HabSim-HMS: A Systems Testbed to Investigate Situational Awareness for ExtraTerrestrial Habitation

R Murali Krishnan *, Zixu Zhang †, Kairui Hao ‡, Sreehari Manikkan §, Paul Parsons ¶, Shirley J. Dyke ‖, Ilias Bilionis **
Purdue University, West Lafayette, IN, 47906

Jiachen Wang ††, Chuanyu Xue ‡‡, Song Han §§
University of Connecticut, Storrs, CT, 06269

Mohsen Azimi ¶¶
Mississippi State University, Starkville, MS, 39762

Deep-space habitation presents unique challenges in operating complex control-based systems in distant and disruptive environments far from Earth. The capability to handle unforeseen situations and manage system health is crucial for resilient operations in deep space. Trusted health management strategies operational on the International Space Station (ISS) heavily rely on ground control support. These strategies must adapt to cope with sizable communication delays imposed by the infrastructure limitations or light distances that separate ground control and the habitat. Alternative approaches that increase reliance on artificial intelligence-based onboard autonomy are constrained by limited onboard computing and risks associated with human-system integration. There is an increased need to study the design and architecture of the Health Management System (HMS) of such complex systems to address autonomy challenges for deep-space habitation. This paper presents a model-based HMS testbed developed at the Resilient Extra-Terrestrial Habitats Institute (RETHi) that is part of the HabSim, a System-of-Systems simulator of a Smart Habitat. The testbed is designed to facilitate performance assessment of critical health management functions essential to enable fault-management autonomy, situational awareness, and interface-based mission control. In this paper, we motivate and place the HMS testbed within HabSim, detail the architecture of its

*Ph.D. Student, School of Mechanical Engineering, West Lafayette, IN, 47906, AIAA Student Member, mrajase@gmx.com (Corresponding Author)

†Ph.D. Student, Department of Computer Graphics Technology, West Lafayette, IN, 47906, zhan1575@purdue.edu

‡Ph.D. Student, School of Mechanical Engineering, West Lafayette, IN, 47906, hao55@purdue.edu

§Ph.D. Student, School of Mechanical Engineering, West Lafayette, IN, 47906, smanikka@purdue.edu

¶Associate Professor, Department of Computer Graphics Technology, West Lafayette, IN, 47906, parsonsp@purdue.edu

‖Donald A. and Patricia A. Coates Professor of Innovation in Mechanical Engineering & Professor of Civil and Construction Engineering, West Lafayette, IN, 47906, AIAA Member, sdyke@purdue.edu

**Professor, School of Mechanical Engineering, West Lafayette, IN, 47906, ibilion@purdue.edu

††Ph.D. Student, Department of Computer Science and Engineering, Storrs, CT, 06269 jc.wang@uconn.edu

‡‡PhD Student, Department of Computer Science and Engineering, Storrs, CT, 06269, chuanyu.xue@uconn.edu

§§Associate Professor, Department of Computer Science and Engineering, Storrs, CT, 06269, song.han@uconn.edu

¶¶Assistant Professor, Michael W. Hall School of Mechanical Engineering, Starkville, MS, 39762, azimi@me.msstate.edu

components, and demonstrate the testbed features through an illustrative example. We also discuss the lessons learned in developing this testbed that may be useful for others seeking to solve similar problems.

Nomenclature

AOS	=	Autonomy Operating System
BIT(s)	=	Built-in Test(s)
C2	=	Command and Control System
cFS	=	core Flight Systems Application
CDHS	=	Communication & Data Handling System
EMS	=	Electro-Mechanical System
FDD	=	Fault Detection and Diagnosis
FFM(s)	=	Functional Fault Model(s)
GCC	=	Ground Control Center
GCR	=	Galactic Cosmic Radiation
HCI	=	Human-Computer Interface
HMS	=	Health Management System
ISS	=	International Space Station
ISHM	=	Integrated Systems Health Management
LEO	=	Low-Earth Orbit
MCVT	=	Modular Coupled Virtual Testbed
RETHi	=	Resilient Extra-Terrestrial Habitats Institute
RHC	=	Reference Habitat Concept
SmartHab	=	Smart Habitat

I. Introduction

Sustainable deep-space exploration requires the development of resilient habitation systems capable of operating safely in harsh, resource-constrained environments [1]. Lessons learned from human spaceflight operations on the ISS highlight the challenges of managing risks in Low-Earth orbit (LEO) [2]. Current LEO-based spaceflight operations handle challenges related to human-systems integration risks [3–5], and others unique to the space environment, like the risks posed by increased galactic cosmic radiation exposure (GCR) to crew [6, 7] and onboard electronics [8–10] without loss to mission or crew. As future manned missions extend beyond LEO, reliance on ground control becomes

impractical due to intermittent and delayed communication [11, 12]. Independence from ground control would affect system-level health management of the engineered systems that support them, as active health and state management of mission-supporting systems is primarily managed by ground-control [13].

Increasing onboard autonomy has been recognized as an essential enabler for safe, resilient, and cost-effective mission operations. Onboard autonomy can enhance system-level situational awareness and automate health management in deep-space habitation systems [14], particularly during transitions and dormant operations [15]. To address this, advanced system-level autonomy technologies, such as Integrated Systems Health Management (ISHM), are essential for automating system-level health management and enhancing situational awareness [16].

However, integrating autonomy into complex systems introduces new challenges, particularly the heightened risk of accidents [17, 18]. Bolting on automated control behaviors to aid health management in these complex systems increases the risk of loss and accidents while operating them [19, 20]. Thus, modern systems engineering processes designed for complex control-based systems initiate the development, verification, and validation of ISHM platforms early in the design process [21]. Enabling system-level health management through ISHM technologies requires a system-level overview and requires effort from stakeholders distributed across geographies and expert disciplines. Modern state-of-the-art ISHM platforms follow a modular and hierarchical functional decomposition [22–24]. While many ISHM frameworks have been developed and successfully deployed in various LEO-based aerospace applications, there is limited literature on the requirements for developing system-level health management applications for deep-space habitation. Researchers developing system-level health management frameworks for deep-space habitation do not have access to existing ISHM frameworks as they are typically proprietary. The lack of accessible testbed platforms hinders establishing realistic design, operational, and developmental requirements of increased autonomy needed to support system-level health management required for deep-space habitation.

Recent literature points to the reliance on simulations and testbeds for examining the complex interactions at system-level operations, particularly for handling complex operational vulnerabilitiesin deep-space exploration [25]. For example, NASA's Lunar Gateway project created Gateway in a Box, a low-fidelity model emphasizing software emulation, to conduct flight control simulations. As a next step for flight-readiness tests for the Gateway, Gateway in a Rack was implemented as a medium-fidelity hardware emulation of the Gateway [26]. Simulations provide a low-risk method to test and enhance operational techniques and technologies in the lab before deployment in space. Therefore, this paper introduces HabSim-HMS, an open-source health management system (HMS) prototyping framework within "HabSim" developed at the Resilient Extra-Terrestrial Habitats Institute (RETHi) for studying autonomy and situational awareness challenges in deep-space habitation by emulation through software. HabSim is a System-of-Systems simulator of a Smart Habitat on the Moon. HabSim supports research in evaluation of control effectiveness of safety controls [27], resilient design of habitat subsystems [28–31], and in the development of fault-detection and diagnosis algorithms [32, 33], all of which are part of this virtual collection. HabSim-HMS is designed to study unique autonomy

and situational awareness challenges inherent in deep-space operations. By providing an accessible platform for academic research, HabSim-HMS aims to bridge the gap between industry advancements and academic exploration, fostering a deeper understanding of prototype design processes and enabling the development of robust ISHM platforms.

Herein, we describe the conceptual design of HabSim-HMS and articulate the processes followed to develop its high-level functional requirements. Then, we detail the capabilities of each HabSim-HMS component, offering insights into the systematic approach taken during the framework's design. Through an illustrative example, we explore the features of HabSim-HMS, demonstrating its utility in studying challenges related to autonomy and situational awareness in deep space. The open-source nature and ease of setup of HabSim-HMS make it a valuable resource for researchers, helping them to develop better requirements for a habitat's HMS early in the mission design phase. We also discuss the lessons learned during the development of HabSim-HMS, a product of over four years of research and developmental effort. These lessons may provide valuable knowledge for researchers seeking to develop similar test frameworks for system-level health management in complex systems. Finally, we hope that this paper contributes to the growing knowledge in deep-space habitat design by providing a model-based solution to the challenges associated with early-stage development and testing of an ISHM platform.

II. Architecture of HabSim's Health Management System

Model-based resilience assessment of a system requires system models that capture its performance level during damage propagation and recovery when subjected to disruptive events [34, 35]. These system models should also capture safety controls that can recover system performance from damage-induced losses or accidents [36]. The conceptual architecture and functional modeling hierarchy of RETHi's resilience assessment testbed are derived from a Reference Habitat Concept (RHC). The RHC is an architecture of a habitat system operating on the Lunar surface used to represent the gross level structure of the habitat system and the relationship among its components.

The RHC has an interior environment (IE) protected by a rigid structure (ST) covered with a structural protective layer of regolith (SPL). An Environmental Control and Life Support System (ECLSS) controls the temperature and pressure of the IE. An extra-terrestrial microgrid, the power system (PW), supplies power to all functional components of the habitat that require electric power for operation. These electro-mechanical systems (EMS) work together to meet the operational temperature and pressure set points of the RHC. However, internal or external disruptions may damage components of these systems and degrade the system's ability to meet the temperature and pressure set points based on mission-phase requirements (performance of the habitat). Externally, the RHC is assumed to face operational risks due to seismic activity [37], electro-statically charged dust [38], and micro-meteorite impacts. Internally, the RHC is assumed to experience system performance loss due to component failure and fire-related hazards. Component failures are inevitable during the long operational life of the habitat. Functional loss of systems due to component failures affect the performance of the habitat in an unforeseen manner due to the tight coupling of their physical states.

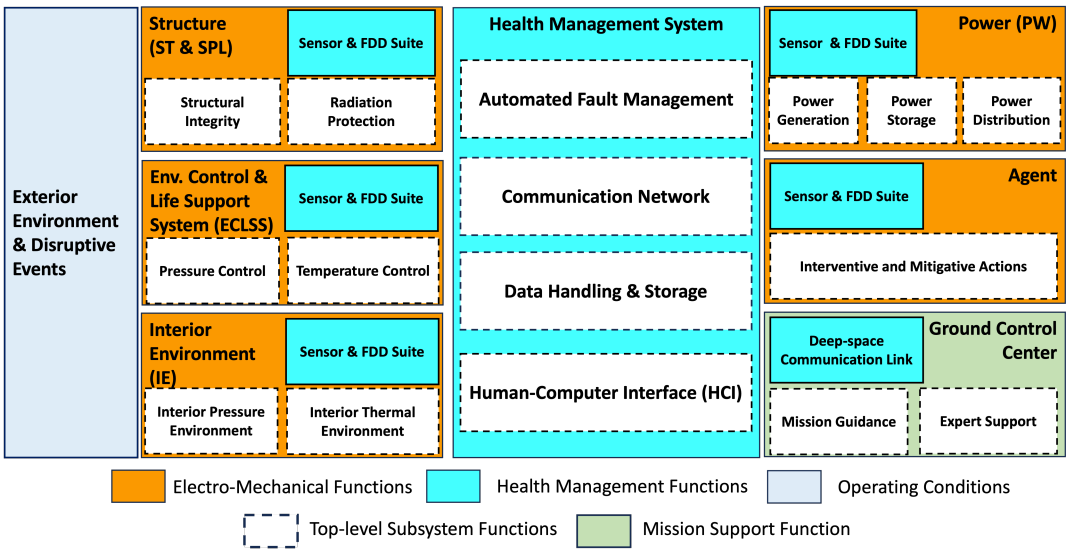

Fig. 1 Reference Habitat Concept (RHC) of the notional SmartHab design modeled in HabSim. HabSim is logically divided into two gross functional families, namely, the Electro-Mechanical functions and the Health Management functions. In this paper, we describe the architectures, standards, data models and tools that we use to implement the Health Management functions in HabSim.

Fire-related hazards are critical to the safety on speca-based habitation, due to the high risk oxygen-rich environment housing electro-mechanical components. Further, fire behaves differently in space due to the micro-gravity environment and controlled airflow, making it harder to extinguish [39, 40]. Thus, it is among the prime concerns for operational risks in inhabited space exploration.

