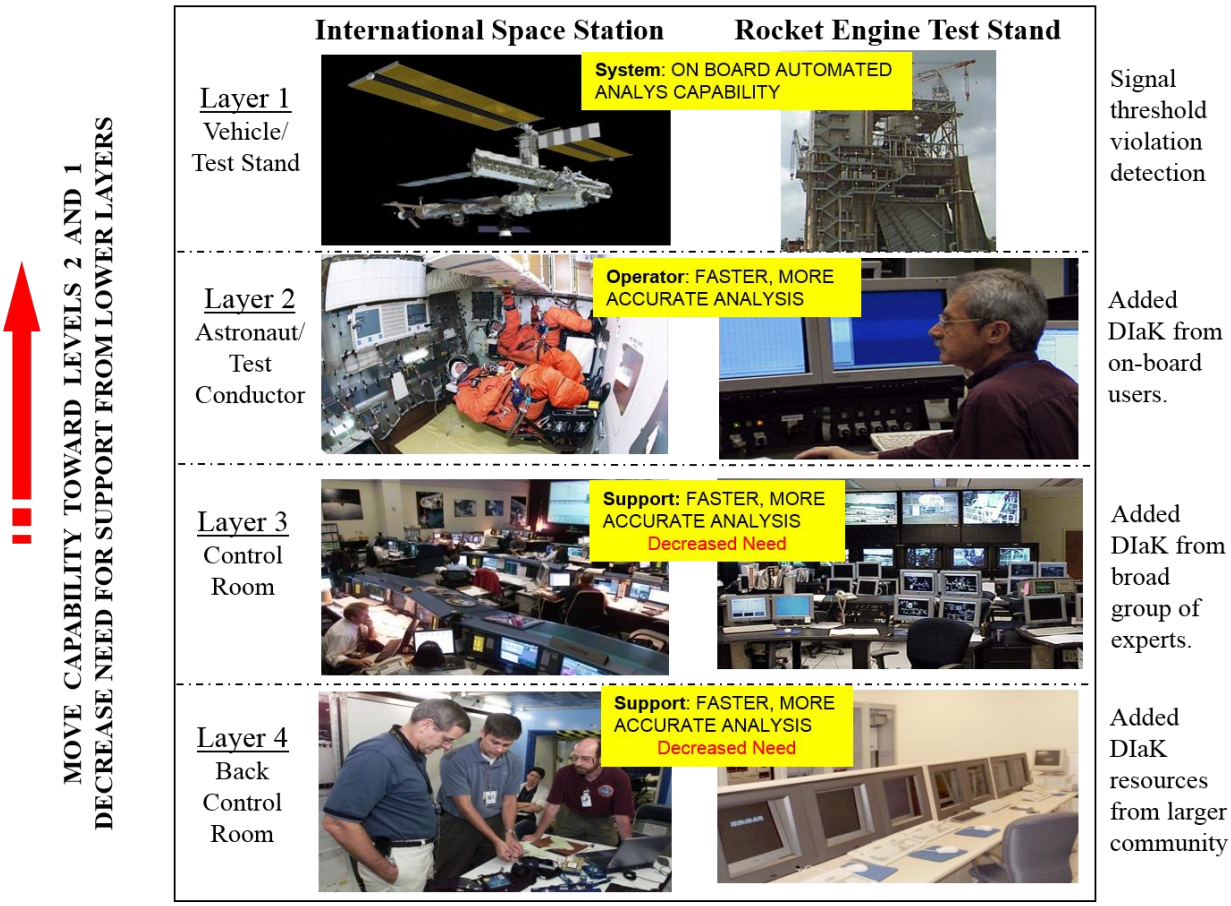

Figure 1. Layered ISHM capability.

A. Anomaly Detection

There are many strategies to detect anomalous behavior. One strategy is to define what nominal behavior is and assign anomaly to any behavior that is not nominal. Another strategy is to define what anomalous behavior is and identify it when it occurs. These strategies imply the use of models of processes that describe the operation of a system (nominal and anomalous processes).

Since systems are designed and analyzed to perform specific functions, models describing nominal behavior of systems exist. Physics-based models for analysis are certainly at the core of any design and operation of a system. For this paper, the implementation of anomaly detection for a "thinking" system that is designed for commodity distribution is described. The capability is part of NPAS and is currently being deployed to make a Nitrogen distribution system autonomous.

