SIMULATION BUILDER, ANALYSIS, AND DEVELOPMENT (SIMBAD) TOOLKIT FOR HUMAN SPACEFLIGHT OPERATION TRAINING USING THE SPACECRAFT SIMULATION PLATFORM

A Thesis

by

WILLIAM CLAYTON YOUNG

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2022

Major Subject: Aerospace Engineering

ABSTRACT

As the scope of human spaceflight continues to expand, the Human Systems Integration (HSI) developed to support complex missions must be robust and efficient. This risk has been outlined in the Human Research Roadmap (HRR) as the "Risk of Adverse Outcomes Due to Inadequate Human Systems Integration Architecture"[1], short name HSIA. One of the most critical elements of any human spaceflight mission is training, which prepares flight operations teams with the resources necessary to carry out that mission. As more distant destinations such as the Moon or Mars are targeted for human spaceflight, ensuring crew have the tools they need to overcome new types of challenges will be a significant focus when developing new training infrastructures. With the nature of such missions, there are several knowledge gaps associated with HSIA that motivate investigating how training should be carried out on such missions.

This research focuses on studying these gaps and using the findings to create a conceptual demonstration for a tool that can be used to assist in the training infrastructure that supports future spaceflight missions. This tool is called the Simulation Builder, Analysis, and Development (SimBAD) tool, which is a User Interface (UI) that utilizes the Space Collaborative Real-time Analysis and Flight Toolkit to build virtual training environments. There are four main objectives that incentivize the development of this tool, the improved collaboration between groups in the flight operations team, a training framework that is capable of being packaged on board a spacecraft, a framework that accounts for dynamic mission parameters, and a heightened level of autonomy for crew on missions. These objectives have been driven by the findings from an examination of current spaceflight training methods, previous research on training for future missions, and elements of the HSIA risk that pertain to training.

The SimBAD tool was designed with features that were motivated by these objectives to effectively create a virtual training facility. These features allow the user to control the environments, systems, procedures, events, and evaluations that are constructed together inside a virtual simulation. Giving this control to users as well as access to the environment through Virtual Reality

(VR) is the overall method through which this thesis argues the objectives of the concept are met. These objectives are determined to be met and results for analysis are created through a demonstration of the concept. For this research the demonstration is done through several scenarios that are constructed in simulations using the SimBAD tool. The first is a simulation of IntraVehicular Activities (IVA) procedures being executed on board the International Space Station (ISS) which demonstrates the tool is able to account for dynamic mission parameters; the second is a simulation of two users inside a Mars habitat performing a comms check procedure that demonstrates capability for improved collaboration between groups on the flight operations team. The UI and VR platform demonstrate the tool is capable of packaging on board a spacecraft as well as increasing the autonomy the crew has during their mission. The elements of SimBAD establish a closed-loop infrastructure as a virtual training facility that offers improvements towards human spaceflight in HSI, particularly for future exploration missions by offering functionality towards the construction of simulated scenarios with procedure capability, dynamic event scripting, and simulation evaluation.

DEDICATION

Whether my dream was to be a paleontologist at five years old or an astronaut at twenty-five, you always encouraged me to follow that dream. Thanks, Mom.

ACKNOWLEDGMENTS

There are a lot of people who helped make this thesis something that I could realistically complete. First of all is Dr. Gregory Chamitoff, without whom I would not have had the guidance, support, and inspiration to pursue this degree. Working with him has been a great experience and I'm grateful for all the opportunities that he helped make sure I was able to take advantage of. To all of my colleagues at the ASTROLab, I also have to give some credit, especially because of the many countless nights I spent pestering people on Slack with questions I had. Specifically Connor, Patrick, Alex, Elise, Colton and Trevor.

I would also like to thank my committee, Dr. Nancy Currie-Gregg and Dr. Ana Diaz Artiles for doing helping guide me despite the challenges of the virtual settings we've all been working in the past couple years. Their expertise and feedback has been paramount in helping me make the most contribution in my work as I could. I would also like to thank Marc Reagan and Kevin Heath, who offered me their professional insight in the early days of my thesis.

Last year I somehow found myself getting connected with a bunch of people from all over the world in different areas of the industry and academia. This was through Twitter, of all places. I'd like to thank countless people there for being very supportive of me during all of this and showing me how fantastic the aerospace community could be. I know I'll see plenty of y'all travel far and wide in our field in the many years to come.

Another unlikely place I found encouragement was the Swampletics YouTube series created by Settled. Dedication and perseverance in the face of struggle are things I'm not sure I would've had enough of if I hadn't had the series playing in the background during all the hours I was working on this. To anyone who may be going through their own trials and tribulations, I urge you to remember these words from the series I kept at the front of my mind: "The comeback is always greater than the setback".

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Drs. Gregory Chamitoff and Ana Diaz Artiles of the Department of Aerospace Engineering and Dr. Nancy Currie-Gregg of the Department of Industrial and Systems Engineering.

The models and environments created in the demonstrations discussed in chapter 5 originated from either NASA public resources or students working out of the ASTROLab. The multi user demonstration was conducted with the assistance of Trevor Weidner.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

Graduate study was supported by a fellowship from Texas A&M University and a fellowship from NASA/Texas Space Grant Consortium.

NOMENCLATURE

AR	Augmented Reality
CHAPEA	Crew Health And Performance Exploration Analog
CPU	Central Processing Unit
CSV	Comma-Seperated Value
Desert RATS	Desert Research and Technology Studies
DoF	Degrees of Freedom
ECLSS	Environmental Control and Life Support System
EM	Event Manager
EVA	ExtraVehicular Activity
FA	Flight Analogs
FCOD	Flight Crew Operations Directorate
FOD	Flight Operations Directorate
GPU	Graphics Processing Unit
HCI	Human Computer Interaction
HEMAP	Human Engineering Modeling and Performance
HERA	Human Exploration Research Analog
HH&P	Human Health and Performance
Hi-SEAS	Hawai'i Space Exploration Analog and Simulation
HRR	Human Research Roadmap
HSI	Human Systems Integration
HSIA	Human Systems Integration Architecture
ICE	Isolated, Confined and Extreme

ISS	International Space Station
IVA	IntraVehicular Activity
JITT	Just in Time Training
JSC	Johnson Space Center
KSC	Kennedy Space Center
LDSM	Long Duration Spaceflight Mission
LEO	Low Earth Orbit
MCC	Mission Control Center
MOD	Mission Operations Directorate
MSFC	Marshall Spaceflight Center
NBL	Neutral Buoyancy Lab
NCW	Notes, Cautions, and Warnings
NEEMO	NASA Extreme Environments Mission Operations
ROI	Research Operations and Integration
SAFER	Simplified Aid For EVA Rescue
SE&I	Systems Engineering and Integration
SimBAD	Simulation Builder, Analysis and Development
SimE	Simulation Environment
SimEval	Simulation Evaluation
SimP	Simulation Procedures
SimSys	Simulation Systems
SIRIUS	Scientific International Research In a Unique terrestrial Station
SpaceCRAFT	Space Collaborative Real-time Analysis and Flight Toolkit
SVMF	Space Vehicle Mockup Facility
UE4	Unreal Engine 4

UI	User Interface
VR	Virtual Reality
VRL	Virtual Reality Lab
XR	Extended Reality

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 Motivation

Human spaceflight is growing more complex as mission objectives evolve and become increasingly ambitious. Autonomy and automation are also concerns as distance increases on these missions [3]. Ensuring crew, vehicle, and mission safety will become more difficult as the systems grow more complicated. Integration between these systems will also intensify, and the interdependence of the systems will need to be considered. NASA's HRR contains a database for categorized risks, evidence reports, and knowledge gaps that apply to human research as it pertains to the advancement of human spaceflight. The HSIA risk describes these concerns and outlines several knowledge gaps to be filled in order to properly mitigate the risk. These gaps highlight different deficiencies in what is known about HSI that are crucial to implementing one that is sufficient for more ambitious human spaceflight missions. Several of these gaps prescribe how future architectures must have the capability for testing and training within the constraints that future missions will provide. The infrastructure necessary to test flight elements; hardware, software, and procedures; as well as train crews on those elements must be sufficiently effective with respect to the complexity of the systems and systems integration.

Applying these concerns to a potential mission to Mars highlight the areas in which the current human spaceflight infrastructure needs improvements. A vehicle capable of transporting a crew to Mars and back would require a dense set of subsystems such as radiation and thermal protection, Environmental Control and Life Support System (ECLSS), power, communication, propulsion, as well as be capable of ExtraVehicular Activity (EVA) and robotic operations, all while carrying a lander for the Martian surface which will then incorporate its own set of subsystems. Logistically, the travel distance and duration between our planet and Mars is also a point of concern. As the vehicle travels further from ground stations on Earth, communications will degrade and the ground's capability to support operations will decrease. Crews will need to be able to remain proficient on the procedures for vehicle systems during landing, surface operations, and launch phases without having access to typical ground training facilities. Bringing these facilities on the voyages would be convenient, but highly impractical. However, developing a virtual facility that offers comparable training could provide a solution that mitigates the problems associated with these missions.

1.2 Background

1.2.1 HSIA

This specific scenario illustrates several issues that will be encountered as human spaceflight programs venture further from the surface of the Earth. This research delves deeper into the HSI challenges that will be faced on these expeditions. Training is one of the elements of a program that enhances the interface between the systems used to conduct a flight and the human operating them. This element is important for giving operators, both ground support and crew, an appropriate amount of context for the systems in order to utilize them effectively. Training also builds situational awareness by increasing the knowledge of the mission environment and how the systems interact with it. Enhancing these improves operational capability and increases the effectiveness and efficiency with which missions can be carried out. The HSIA risk, as currently outlined in NASA's HRR, describes how different facets of missions to the Moon or Mars would see decreased operational capacity without improvements to the current architecture used for HSI. HSIA assigns the perceived adverse outcomes to the need for increased crew independence and operational complexity on future exploratory expeditions. Training is one way in which adverse outcomes can be mitigated as future operations are designed to fulfill those needs. This thesis applies that risk to training infrastructures to answer the question "How can the training element be improved to mitigate or avoid these adverse outcomes?"

Knowledge gaps associated with the HSIA risk are defined in the HRR and help contextualize the what kind of decisions to make when designing elements of HSI. These knowledge gaps have been determined to be the primary paths forward to understanding and eventually mitigating the HSIA risk. They describe the background that defines what still needs to be determined and how it applies to the risk, as well as identify the targets for considering the gap to be filled. Four of these in particular are used in the development of SimBAD and influence design decisions to make the tool suitable for future spaceflight missions, the HSIA-201, 501, 601, and 701 gaps.

HSIA-201 calls for a better understanding of the demands of future missions and systems on individuals and teams and how those demands will affect crew health and performance. The specific challenge faced during these missions that this thesis will focus on is the increased distance from Earth, how that affects crew performance, and what improvements could be made to mitigate that.

HSIA-501 calls for a better understanding of how HSI will be used to develop dynamic and adaptive mission procedures and processes to mitigate individual and team performance on earth-independent exploration missions. This thesis focuses on developing a concept for a tool that enables a more dynamic mission design process in both the pre- and in-flight phases. Accounting for unknown mission parameters in real time and having an adaptable infrastructure that allows for said parameters would be a valuable capability to have for these scenarios.

HSIA-601 calls for a better understanding of HSI training procedures, regimens, and standards for all phases of a mission to help reduce demands on crew; support meaningful work during long-duration missions; and mitigate potential decrements in operationally-relevant performance during future missions. This knowledge gap is the most directly applicable to the material in this thesis. Getting a better sense for how to accomplish these is one of the main drivers for the development of SimBAD. Decreasing demand on the workload assigned to crew as well as the training necessary for the crew to be considered ready will be important as these workloads and training requirements increase. Having the capacity to better prioritize or assign these workloads through an improved infrastructure is necessary for future missions.

HSIA-701 calls for a better understanding for how human-automation-robotic systems can be best utilized so that they provide a comprehensive awareness of the state of the crew. This is important in the context of missions with increasing autonomy and crew independence as they will need to be able to acquire and interpret pertinent information to their own capabilities. This thesis focuses on developing an element of the tool that allows this within the training infrastructure. Incorporating this into training would help establish a closed loop for training regimens by providing feedback to the user that enables them to augment their regimen. Having this capability will increase a crew's level of autonomy and is necessary for future missions.

1.2.2 History of Human Spaceflight Operations

The history of human spaceflight training provides important context here, more specifically, how systems engineering and integration had changed to fill the requirements created by the increasing complexity of spaceflight through recent decades. Human Spaceflight began with the Mercury Program, where the astronaut corps was entirely consisting of test pilots. Test pilot culture drove the development of training infrastructure, as most of the human spaceflight operational capabilities were inherited from those programs in the military. One of the main discoveries made during this period was the need for detailed systems knowledge from the operators, and this drove much of the development of operations through Gemini and into the Apollo program. This is where the need for the engineering support team was discovered and reprioritized flight rules, the regulations and procedures adopted for flying spacecraft in various conditions, as the main operational focus of a mission. Even at these early stages of the development of human spaceflight operations, simulations were a critical function in the process[4].

The primary objectives of the Gemini program were to improve upon the operational capabilities needed to conduct rendezvous and docking between two vehicles in Low Earth Orbit (LEO), conduct a guided reentry of a crewed capsule, and conduct EVAs outside of a vehicle. During Apollo, the last operational capability, performing successful scientific operations and explorations tasks in space, was developed and perfected to complete the trajectory of the flight operations team that had been working on human spaceflight up until this point[5]. Interestingly, no broad formal application of systems engineering was administered to the program up until this point to support the growing complexity of the operations being carried out by operators. This changed during Skylab, where the complexity of subsystems became great enough and the interdependence between them was significant, so the need for a stronger relationship between the engineers and operators at

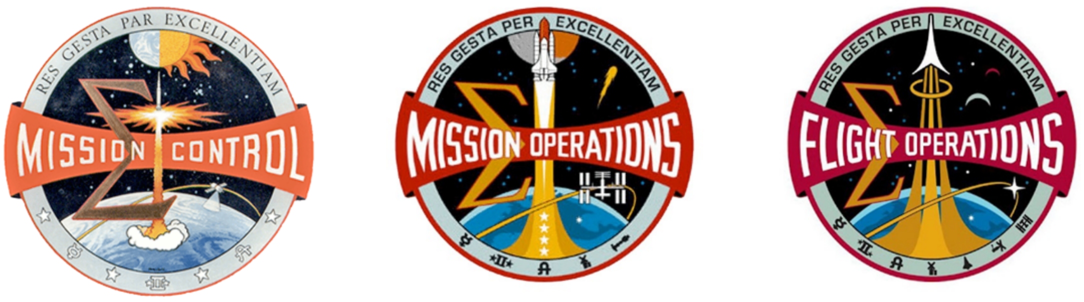

RES GESTA PAR EXCELLENTIAM
MISSION CONTROL
RES GESTA PER EXCELLENTIAM
MISSION OPERATIONS
RES GESTA PER EXCELLENTIAM
FLIGHT OPERATIONS

Figure 1.1: Mission Control, MOD and FOD Logos

every level of the design became apparent. A tangible example of this relationship was between the flight operations team at JSC and the engineering team at the Marshall Spaceflight Center (MSFC), where the teams would conduct reviews of the systems and ensure operational compatibility. The value of this relationship was proven during the various flights of the Skylab program[6].

This would evolve into what would formally become known as the Mission Operations Directorate (MOD) when the Space Shuttle program began development, and later into the ISS program. The Shuttle orbiter was the most systemically complex human rated spacecraft designed up until this point in spaceflight, and so the need for a structure that could provide mission elements such as preparation, training, and real-time operations was evident. Each division within the MOD was responsible for individual integration within itself, as well as horizontal integration with the systems that other divisions were responsible for, as well as with external support teams. After the decommissioning of the Shuttle orbiter, MOD was consolidated with the Flight Crew Operations Directorate (FCOD) into the current group in charge of human spaceflight operations, the Flight Operations Directorate (FOD). The evolution of the logo used to designate each group from each significant era of human spaceflight is shown in Fig. 1.1. This change resulted in a group that was responsible for training the flight controllers as well as the crew. The significant detail of how this structure evolved was that it led to opening up channels towards making the resources between groups more congruent and integrated as missions and the systems that supported said missions became more complex.

1.2.3 The Flight Operations Team

Using the Mission Control Center (MCC) at Johnson Space Center (JSC) as an example for a standard crewed spaceflight program, the flight operations team is composed of the crew, the flight controllers, the training team, and the engineering support teams[7]. In practice there are more organizations that support flight operations such as the Human Health and Performance (HH&P) organization, but the four groups listed are the core that are considered for this research to most directly apply. These groups are all critical to nominal human spaceflight operations for their different roles and responsibilities. Understanding these roles and responsibilities is crucial to understanding how to improve systems engineering, specifically human factors, for future missions. Defining what each role contributes to the program aids in understanding what directions the roles can be taken for future operations to enhance the operations themselves as well as introduce improved Systems Engineering and Integration (SE&I) processes[6].

The crew is the group that receives a majority of support from the other three, being the main operational appendage that carries out the mission. Most if not all aspects of human spaceflight operations are developed around what the crew needs to do their job. The flight controllers are the direct support line to the crew and vehicle. This group consists of the specialists that act as the experts for one or more designated aspects of the mission in support of the overall team. Flight controllers are also responsible for producing operational products, or the procedures, plans, and flight rules for the mission. The training team is responsible for coming up with the training protocols for all necessary team members. This could range from a crew member needing to train for an EVA in the Neutral Buoyancy Lab (NBL), to a flight controller trainee running through an on-console simulation of mission operations. Lastly, the engineering support team is composed of members of the design community as well as mission vendors, whose responsibility is to be able to assist in any situation where context is needed from the source for certain faults or off nominal operations that the other parts of the flight operations group may not have the expertise. Future missions will change the current dynamic by limiting how the crew receives support from the other groups. With increasing distance from the ground, immediate support will become less feasible, which indicates a need for improve the capability of the support available onboard the spacecraft carrying crew.

Increasing the level to which these teams are able to integrate with one another is a necessary change for the flight operations team as missions exhibit increased amounts of operational complexity. As interactions between flight operation systems expand, the systems engineering architecture between these systems will need to be capable of supporting this. Increasing the capacity for which disciplines are able to collaborate with each other during the design process is something that would be beneficial from an operations standpoint. Effectively this would mean increasing the degree to which operators would be able to understand the perspective of their colleagues, which is crucial for when subsystems become more deeply coupled. For example, giving insight into the training team from a flight controller's perspective would make it more likely that the training team would be able to identify faults in current procedures or unnecessary steps. Supplying the engineering support team with a training team perspective would benefit the engineers in designing hardware that is more easily trained on and more intuitive for the crew. There are benefits to be found when collaboration is enabled in every direction inside the flight operations team.

1.2.4 Autonomy, Automation and their relationship to Human Spaceflight

Autonomy and automation are two concepts that are also significant in the discourse surrounding future space missions[8]. In their fundamental terms, autonomy is defined as being attributed to systems that can operate outside of human interaction[9], while automation is defined as the capability for a function to be carried out independently[10]. In the context of a human spaceflight mission, autonomy is defined as the ability for a system to respond to a fault on its own, which can range from the system simply sensing that an aspect of the state of the crew, vehicle or mission has changed, or making a decision on an action to be taken without human input[11]. Autonomy in terms of independence from input from ground support is closer to the scope of this research. Automation in that same context is when any process or function of a mission is carried out by a computer without the crew or ground support. It should be noted that these two terms do not necessarily have to be mutually exclusive, the best example being how an autonomous system could include the capability to initiate an automatic process.

While autonomy is focused on more directly in this thesis, differentiating it from automation is necessary when discussing the human spaceflight systems in this context. There are several dimensions of automation, with four specific stages being defined by Parusuraman, Sheridan and Wickens. These incorporate differing levels of information processing and can vary in levels of individual automation with each stage, ranging from the human making all decisions, to the computer executing decisions based on human approval, to the human being ignored in a process by the computer. The four stages of information processing defined are sensory processing, information perception, decision making, and response selection.

With these terms defined within the scope of this thesis, motivating objectives for the training infrastructure concepts being formulated becomes easier. The knowledge gap defined in HSIA-501 in NASA's HRR most closely describes the way in which autonomy should be conceptualized here. The most relevant distinction to make between the common usage for autonomy and how it'll be used in the context of this research is that autonomy in this context refers to a level of independence from ground support, and not necessarily autonomy from human intervention entirely. This is achieved by creating a training tool that allows crew to build their own simulations and procedures without crew Automation will be present but to a low degree, mainly with the computer executing functions defined by the operator for the compilation of simulations, procedures, and performance data.

1.3 Objectives

This thesis has four research objectives aimed at improving the training infrastructure motivated by the trajectory of human spaceflight operations and supported by an analysis of training within current programs. The final product will be a conceptual demonstration of a virtual training tool and facility that utilizes different elements to meet these objectives:

1. To enable improved collaboration between Flight Operations Teams
2. To have the capacity to account for unknown missions parameters in real time
3. To be packageable on board a spacecraft
4. To include a method for evaluation and monitoring of training performance metrics

With respect to the knowledge gaps from the HSIA risk, HSIA-201 and 601 provide motivation for objective three, HSIA-501 motivates objectives one and two, and HSIA-701 motivates objective 4. The literature review in this paper will build upon these concepts and influence the design of SimBAD, which functions as a UI for SpaceCRAFT[12] and allows users to construct and change parameters in custom simulations. Users are also able to to implement procedures, script events in the simulations, and designate evaluation methods for simulations. Fulfilling these objectives would validate SimBAD as a conceptual demo for a tool to be used as part of a training framework for future human spaceflight missions. With these objectives realized, the contributions of this project are:

- A closed-loop training infrastructure as a virtual training facility
- A simulation tool that provides high fidelity scenarios through scriptable, cascadable events

Closing the loop in this context means having all of the functionality necessary to develop a training regimen that is all operable and executable without ground support. SimBAD offers this by giving the user the ability to create virtual environments, write procedures, script expected conditions, and evaluate simulation metrics for continuous and autonomous iteration of the training loop. Having the functionality of specific events being scriptable within the simulation is a very significant contribution, because with the other elements of the tool it creates a framework that can potentially capture more aspects of the exploration scenarios than other simulation platforms to a larger degree. This creates a unique infrastructure that allows for defining the interdependence between systems and environmental conditions that gives a better situational awareness of a scenario which is especially important when validating operations in the exploration context.