HabSim simulates a subset of the damageability and reparability models of electro-mechanical systems of relevance to smart habitation. Our goal is to outline the components for a comprehensive health management system that can be used to study the resilience of a notional design of the RHC modeled in HabSim. The RHC is conceptualized to be resilient to these external and internal disruptions discussed above. Therefore, it is designed with safety controls that provide passive and active resilience to prevent disruptions-induced performance loss to the habitat [41, 42]. Passive resilience is achieved through the design of the EMS that can maintain the habitat's performance invariant to low-intensity disruptions. Active resilience is achieved through the agent system that can deliver repair to the damaged system component through recovery actions. The agent system modeled in HabSim, when instructed, can perform mitigative and interventive recovery actions to adapt the operation of the habitat system to reach near-nominal performance, providing active resilience to the RHC. Executing these recovery actions consumes resources that are finite and costly to replenish. Thus, the RHC should have a Health Management System (HMS) that enables self-diagnosis of hazards and failures, assesses system-level performance, and engages agents in resource-consuming repair actions.

For system level health management and autonomy, the HMS of the RHC should be able to engage the SmartHab's automated fault management policies through situational awareness and be immune to the risks associated with

human-system integration. Therefore, the HMS should be able to provide a mission control interface to the crew for decision support and a ground control interface for ground-based teleoperations. Classic monolithic health management approaches fail in complex systems because of the explosion of fault-to-effect dependencies that emerge with increasing system complexity. A hierarchical strategy is better equipped to deal with the curse of dimensionality in complex habitat systems [24]. The HMS functions should be able to perform distributed sensing and fault-detection and diagnosis (FDD) for each functional EMS. Enabling distributed state and health awareness of the EMS requires the RHC to have a synchronized Communication and Data Handling Service that collects and stores measurements and telemetry from the sensing systems. Finally, an Earth-based Ground Control Center can provide ground-expert or ground-based-teleoperations support for the habitat. The ability to control the state of the habitat from GCC is essential for resilient habitat operations, particularly in uncrewed habitat operations (dormancy & transitions phase). The conceptual features of the RHC are visualized in Figure 1 and they drive the functional requirements of the system models required to develop HabSim, a simulator for the RHC.

A. Modeling the Reference Habitat Concept in HabSim

RETHi's HabSim is a 1/5th scaled system-of-systems simulation model of the RHC described above. HabSim comprises habitat system models that describe their transition from nominal, hazardous, or accident states depending on the disruptions or recovery actions they are subjected to. HabSim models three groups of functional components of a resilient SmartHab, namely F2) Electro-mechanical systems, F3) Agent System, and F4) Health Management Systems. To study the emergent behavior of the habitat system under external and internal disruptions, HabSim also models the exterior environment and loss-causing disruptions (F1), which are exogenous to the habitat system and follow stochastic processes. They form the basis of HabSim's functional modeling hierarchy, as shown in Figure 2.

Details of the Electro-Mechanical Systems (EMS) and disruptions are discussed in depth in Mohsen et. al [43], and the Agent System is discussed in depth in Krishnan et. al [44]. In this paper, outline the functional components of the Health Management System (HMS) of HabSim. HabSim-HMS models the system functionalities required to facilitate decision-making for health management in HabSim. HabSim-HMS supports automated fault management, provide a mission control interface for the crew, and can support simulations for ground control-based deep-space mission-support strategies. Simulating ground control-based mission support strategies is essential for studying the resilience of the habitat system during uncrewed operations and for testing mission handling strategies for missions where communication delays are significant. To achieve these functionalities, HabSim-HMS has the following top-level functional components: 1) Distributed Sensing, 2) Communication and Data Handling Service, 3) Command and Control, 4) Human-Computer Interface System, and 5) Ground Control Center.

All decision-making functions of HabSim-HMS require sensor measurements and prediction from the subsystem-level FDD modules. Built-in Tests (BITs) are defined on these signatures to aid fault isolation, a critical functionality for

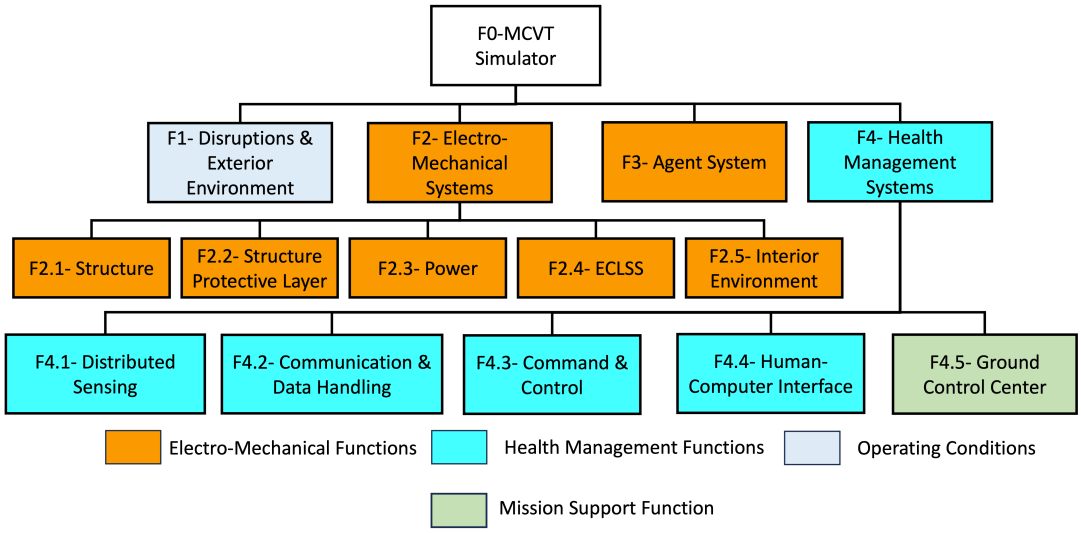

Fig. 2 Function decomposition of the RHC modeled in HabSim. The functional components are divided into four categories: 1) Exterior Environment and Disruptions, 2) Electro-Mechanical Systems, 3) Agent System, and 4) Health Management System. The Health Management System is the focus of this paper.

system-level fault management. Sensors, FDD modules, and built-in tests (BITs) provide awareness of the performance level of each damageable component of the habitat model in HabSim. These sensing modules are distributed throughout HabSim-EMS.

The Communication and Data Handling Service (CDHS) collects data from the distributed sensing component, transmits it in real-time through a communication network, and stores the information in a data repository. The information stored in the data repository should be accessible to fault management and interface applications through a data handling service. The data handling service of the CDHS can handle data requests from multiple health management applications concurrently.

Health management of the habitat system is facilitated by automated fault management and a human-computer interface. Automated fault management should diagnose system-level faults and schedule agent systems for resource-consuming fault recovery actions. HabSim-HMS uses dependency-matrix based FDD algorithms to diagnose system-level faults and schedule recovery actions. During crewed operations, the human-computer interface (HCI) provides a mission control interface to the crew to enhance their awareness of mission operations and a control interface to program low-level system controllers of the habitat. The HCI should visualize state predictions of the habitat under predictive decision scenarios to aid the human expert in making informed decisions through the control interface.

Finally, the CDHS should facilitate the transmission of habitat sensors and FDD measurements to an Earth-based Ground Control Center (GCC). Earth-based GCC receives sensor and FDD measurements from the habitat through a communication channel with limited bandwidth and realistic communication delays, which is essential to consider and study any system health management strategies involving the GCC. GCC should be able to store the telemetry data from

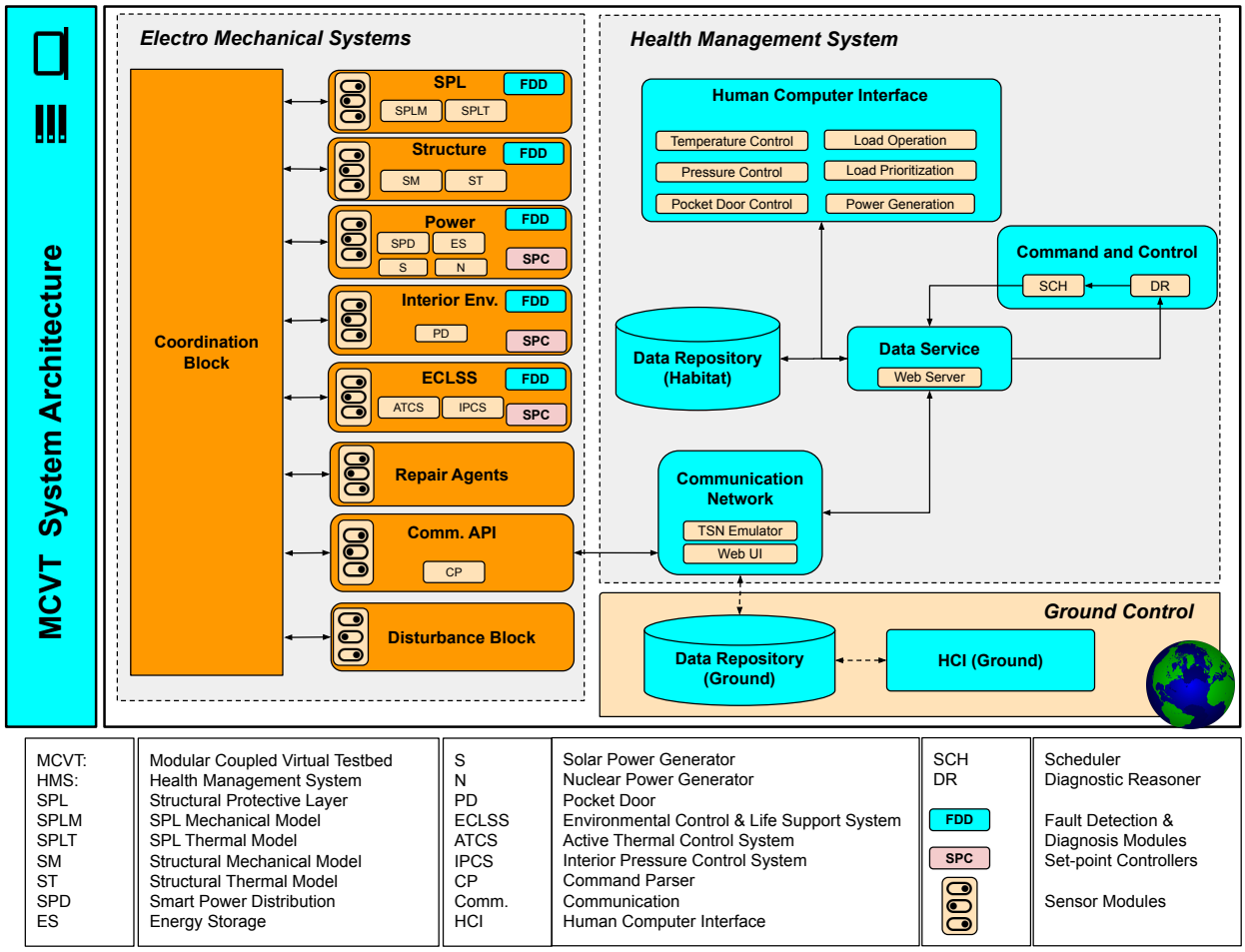

Fig. 3 Architecture of the HMS Testbed in HabSim. In the HMS Testbed within HabSim, the FDD and BIT modules are distributed throughout the HabSim-EMS. The CDHS collects and transmits data from distributed sensors, storing it in a data repository for the HMS to diagnose faults and schedule recovery. The HCI provides a mission control interface for the crew and facilitates data transmission to the Earth-based Ground Control Center (GCC), considering bandwidth and communication delays.

the habitat and provide an interface to ground control experts to provide a mission control interface like that aboard the habitat. The concrete architecture of HabSim-EMS and HabSim-HMS is shown in Figure 3. In the following sections, we explain the detailed architecture of each HabSim-HMS function and describe the capabilities they enable in HabSim.

III. Capabilities of HabSim-Health Management System

HabSim-HMS was developed with capabilities to perform realistic model-based performance assessments of system-level health management strategies for control-based complex space habitation systems. The HabSim-HMS components are a coupled suite of software systems developed to meet the functional requirements specified above. These functions consist of models that emulate system behavior and functional components integrated using Docker-based micro-services architecture, a popular software integration framework for networked systems [45]. As shown in Figure 3

& 2, for performing health management onboard the habitat, we have A) Distributed sensing system, B) Communication and Data-Handling Service (CDHS), C) Command & Control (C2) system, and D) Human-Computer Interface system. We reuse the HCI software and parts of CDHS for GCC-related health management support functions, operating on delayed telemetry information from the habitat's distributed sensing system. This section discusses the detailed architecture of each of these HabSim-HMS components.

A. Distributed Sensing System

Monitoring the state of HabSim-EMS is essential to estimate their operational condition and manage their health. These systems are embedded with a comprehensive suite of sensor and FDD components for observing and understanding the temporal evolution of their physical states. Developing realistic sensing and FDD components for each subsystem behavior can be resource-intensive, particularly during the early design studies for HMS. However, the sensor and FDD modules' performance plays an essential role in characterizing the performance of the HMS. To address these challenges, we develop synthetic sensor and FDD component models that enable us to control their measurement and estimation capabilities and assess their impact on system-level health management. These synthetic models can also simulate damage, an operational risk for control-based systems. Synthetic sensing modules streamline the integration process to system diagnostic tests, which drive system-level diagnostics and decision-making. The diagnostic tests identify exceedances, alarms, and other measurement-processing transformations that diagnostic engines can use at run-time [46]. Integrating a sensing suite in each EMS component of HabSim-EMS is pivotal for providing situational awareness regarding the health condition of the habitat. Comprising a combination of sensors, FDD modules, and Built-In Test (BIT) modules, this suite is embedded within each EMS in MATLAB Simulink to furnish HabSim-HMS with the necessary measurement and telemetry information.