Using NPAS, a comprehensive knowledge-model of a system is created, a Knowledge Domain Model (KDM). Figure 2 shows a portion of a model for the nitrogen system. The foundations of the model is based on the schematics used to build the system. Every part in the schematics is modeled as an object, and then NPAS automatically generates connectivity as the model is built using a graphical interface. Then, once these parts of the software are in place, reasoning involving immediate connectivity among parts can be applied. For example, a type of valve is connected a type of pipe, or that a type of sensor is attached to a type of pipe. Additional relationships among parts are needed to augment analysis and reasoning capabilities. These relationships may be added manually or may be discovered by the system itself. Next, a process of discovery by NPAS is described which determines flow paths from sources to sinks (loads) so that physics-based models and analyses may be applied to perform anomaly detection or predictions.

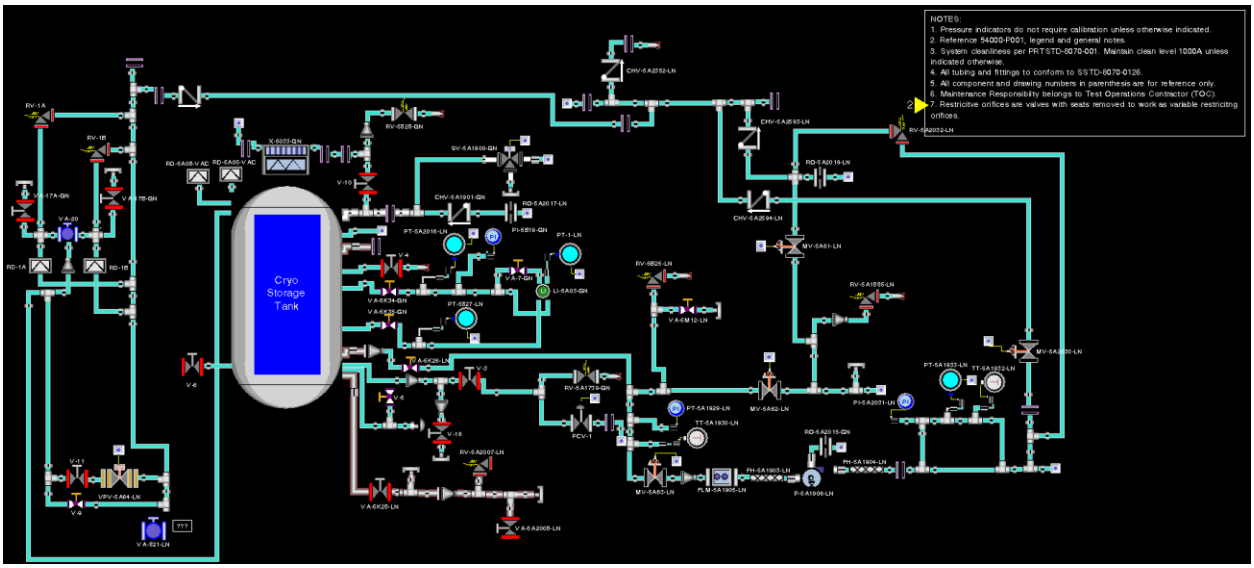

Figure 2. Life schematic encompassing the application knowledge model.

B. Concepts and models for "thinking" flow systems

Typically, flow analysis is performed on systems with pipes, tanks, valves, pumps, sensors, and other items. Therefore, an ISHM capability must "understand" these items in the context of the role they play in a flow system. Furthermore, flow systems must be identified in order to apply models and perform analysis while operational.

NPAS uses the following process to identify flow systems:

- Discover flow paths between sources and sinks - these are paths that are discovered dynamically as system configuration changes due to valve actuation, or to changes in other flow control elements in the flow path.
- Apply process models along flow paths to identify inconsistencies - for example, absence of flow when the model suggests there should be flow.

Figures 3 demonstrates how flow paths are defined as a concatenations of *Flow-Sections* to define *Enabled-Flow-Systems*, and identification of *Flow-Paths* that show open valves (flow control elements) from source (tank) to sink (tank, atmosphere). Note that the discovery of Flow-Sections and Enabled-Flow-Systems is done once at initiation, and is updated only when the schematics change. The determination of Flow-Paths is dynamic and is done every time valve movements occur and at specified rates (Figure 4).