2. LITERATURE REVIEW*

2.1 Previous to Current Training Methods

The established infrastructure for training a crew for human spaceflight operations is effective for supporting the current mission objectives of human spaceflight programs. Those objectives are to support science and maintain a human presence in LEO. The current platform for this used by NASA is the ISS, where astronauts conduct spacecraft-related and science operations that extend our knowledge and capabilities on long duration missions. The ISS is a joint international project between the US, Canada, Japan, Russian, and European space agencies and was constructed using the US Space Shuttle and Russian Proton and Soyuz rockets. The work done on this craft acts as a testbed for new and developing technologies that will take crews to the Moon and beyond. Some of the most important operations carried out are the scientific and extra-vehicular ones in terms of directly contributing to the knowledge base needed for future mission operations. For this reason, many of the training facilities are focused on supporting these aspects of an expedition.

The three facilities discussed in this thesis to describe the current state of crew training methods are the NBL, the VR lab, and the Space Vehicle Mockup Facility (SVMF). There are more facilities used for training than these, and there are others used for training other elements of the flight operations team (such as flight controllers), but these will not be discussed in as much detail because the main focus of this research is to develop a tool for simulation software, and the main capability of these facilities are as simulators. Simulators are historically one of the best ways to train for spaceflight as the conditions of actual spaceflight conditions are difficult to replicate on the ground in a way that important information can be conveyed. Another result of the increasing complexity of spaceflight missions is the broad scope of situations for which a crew must be p repared. Most simulators can only capture a set number of conditions and may be specifically used only to train for specific situations and types of o perations. The NBL and VR Lab focus on teaching EVA and

*Part of the data reported in this chapter is reprinted with permission from "Simulation Builder, Analysis, and Development (SimBAD) Tool through the SpaceCRAFT Platform", William Young, 2022. 2022 IEEE Aerospace Conference (AERO), Copyright 2022 IEEE

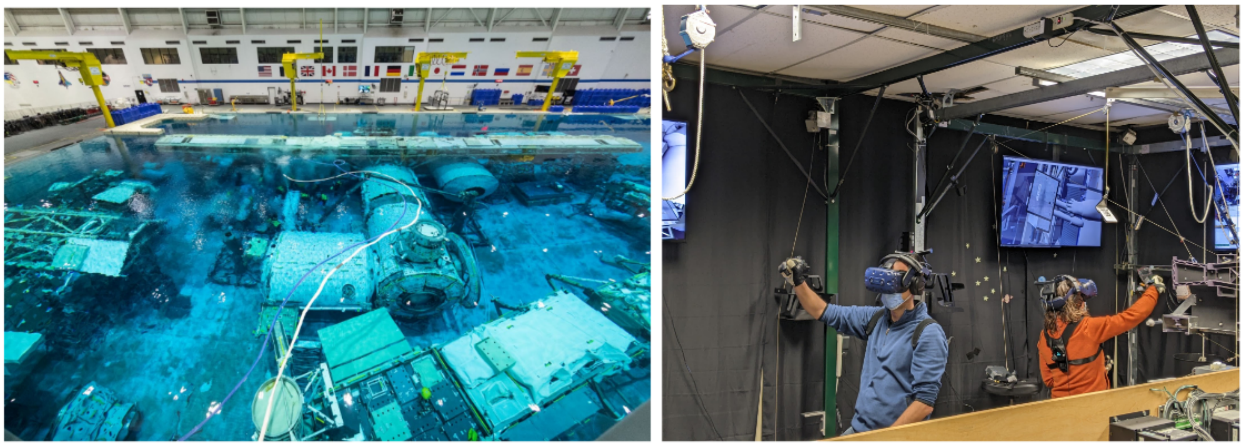


Figure 2.1: NBL at the Sonny Carter Training Facility (left) and the VR Lab at JSC (right)

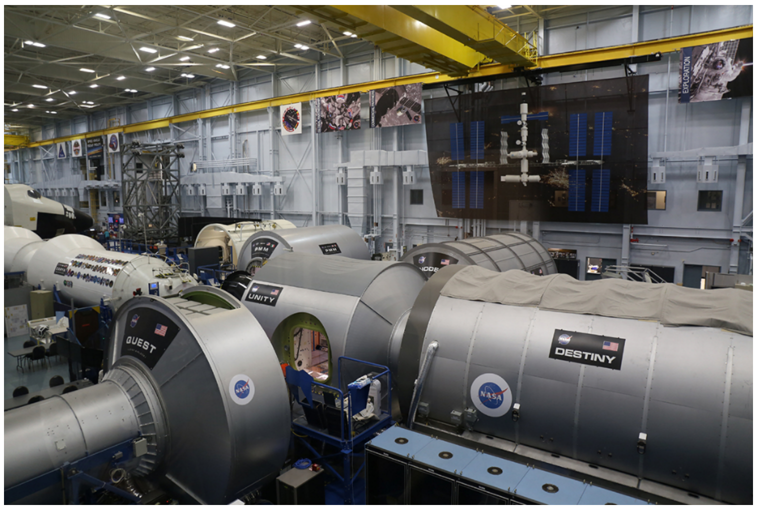


Figure 2.2: The SVMF in Building 9 at JSC

robotic operations skills, while the ISS mockup in the SVMF is useful for payloads, science, and overall vehicle familiarity. Beyond these purposes, these facilities lack fidelity as simulators for the real vehicle and actual spaceflight conditions.

The NBL (Fig. 2.1) is one of the most effective methods for familiarizing astronauts with all the operational aspects of EVAs. Residing in the Sonny Carter Training Facility near JSC in Houston Texas, the NBL contains a 6.2 million gallon pool with a depth of 40.5 ft and a perimeter of 202 ft by 102 ft. This capability is used in the form of submerging large real-scale mockups of space vehicles that can then be used to simulate an environment as close as can be achieved on the ground to a microgravity environment outside a craft. Astronauts in their EVA suits, assisted by a team of divers, can submerge in the pool and use weights to make themselves naturally buoyant, such that the force of buoyancy from the air in the suits is canceled out by the force of gravity. In this configuration, astronauts are able to practice operations that have been planned out for future missions or gain familiarization with EVA operations. Getting familiar with the subtle aspects of an EVA such as the way the suit fits, how to use tools, even translating one's position are all important for crews to gain a better context before flight. There are still some areas of spaceflight that this facility is not able to replicate, specifically actual microgravity as well as IVA operations. While the weights on the suit do keep astronauts fixed to one spot in the water, gravity still affects them inside the suit causing them to remain susceptible to sliding around inside the EVA suit itself. Despite these limitations, the value of the facility in what it offers as a training resource far outweighs those[13].

The VR Lab at JSC (Fig. 2.1) trains for similar elements of spaceflight as the NBL, but focuses on different aspects of those elements which is more suited to the different techniques in simulation. The advantages and disadvantages of this facility are also different from the previous facility, which is another area to compare. Specifically, the VR Lab focuses on supporting EVA and robotic operations training[14]. This is mainly done by displaying a virtual environment for users that can be interacted with with different control input methods for different scenarios. The main advantage of this system is to give the user a high-fidelity three-dimensional representation of the subject of the simulation that is interactable. These physical models and their respective functionalities are developed virtually out of the lab to be displayed inside the sim. This is one of the major differences between the VR Lab and the other two simulators, which utilize the physical mock ups by themselves to enhance the simulator's fidelity. This gives users a more detailed perspective of the environment they're training for, with the virtual environment having a very high level of visual fidelity. This aids in familiarization with the vehicle as well as experiencing different orientations or lighting conditions you'd likely encounter on an actual EVA. Robotic operations can also be simulated in this environment, with virtual models replacing the hardware and the control systems integrated into the simulation so that the movement and behavior is accurately represented. This allows users to familiarize themselves with the robotic controls as well as the context associated with operation inside the ISS, such as line-of-sight from within the cupola module or the dynamics of the robotic arm when commanding movement. Despite all this capability and simulator fidelity, there are still subtle areas in which the facility is limited. One of the biggest problems with doing a spacewalk in VR is that despite the user having all the visual cues for their orientation, their vestibular system will always betray the ocular system by indicating a gravitational pull. While haptic feedback control is available in the VR Lab, accurately designing controls for the feel and touch of virtual objects is still a challenge and finer details, such as the exact shape of hardware, or what the feeling is to press the correct button on an object are almost impossible to replicate in a way that can be done as well as EVA training in the NBL.

The SVMF (Fig. 2.2) is located in one of the buildings at JSC, and houses multiple physical mockups of vehicles as well as some payloads in modules that have been replicated for simulation purposes. The main objective of the facility is to support engineering and mission operations. Besides being a trainer for crews, the facility also acts as a ground analog for systems in orbit, allowing the flight operations team to troubleshoot problems on the ISS from the ground. One type of procedures that are practiced using these mockups is emergency procedures, which are the most important for crew to be able to execute properly. Because the mockups have internal representations of the modules on the ISS, this facility is the most viable for practicing IVA operations compared with the NBL and VR Lab. This is also a facility through which scientific operations can be taught and practiced. One of the issues with conducting science on the ISS is the sheer amount of materials needed to conduct so many experiments. These must be properly stowed and catalogued in order for the procedures that dictate how the science is carried out by the astronauts to be correct. Familiarizing with payload racks and the locations and procedures for different experiments and hardware is important for increasing the efficiency of quiescent station operations and keep crew schedules from becoming unmanageable. Being able to conduct this kind of training and preparation on the ground is very useful to the human spaceflight program when the crew's time is one of the most valuable commodities. There are still limitations to what this facility is capable of in terms of accurate representation of spaceflight conditions, most notably the lack of any kind of replication of a microgravity environment or partial functionality from some parts of the mockups. This does not detract from the capability to help crew familiarize with the internal layout of the ISS or the locations of different hardware and experiments. This also is not an issue for the operational support given through providing resources for comprehensive run-throughs of procedures.

2.1.1 Analog Missions

One of the most prominent methods for evaluating the effects of extreme environments on human systems is by conducting analog missions[15]. There are already multiple examples of analog facilities for different surface operations such as the moon or Mars. NASA's desert Research and Technology Studies (RATS), NASA Extreme Environments Mission Operations (NEEMO), and Hawai'i Space Exploration Analog and Simulation (HI-SEAS) are good examples of current facilities that are working towards simulating these hostile environments and helping develop technologies and operations that can be used when those missions eventually fly. Other facilities focus on simulating conditions like Isolation, Confinement, and Extreme (ICE) environments in order to assess human factors for long duration or long distance missions. The Human Exploration Research Analog (HERA), Crew Health and Performance Exploration Analog (CHAPEA), and Scientific International Research In a Unique terrestrial Station (SIRIUS) programs are examples where the human factors instead of the environment itself are the main focus of the research conducted.

The NEEMO missions in particular have been very valuable experiences for training astronauts to cope with ICE environments while maintaining an operational and scientific focus[16]. The Aquarius habitat (Fig. 2.3), located 12 miles offshore from Key Largo in Florida, is where these analog missions take place. This program was developed by specialists from the astronaut training team in order to give astronauts, engineers, and scientists the opportunity to become "aquanauts" and experience the challenges faced during asteroid, moon, or Mars expeditions. At a depth of 60

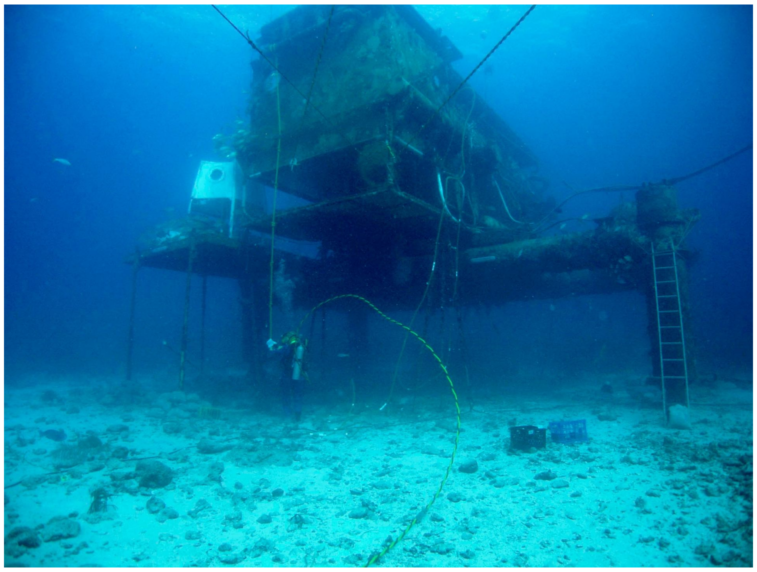


Figure 2.3: The Aquarius habitat used for the NEEMO Program

ft, the base sits at an external pressure of around 2.5 atmospheres, giving a unique added layer of fidelity to the simulated environment, where concerns about the pressure difference inside and outside the habitat exist. The interior of the base is congruent with that of a space vehicle, and is about the same size as the US Laboratory module on the ISS. The habitat is able to support operations from the inside for activities ranging from habitat science to EVA procedures. Recent missions have begun to use this testbed to experiment on different operational techniques for situations with time delays like would be experienced during a Mars Mission[17]. While the results support the claim that operations such as EVA can be conducted meaningfully with these time delays, it is also recommended that methods for planning EVAs that include a degree of crew autonomy are developed.

Other analogs such as the HERA project have done research that give insight into how the human condition is affected by ICE environments and what kind of ways these missions need to be planned and supported[18]. The HERA facility is run through the Human Research Program (HRP) by the Flight Analogs (FA) team of the Research Operations and Integration (ROI) (formerly ISS Medical Projects) element. The facility is a two-floor, four part cylindrical habitat (Fig. 2.4) consisting of a core, a loft, an airlock, and a hygiene module. Communication and autonomy

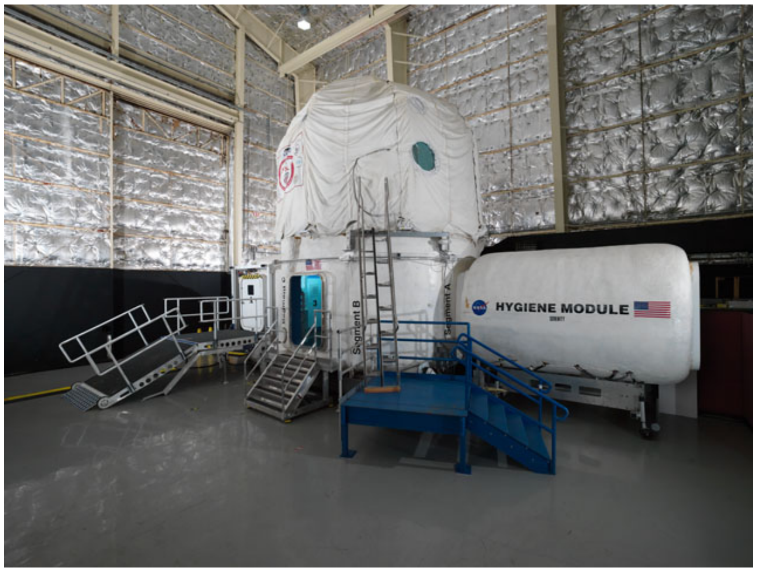


Figure 2.4: The habitat used for the HERA Program

studies are one of the types conducted at this facility as stated in NASA's HERA facility and capabilities information document (2019). Another similar facility to this is SIRIUS, which is located in Russia and while also consists of four main habitation/experimentation modules, the total habitat space is about four times as large (450 m^3 compared to 148.6 m^3 for the HERA habitat). The research objectives are similar, with behavioral studies being done in order to develop countermeasures for the adverse effects of ICE environments, such as in this 2011 paper[19] which looked at the effects of low- versus high-level autonomy in a 105-day Mars mission analog. The results of which showed an interesting effect on the mission control personnel after high levels of autonomy was given to the crew. While the astronauts managed to complete their tasks without a discernible loss in performance, the flight operations team supporting the mission reported higher levels of anxiety. The effects of autonomy on things like emergency procedures was not within the scope of this research.

2.2 Future Training Methods

As more radical changes are made to the infrastructure used to conduct human spaceflight missions, upgrades and new methods will be needed to the training infrastructure in order to support that infrastructure. The two most pertinent concerns to training and testing required are the level of complexity and interdependence of the systems and the lack of communications capability as the distance from Earth increases. With the first concern there are two potential solutions to this based around the same assumption, which is that in some form the knowledge to sustain the vehicle and her systems will all need to be on board. While a connection to the ground will likely be maintained even despite large gaps in time between data going to and from the vehicle, any fault with the communications system would leave the vehicle unable to sustain itself if any high priority operational capability is left on the ground.

The first solution is to develop an onboard infrastructure that allows for the vehicle to be operated autonomously as well as carry out automated procedures. To briefly reiterate, the vehicle's capability to operate autonomously would mean being able to maintain an awareness of the status and health of the vehicle and react to changes, while automation in this sense means the capacity for a procedure to be carried out without manual aid. There are also two different levels of autonomy that could be discussed in this situation, which will be elaborated on in the procedure design section of this thesis. This solution is more dependent on the vehicle's hardware itself and the design than the crew, which would lessen the operational load on the crew and allow them to focus on more important crew dependent tasks. The other solution would be to develop a system that relies more on the crew in the sense that most vehicle functionality will become a crew responsibility, especially once a considerable communications lag is being experienced. Realistically a combination of these solutions would be best, considering the benefits and drawbacks of automatic vs manual operation in this context. Taking as much operational load off of the crew as possible would decrease the amount of training needed to prepare crews for flight, but would increase the complexity of the system managing autonomy and automation for the vehicle.

One important component of training for these future missions is a result of the intertwined human factors associated with future spaceflight missions, particularly the difference between team and individual training. SimBAD offers the potential for increased crew independence from ground support for training. Having a closed loop for training and iterating through different regimens would decrease the dependence on personnel being necessary to support training conducted in-flight. Crew cohesion will be an important human factor for these missions. Like with the subsystems of a vehicle, the crew themselves will likely be interdependent on each other which provides its own set of questions to be answered. A 2015 NASA technical report[20] identifies different competencies tied to different training requirements, those being a combination of task and team specific and generic requirements. The work in this thesis will set up a testbed for supporting some of the training guidelines identified and listed in the study, particularly computer-, simulation-based, and self-paced training.

One likely augmentation to training for future missions is to conduct more comprehensive training while in transit. A key logistical note to make is that a vehicle carrying a crew to another celestial body will likely prioritize transportation as opposed to science such as with the ISS. Especially during longer missions, there will likely be significant downtime during quiescent ops. This is a good opportunity to utilize on-board training methods, which can help hone skills and techniques of crew members for different operations, teach new information, or assist in Just-in-time training preparations. A potential method for this is gamification[21][22], where crews would use game-like elements in order to supplement training and knowledge retention. Utilizing techniques like these within the limited space and capability afforded by a space vehicle would compensate for the setbacks encountered from the environment.

2.2.1 VR Training

Simulators that utilize mockups are historically a very effective way at conveying significant information, familiarizing users, and practicing skills on hardware. Some of the limitations of this method for achieving these objectives, specifically for the NASA human spaceflight program or government-funded programs in general, are the financial and logistical risks they pose as they become more viable for training. For example, a simulator like the NBL gives EVA operational experience that is not replicable by any other simulator where astronauts can don the EVA suits and simulate motion like that of a real spacewalk. The drawback is that the facility requires a large amount of resources to maintain, from the space needed to house the pool and the vehicle mockups to the manpower required in the form of divers who support the simulated spacewalks. The biggest advantage of using VR to simulate space environments is that the resources needed to maintain a virtual environment is significantly less compared to a facility like the NBL. This is the purpose for having the VR lab at JSC, as well as expressing the capability to test and train other hardware that cannot be simulated in other situations such as the Simplified Aid For EVA Rescue (SAFER) system, the mass handling system Charlotte, and robotic environment[23].

Research in the field of VR training has been done to support the conclusion that it suffices as a viable training method, particularly in the ways that it is currently being utilized for. Virtual EVA training with a haptic feedback system in order to simulate the sensation of handling mass while in a microgravity environment[24][25] is one example. Another paper studying the usage of VR in conjunction with telerobotic operations as it pertains to refresher training [26] builds upon that training done at JSC's VR lab by applying it in a way that could be useful to long-duration missions where refresher training would improve knowledge retention. Other research is also being done outside the capability that the VR lab typically developed, such as in the area of IVA operations as opposed to EVA and robotic ops. Cubesat building was used as an example in a 2020 paper[27], which demonstrated the usefulness of preparing for a simpler task such as building a cubesat. This research also supported the conclusion that VR technology could be used to support on-orbit refresher training. Other elements being researched that would provide addendums to the practicality of VR in the training applications for human spaceflight are collaborative training and the simulation of failure modes[28]. Understanding the different uses of virtual simulations will help build a basis for the development of the SimBAD VR training platform in this thesis.

While there is plenty of research supporting the use of VR in general in different training scenarios, one major aspect of this emerging technology that should also be considered is the nature of the virtual environment itself. A VR headset is not the only way to display virtual information, different methods of doing so and understanding the pros and cons of each can provide insight that helps design a better environment for whatever virtual training method gets flown. One example is a virtual environment using walls and projectors instead of a headset[29], which while is not as desirable given the space requirements for flight conditions, does present an alternative option for virtual training on the ground, which could be useful considering most VR headsets have historically been known to cause motion sickness due to the brain processing the different signals coming from the ocular and vestibular systems. This study also reinforced the necessity of high levels of "presence", which will be elaborated further in section 2.3. Another example of a unique application of a virtual environment was the Human Engineering Modeling and Performance (HEMAP) lab[30], which was run at Kennedy Space Center (KSC), and was developed in order to investigate the interaction between crew, ground support, and spaceflight hardware. The facility works very similarly to a motion capture studio, where users put on black suits with white dots on them, and the environment is constructed out of PVC piping. Sensors track the environment and the personnel, and are able to send data to computers that interpret what's been captured and construct a virtual environment with it. The biggest factor here that is different from regular VR is that the ones who are viewing the virtual environment are not the crew themselves, but the support engineers running the simulation. In this situation, the users are effectively the support engineers and not the ones interacting with the environment directly. This is another case where the technology is better suited for ground than in-flight operations. The significance to be taken from this technology is that this kind of system invokes a more involved amount of cooperation between support engineers and the crew who is actually embarking on the task, where the support team has more insight and a more direct perspective on the procedures and gain a more valuable operational awareness.

In environments dedicated specifically to educating users on tasks, particularly technical ones, VR has been shown to allow for higher usability than with standard teaching methods [31]. This area of training gets more into the difference between classroom training and simulations. A lot of knowledge is transferred to crew through classroom settings, where material is presented directly to the students, normally with checks for comprehension at certain milestones such as quizzes, tests, or checkouts. However, this is not the most effective method at knowledge transfer, and is conducted due to limited time and resources for planning any other method. VR offers a solution helping users learn the same material with a much more interactive and engaging experience and could be used to educate crew members in orbit on topics that normally would have been learned in the classroom setting on the ground, which would save the flight operations team time and resources.