In the following sections, we first describe the synthetic models of sensor and FDD components used for monitoring the states of the EMS models in HabSim. Then, we define the BIT modules used to inform the run-time diagnostics engine of HabSim-HMS.

1. Model-based Sensor Modules

Sensor modules are statistical models that produce noisy measurements of relevant physical quantities to help monitor and control subsystems. Each sensor module fulfills three additional functional requirements for simulating sensor malfunctioning. These are simulating sensor noise and continuous drift due to aging, simulating sensor failures due to disturbances, and interacting with agent models to replace the sensors.

First, we describe the modeling of sensor noise and drift at time t. We use the function $x(t)$ to represent the time evolution of a physical quantity to be measured. The sensor measurement is a noisy version of $x(t)$ by adding sensor measurement noise and sensor drift. Sensor measurement noise is an additive scaled white noise process signal $\sigma_y Z(t)$,

where $Z(t)$ is the standard white noise process and σ_y is the scaling factor. Sensor drift is a continuous random process $X(t)$ that degrades the sensor performance by increasing output bias. We model the drift using stochastic differential equations $dX(t) = at^k dt + \sigma_d dB(t)$. This equation assumes that the drift magnitude follows a power function in time. The notation $B(t)$ stands for the Brownian motion and σ_d is the diffusion coefficient for the senor drift. Adding the Brownian motion term allows the simulation of a random drift. At time t, the sensor measurement $Y(t)$ is the random variable defined by $Y(t) = x(t) + X(t) + \sigma_y Z(t)$.

Next, we explain the modeling of sensor failures. We use the binary stochastic process $X^h(t)$ to represent the health state of a sensor. $X_t^h = 0$ means the sensor is functioning properly at time t and $X_t^h = 1$ means the sensor is not outputting any measurements. We consider two mechanisms that can flip the value of $X^h(t)$ from 0 to 1. The first one is the random equipment failure that follows the Weibull probability distribution and the second one is due to deterministic external disturbances, such as fire. We use the random variable Time to Failure (TF) for the first mechanism to characterize the random equipment failure. The TF random variable follows the Weibull density with the scale parameter λ and the shape parameter k. We can use the mean time to failure to determine the values of λ and k.

The indicator function $\delta_1(u)$ models the deterministic external disturbance for the second mechanism. The sensor health state $X^h(t)$ will flip from 0 to 1 if $\delta_1(u) = 1$. In the last functional requirement, the agent system can interact with the developed sensor models by replacing the faulty sensors. The agent system can model faulty sensor replacement by resetting the sensor drift $X(t)$ to 0 and flipping the health state $X^h(t)$ from 1 to 0.

2. Model-based Fault Detection and Diagnosis Modules

The role of an FDD component is to infer the underlying damage state from the available sensor data. Estimating the damage state can be done by comparing the sensor data to data from the nominal condition or fault states (data-drive) or using a physical model of the system (not necessarily a high fidelity one) and statistically estimating the damage state, e.g., Kalman filters, hidden markov models. Developing such FDD components typically requires domain knowledge, statistical expertise, and experimental data from the actual system of interest.

In this work, we simulate the behavior of FDD components without loss of generality; we can think of an FDD component as a computer program that returns the probability of damage, given the sensor data. FDD components estimate the probabilities of system component damage severity. Each system component has a deterministic categorical health state variable s that takes a value from the set $\{s_1, \cdots, s_n\}$ representing n levels of damage severity. We use s_1 and s_n to denote minimum and maximum damage, respectively. The FDD model is a statistical model that treats the categorical health state variable s as a random variable S. At each time t, the synthetic FDD maps s to a multinomial random variable that represents the probability mass function $P = (P_t [S = s_1], \cdots P_t [S = s_n])$ for S. The probability mass function P is an input parameter to a simulated FDD, so it can be set to different values for characterizing the fidelity of an FDD. We also add a white noise process to P to simulate FDD estimation fluctuation.

3. Built-in Test Modules

BITs identify off-nominal trends in the measurement signatures of the synthetic FDD components for each HabSim-EMS. The synthetic FDD modules compute the likely damage severity of a disruption-induced failure mode to a system component. For example, if a component has a health state with a health state with n damage severity levels, it computes $P_t\ (S = s_1, S = s_2, \cdots, S = s_n)$. We use these synthetic FDD outputs to estimate the expected damage severity level,

$$\mathbb{E}[S_t \mid \text{FDD measurements}] = \sum_{i=1}^{n} s_i P_t(s_i) \tag{1}$$

In HabSim-HMS, exceedance tests on the estimated damage severity levels constitute the system BITs, similar to the Limit Checker in NASA's cFS application [47]. The results from the synthetic FDD modules fluctuate with time, as the estimation mechanism is modeled as a stochastic process. Hence, they are smoothed using a discounted-averaging procedure controlled by a parameter, γ, before being fed into the BIT mechanism. The "smoothed" estimate of the damage severity level is calculated according to this recursive procedure,

$$\hat{s}_t \leftarrow \hat{s}_{t-\delta t} + \gamma(\hat{s}_{t-\delta t} - \mathbb{E}[S_t \mid \text{FDD measurements}]) \tag{2}$$

BITs check if the estimated damage severity levels are below the user-defined threshold and produce a binary-valued output. If the damage severity level is below the threshold, the BIT returns a "Good" value; otherwise, it returns a "Bad" value. These test values help to isolate faults observed in HabSim-EMS as they face failure-related performance loss due to disruptive events.

B. Communication and Data Handling Service (CDHS)

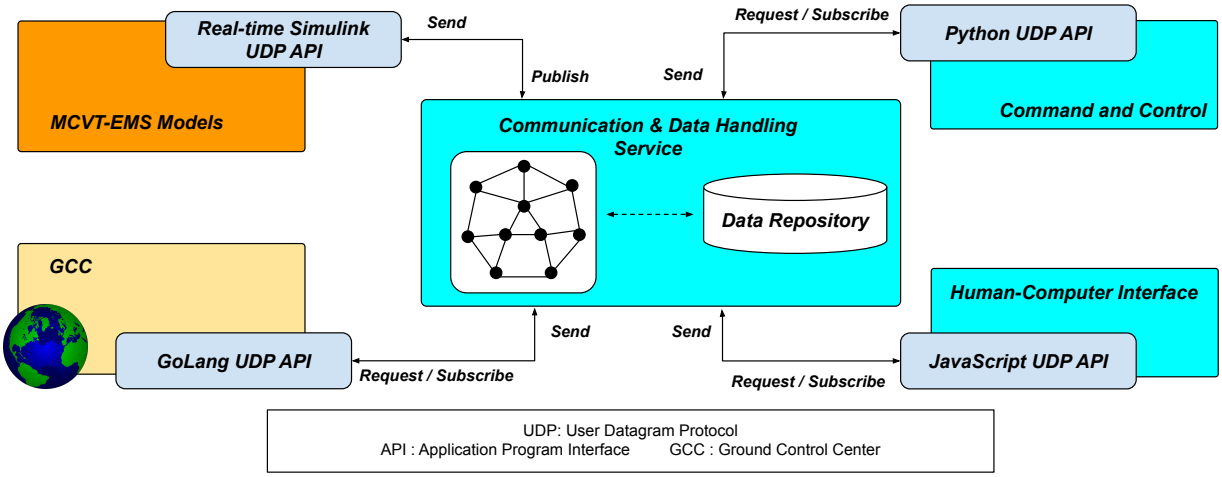

Fig. 4 Architecture of the Communication and Data Handling Service of HabSim-HMS. The CDHS is responsible for managing the data exchange requests among the HabSim subsystems and the data repository.

CDHS supports real-time and reliable data transfer among networked subsystems in HabSim (Figure 4). To meet the data flow requirements within HabSim-EMS and between HabSim-EMS and HabSim-HMS, adequate bandwidth must be allocated for individual subsystems to satisfy their data rate requirements. For example, the communication between HabSim-HMS and HabSim-EMS requires a minimum bandwidth of 445.3 kbps. The communication within HabSim-EMS requires a relatively larger bandwidth of 4.7 Mbps because of the higher frequency (200 Hz) sensor data required by local controllers of the ECLSS model. Network-wide time synchronization must also be maintained with a synchronization error no larger than 100 nanoseconds. Ultra-high reliability and bounded low latency must be guaranteed for data transfer among subsystems.

1. Ethernet-based TSN Emulator

Traditional wired communication technologies designed for mission- and safety-critical systems are mostly bus-based or Ethernet-based. The widely used CAN bus communication system in the aerospace industry can, at best, provide 5 Mbps data throughput with the newer CAN-FD standard, which barely meets the bandwidth requirements in HabSim. Another popular communication network in avionics applications is the Time-Triggered Ethernet (TTEthernet) [48], which is improved from the Avionics Full Duplex Switched Ethernet and has been developed as the communication system in some space projects, e.g., NASA's Orion Multi-Purpose Crew Vehicle and Ariane 6. However, the fixed slot-based scheduling for individual packets at each node in TTEthernet leads to considerable delay and is not flexible enough to adapt to network configuration changes. To design the communication network subsystem for HabSim, the feasibility of applying Time-Sensitive Networking (TSN) [49] technologies is investigated to support ultra-high-speed real-time and reliable communications among subsystems in HabSim. TSN provides several essential features for mission-critical applications, such as bounded latency, global time synchronization, independence from physical transmission rates, fault tolerance without additional hardware, and the interoperability of the solutions from different vendors. Further, existing TSN protocols are mostly based on the Ethernet standard; it automatically scales with the Ethernet, meaning that the bandwidth or other performance criteria of the technology are not restricted. Based upon the feasibility study, a TSN-enabled communication network simulator (Figure 5) is developed to serve as the communication infrastructure within HabSim-EMS and between HabSim-HMS. Each TSN switch is equipped with multiple input/output gates and multiple priority-based packet queues, and thus can guarantee the timing behaviors of critical traffic while maximizing throughput of regular traffic. There are three main components of the TSN switch: (1) **packet classifier** that classifies packets based on their traffic class and priority and passes them to the corresponding queues in the scheduler; (2) **packet scheduler** that schedules the packets in queues according to a certain policy; (3) **packet router** that routes packets to its destination. In this Emulator, all these critical components are implemented along with some selected TSN protocols, e.g., IEEE 802.1Qbv Time-Aware Scheduler for real-time packet scheduling, Dijikstra's shortest path routing algorithm and IEEE 802.1CB Frame Replication and Elimination for Reliability for

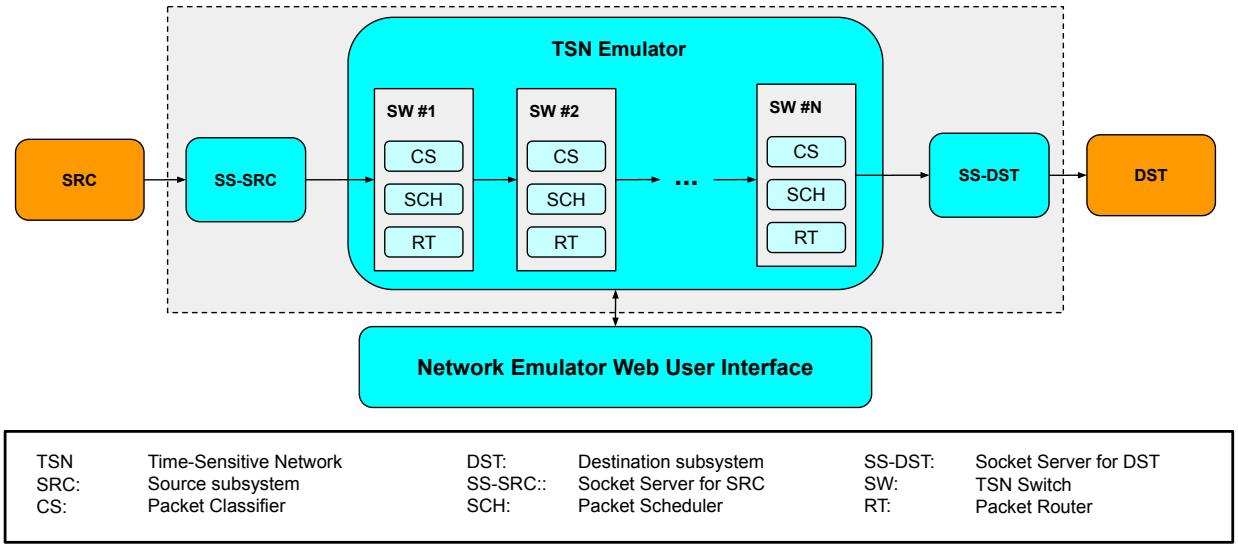

Fig. 5 Data-flow through the communication network emulator built on Ethernet-based TSN technology

reliable packet transmission. The emulator's core (packet processing and forwarding functions) is developed in Go language, and the Web-based user interface for network management is built on Vue.js.