Flow-Paths are then defined as ordered collections of elements (parts) that originate at source elements and end at sink elements (membership sequences). Then, since the system is always aware of all flow paths, a broad range of analyses is done persistently and comprehensively in all flow paths existing at any given time. The analyses is done using models that are consistent with the physics of the processes taking place.

1. *Apply flow models to flow-sections, enabled-flow-systems, or the entire flow path or to elements or portions therein*

Once the system recognizes the concepts above (flow-section, enabled-flow-system, flow-path), a wide range of models can be applied consistent with the configuration and processes taking place in the system. Below are some example models that show how typical operator strategies for analysis can be readily implemented. The reason is because the system is able to speak the language/concepts used by operators when applying models.

Model: "Throughout each flow section there is one commodity, and some physical variables of the commodity, for example, temperature should be similar throughout the piping section." Similar value set 1: T3, T4. This model enables comparing temperature measurements to increase or decrease belief in the sensors health.

Model: "Flow rates along sections shared by all branches defining multiple flow paths are the same." In Figure 5, F1 provides flow measurement through V5 and V1, since these valves belong to all paths in this particular configuration. This is a model to transfer measurements from one part of the system to another for applying models in a piece-wise manner. In this case, it can be seen that F1 can be used to apply a model for flow through Valve V1. In this way, models of flow through valves are applied persistently and comprehensively for all valves where measurements can be transported to provide pressure drop and flow.

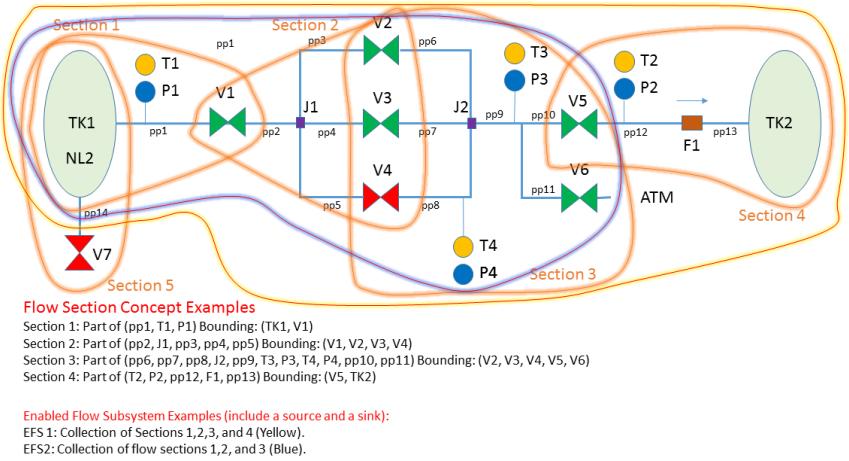

Figure 3. Concepts describing Flow Section and Enabled Flow Subsystem.

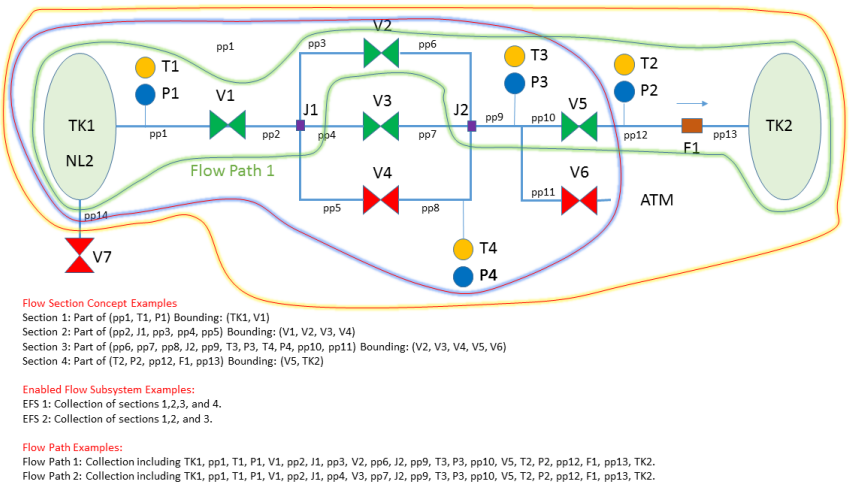

Figure 4. Concept of Flow Path.