2.3 Simulations

Simulators have several aspects about them that make them viable for different situations when training operators. Knowing what those aspects are helps design more effective simulators that fulfill different elements of a mission, whether that be procedure familiarity, environment familiarity, or knowledge of a specific skill or technique. The medical field is one in which simulators are widely used and the level of expertise being taught is typically on par with the skills and techniques being taught to astronauts. Operators in both fields need to have a well-refined skill set in order to be able to carry out their jobs, and also be able to use these skills in high-intensity situations. For this reason, studying the way in which professionals in the medical field define their simulation methods is a good way to understand techniques that can be generally definitive, and then reapplied to human spaceflight applications.

A good place to start defining a simulator is with the five main aspects that compose a simulation. These are Fidelity, Reliability, Validity, Learning, and Feasibility [32]. Fidelity is defined as the degree to which the simulation resembles the real-life experience. Reliability is the measure to which the performance can be consistently recorded through each iteration of the simulation. Validity is the whether the method of simulation has construct, concurrent, or predictive validity, or how much trust can be put in the method for providing results that will correlate to actual performance. Learning is simply defined as whether or not the trainee actually learns what they were supposed to from the curriculum. Finally, Feasibility is the degree to which the simulation method is affordable to implement considering the resources required for the specific method.

Another paper introduces an interesting concept known as the “Miller’s Pyramid” [2], shown in Fig. 2.5, which is a categorization technique used to discern the level to which a trainee has acquired a skill or identify what a certain simulation method is meant to convey. The bottom of

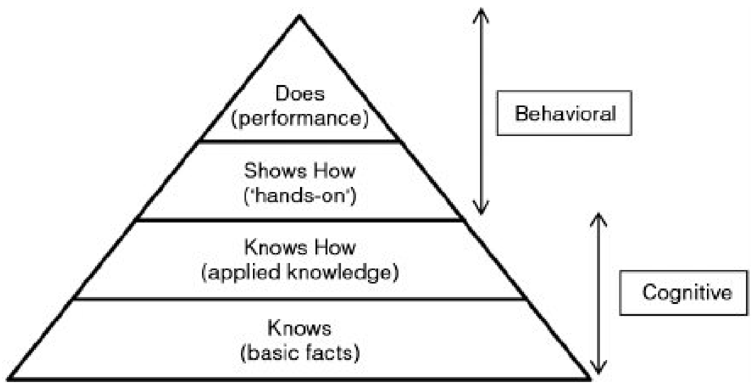


Figure 2.5: Miller's Pyramid, which categorizes different levels of understanding [*Reprinted with permission from [2]

this pyramid is the "knows" region, where the most basic knowledge of a concept resides. Further up is the "knows how" region, where the knowledge of how a concept works exists. Second from the top of the pyramid is the "shows how" area, where actual demonstrations of a concept would happen. At the top is the "does" section, this is where the trainee would be able to actually execute a concept. To relate these to spaceflight concepts, the "knows" and "knows how" regions would be covered by training materials such as classroom learning, quizzes, tests, and checkouts. The "shows how" section would be demonstrated through simulations or other scenarios in which a trainee would be required to exhibit the knowledge they're supposed to have learned in an environment that is not the real operation environment. Which leaves the top of the pyramid, the "does" block, which would be the trainee taking their knowledge and experience and applying it in the real life situation. Another paper on clinical skills produced some interesting findings with the correlation between the level of expertise of a user and the level of complexity of a simulation relative to the user's experience level[33]. Simply put, simulations developed for professionals would not be effective in the hands of amateurs, and simulations developed to be simple for beginner-level users would not be useful for professionals. This is significant when considering astronaut training methods, because astronauts, even astronaut candidates, are highly-qualified individuals. However, some skills still require an acknowledgement that the users will be beginners and the simulations must be tailored accordingly. This means that the level of complexity of a simulation must be honed to a more sensitive degree, such that astronauts with more experience can find the simulation useful, and the astronauts with less experience are able to learn and develop their skills.

2.4 Procedure Design

In human spaceflight, procedures have become some of the most important forms of documentation for programs and an integral part of the flight operations team's nominal routine. Between the three main operations products that are developed by flight controllers, procedures are the ones that deal with the most day-to-day operations. The purpose of Flight Rules are to dictate in what fashion a flight will be conducted with constraints and requirements, and flight plans indicate the pre-determined schedule for which activities in the flight will take place and in what order. Procedures describe in detail how these activities need to be carried out and give the highest level of detail for human spaceflight operations. The development of procedures is normally dictated by specific activity requirements and Flight Rules, which eventually propagates down to operations. Designing procedures for singular subsystems is relatively simple, but when developing them for more complex subsystems that require a high level of cross-system knowledge, development becomes more difficult proportionally.

Procedure design is where the topic of autonomy and automation are also brought back up for elaboration. While both concepts have been defined, there is a lower level of definition for different types of autonomy and automation that gives a better understanding of the potential operational conditions that could be experienced during Long Duration Spaceflight Missions (LDSM). Specifically, there are two levels of autonomy to be discussed in order to understand how different procedures could be designed for different types of activities. The first is a relatively high level of autonomy where the vehicle and the crew are autonomous, where the crew is considered part of the vehicle's operational capacity. The second is a lower level of autonomy where the vehicle is considered separate from the crew and would be able to notice and react to faults on its own without crew intervention. Understanding the difference between these two levels of autonomy is helpful to understanding and categorizing different procedures based on the necessity of the crew, therefore optimizing their time for more crucial duties. In situations where the higher level of autonomy is administered where the vehicle and crew are independent of ground support for however long, the scenario where the crew themselves design a procedure becomes a possibility for consideration. Unorthodox situations like this should be considered and reconciled before these missions fly.

With regards to automation, infrastructure for running procedures without support from a crew member has existed on the station for most of its operational lifetime. A vehicle that complex being run completely manually would require more personnel than can fit on the vehicle. This alone proves the need for reliable and widespread automation on a vehicle traveling long distances. For the Shuttle and ISS, the timeliner tool[34] has historically been the means through which procedure automation is carried out, typically with payload and core systems. Flight Controllers used this tool to write scripts for sequences of actions and events that would be sent as a chain of command to the hardware onboard the shuttle/station which would be able to then read and enact those commands automatically and in proper succession. The tool is a scripting language that has syntax similar to that of what would be encountered in a manual procedure, making readability and learnability easy for users already familiar with that format. This characteristic of the tool is important when considering other tools that build procedures, even ones that aren't automatic. Creating a user interface that is congruent with existing procedure nomenclature is important for the tool's usability, especially for a human spaceflight program such as NASA's. In a similar vein, procedures that can be built by the crew members themselves would also be something useful for a human spaceflight mission, and so half of that struggle would be creating something that is robust and easy to use. A 2018 thesis noted several aspects of designing electronic procedures that would make them useful for in-flight situations, specifically ones that would require Just-In-Time Training (JITT) [35].

2.4.1 Emergency Procedure Design

In the hierarchy of what elements of human spaceflight take the most priority, ensuring crew safety will always come first, secondary to that is the vehicle health and status, with mission success being the third and lowest priority of those three. With this hierarchy in mind, emergency procedures become some of the most important operational products that come out of the flight operations team. Emergency situations are difficult to anticipate and properly prepare for, and crews are also not likely to have to deal with them in flight. Because vehicles are designed with avoiding hazardous situations in mind, these situations rarely occur but are frequently trained for. This causes a potential for a disparity in the crew reaction during a procedure and the crew reaction during an actual event. There are several ways to avoid this. One way is to simulate an emergency situation randomly inside of a simulation of nominal operations, catching the crew off guard and eliciting a genuine response. One of the best ways to prevent that is to create a simulation that is as close to the hazardous event without becoming actually dangerous, in order to encourage the proper reaction and state of mind from the crew for them to deal with events. VR is a good method for doing this, as sights and sounds can all be simulated with a high degree of fidelity and incur the proper response from crews and prepare them for the situation they might end up encountering.

2.5 Long Distance Spaceflight Missions Human Factors

In long-duration transits, there are several physiological effects that should be noted when designing missions to far-off destinations in order to properly ascertain the state of the crew during all phases of the expeditions. The two most significant risks when it comes to crew performance are crew behavioral and mental health, and physiological health detriments in the form of muscle atrophy caused by the microgravity environment. In both cases, the ability of the crew to perform their job is diminished and affects mission outcome. While the effects of microgravity on the muscles and bones is more well known and counteracted by consistent exercise and simulated loading on specific muscle and bone groups, the mind is less well understood. Countermeasures are currently being researched, but likely one of the most helpful methods for counteracting a decline in the mental state is consistent exercise, similar to how muscle atrophy is mitigated. ICE environments are particularly stressful on a human being when it comes to mental health, which is why carefully designing the environment itself is important to ensure the crew can manage themselves for the entire duration of the mission. This means everything from the subtle differences in different lighting conditions to stimulate the body's circadian rhythm, to choosing crew members that engage with the other members well in professional, personal, and even cultural senses. Because the environment is so important, it follows that manufacturing a virtual one for the crew is a potential countermeasure being considered. Showing users images that aren't of a cramped, sterile spacecraft but of their home or a more natural environment could help offset some of the negative effects on mental state induced by spaceflight. VR could also be used to maintain mental acuity, especially for tasks that might be lower priority for the mission, and therefore easier to forget along the way to the destination.

2.6 VR Hardware

In the last several decades, the capability of graphical representation of virtual objects has exploded exponentially and opened up many opportunities as to what they can be utilized. Understanding the hardware for VR is necessary to properly ascertain what the capability is for use in spaceflight scenarios, and arguably more importantly is analyzing how this hardware could integrate with a vehicle and become flight certified. Two of the biggest concerns with the hardware are power draw and thermal output. VR systems utilize some of the most visually acute graphics systems to date. The Graphics Processing Unit (GPU) of the system dictates the visual acuity of the simulation, but while a better GPU will result in a higher level of fidelity, it will also result in a higher power draw. This is also true for the Central Processing Unit (CPU), which performs the computational work for the system. The effect on the power is similar, but the contribution to the simulation is different. Instead of enabling better visual acuity, the CPU will dictate how fast the simulation can run, and how many subsystems in the background are able to be simulated. A worse CPU means you might not be able to properly perform the computations needed to propagate a simulated ship's movement, for example.

Most VR systems, like the ones used for some activities carried out in the VRL, are composed of three components: The headset, controllers, and infrared "lighthouses" that track the position of the headset and controllers. The Vive system depicted in Fig. 2.6 shows how this setup can look. A VR system translates the infrared sensor data communicated between the headset and controllers to output position data. That position data is then sent to the computer to be interpreted in a virtual environment and displayed back to the headset through the GPU. Depending on the system a user

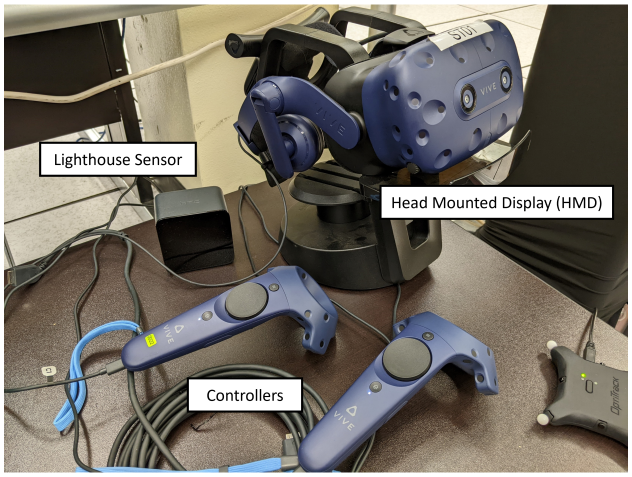


Figure 2.6: Vive Pro 2 wireless VR system with hardware elements annotated

may experience up to 6 Degrees of Freedom (DOF) within the virtual space. A system with 6DOF tracking shows the full 3 dimensional position and yaw-pitch-roll orientation of the headset and controllers.This assists in creating a sense of presence as minute head movements can be tracked and displayed to the user. Other technologies such as eye tracking or haptic feedback can help mitigate issues with nausea or increase the amount of sensory immersion inside a simulation.

2.7 SpaceCRAFT

The simulation platform that the SimBAD tool is based on requires some definitions and context in order to have a proper discussion of what SimBAD is adding to the current platform. This context is also useful to comment on what can be upgraded in future versions of SpaceCRAFT to make it more useful for the objectives that SimBAD is focused on fulfilling. SpaceCRAFT can be defined with two vital components and the supplemental architecture that supports those components. These two components are the Compute_Server and the Unreal Engine 4 (UE4) client. The Compute_Server is the code responsible for running the simulations from a simulation configuration, or ‘sim config’ file, as well as performing all of the necessary calculations prescribed by the user set systems and entities. The UE4 client element of the platform utilizes the graphical processing power of UE4, which visualizes all of the models and environments. The interaction between these two components is driven by the Compute_Server, which sends data to entities in-

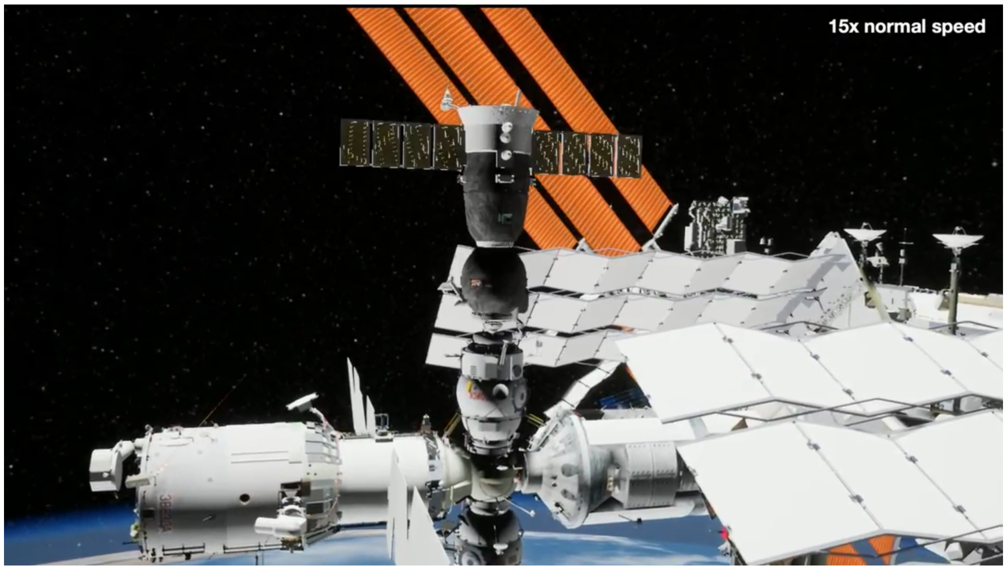


Figure 2.7: SpaceCRAFT Simulation with ISS and Soyuz models for Proximity Operations Analysis

side the simulation, which affect different entity parameters based on the situation being simulated. This interaction is shown in Fig. 2.8. SpaceCRAFT also has a multi-user capability that SimBAD takes advantage of to apply the simulation builder concept to a collaborative level, which is one of the objectives for this research

SimBAD is adding onto the current capabilities of SpaceCRAFT in order to help develop a professional version of the platform. The main goal of the tool is to create a functional user interface for creating sim config files. Currently the only method for creating these files is manually, which is a non-intuitive process given SpaceCRAFT has a specific .JSON file schema that the Compute_Server can interface with. Adding this functionality is essential to meet the requirement of allowing users to have the freedom to build their own space environment simulations. The other goals are secondary, and deal more with adding operational functionality to the platform from several different angles. There are three secondary components that SimBAD adds, procedure management, failure mode management, and procedure evaluation methods management. While SpaceCRAFT is able to simulate a myriad of different situations under different conditions, there is currently not an infrastructure for adding in procedures to run through in simulations. Adding

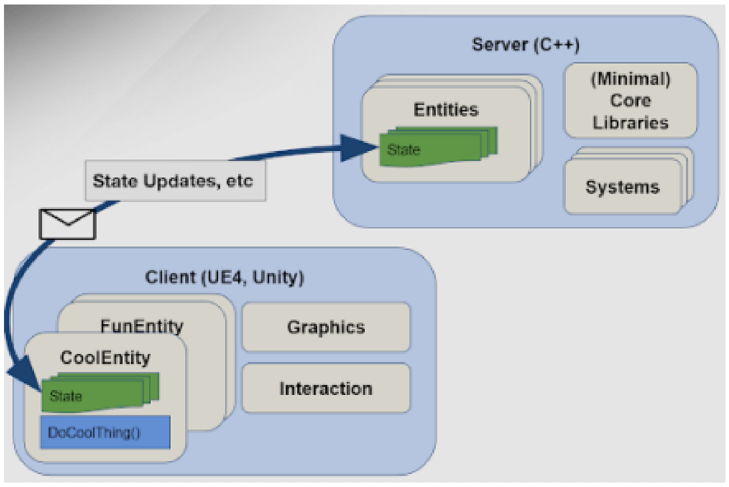


Figure 2.8: Diagram of SpaceCRAFT Platform Architecture

this would give the platform a lot more operational flexibility in what it could potentially be used for in the realm of human spaceflight systems integration and training. A real-time event manager would further increase this flexibility, adding onto the platform's event-driven system that could be responsive to different environmental scenarios affecting the systems in a sim and adding a necessary level of realism. Procedure Evaluation methods would be a useful addition for tracking different metrics inside a simulation, and allow for the post-processing of data as well as easier quantification of user performance while executing a procedure or mitigating a failure inside of a simulation. Overall, these secondary additions would greatly append onto the operational capability of the platform and increase the viability of general use for public and private space entities who are aiming to expand their human spaceflight program.

3. SIMBAD CONCEPTUAL DESIGN*

SimBAD is a toolkit that utilizes multiple smaller individual tools that compartmentalize the overall functionality into more manageable and understandable groups. These are all composed of UE4 UI, UE4 C++, SpaceCRAFT API, and JSON components to different degrees depending on the objective of the element. Splitting up these elements into parts helps manage the workload for the project, as well as support nomenclature that is easier and more digestible for users. Having each element contain its own separate sub-functionalities is a helpful choice when debugging problems inside the codebase itself and help construct a more focused approach to the development of the element. The framework this helps install is also useful for helping users understand the SpaceCRAFT platform and how sim config files are used inside the architecture, as well as assist in visualizing how SimBAD works at a high level. Having an improved method for Human-Computer Interaction (HCI) would assist implementing an effective HSI on future spaceflight missions that would benefit from increased levels of autonomy.

One facet of the current state of SpaceCRAFT is how simulations are initialized inside the current front end of the platform, which takes pre-built sim configs. This is a limit on users as there is not currently any method for editing these configs inside any of the UI in SpaceCRAFT. Users previously were required to understand the JSON schema used by the platform that gets parsed into the Compute Server and edit those files directly. This added more layers of complexity than is viable to SpaceCRAFT's objective of being more usable than current space environment simulators. SimBAD adds functionality that takes those layers of complexity away from the user's experience by appending onto the front end a UI that takes user input that creates and edits sim configs before the simulation has started. The functions in SimBAD are called using the UE4 C++ API, and depending on the specific function uses other libraries for more specific actions such as reading data, writing or formatting data.

*Part of the data reported in this chapter is reprinted with permission from "Simulation Builder, Analysis, and Development (SimBAD) Tool through the SpaceCRAFT Platform", William Young, 2022. 2022 IEEE Aerospace Conference (AERO), Copyright 2022 IEEE

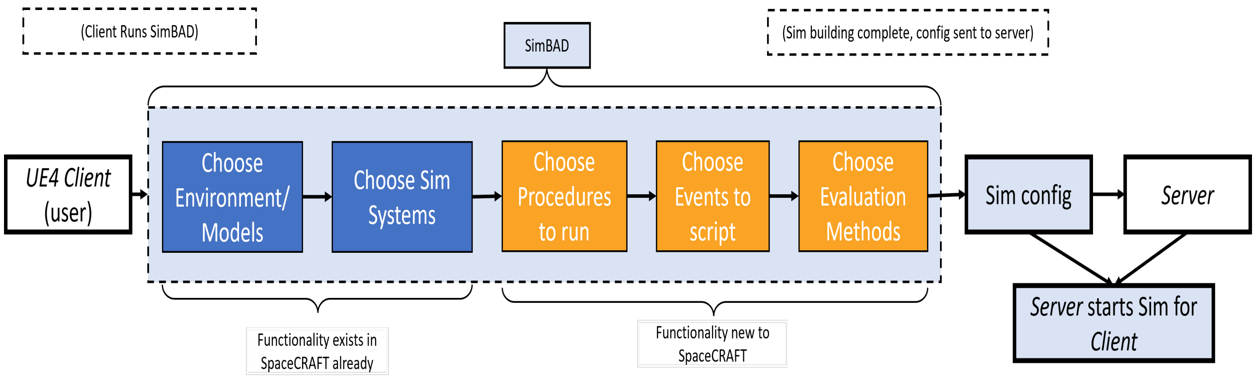

(Client Runs SimBAD)
SimBAD
(Sim building complete, config sent to server)
UE4 Client (user)
Choose Environment/ Models
Choose Sim Systems
Choose Procedures to run
Choose Events to script
Choose Evaluation Methods
Sim config
Server
Server starts Sim for Client
Functionality exists in SpaceCRAFT already
Functionality new to SpaceCRAFT

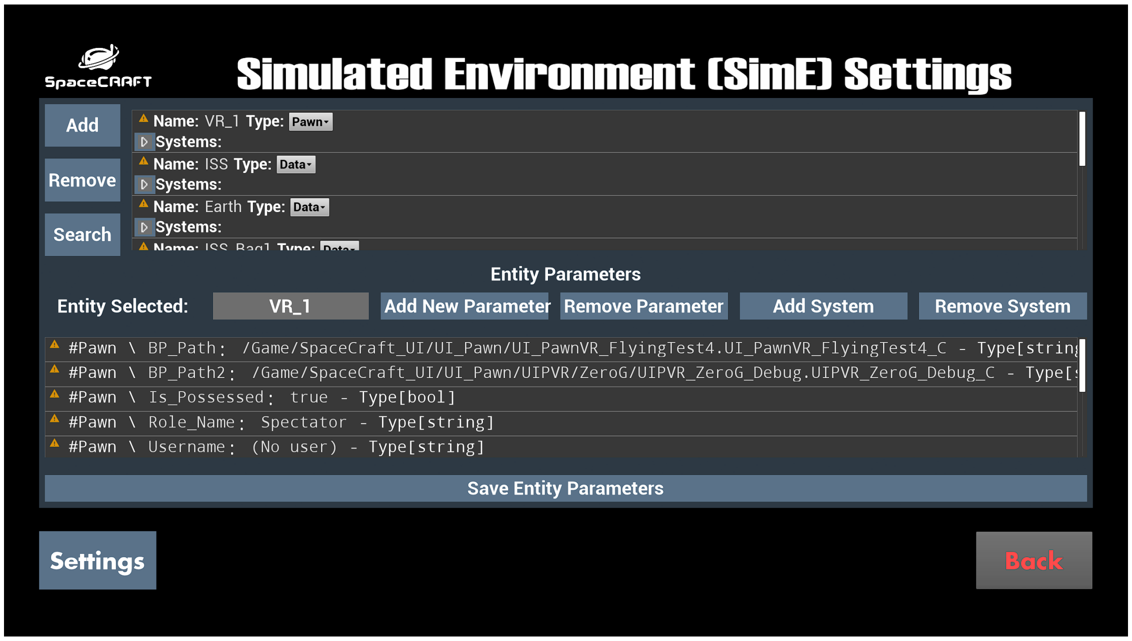
Figure 3.1: Block Diagram showing the place SimBAD fits in the architecture used to run simulations

The timeline for how simulations are initialized and run inside the platform is also important for understanding at what point along that timeline SimBAD fits best. As an example, the current approach for starting simulations is exhibited in the Space Teams activity, which is a program that uses SpaceCRAFT to teach students about spaceflight and exploration using different pre-built simulations loaded onto a compiled version of the platform. On the surface level this version mainly consists of the graphical side of the platform, or the UE4 front end and simulations themselves. The difference between the platform here and other projects built using UE4 such as video games is the Compute Server. This has been compiled separately but lives in the same file directory as the packaged SpaceCRAFT executable. The Compute Server is able to override the game engine's computational processes in places where they incur a large amount of approximation error that is not conducive for scientifically accurate simulations. In the Space Teams activity, the user is presented with a front end that allows them to choose between the different available simulations. When a simulation is chosen, a command is sent to both the game engine and the Compute Server to start the simulation on both sides of the platform. At this point the Compute Server is reading and parsing the sim config data, preparing systems to run their processes and wiring those to entities and their data before sending those entities back to the game engine to be loaded and rendered. The game engine, after running the command, loads a different, empty level that has an engine-based Entity Manager object loaded into it that takes in the entity data from the compute server and then loads in the corresponding engine assets and renders them for the user. At this point the simulation has begun and the quiescent backend processes handle the behavior of the objects inside the simulation.