2. Data Handling System

The data handling system manages the data exchange requests among the HabSim subsystems and the data repository. As shown in Figure 6, four types of tables in the SQL-based database are connected by a predefined unique `DATA_ID`. The `META` table specifies the meta information for each data stored in the database. It creates a unique `DATA_ID` for each measurement signature, which is key for writing or retrieving data and building data relationships. The data handling system requires that all incoming data be registered in the `META` table before storing them in the database. The `RELATIONSHIP` table maintains the relationship among data. The relationship is extracted from the dependencies among data flows in HabSim. The `INTERACTION` table describes the relationship among HabSim-EMS. The information stored in this table augments the `META` and `RELATIONSHIP` tables with information from HabSim's design structure matrix. Lastly, `RECORD` table stores the time-series data with a primary key `SIMULINK_TIME` generated by HabSim-EMS. Furthermore, the physical time (wall clock) when data is transmitted from subsystems is also recorded to ensure synchronization between HabSim-EMS and HabSim-HMS. Each record table corresponds to one data flow, and the number of columns for each record table is determined by the `DATA_SIZE` information stored in the meta table.

The data handling system provides reliable and easy-to-use back-end Application Programming Interfaces (APIs) to facilitate data transfer and storage. The service APIs are developed for Go, Python, JavaScript, and Simulink languages to facilitate data exchange between HabSim subsystem models. The data handling system provides four basic data-exchange operations in HabSim, namely `SEND`, `REQUEST`, `SUBSCRIBE`, and `PUBLISH` [50, 51]. The `SEND` and

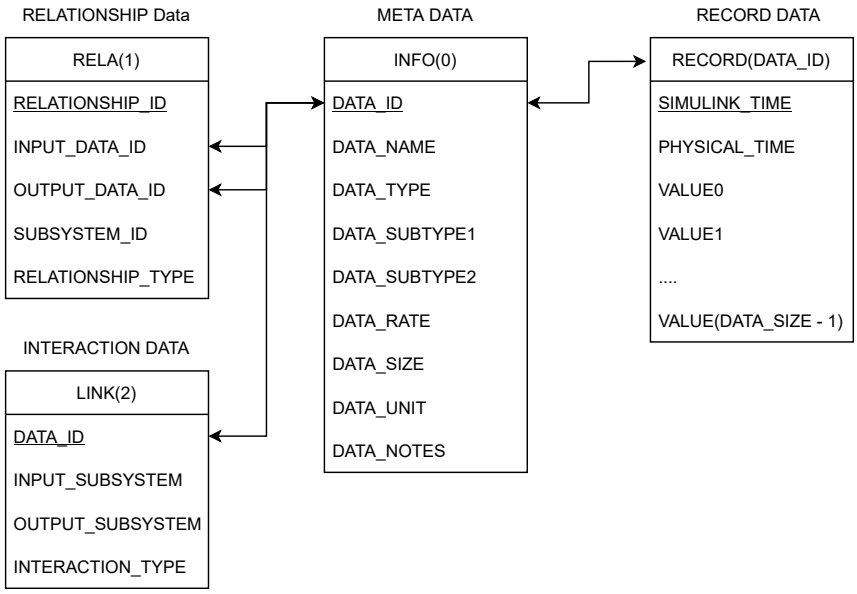

Fig. 6 Schema of the data repository

`REQUEST` operations are based on the client-server paradigm typically used for event-triggered communications. To achieve lower latency, `SEND` and `REQUEST` operations are implemented in UDP protocol at the cost of less reliant data transfer at the application layer. `PUBLISH` and `SUBSCRIBE` use a specialized messaging paradigm where publishers do not program the messages to be sent directly to specific receivers. Instead, published messages are categorized into classes, facilitating consistent data exchange to a subset of subscribers. This messaging paradigm is used for the HabSim-HMS applications where the subscribers require continuous data flow with stable frequency from a publishing source.

C. Command & Control (C2) System

The C2 automates decision-making for scheduling mitigative actions required to actively manage a failure in HabSim-EMS. It isolates the failure experienced by HabSim-EMS using model-informed diagnostic reasoning and schedules failure-mitigating actions based on user-defined heuristics. It commands the agent system to initiate a mitigation action program that consumes resources. It uses the `SUBSCRIBE` data handling API to access telemetry for running the diagnostic reasoning application and the `SEND` API to communicate with the agent system.

As shown in Figure 7, HabSim's C2 has two functional components: the Diagnostic Reasoner and the Heuristic Scheduler. The Diagnostic Reasoner isolates faults using information derived from the system-level functional fault model (FFM) of the HabSim-EMS. As a hazard warning implied from telemetry may have many causes, we follow user-informed heuristics to prioritize mitigative actions toward potential causes. The Heuristics-based Scheduler operates according to a user-provided quality metric for each mitigative action informed by resource constraints. The

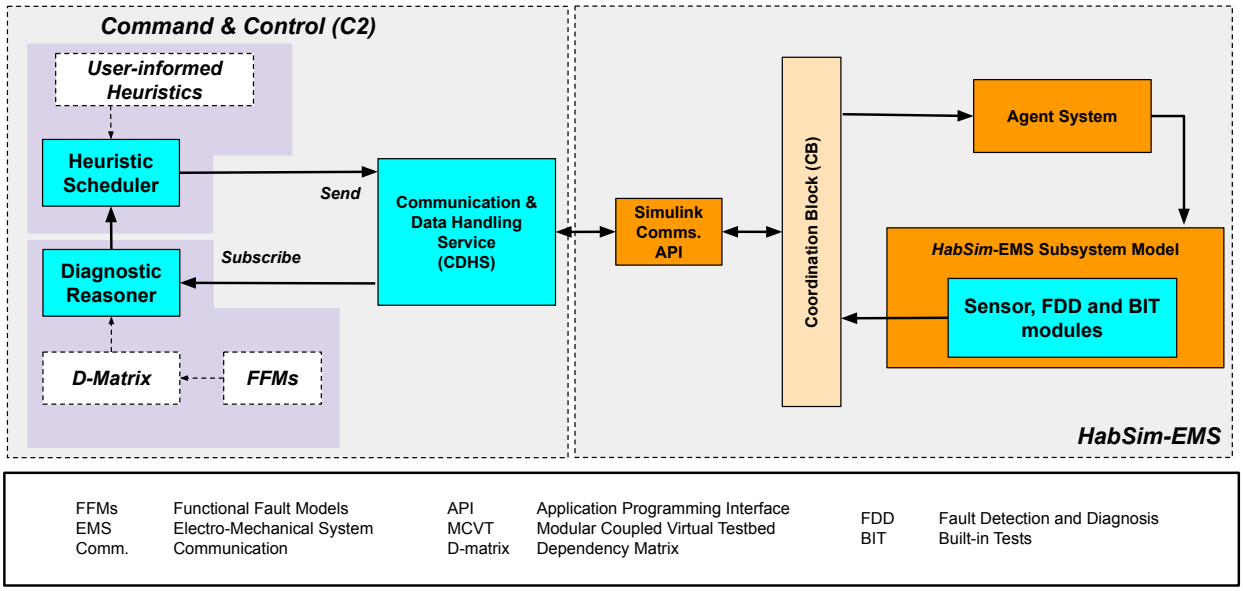

Fig. 7 Data-flow through the Command and Control system. The Diagnostic Reasoner isolates faults using the system-level functional fault model (FFM) of the HabSim-EMS, while the Heuristic Scheduler operates based on user-provided quality metrics and resource constraints. Mitigative actions are executed by an agent system, which performs the sequence of sub-actions for overall fault mitigation. The Command and Control system communicates the intended mitigative-action program to the agent system using the `SEND` data-handling API.

mitigative actions are carried out by an agent system [44], which executes the sequence of sub-actions to perform the overall fault-mitigative action, an active safety control. Once C2 decides the mitigative action to be executed by the agent, it communicates the intended mitigative-action program to the agent system using `SEND` data-handling API. As the agent system delivers repair to the affected HabSim-EMS components, the performance of the modeled habitat reaches near-nominal levels. This section will explore the reasoning-based fault-isolation and heuristics-based policy creation for mitigating actions.

1. Diagnostic Reasoner: Automated Fault Isolation

Reasoning technologies enhance system-level health management in complex systems [52]. Fault management in complex engineering systems requires reasoning out the potential failures that cause anomalous telemetry signatures. Automating fault management typically relied on rule-based expert systems. More recent approaches derive reasoning behaviors using domain knowledge, schematics, failure, and criticality information to enable automated onboard fault management e.g., Orion [53] and Gateway [54–56]. These failure cause-effect models, called functional fault models (FFMs), represent the causal effect of a failure on a system's observations (sensors, FDDs, and BITs). FFMs are a fault management technology used in ISHM applications of complex aerospace systems for real-time diagnostics and design analysis [57, 58]. Further, FFMs can derive the fault-to-effect dependency matrix (D-matrix), which informs onboard real-time fault diagnosis reasoning applications [59]. We adopt the diagnostic reasoning algorithm implemented in

NASA's Autonomy Operating System (AOS) [60]. C2's diagnostic reasoning application implicates the potential failure modes that caused BITs to "`Fail`". If a subsystem BIT produces a "`Fail`" outcome, the reasoning algorithm implicates at least one identified failure mode of the habitat model. If the BIT outcome is "`Pass`", it is additional evidence to exonerate a failure mode "`Pass`" -ed by all its child observations.

2. Heuristics-based Scheduling

Once faults are identified, the C2 decides whether to schedule agent systems to initiate failure mitigative actions that restore system performance. The agent system embedded in HabSim is equipped with policy functions specifically designed to address and rectify performance losses resulting from identified faults. This includes the ability to prioritize and execute various fault-recovery procedures. Hierarchical temporal networks are widely adopted for automating functional workflows in space-based aerospace assets using goal-based AI planning and scheduling technologies. Planning systems that automate the scheduling of agent actions to achieve the goal states have been instrumental in space robotics [61, 62]. Recent successful space-robotics missions have enabled extraterrestrial exploration beyond the scope of traditional orbital missions [63]. While AI-based scheduling has become ubiquitous in space (and terrestrial) robotics development, the realm of automated decision-making for system health management has not seen parallel advancements [64]. Drawing upon their extensive experience, ground control experts traditionally assume the responsibility of making system-level health management decisions. They effectively manage risks for engaging in mitigative actions while facing operational risks.

In HabSim-HMS, C2 automates the decision related to fault management through user-defined heuristics for scheduling repair actions. This feature empowers users to assign priorities to various fault-recovery procedures that the agent system may engage in, depending on the nature of the disruption scenarios and the operating levels of the system. This capability allows for in-depth studies related to sizing and design considerations, as evidenced by works such as those cited in literature [65, 66].

D. Human-Computer Interface System

HabSim-HMS's HCI component provides a mission control interface to HabSim, enabling an operator to provide supervisory control to HabSim-EMS. Operators can monitor the health state of various critical HabSim-EMS components, such as the dome structure, power generators, thermal and pressure controls, and the agent system. Operators are also presented with a control interface to specify performance requirements (set-points) for the local temperature and pressure state control loops modeled in HabSim-EMS. Since HabSim is complex and highly coupled, changing local control performance affects the overall habitat performance due to other seemingly unrelated systems In HabSim, local state control loops require power to maintain changing performance requirements. HabSim's HCI provides model-based predictions of the power system states that the operator needs to be aware of when using the control

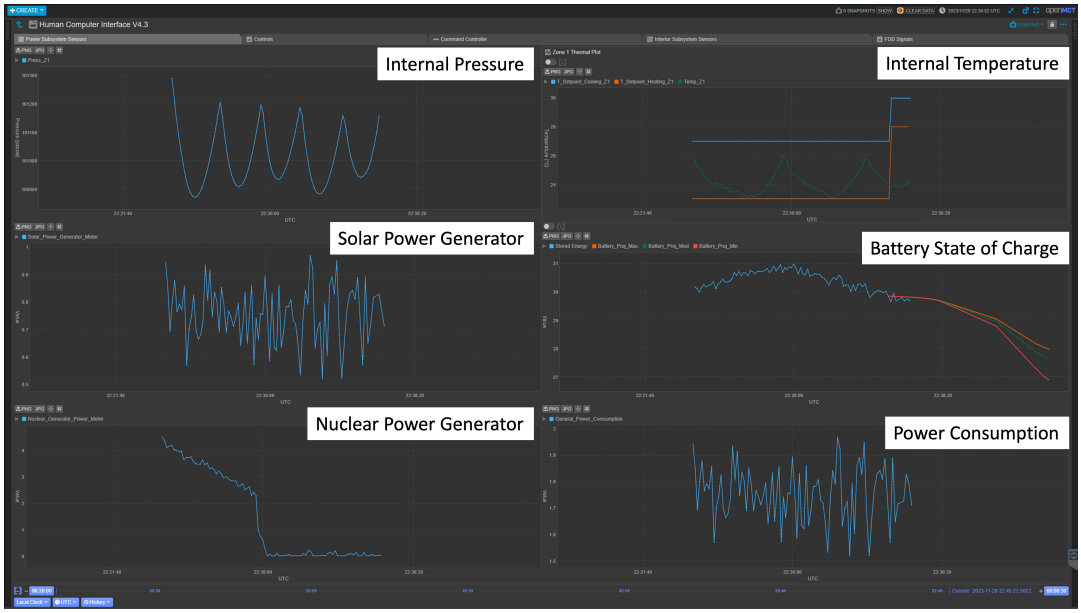

Fig. 8 Dashboard Layout of HabSim's HCI

interface. HabSim-HMS collects the telemetry and exchanges data with the HCI through the CDHS.