2. Concepts and models for "thinking" valve operations

NPAS applies similar analysis to the operation of valves. For this situation, NPAS resolves the valve state by a given command and feedback from open and closed indicators. This is simply a state-machine with some enhancements that the platform enables. In addition to resolving the valve state, the software also generates events that describe the analysis justification associated with particular events, which can then be used to help diagnose possible causes of inconsistencies among the command and feedback indicators.

3. Diagnostics

Diagnostics is implemented based on failure modes and effects analysis (FMEA). Typically FMEA is somewhat generic when it refers to elements of a system (e.g. valves, pipes, and pumps). That is, a valve may fail-open, fail-closed, fail-stuck in position, or leak. These failures are related to possible causes. A valve that fails stuck in position, could be due to the fact that the actuator has failed, or that an object is impeding its movement, or that the control signal can't be changed. These are generic causes that apply to classes of valves and can be re-used. However, system-

wide FMEA is typically done by analysis off-line, where interdependencies among system elements are studied and cause-effect reasoning is done by experts, off-line. Conducting this type of analysis on-board, by a "thinking" system, is possible when the system understands and uses concepts and models that the experts use; similarly to as was described above, in the Anomaly Detection Section. This type of "thinking" enables employing diagnostics strategies that are generic and can be applied to a broad range of systems.

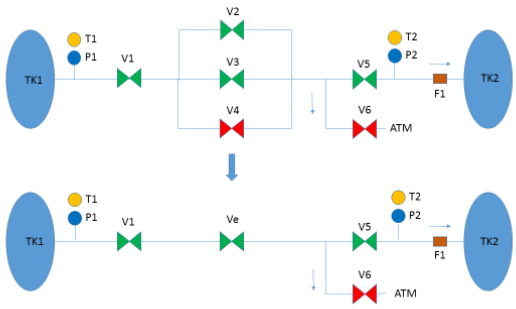

Figure 5. Transfer of flow rate along flow paths.

One typical and powerful diagnostics strategy pertains to channelization. Channelization consists of defining the path for distribution (flow) of commodities (fluid, electric, or other). This is typically done manually by following paths from sources to sinks on schematic drawings. Therefore, automating channelization is paramount to defining system-wide FMEA on-board and for automatic diagnostics. In the Anomaly Detection Section, Enabled-Flow-Systems provide channelization, and Flow-Paths are active channels of flow that are determined in real time.

Given that the system knows all flow paths, flow models may be applied that encompass multiple systems (multiple elements in various systems) and anomaly events in FMEA graphical causal trees, which then may be used to assess and to generate diagnostics. Several examples of this are provided next.

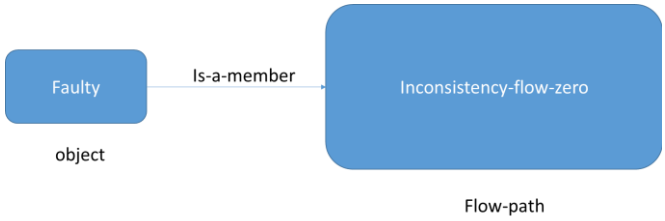

Figure 6. Generic cause-effect graph that assigns suspicion to all members of a flow path whenever an inconsistency is determined.

Model: "Check flow indicators and determine if they satisfy flow conditions." This is a seemingly simple, but very powerful model to assess state (true, false, suspect) of events such health of an element (e.g. sensor, valve, or pump). The strength comes from the multiple ways to define "flow indicators." Some examples are shown below:

- Any flow sensor along a flow-path should not measure zero. If that happens, then all elements belonging to the flow path become suspect of failure. This is represented in a generic graphical causal tree with events as shown in Figure 6. This figure represents actual graphical code. Any time a flow sensor measures a value of about zero, the event *Inconsistency-flow-sensor* is set to TRUE. The graph determines that any member's *Faulty* condition may be the cause, so, if multiple members exist, the *Faulty* event for each is set to SUSPECT. If only one member exists (not reasonable, since the flow system would have to be the flow sensor by itself), its *Faulty* event is set to TRUE. So, with this simple graph, the entire domain is analyzed, ascertaining evidence that some objects may be SUSPECT of being Faulty as a result of evidence indicating no-flow.
- Any valve in the flow-path may be analyzed with a physics flow equation relating flow, pressure drop, and valve opening. If flow determination is reliable, then any inconsistencies lead to assessing anomaly events in valve sensors or command. These events form part of larger graphical causal trees where assessments from other models are inserted.