While SimBAD is built upon the SpaceCRAFT platform, SimBAD is a higher level tool that provides the functionality necessary to implement a virtual training facility. SimBAD utilizes resources from the SpaceCRAFT platform, appends new components, and creates new functionality that enables integrated virtual crew training on the ground, in space, or simultaneously from remote locations. Specifically with regard to entities and systems, SimBAD adds new methods through which users can interact with them as well as utilize them within the other elements. Some unique features that are core to SimBAD's capabilities are procedure scripting and integration with VR simulation, complex scenario design and execution using a new Event Manager, and simulation evaluation tools to assess performance during a particular simulation. The amalgamation of these elements creates a closed-loop infrastructure that operates as a virtual training facility. This is the essential contribution of this work. The SpaceCRAFT platform is the vehicle through which SimBAD achieves this, but the concept could be applied to other platforms. Like with coding projects, although a script in one language can't be used in another, the overall algorithm can be transposed as long as the underlying concept is understood.

3.1 Simulation Environment (SimE)

The SimE element is composed of the simulation "entities", which is a term that comes from the SpaceCRAFT API and refers to all objects that will be spawned inside of a simulation. The SimE manages these entities as well as the relationships between them, or with the systems that are declared in the SimSys element of the tool. This occurs before the simulation startup as mentioned in the previous section as editing the entities focuses mostly on the sim config file itself, which cannot be edited mid-sim. This means that the user engaging with SimBAD occurs after the program has started running but before any simulation has begun. The UE4 client is the component that contains the UI for the user to engage with and displays the current sim config information and has controls that can change this information.



Name: VR_1 Type: Pawn
Systems:
Name: ISS Type: Data
Systems:
Name: Earth Type: Data
Systems:
Name: ISS_Bag1 Type: Data

Entity Selected: VR_1

```
#Pawn \ BP_Path: /Game/SpaceCraft_UI/UI_Pawn/UI_PawnVR_FlyingTest4.UI_PawnVR_FlyingTest4_C - Type[strin
#Pawn \ BP_Path2: /Game/SpaceCraft_UI/UI_Pawn/UIPVR/ZeroG/UIPVR_ZeroG_Debug.UIPVR_ZeroG_Debug_C - Type[
#Pawn \ Is_Possessed: true - Type[bool]
#Pawn \ Role_Name: Spectator - Type[string]
#Pawn \ Username: (No user) - Type[string]
```

Figure 3.2: SimE menu showing listed entities in the upper dialog and their parameters in the lower dialog

Entities are data structures that consist of a list of parameters that describe the state of the system. Some of these parameters are required such as the name, entity type, and system name tags, which are present in all entities regardless of their type or purpose. Besides these and some others used for engine purposes, parameters are specific per entity and can be customized based on their function in the simulation. For example, an asteroid in a simulation may have a set location and velocity to dictate its motion, and a spacecraft in the simulation may have a relative location and velocity as well as other parameters not necessary in the asteroid entity such as power, oxygen, thermal properties, etc. Displaying entities information in the UI requires for the background functions of the tool to parse that information from the JSON file and put it into a format that can be read from, Sim Config, to SpaceCRAFT code, to the UE4 UI.

There are two areas in which the entity data is displayed. The first is a list of entities and their required parameters (name, type, and system name tags), which gives a concise view of what is going into the simulation. The second is a list of the parameters given a selected entity that the user can alternate between and edit at any point in this stage of the simulation building process. Both of these lists in the UI are editable, so the user can create, edit, or remove entities or their parameters at any time.

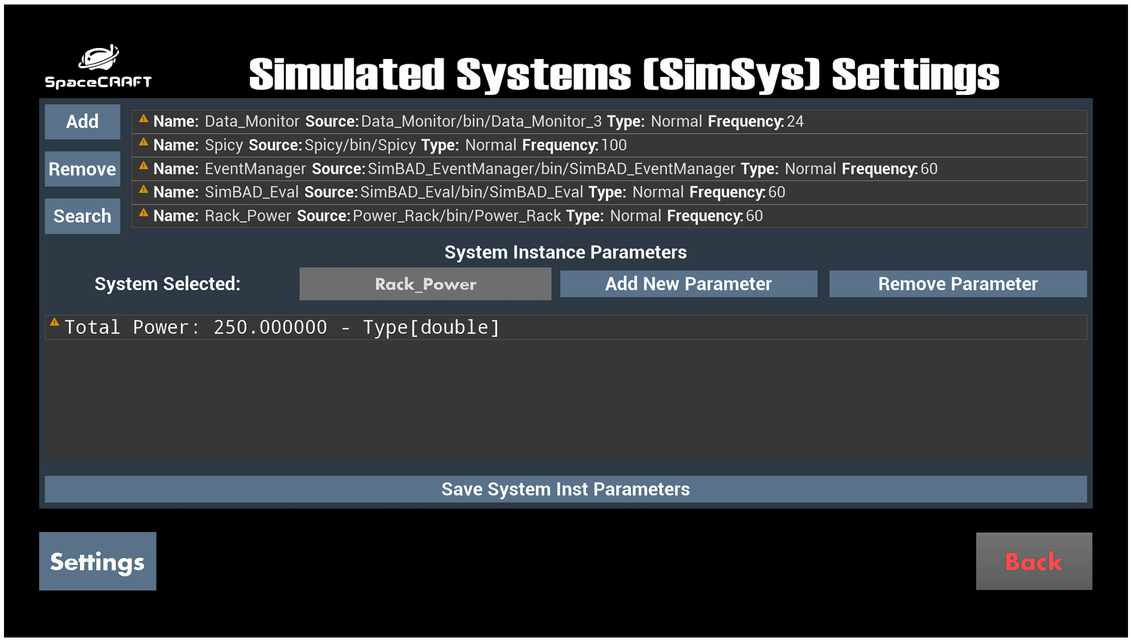
3.2 Simulation Systems (SimSys)

The SimSys element of SimBAD is mostly similar to the SimE element, with the key difference being the focus on the SpaceCRAFT Systems as opposed to the entities. SpaceCRAFT Systems are scripts containing custom functionality that will be carried out by the Compute_Server during runtime of the simulation. Systems dictate how the entities interact with one another and enable the functionality to allow the environment to operate as written by the user. Examples of the kinds of behaviors that systems are responsible for is power being supplied to a module, CO2 input and O2 output from an oxygen generation device, or the gravitational field of a celestial body. The SimSys element manages these systems and their relevant attributes and displays their info in a way for the user to easily access and edit these during simulation or procedure testing and iteration. Like with the SimE UI, the top dialog has users select, add or delete systems from the config with the bottom dialog giving users the ability to select, add, delete, or modify the system's instance parameters.

3.3 Simulation Procedures (SimP)

Enabling the use of procedures in the simulation adds a high level of viability for the Space-CRAFT platform and SimBAD as a tool. The SimP element of the tool is used to allow users to create and edit procedures of their own to use inside of the simulations themselves. This is one of the new additions to the SpaceCRAFT platform and enables users to use it as a means to assist in the development of ops products. This helps achieve one of the main goals of the platform, which is to be a resource in human spaceflight operations. The format of how SimBAD incorporates procedures is through a new JSON schema, which is separate from the sim config JSON file. This element deviates from the first two in that way since it is independent of the sim config schema, but because it is still a JSON file much of the same logic can be used in the functions that read and write to the procedure files.

The SimP dialog (Fig. 3.4) allows users to create and edit their own procedures to be used

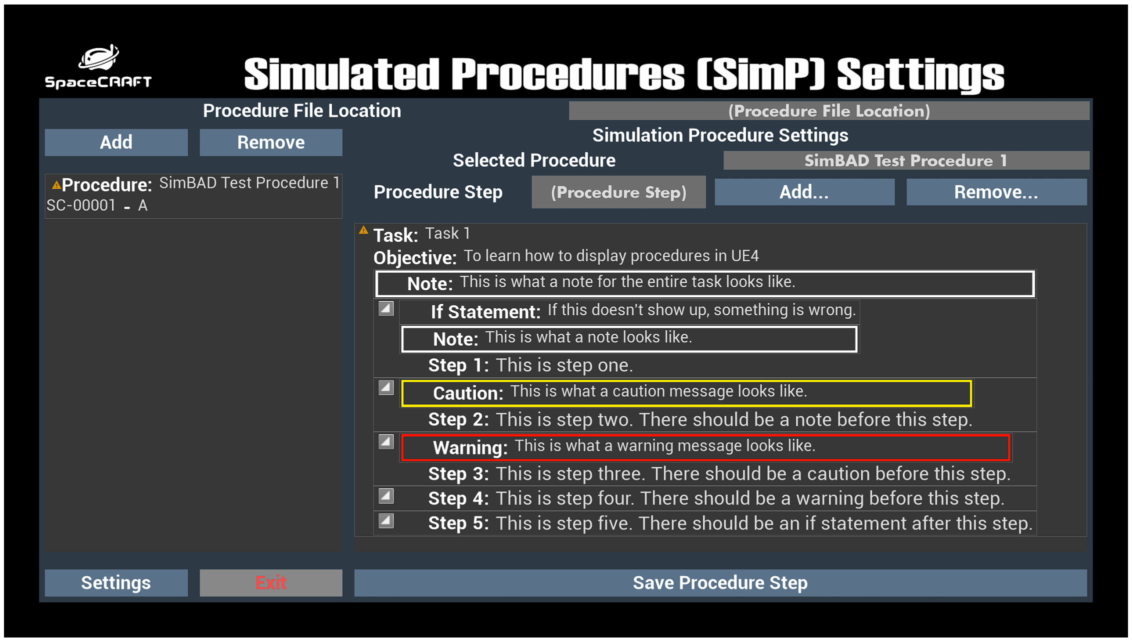

SpaceCRAFT
Simulated Systems (SimSys) Settings
Add
Remove
Search
System Instance Parameters
Add New Parameter
Remove Parameter
Save System Inst Parameters
Settings
Back

Name: Data_Monitor **Source:** Data_Monitor/bin/Data_Monitor_3 **Type:** Normal **Frequency:** 24
Name: Spicy **Source:** Spicy/bin/Spicy **Type:** Normal **Frequency:** 100
Name: EventManager **Source:** SimBAD_EventManager/bin/SimBAD_EventManager **Type:** Normal **Frequency:** 60
Name: SimBAD_Eval **Source:** SimBAD_Eval/bin/SimBAD_Eval **Type:** Normal **Frequency:** 60
Name: Rack_Power **Source:** Power_Rack/bin/Power_Rack **Type:** Normal **Frequency:** 60

System Selected: Rack_Power

```
Total Power: 250.000000 - Type[double]
```

Figure 3.3: SimSys menu showing listed Systems in the upper dialog and their instance parameters in the lower dialog


SpaceCRAFT
Simulated Procedures (SimP) Settings
Add
Remove
Simulation Procedure Settings
Add...
Remove...
Settings
Exit
Save Procedure Step

Procedure File Location: (Procedure File Location)
Selected Procedure: SimBAD Test Procedure 1
Procedure Step: (Procedure Step)

Procedure: SimBAD Test Procedure 1
SC-00001 - A

Task: Task 1
Objective: To learn how to display procedures in UE4
Note: This is what a note for the entire task looks like.
If Statement: If this doesn't show up, something is wrong.
Note: This is what a note looks like.
Step 1: This is step one.
Caution: This is what a caution message looks like.
Step 2: This is step two. There should be a note before this step.
Warning: This is what a warning message looks like.
Step 3: This is step three. There should be a caution before this step.
Step 4: This is step four. There should be a warning before this step.
Step 5: This is step five. There should be an if statement after this step.

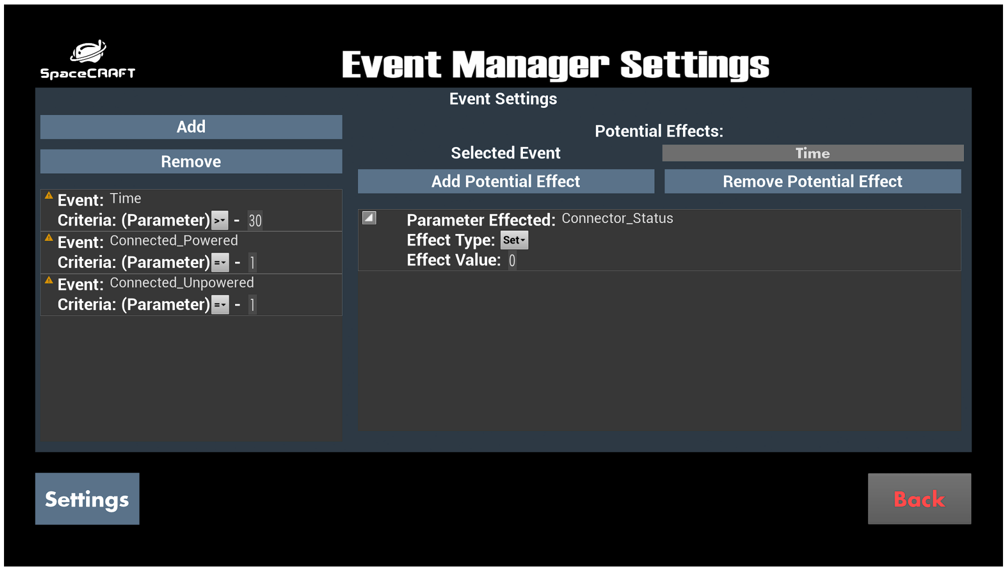
Figure 3.4: SimP menu showing available procedures in the left dialog with details to the right

inside of a simulation. The first dialog shows users the procedures that are available to the user. The second loads up the details of a procedure for editing and adding elements. Each procedure consists of tasks, with each task containing its own checklist of steps to be carried out with Notes, Cautions, and Warnings (NCWs) added to the front of the task. Each step contains a description of the action to be taken as well as an NCW if one is specified. This format was motivated with NASA's format for human spaceflight procedures in mind. This sets a solid baseline and allows for easy formatting changes in future situations.

3.4 Event Manager (EM)

The EM is another new addition to SpaceCRAFT along with the Simulated Procedures element that is necessary functionality towards making the platform more viable as a simulation tool. The purpose of this element is to set up event-driven simulations that allows for more realistic levels of interaction between entities in the simulation, the systems that drive the behavior of the entities, and the users. Adding this level of fidelity to simulations would increase the usefulness of the platform as one that can account for the unpredictable nature of spaceflight. One of the most important aspects of a good space environment simulator is the ability to be able to train astronauts for unexpected situations because being able to react calmly in any situation could make a huge difference at any given moment during an actual mission. Scripting a sequence of events to occur and have the trainees inside the simulation to deal with is an excellent way for trainers to create training regimens that teach crews the specific actions and reactions the regimen is attempting to condition the crews for.

The EM is a comparable element to SimSys in that it deals with the behavior of elements inside the simulation. The distinction between this element and SimSys is that while SimSys deals with the explicit behaviors of individual objects or systems in a simulation, the EM allows for more implicit behaviors being scripted. The most pertinent description of this would be with anomalies. SimSys helps users dictate what the nominal behavior inside a sim would be, while the EM is better for dictating off-nominal behavior. This helps to define the simulation as a more realistic scenario than one that simply shows the expected behavior of simulated elements, and being more



Event: Time
Criteria: (Parameter) > - 30
Event: Connected_Powered
Criteria: (Parameter) = - 1
Event: Connected_Unpowered
Criteria: (Parameter) = - 1

Parameter Effected: Connector_Status
Effect Type: Set
Effect Value: 0

Figure 3.5: EM UI showing dialog for Criteria list on left and Event list on right

viable as a training tool through that functionality.

There are several different situations in which the event manager is useful and adds novelty to the SpaceCRAFT platform. Systems faults are a great example of how this functionality could be utilized. Adding in the possibility for systems to fail along with adding their normal functionality is important for making the virtual versions of the systems as realistic as their actual counterparts which properly familiarizes users with the behavior of said systems. Being able to prevent, mitigate or resolve failures is incredibly important in spaceflight and having a simulation platform that can replicate failure conditions in a robust and consistent way is crucial for adequately preparing a crew for a complex mission. Another situation this kind of event-driven system is useful for is one where there could be a cascading series of events caused by each other. This could be a set of failures such as a power failure leading into an oxygen generation failure, or something such as a power overcurrent issue resulting from multiple systems in use at the same time due to an improper scheduling of procedures. In cases like these, it also shows how having a simulator with this level of realism would also be a significant resource as a planning and logistics tool for a flight operations team. Planning specific tasks to be carried out at different times during a mission is already an important part of human spaceflight, with certain procedures having requirements

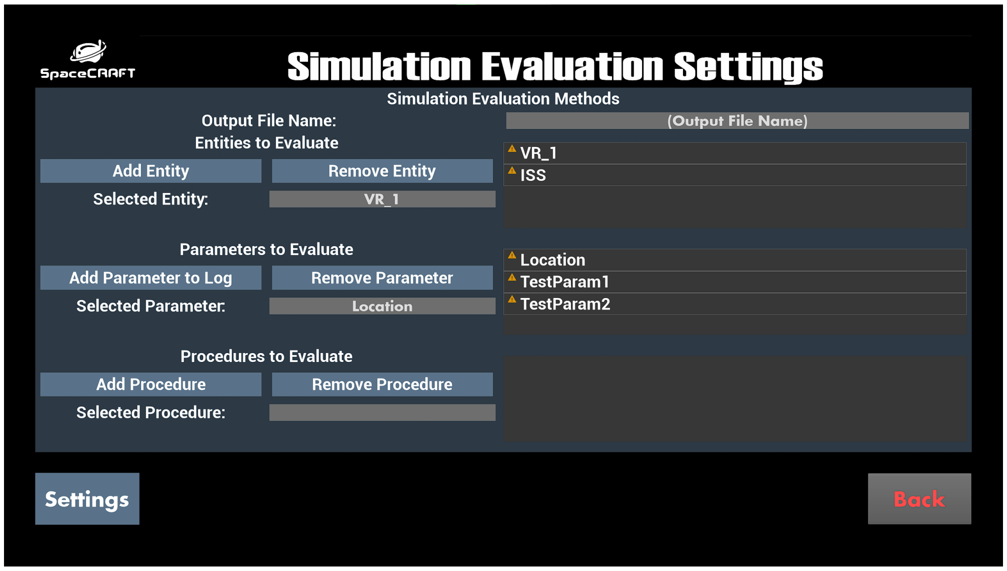


Figure 3.6: SimEval UI showing the three dialogs used to get data from simulations: Entities, Parameters, and Procedures

that frequently clash with one another and require special attention to make the flow of events as efficient as possible.

3.5 Simulation Evaluation (SimEval)

The SimEval element of the SimBAD toolkit is another new addition to the platform that helps users quantify performance inside of a simulation and use that to iterate through different scenarios and under different circumstances. Being able to review performance data has always been an important aspect of running simulations for training members of the flight operations team. Being able to look back on a simulation that was run in order to gain an understanding of where mistakes were made or where certain actions could have been improved is a valuable attribute for a simulation platform to have. This would give opportunities for several different techniques of monitoring crew training, designing procedures, or improving human systems integration.

Crew training could be monitored and give a training team the information to gauge how well any given crew member is doing in a situation where the trainee is being trained on a procedure they aren't familiar with. Not only would this help the trainee understand in what areas they might need to improve, the training team could also use that data to improve a particular training method to compare between performances. This situation is also similar to one where training methods were researched in a study [17] where times were recorded to quantify trainee performance. Another situation where this functionality would be viable is where a mission planning team could run through different combinations of scenarios in different circumstances and determine the best timeline to send on the mission. Historically, timeline development like this has been estimated to a certain degree by the planning team and then rectified after the mission during crew debriefs. Having this resource available to the planning team so they have crew perspective beforehand could provide important improvements in timeline efficiency without having to put a crew through a sequence of tasks in the first place. This is also valuable in a situation such as a Moon or Mars mission, where crew debriefs may not be as valuable considering the time between Shuttle missions and ISS expeditions were relatively short meaning the turnaround for amending a timeline was short. This may not be afforded in longer missions with larger times between missions.

4. TECHNICAL APPROACH*

This section of the paper describes the details of the different elements of SimBAD in the context of how they function within the backend side of the tool. Each section details the different methods and data structures that are used by the tool for each individual element. The purpose of this section of the thesis is to effectively give readers instructions on how to format data in a simulation platform to create other tools like this one. The objective is for this concept to enable others like it to be created and developers of training tools based in virtual simulations a framework to work off of and potentially avoid unnecessary “reinventing the wheel”.