1. Mission Control Interface

HabSim's HCI was designed and built on OpenMCT, an open-source web-based mission control platform [67]. Open MCT has broad end-user device compatibility, supports multi-user collaboration, and is highly customizable depending on the use case. Scalable service delivery to many decision terminals can potentially shorten the handover period between crew members through mobile-based remote monitoring interfaces.

OpenMCT represents data and views as objects that can be composed differently. In OpenMCT, telemetry points can be composed into plots, tables, and other views. Views can be composed to create layouts of multiple display elements sized and placed by the user. Data can be displayed with rich graphical views such as plots and imagery. Composition uses direct and intuitive drag-and-drop operations. Last but not least, the core functionality of OpenMCT can be customized with plugins to support the specific needs of missions across multiple domains. By leveraging these OpenMCT capabilities, HabSim's HCI can benefit from a robust and established software framework for dashboard development.

2. Dashboard layout

The HabSim-HMS HCI dashboard is a central hub for operators to monitor, analyze, and control HabSim-EMS. The primary display layout (Figure 8) shows critical system telemetry, providing awareness of the system-level performance of HabSim-EMS. The current dashboard design of the HCI provides a comprehensive view of the telemetry data coming from HabSim-EMS. The layout is organized into distinct sections, each dedicated to a subsystem. Control widgets are

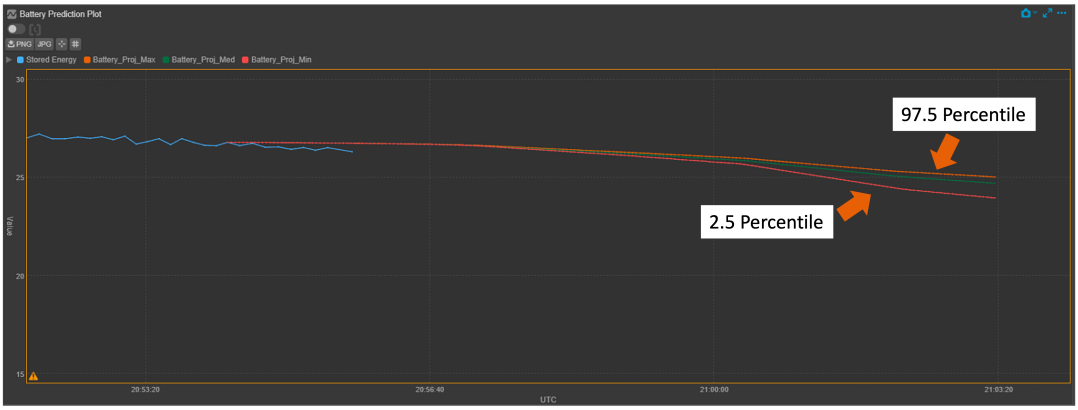

Fig. 9 Predictive uncertainty is overlayed on the state of battery charge telemetry using reduced-order model

incorporated into the dashboard design, allowing users to interact with and manipulate set points of HabSim's ECLSS model. They enable users to adjust the interior temperature and pressure within the habitat. Further, it enables the control of HabSim's pocket door in case of fire-related hazards. The pocket door can be engaged to isolate the spread of fire to a confined zone in the habitat. These control widgets ensure that users quickly understand and interact with them to effectively manage the habitat's environment.

3. Reduced order model for state predictions

A reduced-order model of the HabSim-EMS was created to forecast the energy consumption of stored energy during the meteorite impact scenario. The model employs first-order state space models of power generation, energy storage, power consumption, and interior environment systems. The thermal dynamics of the interior environment are modeled using a lumped RC model. An ON/OFF controller logic is employed for thermal management. The controller can operate in three modes: "heat", "cool", and "auto". In "heat" mode, cooling does not happen when the temperature of IE goes beyond the cooling set point. Similarly, in "cool" mode, heating does not happen when the temperature of IE drops below the heating set point. In "auto" mode, heating and cooling happen as needed to maintain the IE temperature between the heating and cooling set points.

The model takes the initial time, initial values of interior environment temperature and other states, current operating mode, and initial heating and cooling set points. A probabilistic modeling approach is used for the reduced order model to represent uncertainty in the model parameters using probability distributions. HabSim simulation data is used to calibrate the mean and variance of the parameter distributions. The energy consumption prediction is performed through a Monte Carlo sampling approach. Parameters are jointly sampled from the distributions, and energy consumption prediction is done for each sample of the parameter set. Using the set of energy consumption predictions, the forecast is done with quantified uncertainty. The forecast contains median energy consumption (green line in the plot), 0.025 (red line in the plot), and 0.975 (orange line in the plot) quantiles (as is shown in Figure 9).

IV. Exploring Testbed Features Through an Illustrative Example

This section discusses a practical demonstration of HabSim-HMS, showcasing its capabilities through a detailed illustrative example. This approach allows us to illustrate the complex coupling between the HabSim subsystems and the robustness of its health management strategies under simulated space habitation scenarios. By walking through a specific, hypothetical situation, we aim to provide a clear, real-world context to the theoretical constructs and technical specifications discussed earlier. This narrative not only serves to illuminate the practical applications of HabSim-HMS but also highlights its adaptability and resilience in the face of typical challenges encountered in extraterrestrial environments. The ensuing example is designed to encompass a range of functionalities, from system initialization and parameter setting to real-time monitoring, fault detection, and automated response mechanisms, offering a comprehensive view of HabSim-HMS's operational effectiveness and versatility.

The scenario chosen for this demonstration involves a meteorite impact occurring 200 seconds into the simulation, directly striking the nuclear power generator of the space habitat.

A. Simulation Environment and Setup

HabSim-EMS runs in MATLAB Simulink, and HabSim-HMS runs in a Docker container as a networked application with different images. To simulate a damage scenario, configure the HabSim-EMS models using HabSim's configuration files. We then `docker-compose` the HabSim-HMS docker application that builds the HabSim-HMS application. Once the HabSim-EMS model runs in Simulink, it generates subsystem telemetry processed by the waiting HabSim-HMS application. HabSim-EMS can receive an operator's control and command inputs through the OpenMCT-based HCI. HCI can visualize all telemetry is downlinked to the data storage from HabSim-EMS using the data service APIs discussed in Section III.B. HCI uses a reduced-order habitat model, combined with control inputs, to form predictions on key habitat states, providing decision support for an operator. The operator can provide manual input to HabSim-EMS to control its performance based on the telemetry data and system predictions received by HabSim-HMS. Figure 10 shows the data flow through all components of HabSim-HMS.

B. Simulation Scenario Configuration

The simulation is initialized with baseline parameters that define nominal operating and disruption conditions in the habitat. HabSim was configured to run for 3600 seconds, simulating a high-intensity (level 4) meteorite impact damaging the habitat structure. For this simulation, the total nominal power output from the power system is 2 kW, with the nuclear generator having a maximum nominal power output of 1.5 kW. Battery systems are configured to have a maximum capacity of 150 kWhr, with an initial state of charge at 29%. Dust accumulation increases near the nuclear power generator panels from 0.15 gm/seconds/m^2 to 5 gm/seconds/m^2. We edit the Input, Configuration, and Simulation settings files to configure HabSim to simulate this disruption scenario.

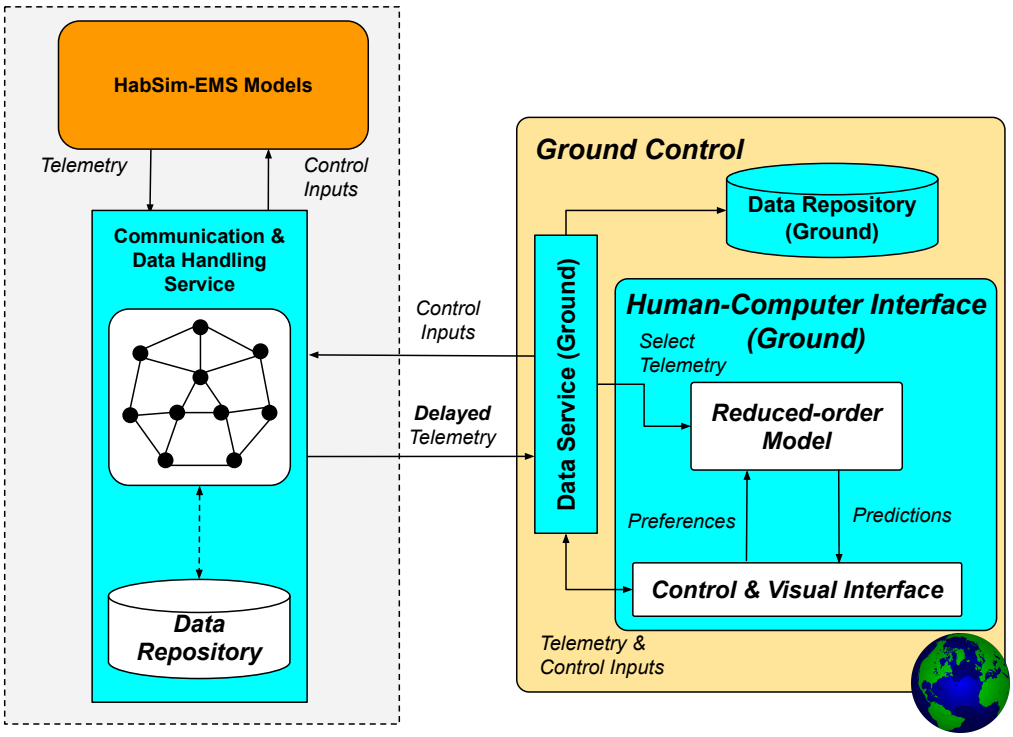

Fig. 10 Data Flow for the Illustrative Example demonstrating features of HabSim-HMS

This event is chosen for its potential to cause a sudden and significant disruption to the habitat's power supply, a critical aspect of space habitation. The meteorite impact is expected to result in 1) Immediate loss of functionality of the nuclear power generator, 2) Increased dust accumulation rate on solar panels and other related subsystems as a secondary effect of the impact, and 3) Major deficiency in power storage. These outcomes present complex challenges, testing the resilience and adaptability of the habitat system modeled in HabSim by maintaining safe performance while withstanding sudden power loss and environmental disruptions. The scenario reflects a realistic but high-stakes situation we can anticipate in deep-space habitation, providing valuable insights into the system's performance under extreme conditions.

C. Simulation Process

The simulation can be started by running HabSim-HMS in Docker and HabSim-EMS in Simulink. Figure 11 visually communicates a sequence of real-time decision-making and control actions following the disruptive event in the simulation, demonstrating how HabSim-HMS drives response through telemetry data and predictive analytics to maintain system performance.