- One procedure used to assess valve state is to determine how the command and feedback sensors relate. In the case of valves that may be commanded (CMD) to OPEN or CLOSE, which have an OPEN-INDICATOR (OI) and a CLOSED-INDICATOR (CI); all combinations of CMD, OI, and CI are analyzed. Figure 7 shows a graphical causal tree for an event related to this procedure, and an additional event that tracks history of a valve following the command.
    - The event is Inconsistency-cmd-open-OI-open-CI-closed (valve commanded to open, the OI says it is OPEN and the CI says it is closed). The inconsistency may be caused by combinations of the three Faulty events and CMD-not-reaching-valve event as shown in the graph. So, a value of SUSPECT is assigned to combinations that reach the inconsistency event (3 blue arrows). If the additional event indicator-history-not-following-command becomes TRUE for the CI-sensor, then *CI Faulty* becomes TRUE, and is asserted as the cause for the valve-state inconsistency. So, the diagram enabled diagnosing that the CI-sensor is faulty and the conclusion that the valve is open is a correct one.

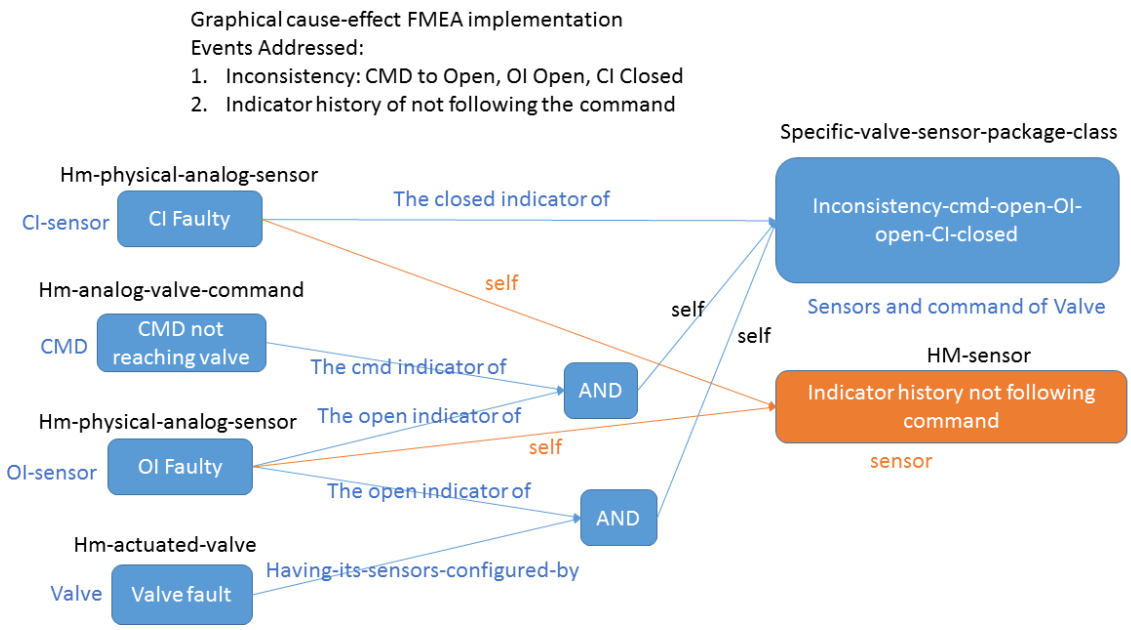

Figure 7. Cause-effect graph for generic reasoning associated with valve anomaly analyses.

There are, of course, a large number of models that may be applied, including strategies to detect leaks. Models may be applied "piece-wise" throughout the domain whenever conditions are appropriate; and additionally, appropriate conditions are established dynamically as needed. "Piece-wise" implies that the autonomous system chooses portions of the domain to conduct reasoning by evaluating constraints and availability of information. The purpose of this paper is not to cover all possible thinking strategies for ISHM, but to present the power of enabling a system to "think" and analyze based on concepts and models that it understands and discovers in the context of the application knowledge model being programmed.

IV. Prognostics

Prognostics can be treated as diagnostics in the context that an indicators of failures may be a prognostics indicator that can predict future failure, rather than assess that failure has occurred. Hence, prognostics, instead of diagnostics analyses strategies and models are used to determine events that indicate future anomalies. The directed graphical cause diagrams are created using prognostics events (future anomaly indicators) rather than diagnostics events (present anomaly indicators). Therefore, the treatment of anomaly detection and diagnostics described in previous sections also apply to prognostics indicators and prognostics analysis.