Understanding how SimBAD works also requires an understanding of how the different components of the SpaceCRAFT platform operate in the backend. Most of the SimBAD elements require access to core elements of the code base and therefore understanding how the backend works so that the tool is developed parallel to the current functionality and appending onto it in a way that is intuitive to developers who are already familiar with the current code. The most significant characteristic of SpaceCRAFT that applies to SimBAD is the relationship between the UE4 client and the Compute Server. The client is the interface that the user will be interacting with and is composed as a customized platform version of the UE4.27 game engine, and the Compute Server contains the backend processes taking input from the user and refreshes the state of the simulation based on how those inputs react to the entities and systems inside the simulation.

The C++ functions that cover the functionality utilized by these elements are split between several different files that contain different classes and data structures to best divide the potential work as efficiently as possible. On the SpaceCRAFT UE4 engine side, the SimBAD_Functionality C++ is a class which consists of a source and header file inside the UE4 project’s source code directory. The majority of functions that are kept here are for writing nodes that can be used in Blueprint code. Blueprint coding is a form of coding specific to the Unreal Engine that uses nodes with different

*Part of the data reported in this chapter is reprinted with permission from "Simulation Builder, Analysis, and Development (SimBAD) Tool through the SpaceCRAFT Platform", William Young, 2022. 2022 IEEE Aerospace Conference (AERO), Copyright 2022 IEEE

functions in order to dictate the behavior of client assets. While this style of coding is relatively simple compared to C++, it's also more restrictive and generally slower, which is why as much functionality is inside C++ classes instead. However, to interface with UE4 through a UI, some blueprints are still created in order to better interface with menu-specific functionality and actions, normally in the form of setting buttons on menus to specific functions that are referenced from the SimBAD_Functionality class. Apart from the client side of the functionality, there is also Compute Server functionality that is being added that will be separate from other classes. The first and most significant file on the server side of the platform is the SimConfigEditor, which is a SpaceCRAFT API C++ class that holds functionality that operates on data types that aren't available in the UE4 C++ API. The most common occurrence this class is used is when input data from the user is sent back to be written to the JSON config. Along with this, the SimBAD_EventManager and SimBAD_EvalMethods are SpaceCRAFT API System classes that correspond to the fourth and fifth elements of SimBAD.

4.1 SimE & SimSys

The entity and system data held within the sim config file are not immediately capable of being sent to the UI.This requires a custom SpaceCRAFT UE4 engine C++ data structure that can have all of the necessary server-side entity data written to it to be parsed into UE4. This data is read using the SimBAD_Functionality class but written using a combination of this class and the SimConfig Editor. Fig. 4.1 shows a flow diagram for the sim config data. Once data is read from the JSON file through the SpaceCRAFT code and into the UE4 C++ class that contains the functionality that interfaces with the UI and displayed to the user, the user is able to interact with this data.

Automating this process in a way that is robust and can be broadly applied to all kinds of simulations requires a relatively complex logic. One of the main issues come from converting data from JSON structures to structures that can be read into UE4, and another comes from the problems that arise when dealing with the multitudes of data types that SpaceCRAFT can handle and pass through the Compute Server from sim configs. One of the complications when dealing with the client side of the platform is that the Unreal Engine uses its own set of data types and

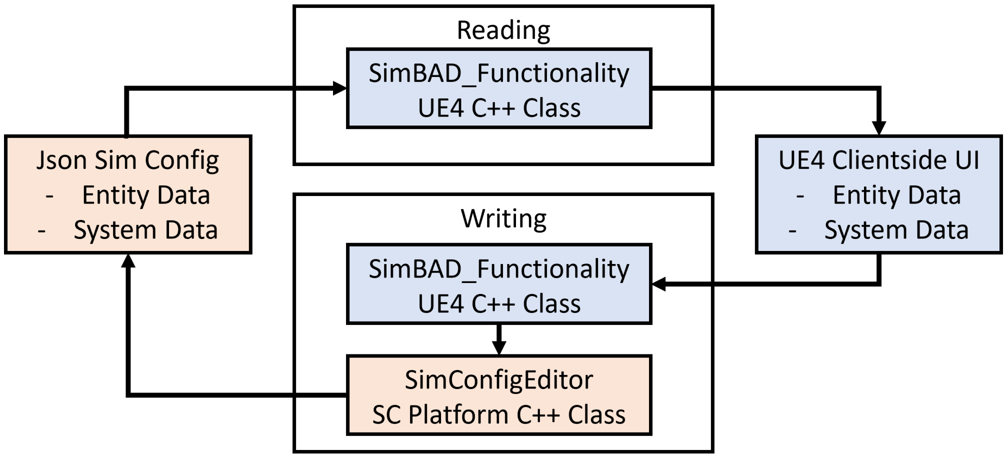


Figure 4.1: Flow Diagram showing the direction in which data is read/written. Blue boxes use UE4 C++ and red boxes use SpaceCRAFT API

structures, not the standard ones available in vanilla C++. The SpaceCRAFT API is also using its own set of custom data types, which means there has to be a lot of conversion between things as simple as a platform string, and a UE4 string that can be displayed in a UI. The SpaceCRAFT API has functionality to read the platform data types of individual parameters and convert them into more conventional data. However, no conversion for the entire entity data structure into a format readable by UE4 existed until SimBAD. The structure is referred to as FEntityConfig, shown in Fig. 4.2, which is congruent with UE4 data types such as FString and FVector. The structure is composed of several elements that are populated when the JSON config is parsed and read in C++ code, after which the entire structure is sent through to the blueprint side of UE4 code where the struct is read into its different components so it can be added to the displays.

When data on the display is changed and committed, the input gets sent back through the blueprint and back to the SimBAD_Functionality class. Because the data needs to go back to the original JSON config file, the data has to be directed to the SimConfigEditor class on the server side of the platform. This is split up in this way such that editing config data is possible outside of the SimBAD functions, having these methods restricted to a UE4 C++ class would make writing to a sim config inefficient if another server process needed to perform that action but had to reference the Unreal Engine library first. This highlights the main advantage of having different

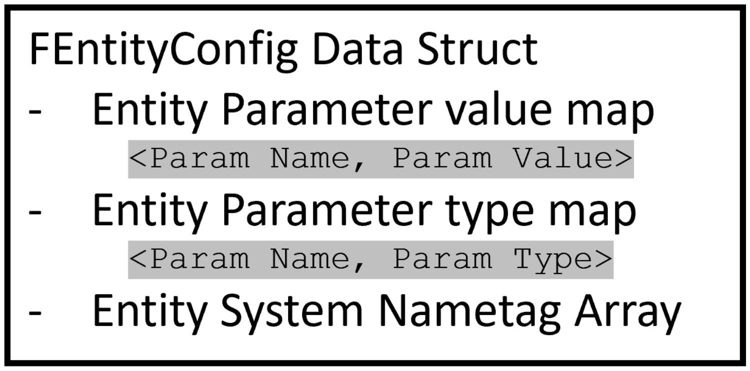

FEntityConfig Data Struct
- Entity Parameter value map
  `<Param Name, Param Value>`
- Entity Parameter type map
  `<Param Name, Param Type>`
- Entity System Nametag Array

Figure 4.2: FEntityConfig Data structure used by the SimE backend

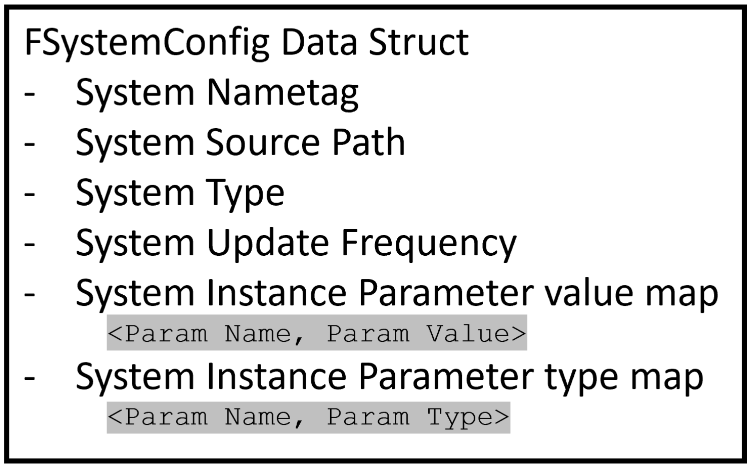

FSystemConfig Data Struct
- System Nametag
- System Source Path
- System Type
- System Update Frequency
- System Instance Parameter value map
  `<Param Name, Param Value>`
- System Instance Parameter type map
  `<Param Name, Param Type>`

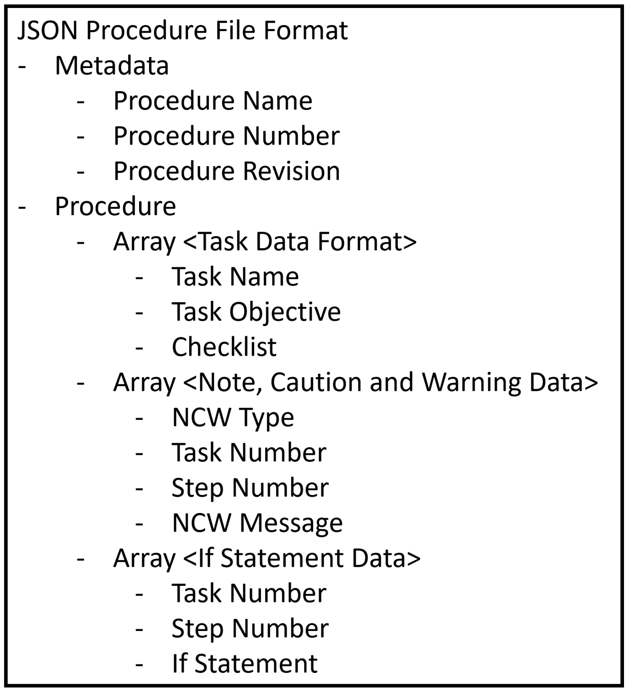
Figure 4.3: FSystemConfig Data Structure used by the SimSys Backend

functions separated into different parts of the platform in a way that makes the data management most efficient. Once data has been sent back through the SimBAD_Functionality class on the UE4 side to the SimConfigEditor class on the server side, the function uses other methods from the RapidJSON library, which is an API that interfaces with JSON files with both read and write functionality. To add back in the parameter to its proper location, the entire config is read and loaded, then finds and gets a reference to the parameter being changed. The config is loaded again but this time in order to write to the file as opposed to read, before being closed and saved. Once this function runs, the UI will refresh itself and display the change that was just committed.

As with the SimE element, a data structure FSystemConfig was created in the SimBAD_Functionality class to define system data in a way that can be interpreted and read in by the SimBAD UI. The process for getting this data from the config is also similar, but unlike entities, SpaceCRAFT does not have functionality for interpreting systems into usable data types. However this is the only difference, and system information is otherwise read from the JSON, structured into the FSystemConfig and sent through the SimBAD_Functionality class and back into the UI to be displayed to the user. Similar to the UI display for entities, a system has its most important information displayed in a list with other systems, along with a second list that gives Instance Parameters of the selected system. The main difference is the existence of Instance Parameters as opposed to normal parameters for entities. While regular entity parameters can be changed by systems during a sim, system instance parameters are constant and set during the initialization of the class when the Computer Server starts. However this difference does not encourage any new methodology for committing edited values to the sim config, and so the code logic used for these parameters is relatively the same as with Entity parameters. Values are passed from the UI through the SimBAD_Functionality class in SC engine code to the SimConfigEditor class on the server side and then written to the actual file.

4.2 SimP

Procedure files are a new addition to SpaceCRAFT and require their own files similar to a sim config, using the JSON formatting to contain data. The structure of a procedure file is more complex than either the entity or the system data structures used in the sim config. There are two JSON objects in a procedure file, one for the procedure metadata which consists of the procedure name, number, and revision and a second for the procedure data itself. Procedure data is structured using a JSON object that contains three nested data arrays, one for Task data, one for NCW data, and the final one for If Statement data. The Task data array is composed of a task name, objective, and checklist which is itself another nested array of the actual steps that describe how to complete a task. Because this Task object is an array, there is room for multiple different tasks in one procedure. The NCW data array contains elements that are structured by assigned the type, task number, step number, and message. The type indicates whether the message is a note, a caution, or


JSON Procedure File Format

- Metadata
  - Procedure Name
  - Procedure Number
  - Procedure Revision
- Procedure
  - Array <Task Data Format>
    - Task Name
    - Task Objective
    - Checklist
  - Array <Note, Caution and Warning Data>
    - NCW Type
    - Task Number
    - Step Number
    - NCW Message
  - Array <If Statement Data>
    - Task Number
    - Step Number
    - If Statement

Figure 4.4: Procedure File Format used in Procedure JSON files

a warning and each is displayed differently inside the UI to help the user more easily differentiate between them. Assigning task and step numbers indicates at what point the note will appear in the procedure, and the message data contains the string that will be displayed. Lastly the IF statement contains an array of data objects that is structured similarly to the NCW elements, but lacks a "type" object since there is no need to differentiate between types of if statements. While this is structured similarly to the NCW data, it is not included as that part of the procedure structure to leave room for future plans to add data that can add references to other procedures, which would be a useful feature in an if statement of a procedure.

This procedure JSON structure, like with the entities and systems from the sim configs, needs a SC engine side data structure counterpart to convert the data into a format that can be interpreted by the client and displayed on the UI. The Procedure Display is similar again to the SimEn and SimSys displays in that the metadata of the procedure is displayed as a list of the available procedures being used, with the specific data per procedure listed below with tasks, steps, and their accompanying messages. While the client side data structures FEntityConfig and FSystemConfig

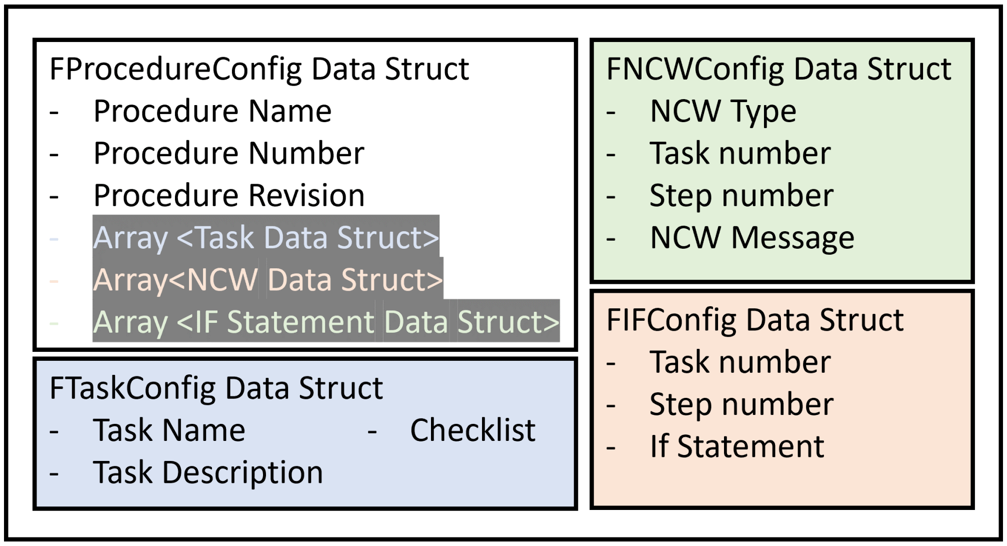


Figure 4.5: FProcedureConfig Data Structure with corresponding Data Structures for the nested structure format used

are self-contained, the structure for procedures contains other nested structures for each of the three data arrays in the JSON. The main structure is referred to as FProcedureConfig, which contains the metadata as three strings, and then a UE4 TArray of more custom structures. They are referred to as the FTask Config, and FNCWConfig, and FIFConfig data structures and are structured respective to the way the corresponding arrays contain data objects. Having this nested container structure allows for iterating through the procedure's data and adding items to the UI in a nested list.

4.3 EM

Creating events in SpaceCRAFT requires a specific methodology that utilizes the functionality that exists within the API already. Entities and their parameters are the target in this current version of logic implemented in SimBAD. The overarching concept driving the logic is that events are caused by parameters in specific entities reaching thresholds that motivate some defined event. Events are implemented as a specified "effect" that is applied to pertinent parameters wherever it can be found in other entities in the simulation. This concept in mind covers a wide range of potential events, for example, if a connector entity is not connected to a power source, the current parameter of that entity would be 0 and would be below a threshold value for delivering power to

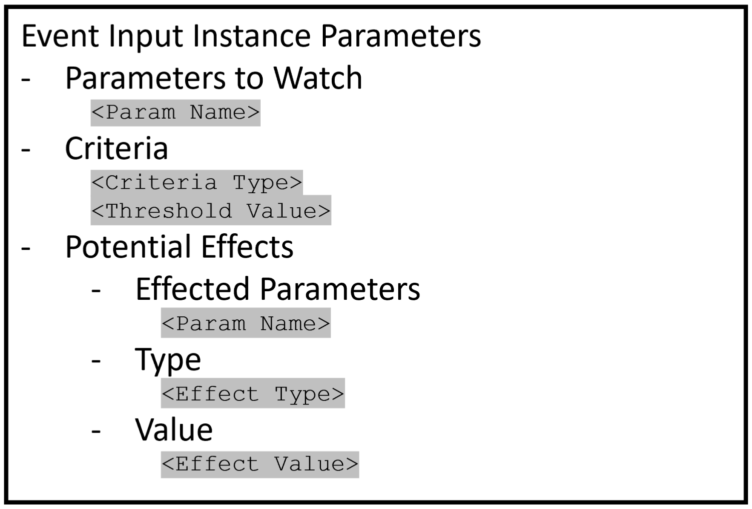

Event Input Instance Parameters
- Parameters to Watch
  `<Param Name>`
- Criteria
  `<Criteria Type>`
  `<Threshold Value>`
- Potential Effects
  - Effected Parameters
    `<Param Name>`
  - Type
    `<Effect Type>`
  - Value
    `<Effect Value>`

Figure 4.6: Input JSON data used to define events

a rack entity, which contains hardware that relies on that power, and the applied effect would be to set the value of the power parameters of those entities to 0.

Having a methodology to apply in a simulation is the first step towards adding this functionality, from here understanding what data to have and how to structure it is the next step in making the event manager functional. Having the concept already expanded upon in the context of the SpaceCRAFT API helps make this part easier as the input and output data has been stated. The event manager is actually a SpaceCRAFT system, so it can utilize instance parameter objects in the sim config JSON file. Event data is input with three objects: Parameters to watch, event criteria, and potential effects. The first is a list of strings that contains the names of parameters in the sim that need to be watched. Criteria is dictated by two nested objects, the criteria type and the threshold value. Currently the types of criteria established in this version are less than, equal to, and greater than. The potential effects object is also characterized by a group of nested objects. The first object is a list of effect parameter names for the script to search for to apply an effect. With each parameter a corresponding type needs to be specified that tells the script how that parameter is affected. The current effect types that are supported are scale, set, and add. Lastly, the value of the effect is given.

While a sim is running, a script with the logic shown in the diagram in Fig 4.7 is executed at a

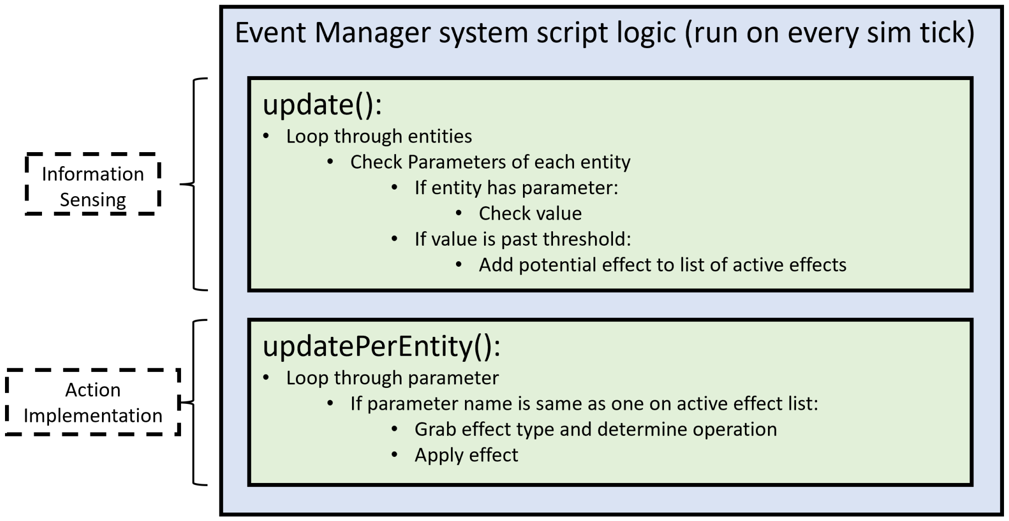

Event Manager system script logic (run on every sim tick)
update():
• Loop through entities
• Check Parameters of each entity
• If entity has parameter:
• Check value
• If value is past threshold:
• Add potential effect to list of active effects
updatePerEntity():
• Loop through parameter
• If parameter name is same as one on active effect list:
• Grab effect type and determine operation
• Apply effect
Information Sensing
Action Implementation

Figure 4.7: Diagram showing Event Manager scripting logic

specified frequency from the user. This part of the code uses some of the information processing nomenclature described in section 1.2.2 of this paper. Specifically, the methods this script runs collectively act as an automatic process that uses information sensing and action implementation in order to execute events. Contextualizing what the script is doing in this way is a good way to motivate the development of the logic and account for different cases. The capability of the event manager in this initial version of the system is not able to handle too many different cases. It shows the functionality is available in SpaceCRAFT and having a tool like SimBAD can make that usable in the context of a training tool.

4.4 SimEval

The Evaluation Methods element is also written as a SpaceCRAFT system like with the Event Manager. This means that the input data that instructs the script what to do is given in instance parameters in the sim config. There are three array objects that define what gets evaluated in a simulation being run. The first object is a list of entities, the second is a list of parameters, and the last is a list of procedures. These are all things that the script looks for and logs to a Comma-Separated Value (CSV) file. With entities and parameters, for each simulation tick the script updates, checking for entities on the list it was given. If that entity is on the list, the script

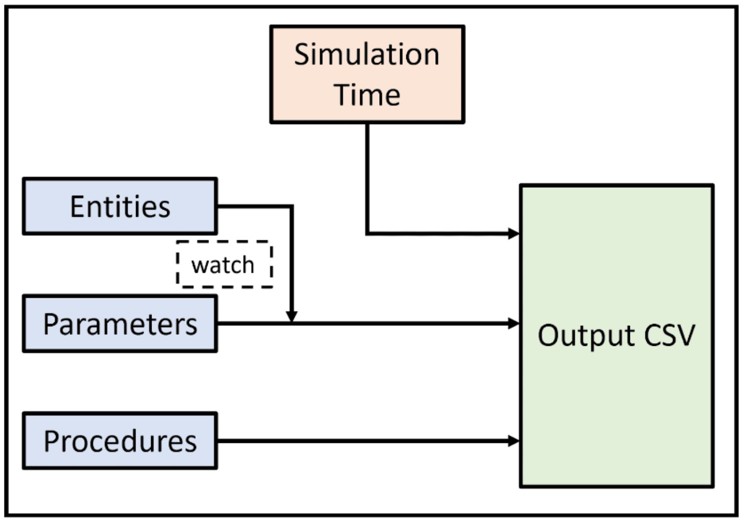


Figure 4.8: Flow Diagram showing how data is sent from the simulation to the output file

steps in and checks the parameters. If a parameter is on the second list, then the value of that parameter is logged to the specified CSV file. With procedures, the script watches one entity in particular that is specifically communicating with the pawn in the client that is utilizing a procedure UI in the sim. When the user steps through a procedure, a timestamp is sent to that entity on the server which is then read and logged by the script.