In this depicted scenario, the simulated habitat experiences a meteorite impact at the 200-second mark, explicitly targeting the nuclear power generator. The ground-based participant receives telemetry and detects an abnormality

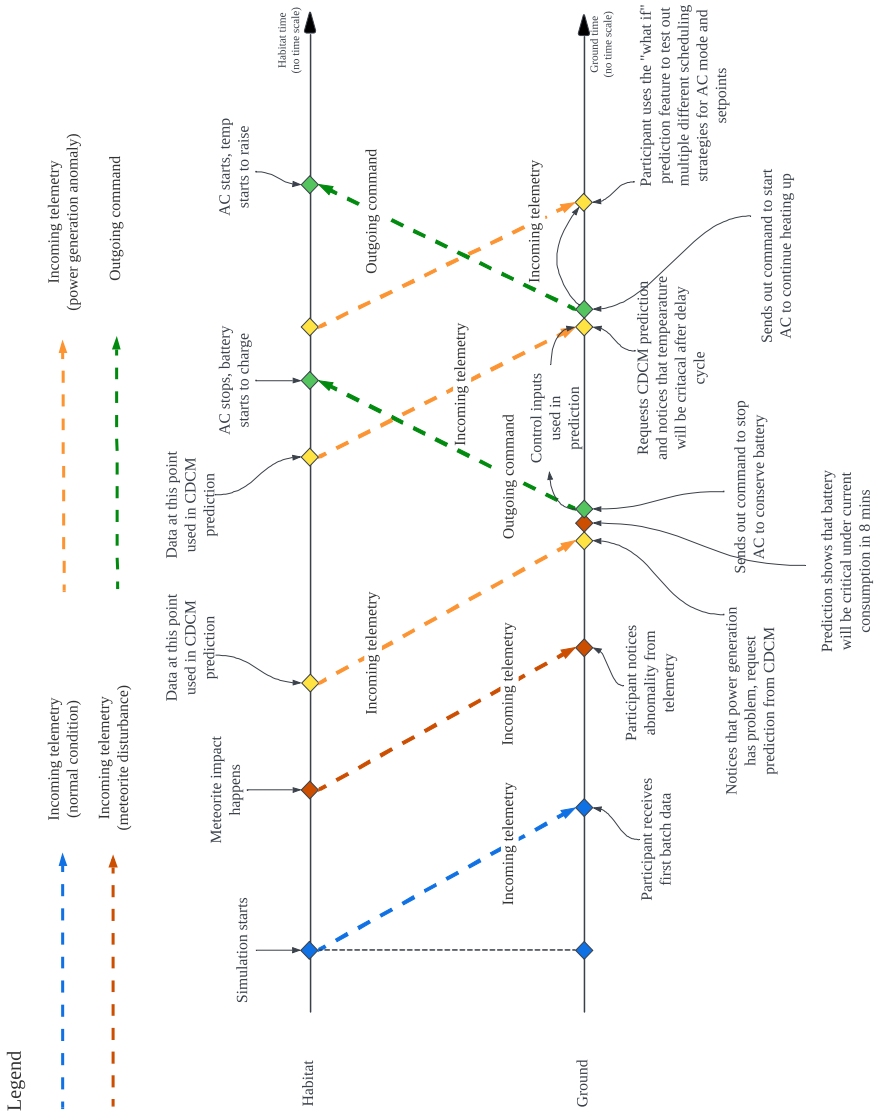

Fig. 11 Scenario handling and system response timeline for the human-systems integration experiment. The timeline shows the sequence of events and operator actions following the meteorite impact scenario on the nuclear power generator in HabSim. In this scenario, the operator uses the HCI to monitor telemetry data, predict system performance, and make control decisions to maintain the habitat's operational status under simulated communication delays.

indicating a failure in power generation. Since the telemetry reaches the ground with a delay, the ground-based participant can use the predictive capabilities of HCI's reduced-order model to determine the depletion times of the battery reserves under the current power consumption rate. Figure 12 shows an example of the telemetry data coming in from Habsim-EMS to HCI and represented for the operator to view. As we can see from the plots of the power subsystem, the nuclear power generation stops producing power in the early stage of the simulation due to the simulated meteorite-impact damage. The operator evaluates the available information against response protocols to determine the most effective action. In response to the loss of nuclear power generation, the HMS and habitat operation must rely on solar power generation. As a proactive measure, the operator must adjust the interior temperature set points and Active Thermal Control System (ATCS) controller operating modes accordingly to compensate for the power deficit (as shown in Figure 8). This action complements automatic power distribution that optimizes load management to ensure essential habitat functions remain operational. However, subsequent predictions highlight an impending critical temperature scenario following this intervention. In response, the operator reactivates the ATCS controller to avert temperature-related issues. To optimize system response, the operator employs a "what if" analysis schema, testing various ATCS controller scheduling strategies using the predictive model embedded in the HCI to achieve a balance between conserving battery life and maintaining temperature control, thereby illustrating the interactive process of managing a habitat's life-support systems in the wake of a significant disruption.

Throughout the impact-recovery process, the human operator closely monitors the stored energy left in the main battery to estimate and control the habitat's interior environment status to maintain a habitable environment and conserve the necessary amount of energy should any consequential incidents happen. In this case, the human operator works with the habitat's automated system as a team to continuously assess the changing conditions and adapt its strategies accordingly, showcasing the Human-AI team's dynamic response capability in the face of a complex and evolving challenge. All data generated during the simulation, including the telemetry and health status FDD data, are recorded automatically for data review and retrieval for future analysis. Operator-specified control actions initiated from HabSim's HCI will also be recorded in the data repository. Thus, we can "replay" any simulation scenarios to make sense of the event timelines behind this scenario handling and system response experiment.

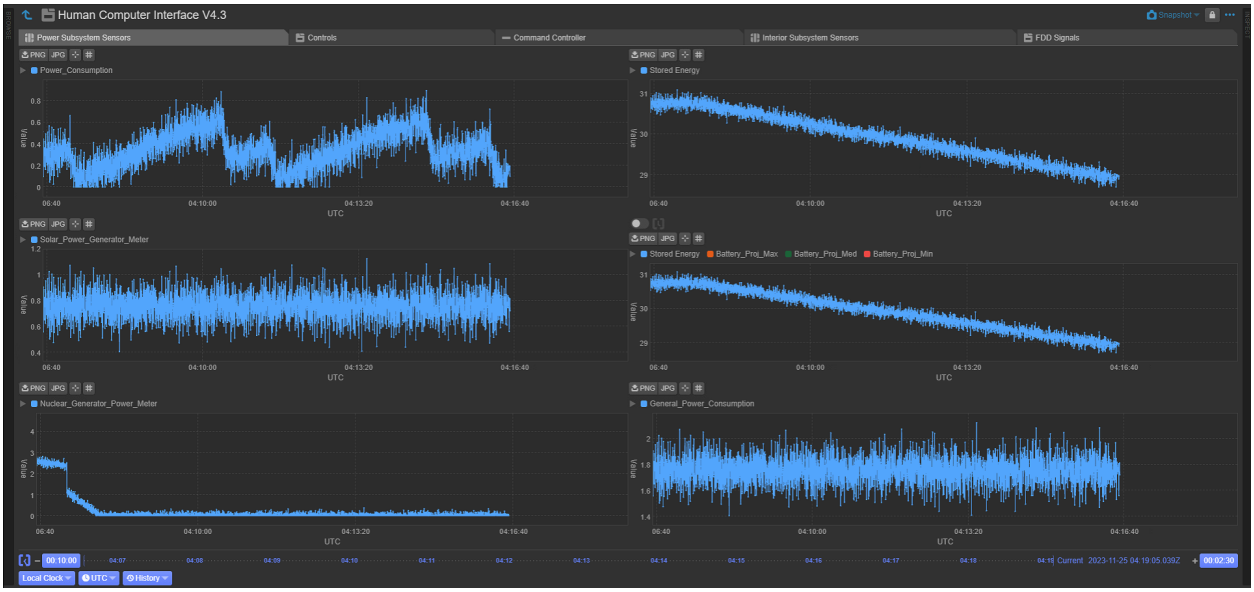

Fig. 12 Telemetry from HabSim Power Subsystem is visualized in the OpenMCT-based HCI. The operator can view the power subsystem's performance and make decisions based on the telemetry data. When predictive models are enabled to calculate the uncertainty bounds of performance, like in Figure 9, the operator can make informed decisions about the system's health and performance.

V. Conclusions & Lessons Learned

Our experiences in developing and testing the HMS testbed within HabSim provided some valuable lessons and experiences that apply to integrated systems health management development. Early versions of HabSim-HMS were primarily focused on commissioning the CDHS that could handle real-time streaming telemetry from HabSim-EMS. In the first iteration of HabSim-HMS, we developed the TSN emulator using OMNET++, a discrete event simulation framework [68]. In this version, C2 performed as a rule-based expert decision-making system [69–71] that automated fault management for the HabSim-EMS. The communication network stored telemetry information in a MySQL database, and C2 extracted the required telemetry information using Python APIs to make decisions about repair interventions to recover system performance. As more disruption scenarios and faults were modeled into the HabSim-EMS, the OMNET++-based TSN emulator showed performance degradation due to the increased volume of data that needed to be handled by the communication network. Hence, we developed an in-house TSN emulator in the Go Programming Language [72], demonstrating significant improvement in run-time performance. Further, a dedicated data handling service was required with the explosion of telemetry information handled by HabSim-HMS. The data handling service was designed and developed to provide a consistent schema to store telemetry information in relational databases and provide data-exchange APIs for accessing and writing data from the databases for other HabSim-HMS applications.

As more complex failure behaviors were introduced into the HabSim-EMS, the decision-making backend of HabSim-HMS faced an explosion of rules required to isolate and handle system failures. The rule-based expert decision-making system that supported automated fault management in C2 was augmented with an inference engine that could isolate faults using information obtained from FFMs of HabSim-EMS. The fault-to-effect D-matrix, derived from the system's FFM, successfully drives all failure modes modeled in HabSim-EMS.

Another hurdle for our team during the development of HabSim-HMS was the "software dependency hell" [73, 74]. Each component of HabSim-HMS was developed with software frameworks with broad community and online support for the functional components of HabSim-HMS (Figure 13). Components of the CDHS were developed in macOS, with the backend developed in Go language and the front end using Vue.js. C2 was developed in macOS with the Python programming language. HabSim-EMS was developed in MATLAB Simulink in the Windows Operating System. While this reduced the development time for individual systems, significant time and effort were spent resolving software dependencies when integrating with the Windows Operating System. As we sought to integrate more functionalities into HabSim-HMS, namely the HCI and the GCC node, we needed to adopt a platform that retained developer convenience but alleviated the challenges during system integration. We adopted Docker, a Linux-based virtual containerization technology that developers and system administrators widely use for developing and maintaining continuous improvement and development pipelines for web-based applications [75]. Docker containers are lightweight [76, 77] and improve reproducibility in software engineering research [78]. It is being increasingly adopted for health management for complex systems [79] and developing hierarchical learning in edge-computing architectures [80]. HabSim-HMS consists

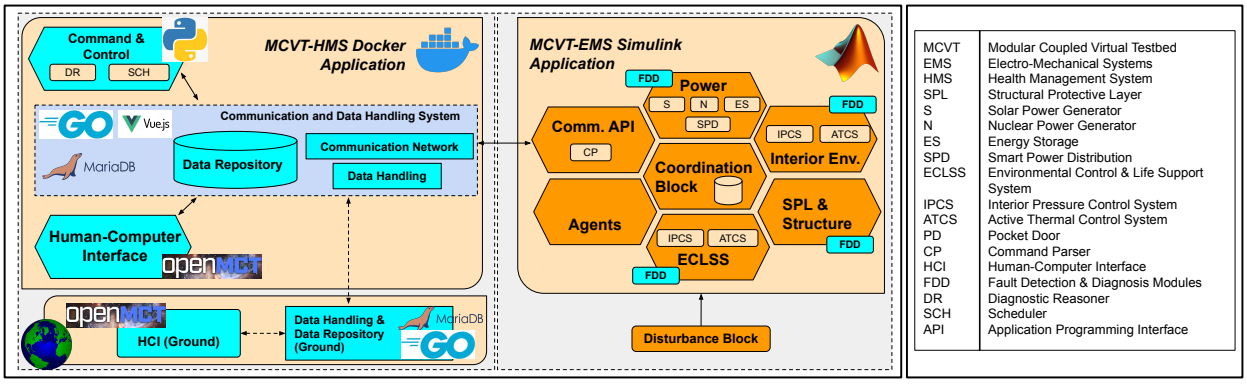

Fig. 13 Docker enables seamless integration of different software frameworks required to run the HMS Application with HabSim-EMS.

of six micro-services, namely the communication network service that functions as a TSN emulator; the Habitat and Ground database service where all telemetry information is stored; a data handling service that handles all data-exchange requests between the data repository, publishers, and subscribers to this data; a C2 service that performs automated fault management and heuristics based scheduling; and the HCI service that provides mission control interface to users. Containerizing components of HabSim-HMS into modular docker containers has significantly improved the efficiency of our systems integration process and has enabled us to accelerate the number of prototypes we developed and iterated successfully.

Through HabSim-HMS, we found it possible to enable situational awareness and autonomous fault-management of a control-based complex system through distributed and hierarchical health management. The testbed architecture described in this paper supports automated fault management of 30 failure modes and provides situational awareness with access to over 50 telemetry signatures. At RETHi, HabSim-HMS is primarily for the study of resilient control and health management strategies for deep-space habitation under the constraints of communication delays and limited onboard computing. HabSim-HMS supports RETHi's research initiatives into cyber-physical systems, artificial intelligence, and human-system integration for deep-space habitation.