V. Autonomy

Interest in autonomy continues to steadily grow due to NASA's plans for exploration beyond low Earth orbit. More recently, plans to establish Moon or Mars bases have highlighted requirements for autonomous space habitats (transport and settlement, i.e. Deep Space Transport and Deep Space Gateway), autonomous systems for ISRU, robotics and systems necessary for extra-planetary surface exploration and long term human survival. The goal for these missions is to have systems that can, as much as possible, function without human intervention; with the understanding that some shared-autonomy or assisted-autonomy will still be desired or required.

A report by Carnegie Mellon University "Technology for Autonomous Space Systems (CMU-RI-TR-00-02, September 2002)" [9] presents a survey of the state of the art at that time. The report addresses underlying technologies, component technologies, and space systems. It indicates that "…the preponderance of systems in this survey were directly, albeit remotely, controlled," *not autonomously controlled*. Although the report is from 2002, it describes autonomy content associated with many NASA projects that are currently operational, and others that are still in the planning phase. The report identified NASA Deep Space 1 (DS-1) as the first highly autonomous space mission (DS-1 included the Remote Agent planner that autonomously plans course corrections to achieve goals and deploys or enables science instruments at locations appropriate to the specified science targets)[1].

Autonomy has historically been implemented as "brute-force autonomy" (BFA), as opposed to "thinking autonomy" (TA). BFA attempts to consider all possible cases for decisions and applies strategies to generate solutions offline. The decisions for cases, and only those cases, are incorporated into the software, and the processor simply implements predetermined actions for a correctly predicted case. This method can never be comprehensive since there will inevitably be cases that are not apparent, imagined and/or just missed. BFA has limited reusability and potential for sustained evolution of autonomy, both are critical criteria required for the affordable development of autonomous software. BFA inherently limits the degree of functional autonomy while also significantly increasing the cost to develop and deploy autonomous systems. Thinking Autonomy (TA), in contrast, implies that the system 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 DS-1 Remote Agent (RA) implementation of autonomy is an example of how TA could be achieved, and employs a paradigm similar to that used for the development of the NASA Platform for Autonomous Systems (NPAS), created by NASA SSC.

The future of "true" autonomous operations requires systems capable of independent reasoning so the need for persistent updates is eliminated, human oversight and errors are minimized, but rapid comprehension and action by operators is enhanced. Important developmental gaps include availability of advanced software platforms for intelligent applications; development of standards; formal methods for software validation and verification; ontology and language formulation to enable "thinking," analysis, planning, and operations at high levels of abstraction; and to advance TRL of implementations to support NASA missions.

NPAS is evolving to incorporate "thinking" autonomy. That is, the ability to create and execute plans that encompass autonomous operations addressing multiple missions, including sustainment of the system and achieving of exploration and science objectives. References describing implementation of autonomous operation using NPAS and preceding versions include [10-12]. The autonomous system must apply strategies that are materialized as plans and execution of plans change, and this must be accomplished on-board and in real-time. As with ISHM, it is key to develop an ontology and languages that describe concepts for operation. Ontology may be defined as a set of concepts and categories in a subject area or domain that shows their properties and the relations between them. For example, a fundamental concept in autonomy strategies is the concept of "redundancy." The autonomous system should be aware of every "redundancy" case in the system at any given time. NPAS, for example, understands the following definition of redundant sensors: "2 or more sensors that measure the same parameter are redundant if they are connected to the same flow-section." The reasoning is that the flow-section is filled with the same fluid at the same conditions throughout the flow-section. With this definition, NPAS autonomy navigates the system and discovers every case of redundancy, making it available for use in all strategies for planning.

It is also critical to develop ontologies for every subsystem of a system (e.g. power, propulsion, and other subsystems of a habitat module), as well as for the overall vehicle/habitat manager that enables autonomous operations of the module. These considerations leads to the concept of a Hierarchical-Distributed Autonomy (HAD). NPAS is evolving in this direction, and has demonstrated basic implementation of HAD for an autonomous habitat module test article which involved integration of NPAS as a vehicle manager, and associated subsystems including power and avionics.