This functionality enables several different scenarios that are useful in a training setting. Using this with the event manager allows operators to script events, monitor the parameters that dictate the event, and then quantify the reactions the users inside the simulation have. A simple example of this is used for the ISS VR simulation done as one of the validation measures for this thesis. In that case, a connector between a power source and a hardware module is scripted to fail. That behavior is captured by the script and sent to the specified output file. The user then runs through a procedure to mitigate this issue. The output file shows the times at which the user completed each step, which can be used in cases where users are trying to determine the time it takes to complete a procedure, or edit procedures and compare completion times against each other in order to determine a more efficient sequence of events.

5. RESULTS AND DISCUSSION*

Using SimBAD, several simulations have been built as proofs of concept for the several simulator aspects SimBAD wanted to improve upon for the SpaceCRAFT platform and add as viable components of simulations that can be run. The first is a simulation that demonstrates both how procedures can be used inside of a procedure as well as the event manager. Not only does this simulation demonstrate the new functionality added as a procedure development tool, but is taking a step further by using these elements at the same time by having events occur that require the user to carry out procedures in a nested format and prioritize tasks based on the events occurring in real time. The second simulation is a Mars base setting with two IVA astronauts completing tasks inside the habitat. The functionality being demonstrated in this setting is that the two astronauts will be users working together in the same sim, but in different physical locations connected through the multi user functionality of SpaceCRAFT. Together both of these simulations show not only important aspects of SimBAD's contribution to the SpaceCRAFT platform as a training tool on multiple fronts, but the swiftness with which users can create and run different simulations.

5.1 ISS VR Simulation

The first simulation built using this tool consists of a user conducting an IVA procedure inside the ISS in VR and responding to a scripted anomaly. The pawn in the simulation that the user possesses has a camera for a first-person view, with two floating hands to indicate hand placement. The model is not very representative of the user's body because the user would not be able to see it very well anyways and is not necessary for establishing presence. However, the locomotion of the pawn is designed to establish presence in a microgravity environment. The user can propel themselves through the environment by grabbing onto the walls of the ISS interior and pushing themselves in the direction they want to go, which will send them floating through the cabin as

*Part of the data reported in this chapter is reprinted with permission from "Simulation Builder, Analysis, and Development (SimBAD) Tool through the SpaceCRAFT Platform", William Young, 2022. 2022 IEEE Aerospace Conference (AERO), Copyright 2022 IEEE

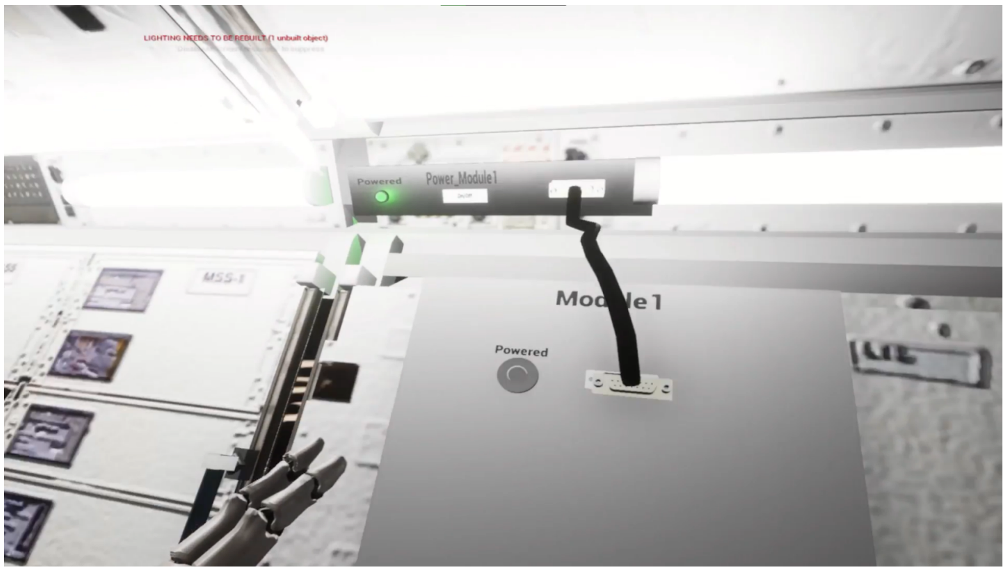
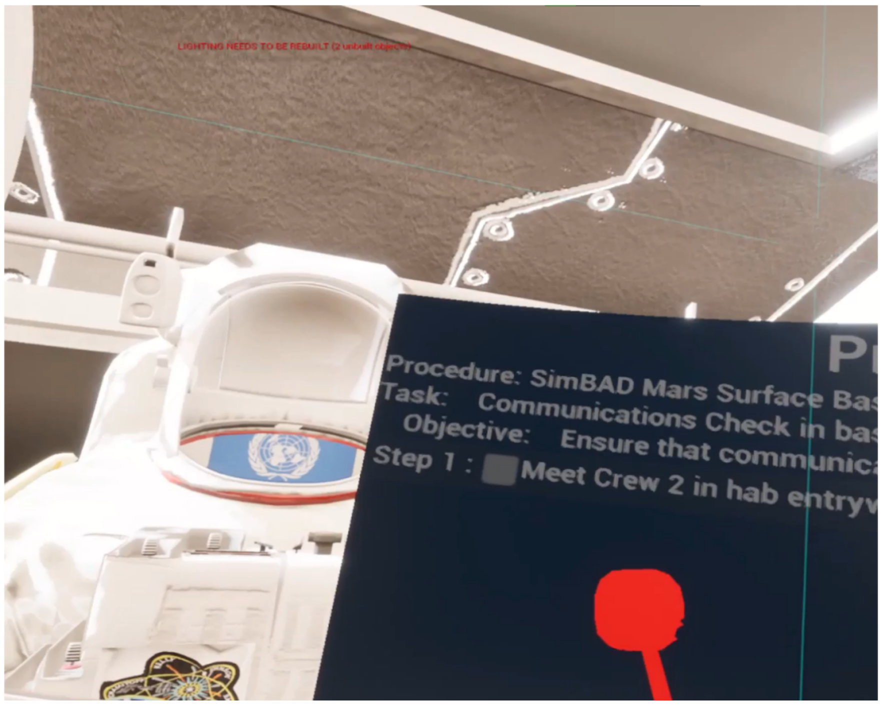


Figure 5.1: User in the ISS VR simulation completing a task from a procedure

would happen in the real ISS. The other objects that the user interacts with also have microgravity physics, so they float and bounce off walls.

The user also has a wrist-mounted menu for navigating a Caution & Warning UI as well as a UI for reading and following procedures. This is the method through which the procedures that were written for this simulation in SimBAD are to be executed by the user. There are three procedures in this simulation. The first is a basic IVA stowage procedure that instructs the user to traverse to a location with stowage bags and relocate them to different locations in other areas of the station. The second procedure is one that is specifically carried out in the event of an anomaly. After 120 seconds in the simulation, the event manager sets a connector (Fig. 5.1) to malfunction. This causes the hardware module to fail and throws a caution to the wrist UI for the user. This caution directs the user to the second procedure, which lists troubleshooting steps to bring the module back online. Before the user traverses to the module's location, they are instructed to retrieve a CO2 probe and make sure the area is not experiencing a buildup of CO2 due to a failed circulation fan. This probe will be malfunctioning as well, and the user will have to follow a third procedure that troubleshoots the device. The third procedure has the user determine that a replacement will be used instead, which works at which point the user returns to the hardware


LIGHTING NEEDS TO BE REBUILT (2 unbuilt objects)

Pr

Procedure: SimBAD Mars Surface Bas

Task: Communications Check in bas

Objective: Ensure that communica

Step 1 : Meet Crew 2 in hab entryw

Figure 5.2: Multiple Users interacting inside the Mars VR multi user simulation

module power troubleshooting procedure. Once the user determines the connector is failed, the user finds a replacement in a designated location. With the functioning connector installed, power is restored to the module and the user carries out the rest of the initial stowage relocation procedure. Once all three procedures are completed, the simulation is concluded.

5.2 Mars Surface Base Multi User Simulation

The second simulation built using SimBAD is a Mars Surface Base simulation with multiple users interacting with each other. The virtual environment depicts a surface base located near the Gale Crater of Mars. A few key differences between the other models in this simulation and the ones used in the previous. First, because there are multiple people, the pawns occupied by the users have full bodies instead of just hands like in the ISS simulation. This is because otherwise it would be more difficult to know exactly where the other user was. The second difference is the lack of interactable models. This is because the simulation only concerns demonstrating the basic concept of having users be able to interact with each other, as opposed to the environment.

The procedure that the users execute is one that requires both users to communicate with each other. This is done by taking advantage of the multi user capability of SpaceCRAFT which includes voice communication. To clarify which user is being referred to, they are given the designations Crew 1 and Crew 2. Crew 1 and 2 start near the entrance of the habitat and verify they are using the same procedure. Crew 2 relocates to a greenhouse module on one side of the habitat. Over the communication system they coordinate back to Crew 1 if their comm quality is adequate. If nominal, Crew 2 then relocates to the command module in the habitat and repeats the process. After this, Crew 1 relocates to a second greenhouse on the other side of the habitat, and calls Crew 2. After affirming nominal communication Crew 1 meets Crew 2 in the command module and confirms that the communication inside the habitat is adequate and coordinates that the procedure has been completed.

5.3 Discussion

5.3.1 Pre- and In-flight training methods and Procedure Design

Both of these simulations help demonstrate that SimBAD can be used as a tool for training in situations like will be experienced in future human spaceflight missions. This concept shows that frameworks like this are conducive to mitigating the HSIA issues that are caused by mission aspects such as the distance from earth, travel durations, and the unknown parameters that exist in the nature of exploration missions. Crews will experience a high level of autonomy on these missions and will need the capability to build augmentable training regimens for themselves, and operate without the assistance of the training instructors.

Increased distance from Earth will eventually result in communication delays that will make real time correspondence with that support impossible. Combining this with the other eventuality that these kinds of missions will provide challenges that can't be accounted for during the preflight phases of the mission prep exacerbates the problem of training being inadequate. Since current methods put so much emphasis on establishing regimens for the pre-flight phase, not enough has been established during the in-flight phase. SimBAD accounts for this by only requiring a small amount of relatively lightweight hardware with low spatial requirements.

This distance also presents the issue associated with crew performance detriments due to knowledge retention. For example, a Mars mission would require a crew to fly on a ship for at least nine months before getting to the surface landing phase of the mission. In that time a pilot's skill will decrease without a device through which they can maintain that skill. An adequate in-flight training tool will need to be able to simulate a landing for that pilot, allow them to carry out procedures, add anomalies, and view performance metrics after the fact. These aspects are all a part of SimBAD's design and will help maintain operational effectiveness.

In the case of unaccountable mission parameters, it's very likely situations will arise that call for procedures to be developed after the mission has launched. In this scenario, giving crews as many tools as they require to develop procedures for new aspects of a mission would greatly increase the chances of success. While grounds will still be able to offer limited support in this regime, the consideration of full loss of communication must be considered. This motivates the need for a tool that supplies a closed loop for procedure. SimBAD offers this by allowing users to create an entire simulation from scratch, have a high amount of control over the environment and the behavior of the sim, and provides methods through which the user can quantify a performance and utilize that information in the development of a procedure.

5.3.2 Collaborative Mission Development

Another aspect of these missions is the long development time needed to design a mission that will be as effective and efficient as possible. Streamlining the processes used to create all of the products and tools used for the missions will decrease the workload of groups in flight operations teams while increasing the quality of the products they output. SimBAD presents a method that will benefit these processes by enabling users to create procedures in high fidelity environments that can simulate a large amount of mission parameters. Increasing the amount of situational awareness that can be afforded to mission designers will support their work by increasing the amount of relevant information afforded to them in an engaging way and providing context that may not be provided in other settings. Not only this, but the capability to do this development with other teams in different physical locations, even individuals who may be in space, allows for a more constructive development space.

6. FUTURE WORK*

6.1 Tool Validation for Training

The next step for SimBAD is quantitative validation towards establishing it as a viable training tool. When developing technologies for the field of human factors, the standard is for using data from subject trials conducted and use case evaluation to move forward towards utilization in a flight environment. Having a more specific idea of a use case regarding a program using this tool would result in a requirements list that could be used to build a test matrix. For example, an entity such as NASA might use SimBAD differently than one like a private launch provider. SimBAD is already equipped with the SimEval element, which would streamline the verification process. It should also be noted that SimBAD has potential in fields other than space but with similar levels of required training. Places in extreme environments such as oil rigs or antarctic outposts would see benefits from a tool like this, and could also provide an adequate proving ground for the technology.

Because SimBAD is designed as a training tool, the ideal comparison to make would be against other training methods. Evaluating the completion of procedures through different methods and comparing the user's preferences with the different simulators is one option for starting this analysis. Getting performance metrics such as procedure task and step durations for crew completion are a baseline. Specific tasks would require matching metrics to properly assess performance and simulator efficacy. For example, some procedures that require precision such as the construction of a structure on the outside of a vehicle. Getting the output values of the locations of the structure elements would give the evaluators the metrics to determine how well the procedure was conducted. Other simulations might assess operational capability; users would be assessed on their ability to properly prioritize an action given a specific situation. SimBAD's EM is an element that would make this a viable avenue for evaluation.

*Part of the data reported in this chapter is reprinted with permission from "Simulation Builder, Analysis, and Development (SimBAD) Tool through the SpaceCRAFT Platform", William Young, 2022. 2022 IEEE Aerospace Conference (AERO), Copyright 2022 IEEE

6.2 Multi User Procedures

One of the most viable areas for future research is to improve the usefulness of this tool in a multi user, collaborative environment. Adding more infrastructure for users to be able to build simulations and procedures from remote locations as well as from within the same building offers a huge advantage when compared to other simulation platforms in that regard. The concepts presented in the demonstration simulations of this thesis can be combined to a higher degree and explore elements of mitigating HSIA risk on a lower level and provide more detailed findings with regards to how multi user procedure development should be conducted.

The area of virtual assistance is also one that should be considered in future iterations of the multi user elements that SimBAD uses. Having the tool allow for remote users to take an observation role and monitor training performances in real time, for example a trainer walking a crew through a new procedure remotely, would be a valuable addition. Increasing the capability for users to augment the simulation in real time is one area in which a multi user sim could benefit. Having the event manager be usable in real time and at the command of a second user in the simulation is one example of how the presented concepts could be improved to meet the objectives of this research to an even greater extent.

6.3 Intuitive Procedure Design

Intuitive procedure design is a specialized process of designing procedures for operators in complex technical environments such as the ones astronauts are exposed to. Simply put, intuitive procedures are ones that ideally require no prior training, and are designed in such a way that the user can follow the steps laid out for them and successfully complete the task on their first attempt. This concept would be incredibly beneficial, especially considering the concerns of mental strain and crew workload on missions that will get more system-intensive and require them to retain more information. Decreasing the overall information necessary for the crew to retain or practice while the amount of systems and knowledge needed would balance the capabilities required from the crew.

This concept is also interesting within the system design phase of the mission itself. While the engineering teams are designing the hardware and software to carry out the mission, the operations teams can design their procedures and attempt to make them intuitive in the sense described. This offers an avenue for the operations teams to make suggestions and collaborate on the design of the systems so that they have intuitive procedures in mind from as early a stage of development as possible. Building this into the systems from the start will make implementing intuitive procedures easier and operations less intense.

6.4 Augmented Reality

While Virtual Reality is a proven method for training in space environments, another medium of Extended Reality (XR) could also be very useful for achieving more effective results in procedure development for space missions, Augmented Reality (AR). The advantage of AR as opposed to VR is that the user can have most if not all of the UI elements previously available in VR, but now is able to merge this setting with a real physical environment. This especially has implications for intuitive procedure design, where instructions and steps can be presented in a more visual sense than just through reading off a checklist. This also has implications, when combined with both an established multi user setting, and intuitive procedure design, for a simulation builder that would let ground teams create and test procedures without crew input in VR. After this, the teams could then package and send that simulation data up to the vehicle, where the crew could don an AR headset and perform the exact same procedure, but in an intuitive sense where no prior experience was required.

Another potential avenue for this technology to be integrated is for the simulation building process in the first place. Instead of interfacing with a UI on a flat screen to set up a simulation, the user could construct their environment in AR and then use that for their sim or procedure. Creating a procedure at this point would also be beneficial, and would save time in the future as these virtual environments and procedures to be carried out would have been developed in parallel. This would increase the likelihood that the procedures are as effective as they can be. Users could also augment their procedures in real time as they were doing them while using AR and making

Figure 6.1: Astronaut and Expedition 65 Flight Engineer Megan McArthur testing an AR headset while on board the ISS - Credit: NASA

improvements and changes as the situation called for them.

6.5 Autonomous Mission Design

Another area to explore in the future, particularly when considering long-distance missions is autonomous mission evaluation. This refers specifically to missions that require a level of crew autonomy in quiescent ops or at the very least as a contingency in the event of some system failure isolating the crew from their ground support teams. The idea posed here is that a tool like SimBAD could be used by crews to design their own procedures and effectively carry part of the job of the flight operations team independently. When considering a vehicle as complex as the one that will be taking astronauts to destinations such as Mars, the subsystems aboard the vessel will require more training than is available to be scheduled for crews before they launch. If there is a tool with SimBAD's capabilities on board with them, crews could use it to prepare for anomalous situations or scenarios for which they might not have had time to train for on the ground.

The main improvement to be made on the framework introduced in this thesis is for the training regimen design loop to be as closed as possible. I.e, the user needs to have as many elements of potential environments as possible, methods in which to manipulate the environment, possibilities for events to be scripted in the simulation, and feedback to be given afterwards. Development loops require an iterative process and doing this for procedures and training plans is no different. One example of a valuable improvement to be made would be for post-processing methods to give more detailed reports of simulation results. Currently, data is presented in a raw form and leaves the processing to the user. Having other tools appended onto SimBAD for different types of analysis would increase the level of autonomy that crews could achieve in this particular area.

7. CONCLUSIONS*

The SimBAD tool presents a novel concept for improving the training element of HSI for future human spaceflight missions by creating a closed-loop virtual training facility. This tool was designed as a UI to append onto the SpaceCRAFT simulation platform to give users more control over VR simulations they can build. SimBAD hands users full control of the objects in a simulation and the systems that dictate their behavior SimBAD also adds functionality to create and edit procedures to be used inside a simulation, script events between entities and systems, and monitor user-defined parameters. The EM in particular adds a high level of fidelity to simulations and is a useful element when considering exploration spaceflight and the nature of missions with relatively low known mission parameters and high operational complexity. The use of these elements altogether creates the concept that SimBAD demonstrates, which is a virtual facility that mitigates the risk of inadequate HSI, specifically the training element in increasingly earth-independent spaceflight.

Four objectives were drafted to validate SimBAD as a viable concept to be used as a training tool in these environments: to be able to travel with crews in flight, to enable higher levels of collaboration between groups on flight operations teams, to have a capacity to account for unknowable mission parameters in real time, and to have a capability for training performance evaluation. These objectives were motivated through an analysis of current flight operations teams, training regimens used for astronauts, and human spaceflight challenges outlined in NASA's HRR. The HSIA risk of the HRR is a significant concern regarding future spaceflight missions which describes the ways in which inadequate HSI will be detrimental to mission outcomes. This issue will need to be mitigated before missions to distant destinations will become feasible. Applying this particular risk to how training infrastructures should be implemented gives the objectives.

To validate these objectives, two simulations were built using this tool to exercise the capability

*Part of the data reported in this chapter is reprinted with permission from "Simulation Builder, Analysis, and Development (SimBAD) Tool through the SpaceCRAFT Platform", William Young, 2022. 2022 IEEE Aerospace Conference (AERO), Copyright 2022 IEEE

for the concept to be implemented in a real world situation. An ISS IVA procedure was simulated in VR which had the user follow several procedures and mitigate a scripted anomaly. Overall this demonstrates the baseline capability of the tool as a training device. At a lower level the simulation validates SimBAD as being able to account for dynamic mission parameters with the event manager having functionality to script events. The second simulation was a communications check procedure on a Mars surface base with two users taking advantage of the multi user capability of the SpaceCRAFT platform. This shows that SimBAD is usable in settings where users that want to collaborate from different physical locations, including situations where one user may be on a vehicle in flight and another is on the ground. This simulation validates the tool as enabling better collaboration between groups in flight operations teams. The development of the tool using VR technology through the SpaceCRAFT platform validates the tool as being packageable on a space vehicle. All of the hardware and software necessary to use this training infrastructure takes up a relatively small amount of space, but does have relatively significant data storage and power usage demands. Having the tool on board a craft also helps validate the capacity for increasing the capacity for the crew to operate autonomously. The evaluation methods element of SimBAD closes the loop on this objective by giving users the ability to produce and analyze their own performance data.

In conclusion, the SimBAD tool explores improvements to training methods in scenarios where crews will experience higher levels of autonomy and mission complexity. The concepts presented in this thesis take steps towards mitigating issues such as increased workload on crew, detriments to crew knowledge retention, and unaccounted mission variability. The overall design of the tool allows for enhancements to the way in which missions are designed during both pre- and in-flight phases. Enabling seamless collaboration between teams and suggesting new methods for iterating through procedure design, as well as giving crew more chances for input in the process is highly valuable for autonomous situations. Maintaining an effective training regimen that can be maintained after a crew has already embarked on a mission and is in an operational environment increases the capability of the flight operations team to complete their mission and achieve their objectives.

REFERENCES

[1] HRR - Risk - Risk of adverse outcomes due to inadequate human systems integration architecture. `https://humanresearchroadmap.nasa.gov/risks/risk.aspx?i=175`.

[2] G E Miller. The assessment of clinical skills/competence/performance. *Academic Medicine*, 65:S63–7, 9 1990.