Acknowledgement

This work was supported by a Space Technology Research Institute's Grant (number 81NSSC19K1076) from NASA's Space Technology Research Grants Program. We acknowledge the software engineering support provided to this work by Mahira Morris.

Credit author statements

R Murali Krishnan: Conceptualization, Methodology, Software, Visualization, Investigation, Writing - original draft; **Zixu Zhang**: Software, Methodology, Investigation, Writing - Review & Editing; **Kairui Hao**: Software, Writing

- Review & Editing; **Sreehari Manikkan**: Software, Writing - Review & Editing; **Jiachen Wang**: Software; **Chuanyu Xue**: Software; **Mohsen Azimi**: Software; **Paul Parsons**: Methodology, Writing - Review & Editing; **Song Han**: Methodology, Funding acquisition, Writing - Review & Editing; **Shirley Dyke**: Funding acquisition, Supervision, Writing - Review and Editing. **Ilias Bilionis**: Conceptualization, Investigation, Methodology, Funding acquisition, Writing - Review & Editing;

References

[1] Crusan, J. C., Craig, D. A., and Herrmann, N. B., "NASA's Deep Space Habitation Strategy," *2017 IEEE Aerospace Conference*, Mar 2017, pp. 1–11. https://doi.org/10.1109/AERO.2017.7943624.

[2] Griffin, G., "Introduction," *Human Spaceflight Operations: Lessons Learned from 60 Years in Space*, edited by G. E. Chamitoff and S. R. Vadali, AIAA Inc., Feb 2021, Chap. 7, pp. 113–128. https://doi.org/10.2514/5.9781624104770.0000.0000.

[3] Zumbado, J. R., Campbell, P., Miller, M., and Witt, E., "Introduction to NASA Human Systems Integration," *Human Systems Integration (HSI) Practitioner's Guide*, NASA, 2015, pp. 1.1–1.13. URL https://ntrs.nasa.gov/api/citations/20150022283/downloads/20150022283.pdf.

[4] Silva-Martinez, J., "Human systems integration: process to help minimize human errors, a systems engineering perspective for human space exploration missions," *REACH*, Vol. 2–4, 2016, p. 8–23. https://doi.org/10.1016/j.reach.2016.11.003.

[5] Boy, G. A., "Aerospace Human Systems Integration," *A Framework of Human Systems Engineering: Applications and Case Studies*, edited by H. A. H. Handley and A. Tolk, Wiley-IEEE Press, 2020, Chap. 7, pp. 113–128. https://doi.org/10.1002/9781119698821.ch7.

[6] Cucinotta, F. A., "Review of NASA Approach to Space Radiation Risk Assessments for Mars Exploration," *Health Physics*, Vol. 108, No. 2, Feb. 2015, pp. 131–142. https://doi.org/10.1097/hp.0000000000000255.

[7] Fogtman, A., Baatout, S., Baselet, B., Berger, T., Hellweg, C. E., Jiggens, P., Tessa, C. L., Narici, L., Nieminen, P., Sabatier, L., Santin, G., Schneider, U., Straube, U., Tabury, K., Tinganelli, W., Walsh, L., and Durante, M., "Towards sustainable human space exploration—priorities for radiation research to quantify and mitigate radiation risks," *npj Microgravity*, Vol. 9, No. 1, 2023. https://doi.org/10.1038/s41526-023-00262-7.

[8] Stassinopoulos, E., and Raymond, J., "The space radiation environment for electronics," *Proceedings of the IEEE*, Vol. 76, No. 11, 1988, p. 1423–1442. https://doi.org/10.1109/5.90113.

[9] LaBel, K., Johnston, A., Barth, J., Reed, R., and Barnes, C., "Emerging radiation hardness assurance (RHA) issues: a NASA approach for space flight programs," *IEEE Transactions on Nuclear Science*, Vol. 45, No. 6, 1998, p. 2727–2736. https://doi.org/10.1109/23.736521.

[10] Schwank, J. R., Shaneyfelt, M. R., and Dodd, P. E., "Radiation Hardness Assurance Testing of Microelectronic Devices and Integrated Circuits: Radiation Environments, Physical Mechanisms, and Foundations for Hardness Assurance," *IEEE Transactions on Nuclear Science*, Vol. 60, No. 3, 2013, p. 2074–2100. https://doi.org/10.1109/tns.2013.2254722.

[11] McTigue, K. R., Parisi, M. E., Panontin, T., Wu, S., and Vera, A. H., "Extreme Problem Solving: The New Challenges of Deep Space Exploration," *SpaceCHI: Human-Computer Interaction for Space Exploration (CHI '21)*, May 2021, pp. 1–5. URL https://human-factors.arc.nasa.gov/publications/McTigue_Parisi_Panontin_Wu_Vera_SpaceCHI.pdf.

[12] Parisi, M., Panontin, T., Wu, S.-C., Mctigue, K., and Vera, A., "Effects of Communication Delay on Human Spaceflight Missions," *Human-Centered Aerospace Systems and Sustainability Applications*, AHFE International, USA, 2023, pp. 64–73. https://doi.org/10.54941/ahfe1003920.

[13] McTigue, K. R., Parisi, M. E., Panontin, T., Wu, S., and Vera, A. H., "How to Keep Your Space Vehicle Alive: Maintainability Design Principles for Deep-Space Missions," *SpaceCHI: Human-Computer Interaction for Space Exploration (CHI '23)*, June 2023, pp. 1–7. URL https://human-factors.arc.nasa.gov/publications/SpaceCHI2023_Maintainability.pdf.

[14] Dyke, S. J., Marais, K., Bilionis, I., Werfel, J., and Malla, R., "Strategies for the design and operation of resilient extraterrestrial habitats," *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2021*, edited by D. Zonta, H. Huang, and Z. Su, SPIE, 2021, pp. 1–12. https://doi.org/10.1117/12.2585118.

[15] Badger, J. M., Higbee, D., Kennedy, T., Vitalpur, S. V., Sargusingh, M. J., Shull, S. A., Othon, B., Davies, F. J., Hurlbert, E. A., Townsend, N., Mauldin, J. M., Nelson, E., Nagy, K., Hurlbert, K., Frank, J. D., and Love, S. G., "Spacecraft Dormancy Autonomy Analysis for a Crewed Martian Mission," *NASA/TM-20180005514*, NASA, 2018, pp. iii–iv. URL https://ntrs.nasa.gov/api/citations/20180005514/downloads/20180005514.pdf.

[16] Bowman, C., Tschan, C., and Zetocha, P., "Advances in Intelligent Systems for Space Applications," *Advances in computational intelligence and autonomy for aerospace systems*, edited by J. Valasek, American Institute of Aeronautics and Astronautics Inc., Jan. 2018, Chap. 1, pp. 1–46. https://doi.org/10.2514/5.9781624104794.0001.0046.

[17] Johnston, P., and Harris, R. L., "The Boeing 737 MAX Saga: Lessons for Software Organizations," *Software Quality Professional Magazine*, Vol. 21, No. 3, 2019, pp. 4–12. URL https://api.semanticscholar.org/CorpusID:195414546.

[18] Koopman, P., Kane, A., and Black, J., "Credible autonomy safety argumentation," *27th Safety-Critical Systems Symposium*, Feb. 2019, pp. 1–7. http://users.ece.cmu.edu/~koopman/pubs/Koopman19_SSS_CredibleSafetyArgumentation.pdf, last accessed on 12/01/23.

[19] Leveson, N. G., *Engineering a Safer World: Systems Thinking Applied to Safety*, The MIT Press, Jan. 2012. https://doi.org/10.7551/mitpress/8179.001.0001.

[20] Leveson, N., "A new accident model for engineering safer systems," *Safety Science*, Vol. 42, No. 4, 2004, p. 237–270. https://doi.org/10.1016/s0925-7535(03)00047-x.

[21] Janasak, K. M., and Beshears, R. R., "Diagnostics To Prognostics - A Product Availability Technology Evolution," *2007 Annual Reliability and Maintainability Sympsoium*, IEEE, Jan. 2007, pp. 113–118. https://doi.org/10.1109/RAMS.2007.328051.

[22] Keller, K., Peck, J., Swearingen, K., and Gilbertson, D., "Architectures for Affordable Health Management," *2010 AIAA Infotech@Aerospace*, American Institute of Aeronautics and Astronautics, Apr. 2010, pp. 1–11. https://doi.org/10.2514/6.2010-3435.

[23] Figueroa, F., Walker, M., and Underwood, L. W., "NASA Platform for Autonomous Systems (NPAS)," *AIAA Scitech 2019 Forum*, American Institute of Aeronautics and Astronautics, Jan. 2019, pp. 1–15. https://doi.org/10.2514/6.2019-1963.c1.

[24] Figueroa, F., Underwood, L., Hekman, B., and Morris, J., "Hierarchical Distributed Autonomy: Implementation Platform and Processes," *2020 IEEE Aerospace Conference*, IEEE, Mar. 2020, pp. 1–9. https://doi.org/10.1109/aero47225.2020.9172572.

[25] Stecklein, J. M., Bielski, W. P., Acevedo, A. B., and Daley, M. E., "Gateway Modeling and Simulation Plan," Tech. rep., NASA, 2019.

[26] Muri, P., Hanson, S., and Sonnier, M., "Gateway Avionics Concept of Operations for Command and Data Handling Architecture," *2021 IEEE Aerospace Conference (50100)*, 2021, pp. 1–7. https://doi.org/10.1109/AERO50100.2021.9438539.

[27] Jain, R., Cilento, M., Ulmer, J., and Marais, K., "Control Effectiveness: Metric Development and Application to Resilient Lunar Habitat Design," *AIAA Journal*, 2024. In press.

[28] Rhee, S., Noble, Z., Park, J., Lial, A., Collazo, L., and Ziviani, D., "Development of a Damageable ECLSS and IE Virtual Testbed Model to Simulate Future Resilient Deep Space Habitats," *Proceedings of the 52nd International Conference on Environmental Systems*, Calgary, Canada, 2023.

[29] Rhee, S., and Ziviani, D., "Development of a Resiliency-Based Evaluation Framework for Deep Space Habitats," *AIAA Journal*, 2024. In press.

[30] Shahriar, A., Montoya, H., Majlesi, A., Avila, D., and Montoya, A., "Coupling Independent Solid Mechanics-Based Systems with a Common Surface Interface in a System of Systems Modeling Framework," *AIAA Journal*, 2024. In press.

[31] Chebbo, L., Gultekin, M., Bazzi, A., Tomastik, R., Pattipati, K., Vaccino, L., Azimi, M., and Lund, A., "Modeling and Operation of Microgrids for Deep Space Habitats Under Environmental Disturbances," *IEEE Power and Energy Conference at Illinois*, 2022, pp. 1–6. https://doi.org/10.1109/PECI57361.2023.10197727.

[32] Chebbo, L., Hasnain, N., and Bazzi, A., "Fault Diagnosis of Power Components with Reliability Assessment in Extraterrestrial Microgrids," *AIAA Journal*, 2024. In press.

[33] Wang, Z., Jahanshahi, M., Azimi, M., and Dyke, S., "Sensor Fault Detection in Smart Extraterrestrial Habitats Using Unsupervised Learning," *AIAA Journal*, 2024. In press.

[34] Ouyang, M., and Wang, Z., "Resilience assessment of interdependent infrastructure systems: With a focus on joint restoration modeling and analysis," *Reliability Engineering & System Safety*, Vol. 141, Sep. 2015, pp. 74—-82. https://doi.org/10.1016/j.ress.2015.03.011.

[35] Maghareh, A., Lenjani, A., Krishnan, M., Dyke, S., and Bilionis, I., "Role of Cyber-Physical Testing in Developing Resilient Extraterrestrial Habitats," *Earth and Space 2021*, ASCE, 2021, pp. 1059–1068. https://doi.org/10.1061/9780784483374.098.

[36] Kitching, R., Mattingly, H., Williams, D., and Marais, K., "Resilient Space Habitat Design Using Safety Controls," *Earth and Space 2021*, ASCE, 2021, pp. 992–1003. https://doi.org/10.1061/9780784483374.091.

[37] Lammlein, D. R., Latham, G. V., Dorman, J., Nakamura, Y., and Ewing, M., "Lunar seismicity, structure, and tectonics," *Reviews of Geophysics*, Vol. 12, No. 1, 1974, pp. 1–21. https://doi.org/10.1029/rg012i001p00001.

[38] Grün, E., Horanyi, M., and Sternovsky, Z., "The lunar dust environment," *Planetary and Space Science*, Vol. 59, No. 14, 2011, pp. 1672–1680. https://doi.org/10.1016/j.pss.2011.04.005.

[39] Guibaud, A., Legros, G., Consalvi, J.-L., and Torero, J., "Fire safety in spacecraft: Past incidents and Deep Space challenges," *Acta Astronautica*, Vol. 195, 2022, p. 344–354. https://doi.org/10.1016/j.actaastro.2022.01.021.