VI. ISHM and Autonomy

Previous sections have stated that ISHM and autonomy must be intelligent "thinking" capabilities embedded on-board systems to operate in real-time. The question now becomes, how ISHM and autonomy are integrated with each other. Two functional interactions must be considered: (1) ISHM as a resource to assist autonomy, and (2) autonomy as a resource to assist ISHM. These functional interactions must also take place in the context of particular applications and particular missions that must be achieved by an application. Figure 8 depicts a conceptual software architecture that has guided the development of NPAS.

In Figure 8, the ISHM and autonomy modules are generic. Each module has strategies related to the functional capabilities they provide. The left is a module that encompasses the knowledge domain model of a specific application (e.g. power system). This module includes tools (primarily graphical) to build the model. The right is where plans are created and executed. These are specific plans that define missions to be executed by the application. The module includes tools to create and execute sequences encompassed by missions.

An explanation of interaction among modules follows. The application domain module provides comprehensive information about every element of the application, including for example, specifications, relationships, and operational constraints. The ISHM module implements the capabilities described in the ISHM Section and updates the domain model with health and availability information. The ISHM module also generates relationships among model elements that are useful to autonomy strategies residing in the Autonomy Module. These could be, for example, collections of elements that are redundant, or that can be used interchangeable; or the sets of alternate paths for flow from a source to a sink. The Autonomous Operations Module inquires the Application Domain Model for availability of elements (resources) needed in a plan, or inhibits assessment of health during certain operations within a sequence.

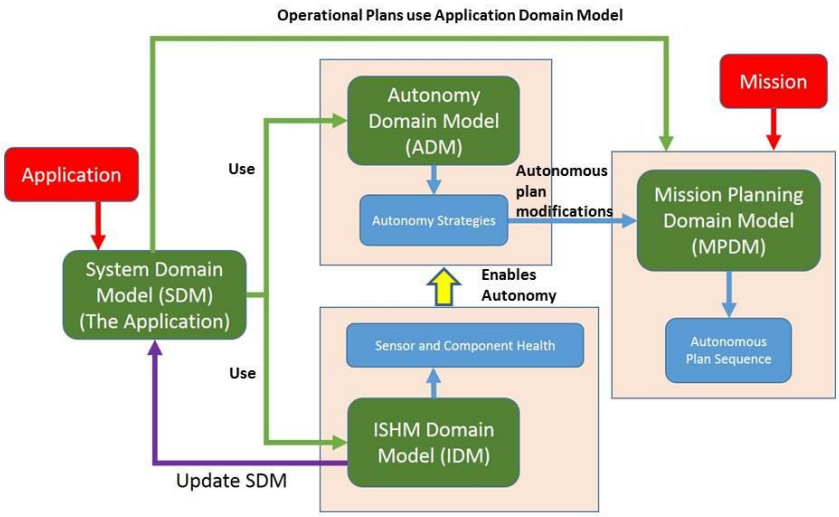

Figure 8. Software Architecture integrating ISHM and Autonomy implemented using NPAS.

The conversation that must happen among modules in the context of autonomous operations requires an ontology and language. NPAS is laying the foundation for this concept, however, this embodiment requires acceptance and development by the autonomous community as standards so that systematic evolution and interoperability of autonomous systems may be possible.

VII. Integrated Awareness

Integrated awareness is about user interfaces that enable the user and developer a comprehensive understanding of the operation of a system. For autonomous systems, awareness interfaces are most important for developers and trouble shooters. The reason behind this statement is because autonomy should require only minimal intervention from

users. This concept is an area that requires substantial research and development. NPAS has been used to develop user interfaces for 3 networked autonomous systems. Primary consideration has been given to show in any system information that relates to interaction with other systems, at a degree of fidelity that is consistent with concepts of operations. For instance, the NPAS vehicle Manager that was developed for integration testing in a habitat test article shows a real-time representation of plans scheduled and being executed by each subsystem of the vehicle. Figure 9 shows the user interfaces for a vehicle manager that was developed. These displays were built in the context of implementation of hierarchical distributed autonomy for a space habitat module. The NPAS system was demonstrated at the Integrated Power Avionics and Software (iPAS) Laboratory at NASA Johnson Space Center in October 2017.