[3] Alan Crocker. Operational considerations in the development of autonomy for human spaceflight. American Institute of Aeronautics and Astronautics, 1 2005.

[4] James A. Lawrence and H. Edward Smith. The role of jsc engineering simulation in the apollo program. *SIMULATION*, 57, 7 1991.

[5] John M Logsdon and Roger D Launius. *Exploring the Unknown: Selected Documents in the History of the US Civil Space Program: Human Spaceflight: Projects Mercury, Gemini, and Apollo.*, volume 7. Government Printing Office, 2008.

[6] Eugene Kranz and Christopher Kraft. Systems engineering and integration processes involved with manned mission operations, 1993.

[7] Brian O'hagan and Alan Crocker. Fault management techniques in human spaceflight operations, 5 2011.

[8] Ryan W Proud, Jeremy J Hart, and Richard B Mrozinski. Methods for determining the level of autonomy to design into a human spaceflight vehicle: a function specific approach., 2003.

[9] Axel Schulte and Diana Donath. A design and description method for human-autonomy teaming systems. In *International Conference on Intelligent Human Systems Integration*, pages 3–9. Springer, 2018.

[10] Raja Parasuraman, Thomas B Sheridan, and Christopher D Wickens. A model for types and levels of human interaction with automation. *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans*, 30(3):286–297, 2000.

[11] Gregory Dorais, R Peter Bonasso, David Kortenkamp, Barney Pell, and Debra Schreckenghost. Adjustable autonomy for human-centered autonomous systems. In *Working notes of the sixteenth international joint conference on artificial intelligence workshop on adjustable autonomy systems*, pages 16–35, 1999.

[12] Connor R. Jakubik, Adam Johnston, Patrick Zhong, Neil McHenry, and Gregory Chamitoff. Spacecraft vr: an event-driven, modular simulation platform with fully-asynchronous physics. American Institute of Aeronautics and Astronautics, 1 2021.

[13] Christian Sinnott, James Liu, Courtney Matera, Savannah Halow, Ann Jones, Matthew Moroz, Jeffrey Mulligan, Michael Crognale, Eelke Folmer, and Paul MacNeilage. Underwater virtual reality system for neutral buoyancy training: Development and evaluation. ACM, 11 2019.

[14] Eddie Paddock. Overview of NASA EVA virtual reality training, past, present and future. `https://nvite.jsc.nasa.gov/presentations/b3/D2_3_Virtual_Reality_Panel_Paddock.pdf`, 2017.

[15] Marcum Reagan, Barbara Janoiko, James Johnson, Steven Chappell Ph.D., and Andrew Abercromby. Nasa's analog missions: Driving exploration through innovative testing. American Institute of Aeronautics and Astronautics, 9 2012.

[16] Bill Todd and Marc Reagan. The neemo project: A report on how nasa utilizes the "aquarius " undersea habitat as an analog for long-duration space flight. American Society of Civil Engineers, 3 2004.

[17] Steven P. Chappell, Trevor G. Graff, Kara H. Beaton, Andrew F. J. Abercromby, Christopher Halcon, Matthew J. Miller, and Michael L. Gernhardt. Neemo 18–20: Analog testing for mitigation of communication latency during human space exploration. IEEE, 3 2016.

[18] Jad Nasrini, Emanuel Hermosillo, David F. Dinges, Tyler M. Moore, Ruben C. Gur, and Mathias Basner. Cognitive performance during confinement and sleep restriction in nasa's human exploration research analog (hera). *Frontiers in Physiology*, 11, 4 2020.

[19] Nick Kanas, Matthew Harris, Thomas Neylan, Jennifer Boyd, Daniel S. Weiss, Colleen Cook, and Stephanie Saylor. High versus low crewmember autonomy during a 105-day mars simulation mission. *Acta Astronautica*, 69, 9 2011.

[20] Kimberly A. Smith-Jentsch et al. 'training'the right stuff': an assessment of team training needs for long-duration spaceflight crews., 2015.

[21] Richard N. Landers and Michael B. Armstrong. Enhancing instructional outcomes with gamification: An empirical test of the technology-enhanced training effectiveness model. *Computers in Human Behavior*, 71:499–507, 6 2017.

[22] Ferdinand Cornelissen. Gamification for astronaut training. American Institute of Aeronautics and Astronautics, 6 2012.

[23] Angelica D. Garcia, Jonathan Schlueter, and Eddie Paddock. Training astronauts using hardware-in-the-loop simulations and virtual reality. American Institute of Aeronautics and Astronautics, 1 2020.

[24] Yuqing Liu, Shanguang Chen, Guohua Jiang, Xiuqing Zhu, Ming An, Xuewen Chen, Bohe Zhou, and Yubin Xu. Vr simulation system for eva astronaut training. American Institute of Aeronautics and Astronautics, 8 2010.

[25] Andrea F. Abate, Mariano Guida, Paolo Leoncini, Michele Nappi, and Stefano Ricciardi. A haptic-based approach to virtual training for aerospace industry. *Journal of Visual Languages and Computing*, 20:318–325, 10 2009.

[26] Lynn Marie Geiger. Investigation of customized refresher training for telerobotic operations in long-duration spaceflight, 2016.

[27] N. McHenry, T. Hunt, W. Young, A. Gardner, B. Bontz, U. Bhagavatula, J. Chiu, G. Chamitoff, and A. Diaz-Artiles. Evaluation of pre-flight and on orbit training methods utilizing virtual reality. volume 1 PartF, 2020.

[28] Miquel Bosch Bruguera, Valentin Ilk, Simon Ruber, and Reinhold Ewald. Use of virtual reality for astronaut training in future space missions - spacecraft piloting for the lunar orbital platform - gateway (lop-g). International Astronautical Federation, 2019.

[29] Jukka Rönkkö, Jussi Markkanen, Raimo Launonen, Marinella Ferrino, Enrico Gaia, Valter Basso, Harshada Patel, Mirabelle D'Cruz, and Seppo Laukkanen. Multimodal astronaut virtual training prototype. *International Journal of Human Computer Studies*, 64:182–191, 3 2006.

[30] Jeffrey Osterlund and Brad Lawrence. Virtual reality: Avatars in human spaceflight training. *Acta Astronautica*, 71:139–150, 2 2012.

[31] Kapil Chalil Madathil, Kristin Frady, Anand Gramopadhye, Jeff Bertrand, and Rebecca Hartley. Integrating case studies in an online asynchronous learning environment: An empirical study to evaluate the effect on community college student learning outcomes and engagement when case studies are presented using virtual reality. ASEE Conferences.

[32] Fadi Munshi, Hani Lababidi, and Sawsan Alyousef. Low- versus high-fidelity simulations in teaching and assessing clinical skills, 3 2015.

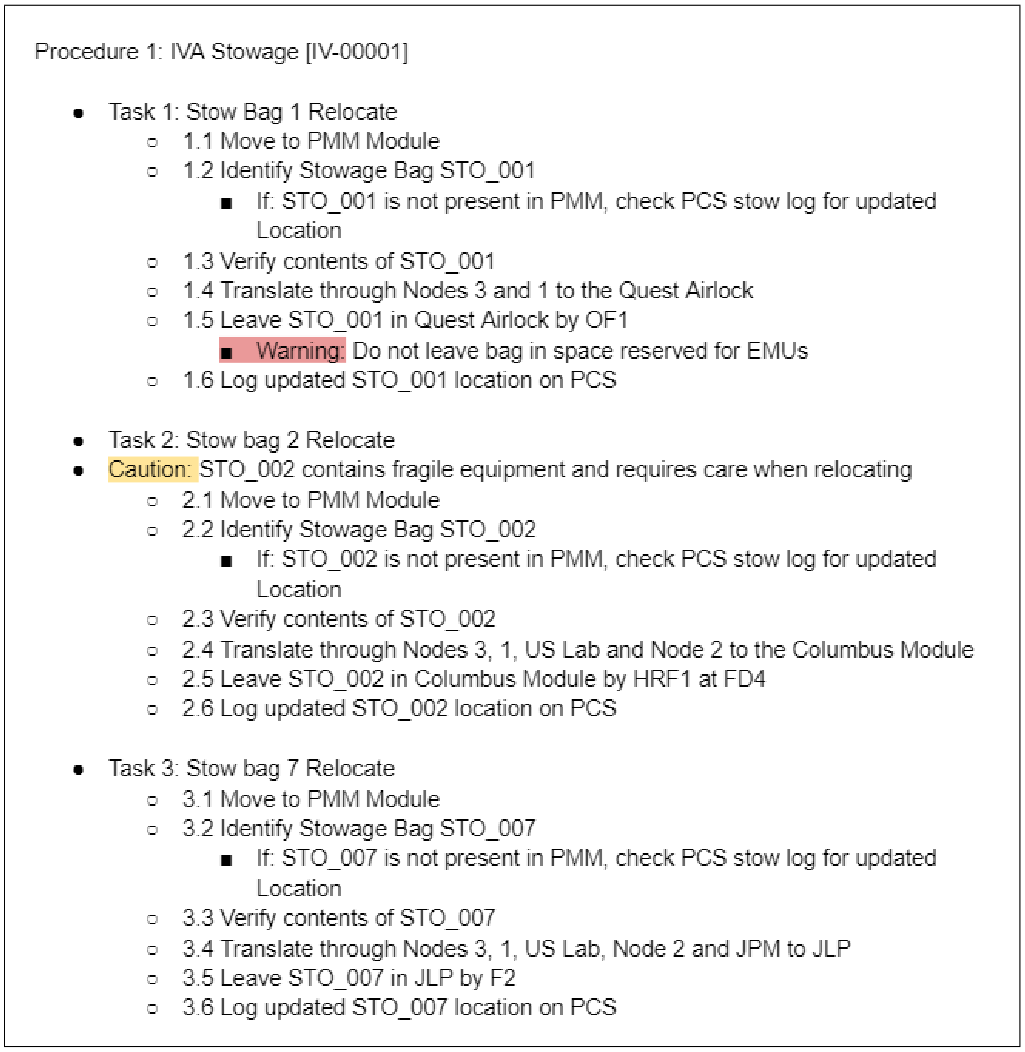
[33] Geoffrey R. Norman, Linda J. Muzzin, Reed G. Williams, and David B. Swanson. Simulation in health sciences education. *Journal of Instructional Development*, 8, 3 1985.

[34] Steven M. Stern. An extensible object-oriented executor for the timeliner user interface language, 5 2005.

[35] Inderraj Singh Grewal. Self-customized electronic procedures for just in time training of space telerobotics, 2018.

APPENDIX A

PROCEDURES USED IN RESULTS



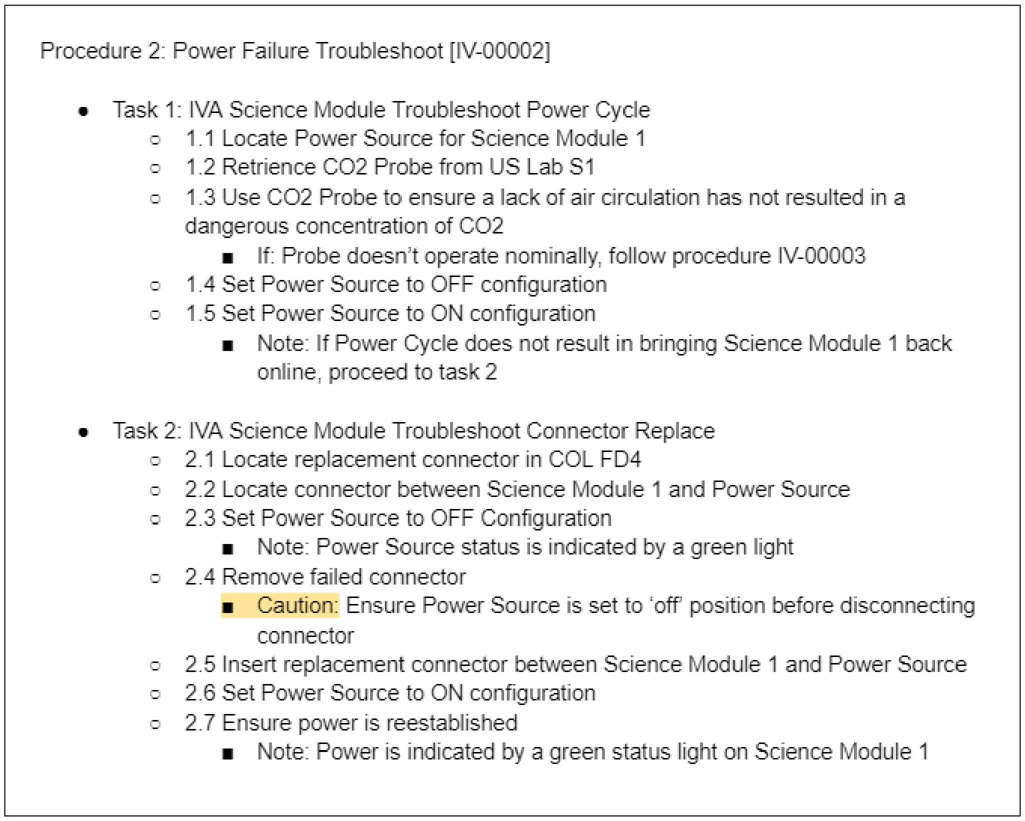
Procedure 1: IVA Stowage [IV-00001]

- Task 1: Stow Bag 1 Relocate
  - 1.1 Move to PMM Module
  - 1.2 Identify Stowage Bag STO_001
    - If: STO_001 is not present in PMM, check PCS stow log for updated Location
  - 1.3 Verify contents of STO_001
  - 1.4 Translate through Nodes 3 and 1 to the Quest Airlock
  - 1.5 Leave STO_001 in Quest Airlock by OF1
    - Warning: Do not leave bag in space reserved for EMUs
  - 1.6 Log updated STO_001 location on PCS

- Task 2: Stow bag 2 Relocate
- Caution: STO_002 contains fragile equipment and requires care when relocating
  - 2.1 Move to PMM Module
  - 2.2 Identify Stowage Bag STO_002
    - If: STO_002 is not present in PMM, check PCS stow log for updated Location
  - 2.3 Verify contents of STO_002
  - 2.4 Translate through Nodes 3, 1, US Lab and Node 2 to the Columbus Module
  - 2.5 Leave STO_002 in Columbus Module by HRF1 at FD4
  - 2.6 Log updated STO_002 location on PCS

- Task 3: Stow bag 7 Relocate
  - 3.1 Move to PMM Module
  - 3.2 Identify Stowage Bag STO_007
    - If: STO_007 is not present in PMM, check PCS stow log for updated Location
  - 3.3 Verify contents of STO_007
  - 3.4 Translate through Nodes 3, 1, US Lab, Node 2 and JPM to JLP
  - 3.5 Leave STO_007 in JLP by F2
  - 3.6 Log updated STO_007 location on PCS

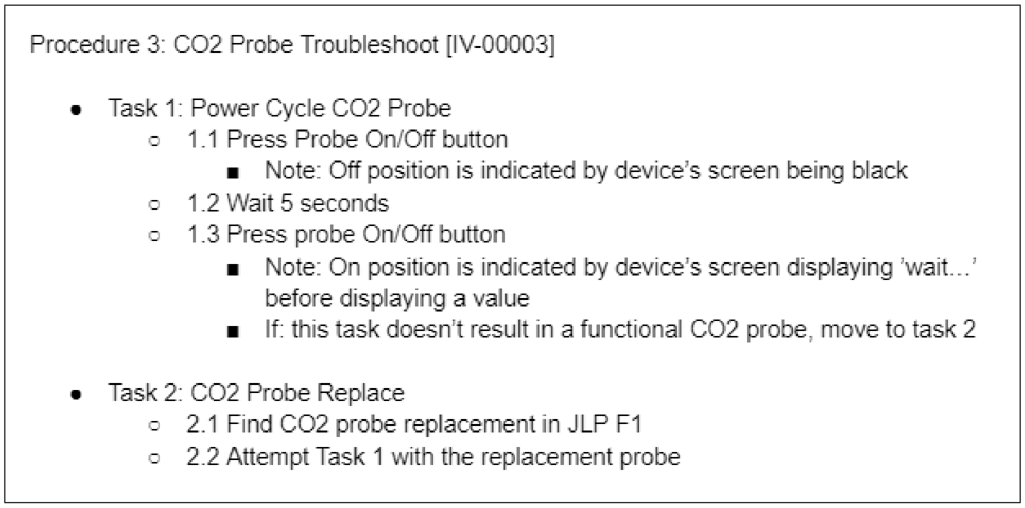
Figure A.1: Procedure 1 for Simulation 1 - IVA Stowage Procedure


Procedure 2: Power Failure Troubleshoot [IV-00002]

- Task 1: IVA Science Module Troubleshoot Power Cycle
  - 1.1 Locate Power Source for Science Module 1
  - 1.2 Retrience CO2 Probe from US Lab S1
  - 1.3 Use CO2 Probe to ensure a lack of air circulation has not resulted in a dangerous concentration of CO2
    - If: Probe doesn't operate nominally, follow procedure IV-00003
  - 1.4 Set Power Source to OFF configuration
  - 1.5 Set Power Source to ON configuration
    - Note: If Power Cycle does not result in bringing Science Module 1 back online, proceed to task 2

- Task 2: IVA Science Module Troubleshoot Connector Replace
  - 2.1 Locate replacement connector in COL FD4
  - 2.2 Locate connector between Science Module 1 and Power Source
  - 2.3 Set Power Source to OFF Configuration
    - Note: Power Source status is indicated by a green light
  - 2.4 Remove failed connector
    - Caution: Ensure Power Source is set to 'off' position before disconnecting connector
  - 2.5 Insert replacement connector between Science Module 1 and Power Source
  - 2.6 Set Power Source to ON configuration
  - 2.7 Ensure power is reestablished
    - Note: Power is indicated by a green status light on Science Module 1

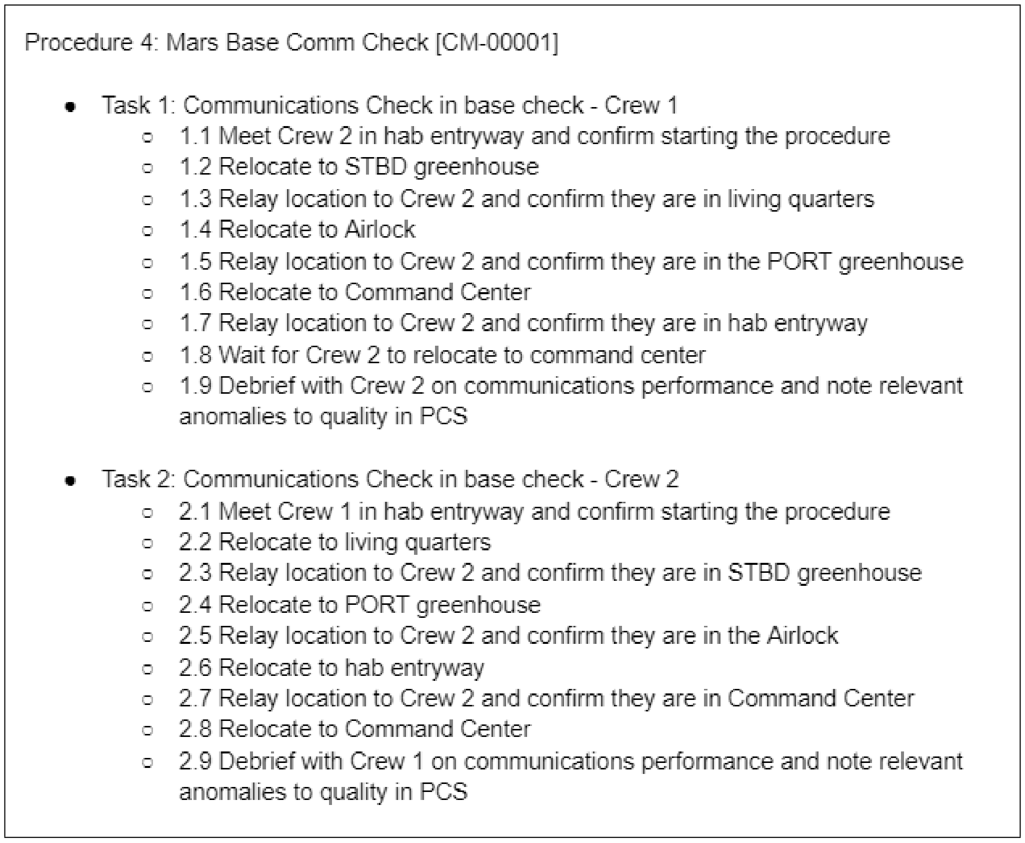
Figure A.2: Procedure 2 for Simulation 1 - IVA Science Module Power Failure Troubleshoot


Procedure 3: CO2 Probe Troubleshoot [IV-00003]

- Task 1: Power Cycle CO2 Probe
  - 1.1 Press Probe On/Off button
    - Note: Off position is indicated by device's screen being black
  - 1.2 Wait 5 seconds
  - 1.3 Press probe On/Off button
    - Note: On position is indicated by device's screen displaying 'wait...' before displaying a value
    - If: this task doesn't result in a functional CO2 probe, move to task 2

- Task 2: CO2 Probe Replace
  - 2.1 Find CO2 probe replacement in JLP F1
  - 2.2 Attempt Task 1 with the replacement probe

Figure A.3: Procedure 3 for Simulation 1 - IVA CO2 Probe Failure Troubleshoot



Procedure 4: Mars Base Comm Check [CM-00001]

- Task 1: Communications Check in base check - Crew 1
    - 1.1 Meet Crew 2 in hab entryway and confirm starting the procedure
    - 1.2 Relocate to STBD greenhouse
    - 1.3 Relay location to Crew 2 and confirm they are in living quarters
    - 1.4 Relocate to Airlock
    - 1.5 Relay location to Crew 2 and confirm they are in the PORT greenhouse
    - 1.6 Relocate to Command Center
    - 1.7 Relay location to Crew 2 and confirm they are in hab entryway
    - 1.8 Wait for Crew 2 to relocate to command center
    - 1.9 Debrief with Crew 2 on communications performance and note relevant anomalies to quality in PCS

- Task 2: Communications Check in base check - Crew 2
    - 2.1 Meet Crew 1 in hab entryway and confirm starting the procedure
    - 2.2 Relocate to living quarters
    - 2.3 Relay location to Crew 2 and confirm they are in STBD greenhouse
    - 2.4 Relocate to PORT greenhouse
    - 2.5 Relay location to Crew 2 and confirm they are in the Airlock
    - 2.6 Relocate to hab entryway
    - 2.7 Relay location to Crew 2 and confirm they are in Command Center
    - 2.8 Relocate to Command Center
    - 2.9 Debrief with Crew 1 on communications performance and note relevant anomalies to quality in PCS