[40] Ruff, G., Brooker, J., and Urban, D., "Fire Safety Technology Development for Exploration Spacecraft and Habitats," *40th International Conference on Environmental Systems*, American Institute of Aeronautics and Astronautics, Jul. 2010, pp. 1–11. https://doi.org/10.2514/6.2010-6240.

[41] Bagchi, S., Aggarwal, V., Chaterji, S., Douglis, F., Gamal, A. E., Han, J., Henz, B. J., Hoffmann, H., Jana, S., Kulkarni, M., Lin, F. X., Marais, K., Mittal, P., Mou, S., Qiu, X., and Scutari, G., "Vision Paper: Grand Challenges in Resilience: Autonomous System Resilience through Design and Runtime Measures," *IEEE Open Journal of the Computer Society*, Vol. 1, 2020, pp. 155–172. https://doi.org/10.1109/OJCS.2020.3006807.

[42] Maghareh, A., Lenjani, A., Dyke, S. J., Marais, K., Whitaker, D., Bobet, A., Ramirez, J., Modiriasari, A., and Theinat, A. E., "Resilience-oriented Design of Extraterrestrial Habitat Systems," *AIAA Propulsion and Energy 2019 Forum*, American Institute of Aeronautics and Astronautics, Aug. 2019, pp. 1–7. https://doi.org/10.2514/6.2019-3972.

[43] Azimi, M., Lund, A., Fu, Y., Montoya, H., Vaccino, L., Krishnan, M., Rhee, S., Chebbo, L., Wang, Z., Maghareh, A., and Dyjke, S., "HabSim: A Modelar Coupled Virtual Testbed for Simulating ExtraTerrestrial Habitat Systems," *AIAA Journal*, 2024. AIAA Journal.

[44] Rajasekharan Pillai, M. K., Alvarenga, B. D., Ravula, P., and Bilionis, I., "Agent-based models for real-time fault-recovery in a System-of-Systems Smart Habitat simulator," *ASCEND 2023*, American Institute of Aeronautics and Astronautics, Oct. 2023, pp. 1–18. https://doi.org/10.2514/6.2023-4752.

[45] Merkel, D., "Docker: Lightweight Linux Containers for Consistent Development and Deployment," *Linux J.*, Vol. 2014, No. 239, 2014. https://doi.org/10.5555/2600239.2600241.

[46] Mathur, A., Cavanaugh, K. F., Pattipati, K. R., Willett, P. K., and Galie, T. R., "Reasoning and modeling systems in diagnosis and prognosis," *Component and Systems Diagnostics, Prognosis, and Health Management*, edited by P. K. Willett and T. Kirubarajan, SPIE, Jul. 2001, pp. 1–7. https://doi.org/10.1117/12.434239.

[47] NASA, "core Flight System (cFS) Limit Checker," GitHub Link: https://github.com/nasa/LC, retrieved 01 December, 2023.

[48] Kopetz, H., Ademaj, A., Grillinger, P., and Steinhammer, K., "The Time-Triggered Ethernet (TTE) Design," *Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)*, IEEE, 2005, pp. 22–33. https://doi.org/10.1109/isorc.2005.56.

[49] Farkas, J., "Introduction to IEEE 802.1: Focus on the Time-Sensitive Networking Task Group," http://www.ieee802.org/1/files/public/docs2018/tsn-farkas-intro-0318-v01.pdf, Mar. 2018.

[50] Pardo-Castellote, G., "OMG Data-Distribution Service: Architectural Overview," *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.*, IEEE, 2003, pp. 200–206. https://doi.org/10.1109/ICDCSW.2003.1203555.

[51] Casini, D., Blaß, T., Lütkebohle, I., and Brandenburg, B., "Response-Time Analysis of ROS 2 Processing Chains Under Reservation-Based Scheduling," *31st Euromicro Conference on Real-Time Systems*, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik GmbH, Wadern/Saarbruecken, Germany, 2019, pp. 1–23. https://doi.org/10.4230/LIPIcs.ECRTS.2019.6.

[52] Mackey, R., Brownston, L., P. Castle, J., and Sweet, A., "Getting Diagnostic Reasoning off the Ground: Maturing Technology with TacSat-3," *IEEE Intelligent Systems*, Vol. 25, No. 5, 2010, p. 27–35. https://doi.org/10.1109/mis.2010.124.

[53] Aaseng, G. B., Sweet, A., and Ossenfort, J., "Performance Analysis of an Autonomous Fault Management System," *2018 AIAA SPACE and Astronautics Forum and Exposition*, American Institute of Aeronautics and Astronautics, 2018, pp. 1–7. https://doi.org/10.2514/6.2018-5149.

[54] Levinson, R., Frank, J. D., Iatauro, M., Knight, C. D., Sweet, A., Aaseng, G. B., Scott, M., Ossenfort, J., Soeder, J., Ngo, T., et al., "Development and Testing of a Vehicle Management System for Autonomous Spacecraft Habitat Operations," *2018 AIAA SPACE and Astronautics Forum and Exposition*, 2018, p. 5148. https://doi.org/10.2514/6.2018-5148.

[55] Aaseng, G., Do, M., Frank, J., Fry, C., and Sweet, A., "Integrating Planning, Diagnosis and Execution for Vehicle Systems Management," *33rd International Conference on Automated Planning and Execution*, 2023, pp. 1–7.

[56] Coderre, K., Edwards, C., Cichan, T., Richey, D., Shupe, N., Soblish, D., Ramm, S., Perkes, B., Posey, J., Pratt, W., and Liu, E., "Concept of Operations for the Gateway," *Space Operations: Inspiring Humankind's Future*, edited by H. Pasquier, C. A. Cruzen, M. Schmidhuber, and Y. H. Lee, Springer Cham, 2019, pp. 63–82. https://doi.org/10.1007/978-3-030-11536-4.

[57] Oostdyk, R., Ferrell, B., Lewis, M., Perotti, J., and Brown, B., "Functional Fault Modeling of a Cryogenic System for Real-Time Fault Detection and Isolation," *AIAA Infotech@Aerospace 2010*, American Institute of Aeronautics and Astronautics, 2010, pp. 1–7. https://doi.org/10.2514/6.2010-3548.

[58] Frank, J., Aaseng, G., Dalal, K. M., Fry, C., Lee, C., McCann, R., Narasimhan, S., Spirkovska, L., Swanson, K., Wang, L., et al., "Integrating Planning, Execution and Diagnosis to Enable Autonomous Mission Operations," *SPARK 2013*, Vol. 545, 2013, pp. 181–6.

[59] Luo, J., Tu, H., Pattipati, K., Qiao, L., and Chigusa, S., "Graphical models for diagnosis knowledge representation and inference," *IEEE Autotestcon, 2005.*, 2005, pp. 483–489. https://doi.org/10.1109/AUTEST.2005.1609185.

[60] NASA, "AOS-DR: Autonomy Operating System - Diagnostic Reasoner," , 2020. Retrieved from: https://github.com/nasa/aos-dr, commit hash: 1d204a25a9ff6721aed89fbe02d6169907628e30.

[61] Chien, S., Sherwood, R., Tran, D., Cichy, B., Rabideau, G., Castano, R., Davis, A., Mandl, D., Frye, S., Trout, B., Shulman, S., and Boyer, D., "Using Autonomy Flight Software to Improve Science Return on Earth Observing One," *Journal of Aerospace Computing, Information, and Communication*, Vol. 2, No. 4, 2005, p. 196–216. https://doi.org/10.2514/1.12923.

[62] Chein, S., Rabideau, G., Knight, R., Sherwood, R., Engelhardt, B., Mutz, D., Estlin, T., Smith, B., Fisher, F., Barrett, T., Stebbins, G., and Tran, D., "ASPEN-Automated Planning and Scheduling for Space Mission Operation," *International Conference on Space Operations (SpaceOps 2000)*, 2000, pp. 1–11. URL https://hdl.handle.net/2014/14437.

[63] Gao, Y., and Chien, S., "Review on space robotics: Toward top-level science through space exploration," *Science Robotics*, Vol. 2, No. 7, 2017. https://doi.org/10.1126/scirobotics.aan5074.

[64] Frank, J., "The challenges of verification and validation of automated planning systems (keynote)," *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, IEEE, 2013, pp. 1–7. https://doi.org/10.1109/ase.2013.6693059.

[65] Pritchard, K., Vaccino, L., Liu, X., Whitaker, D., Dyke, S., and Joyal, B., "Lunar SmartHab Mission Operations and Crew Day-In-The-Life," *International Conference on Environmental Systems, 2023*, 2023, pp. 1–12. URL https://hdl.handle.net/2346/94797.

[66] Vaccino, L., Pritchard, K., Liu, X., Whitaker, D., Dyke, S., and Joyal, B., "Simulation-Based Assessment of Hazardous States in a Deep Space Habitat," *International Conference on Environmental Systems, 2023*, 2023, pp. 1–11. URL https://hdl.handle.net/2346/94795.

[67] NASA, "Open MCT - Open Mission Control Technologies," , 2023. Retrieved from:https://nasa.github.io/openmct/.

[68] Varga, A., "OMNeT++," *Modeling and Tools for Network Simulation*, edited by K. Wehrle, M. Gunes, and J. Gross, Springer Berlin Heidelberg, 2010, Chap. 1, p. 35–59. https://doi.org/10.1007/978-3-642-12331-3_3.

[69] Hayes-Roth, F., "Rule-based systems," *Communications of the ACM*, Vol. 28, No. 9, 1985, p. 921–932. https://doi.org/10.1145/4284.4286.

[70] Grosan, C., and Abraham, A., "Rule-Based Expert Systems," *Intelligent Systems: A Modern Approach*, edited by C. Grosan and A. Abraham, Springer Berlin Heidelberg, 2011, Chap. 7, p. 149–185. https://doi.org/10.1007/978-3-642-21004-4_7.

[71] Abraham, A., "Rule-Based Expert Systems," *Handbook of Measuring System Design*, edited by P. H. Sydenham and R. Thorn, John Wiley & Sons, Ltd, 2005, Chap. 130, pp. 1–7. https://doi.org/10.1002/0471497398.mm422.

[72] Meyerson, J., "The Go Programming Language," *IEEE Software*, Vol. 31, No. 5, 2014, p. 104–104. https://doi.org/10.1109/ms.2014.127.

[73] Seo, H., Sadowski, C., Elbaum, S., Aftandilian, E., and Bowdidge, R., "Programmers' build errors: a case study (at google)," *Proceedings of the 36th International Conference on Software Engineering*, Association for Computing Machinery, 2014, pp. 724–734. https://doi.org/10.1145/2568225.2568255, URL http://dx.doi.org/10.1145/2568225.2568255.

[74] Dick, S., and Volmar, D., "DLL Hell: Software Dependencies, Failure, and the Maintenance of Microsoft Windows," *IEEE Annals of the History of Computing*, Vol. 40, No. 4, Oct. 2018, p. 28–51. https://doi.org/10.1109/mahc.2018.2877913.

[75] Rad, B. B., Bhatti, H. J., and Ahmadi, M., "An introduction to docker and analysis of its performance," *International Journal of Computer Science and Network Security (IJCSNS), ISSN: 1738-7906*, Vol. 17, No. 3, 2017, p. 228.

[76] Potdar, A. M., D G, N., Kengond, S., and Mulla, M. M., "Performance Evaluation of Docker Container and Virtual Machine," *Procedia Computer Science*, Vol. 171, 2020, p. 1419–1428. https://doi.org/10.1016/j.procs.2020.04.152.

[77] N, P. E., Mulerickal, F. J. P., Paul, B., and Sastri, Y., "Evaluation of Docker containers based on hardware utilization," *2015 International Conference on Control Communication & Computing India (ICCC)*, 2015, pp. 697–700. https://doi.org/10.1109/ICCC.2015.7432984.

[78] Cito, J., and Gall, H. C., "Using Docker Containers to Improve Reproducibility in Software Engineering Research," *Proceedings of the 38th International Conference on Software Engineering Companion*, Association for Computing Machinery, 2016, p. 906–907. https://doi.org/10.1145/2889160.2891057.

[79] Xiao, J., Zhang, M., Tian, H., Huang, B., and Fu, W., "Prognostics and health management system for hydropower plant based on fog computing and docker container," *IOP Conference Series: Earth and Environmental Science*, Vol. 121, 2018, p. 042029. https://doi.org/10.1088/1755-1315/121/4/042029.

[80] Jaiswal, K., Sobhanayak, S., Turuk, A. K., Bibhudatta, S. L., Mohanta, B. K., and Jena, D., "An IoT-Cloud Based Smart Healthcare Monitoring System Using Container Based Virtual Environment in Edge Device," *2018 International Conference on Emerging Trends and Innovations In Engineering And Technological Research (ICETIETR)*, 2018, pp. 1–7. https://doi.org/10.1109/ICETIETR.2018.8529141.