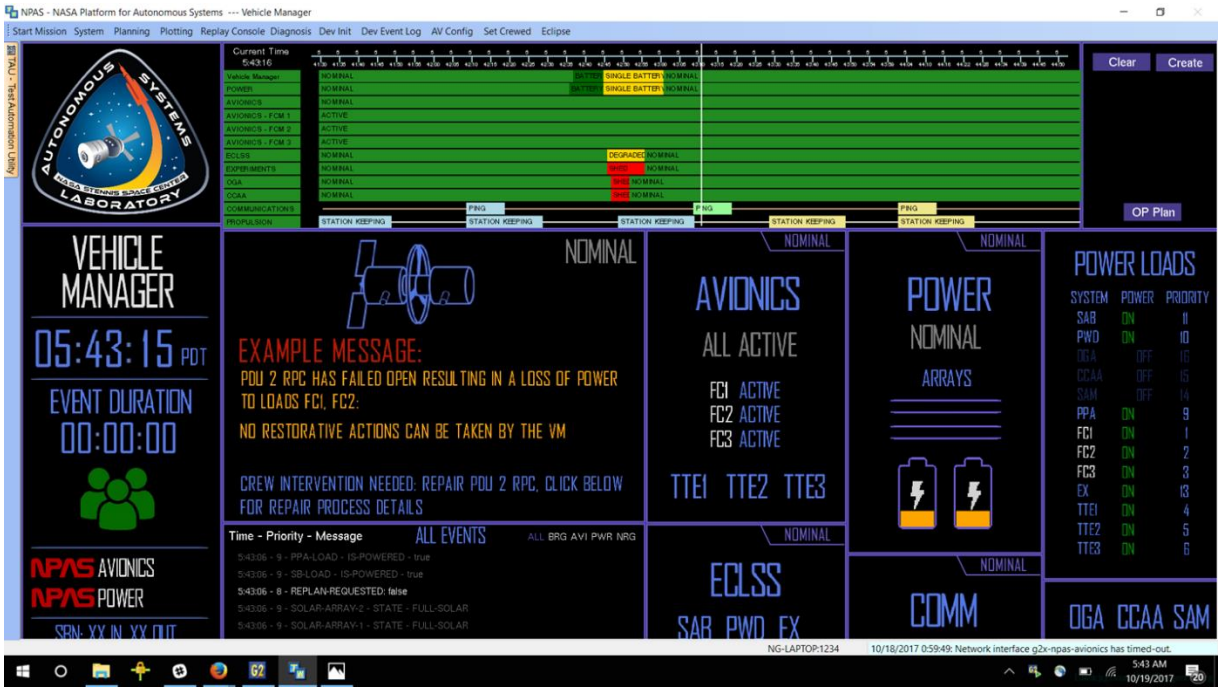

Figure 9. Graphical User Interface of a Vehicle Manager as the member of highest hierarchy of an autonomous space habitat module.

VIII. Conclusions and Recommendations

This paper provides concepts, methodologies, and technologies used in implementation of intelligent autonomy. The emphasis for an intelligent autonomous system is to be associated with the development of "thinking" autonomy as opposed to "brute-force" autonomy. "Thinking" systems require comprehensive domain knowledge models, because comprehensive information and knowledge is needed for analysis required to address a broad range of strategies, and for utilizing a multitude of physics-based models (or other models describing behavior). The paradigm for "thinking" autonomy is being applied by NASA Stennis Space Center with the development of NASA Platform for Autonomous Systems (NPAS) by. NPAS is architected to evolve and incorporate additional capabilities to increment "thinking autonomy." The knowledge models created in NPAS are System Modeling Language (SysML) complete and beyond [13]; these are life models and not merely describing a design, but used in real-time operations.

There are key gaps that limit the advancement of the area of ISHM and autonomy. These include insufficient development of ontologies and languages to enable thinking and reasoning with knowledge models; the lack in most traditional software platforms of tools for creation and management of knowledge models, inference engines, real-time operations, and network enabled interactions; and the lack of standardized concepts of operations consistent with ontologies and languages for autonomy.

IX. Acknowledgements

This work was made possible by funds from NASA's Advanced Exploration Systems (AES). The authors are thankful to Dr. Lauren Underwood from NASA SSC, Center Manager for Advanced Exploration Systems, for her leadership

and support in the projects that have made possible this work, and for improvements to this paper. Our appreciation also to Duane Armstrong from NASA SSC, Chief of Advanced Technology and Technology Transfer for his sustained support. The personnel from D2K Technologies and Kim Wilkins from General Atomics have been involved in the development of NPAS.

X. References

[1] 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.

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

[3] 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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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

[12] 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.

[13] http://www.omgsysml.org/