Figure A.4: Procedure 1 for Simulation 2 - Mars Surface Base Communications Check

APPENDIX B

GOOGLE DRIVE LINKS TO VIDEO CAPTURES OF SIMULATIONS CREATED USING SIMBAD

The material in this appendix is supplemental to the results of this thesis. Specifically, it entails edited videos of the simulations described in section 5, "Results and Discussion". The first video is narrated by the author describing the environment, models, and events occuring during the simulation. The second contains audio recorded between the two users during the simulation as they progress through the procedure. (If links do not function properly, send a message to wcyoung18@gmail.com)

B.1 ISS IVA Simulation

`https://drive.google.com/file/d/1_Kvf2A_sXFF5QaWU3vkh1v-mse-5a16q/view?usp=sharing`

B.2 Mars Surface Base Multi User Simulation

`https://drive.google.com/file/d/1XY3biTOYcuXrW1Pp1CXZ0iTlPeil1Q4W/view?usp=sharing`

APPENDIX C

IN-ENGINE SCREEN CAPTURES FROM ISS IVA SIMULATION

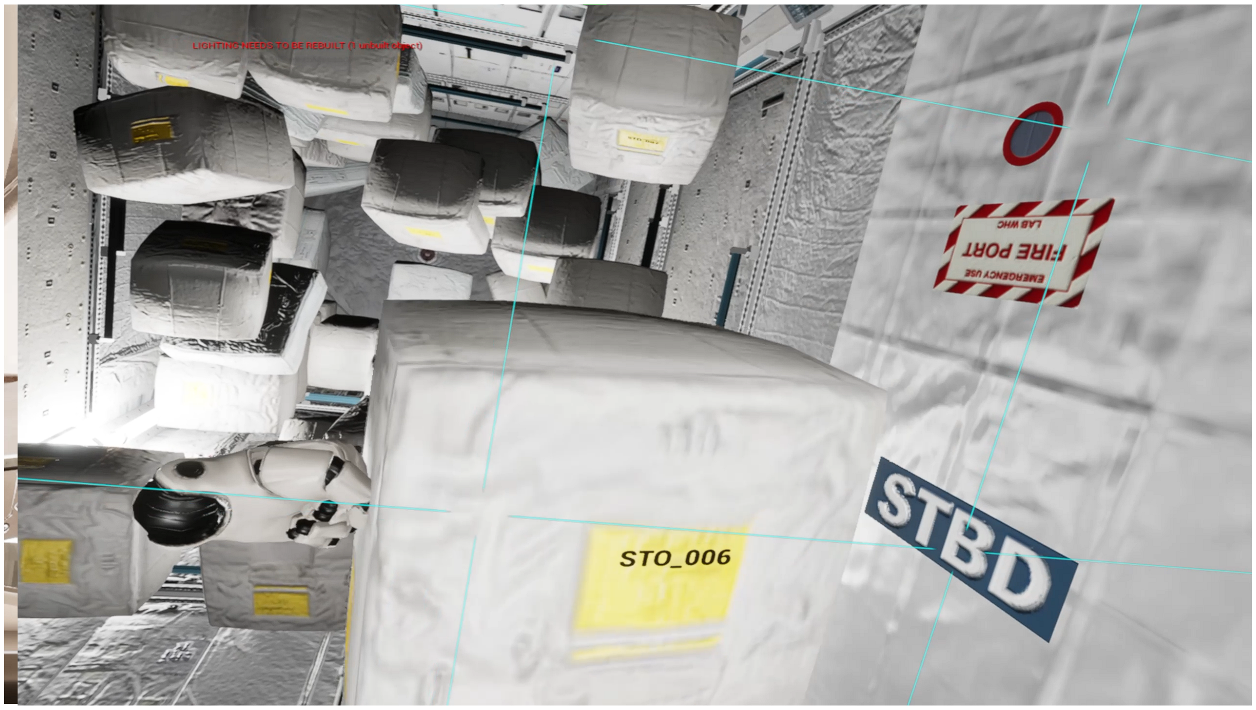


Figure C.1: User in simulation attempting to locate a specific stowage bag in the PMM of the ISS

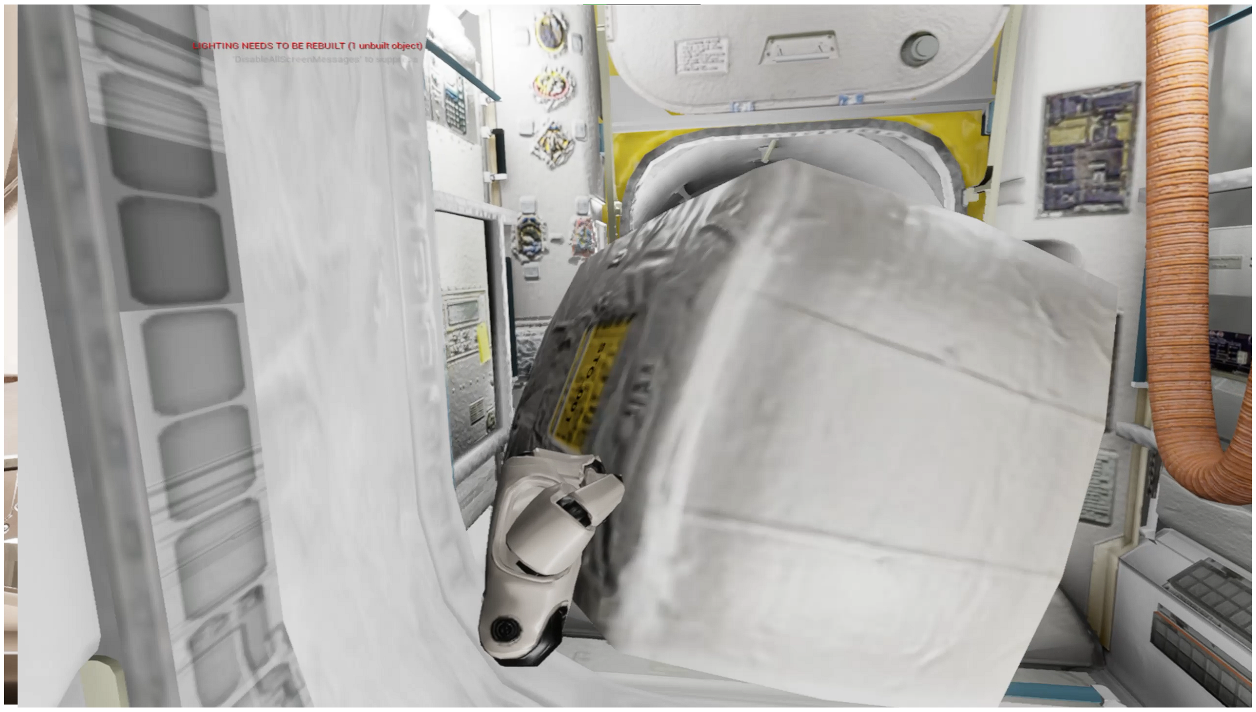
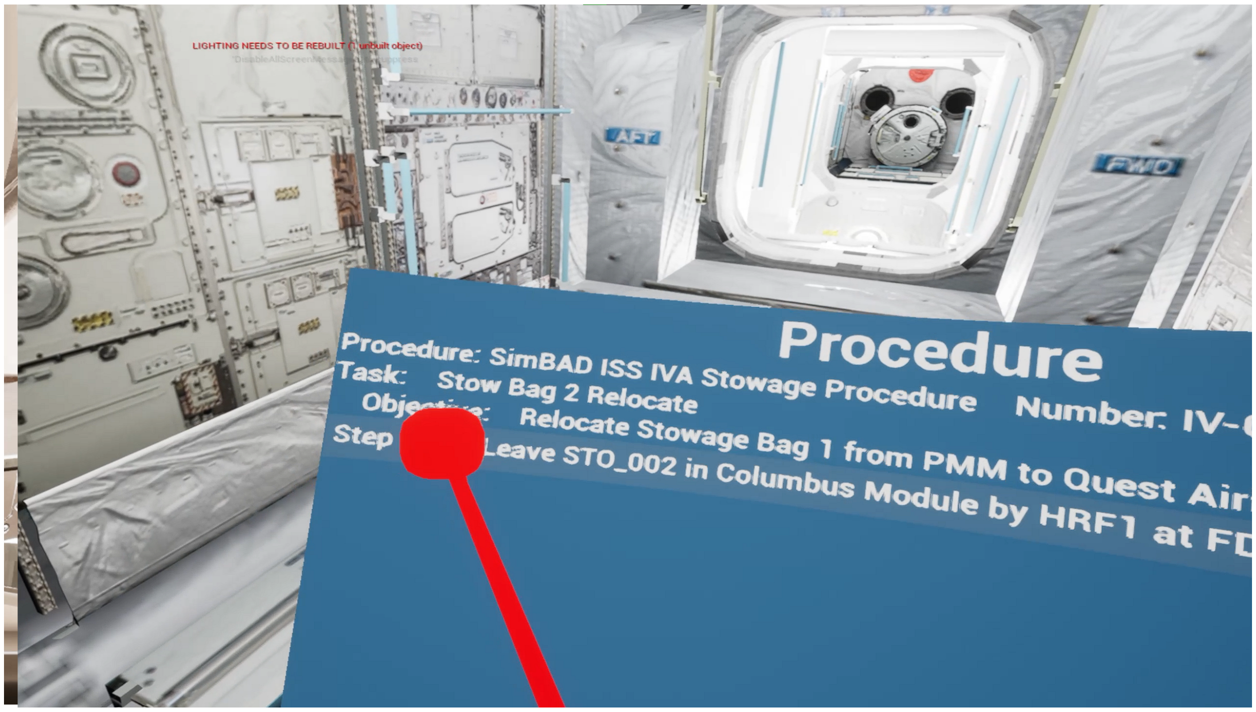

LIGHTING NEEDS TO BE REBUILT (1 unbuilt object)

Figure C.2: User relocating a stowage bag to the Quest airlock


LIGHTING NEEDS TO BE REBUILT (1 unbuilt object)

Procedure

Procedure: SimBAD ISS IVA Stowage Procedure Number: IV-

Task: Stow Bag 2 Relocate

Objective: Relocate Stowage Bag 1 from PMM to Quest Air

Step Leave STO_002 in Columbus Module by HRF1 at FD

Figure C.3: User interacting with procedure UI to track progress and get information for tasks

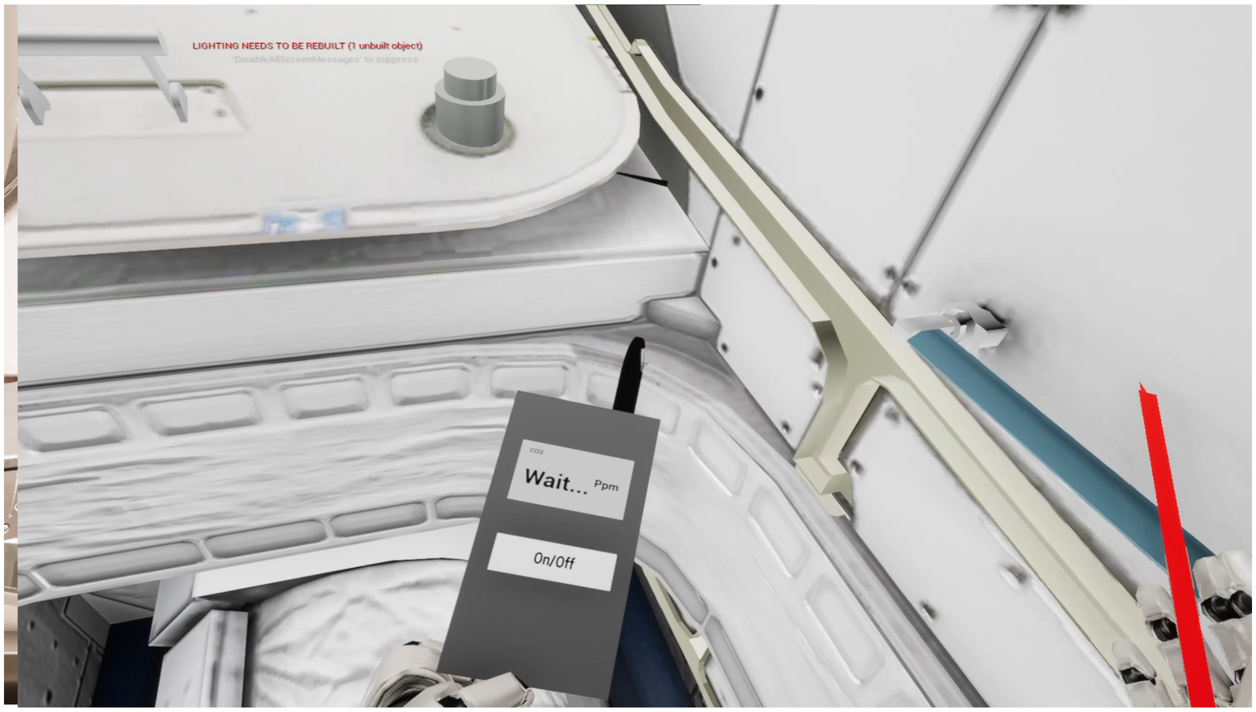

LIGHTING NEEDS TO BE REBUILT (1 unbuilt object)
'DisableAllScreenMessages' to suppress
CO2
Wait... Ppm
On/Off

Figure C.4: User working through power cycle steps of replacement CO2 Probe

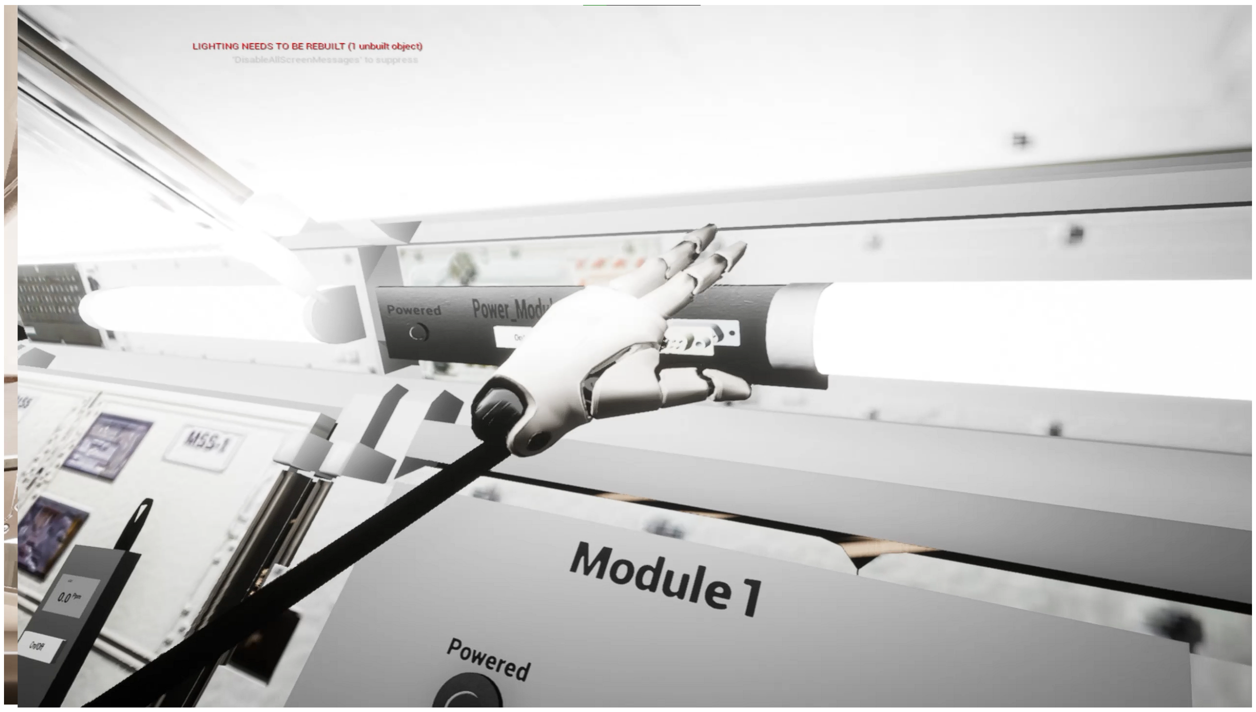
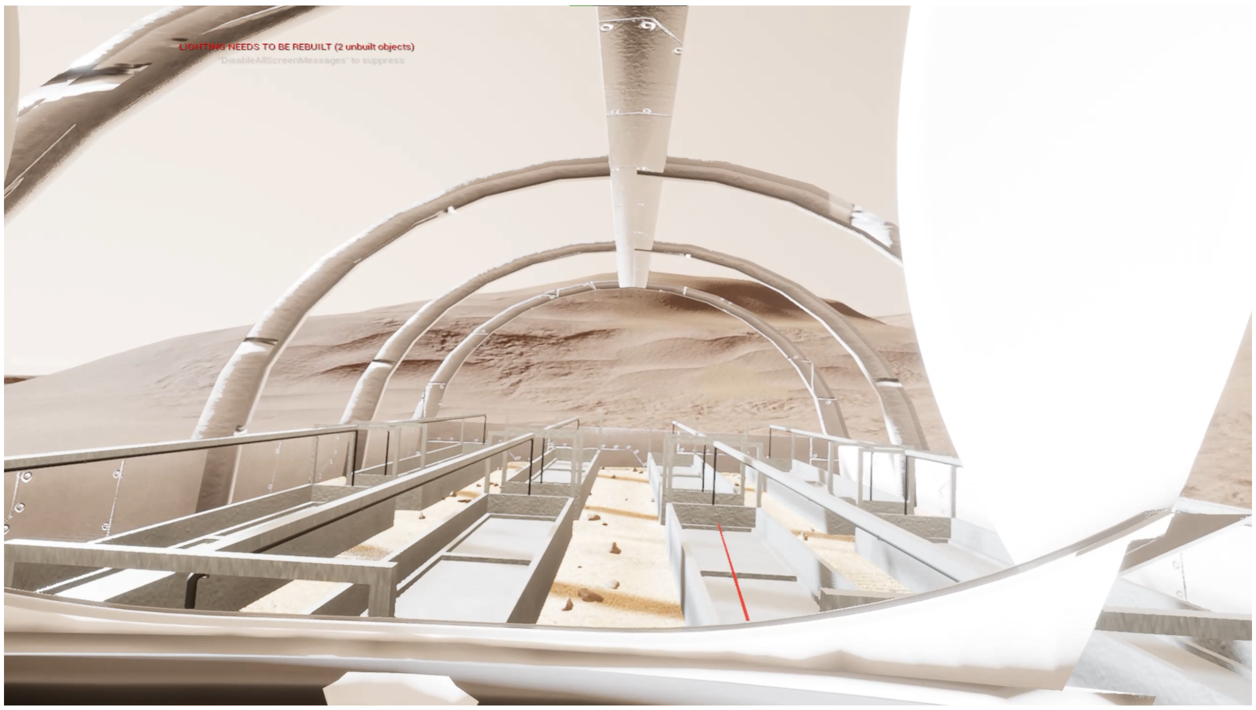

LIGHTING NEEDS TO BE REBUILT (1 unbuilt object)
'DisableAllScreenMessages' to suppress
Powered
Power_Modu
MSS-1
0.0 Ppm
Module 1
Powered

Figure C.5: User interacting with connector being replaced for Science Module Troubleshoot procedure

APPENDIX D

IN-ENGINE SCREEN CAPTURES FROM MARS SURFACE BASE MULTI USER SIMULATION



```
LIGHTING NEEDS TO BE REBUILT (2 unbuilt objects)
'DisableAllScreenMessages' to suppress
```

Figure D.1: User entering greenhouse of custom habitat on the surface of Mars near Gale Crater

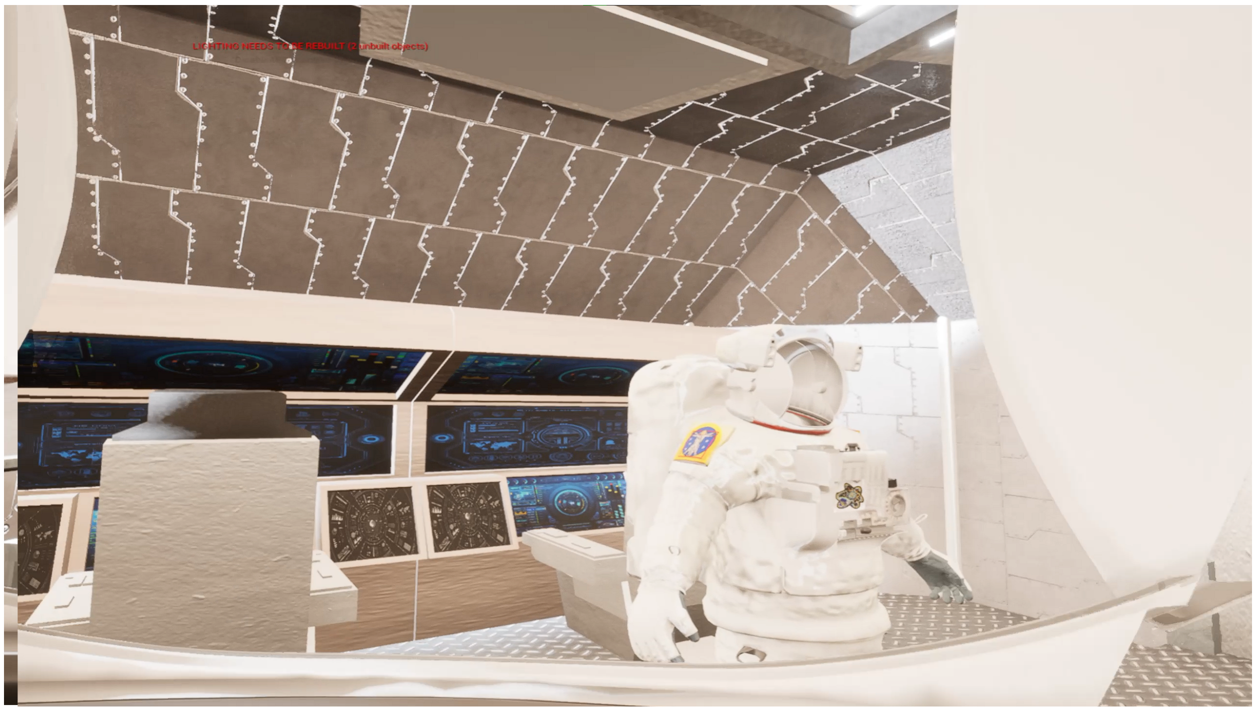

LIGHTING NEEDS TO BE REBUILT (2 unbuilt objects)

Figure D.2: Both users convening in habitat command module to carry out final procedure steps

APPENDIX E

ADDITIONAL UI USED FOR SIMBAD

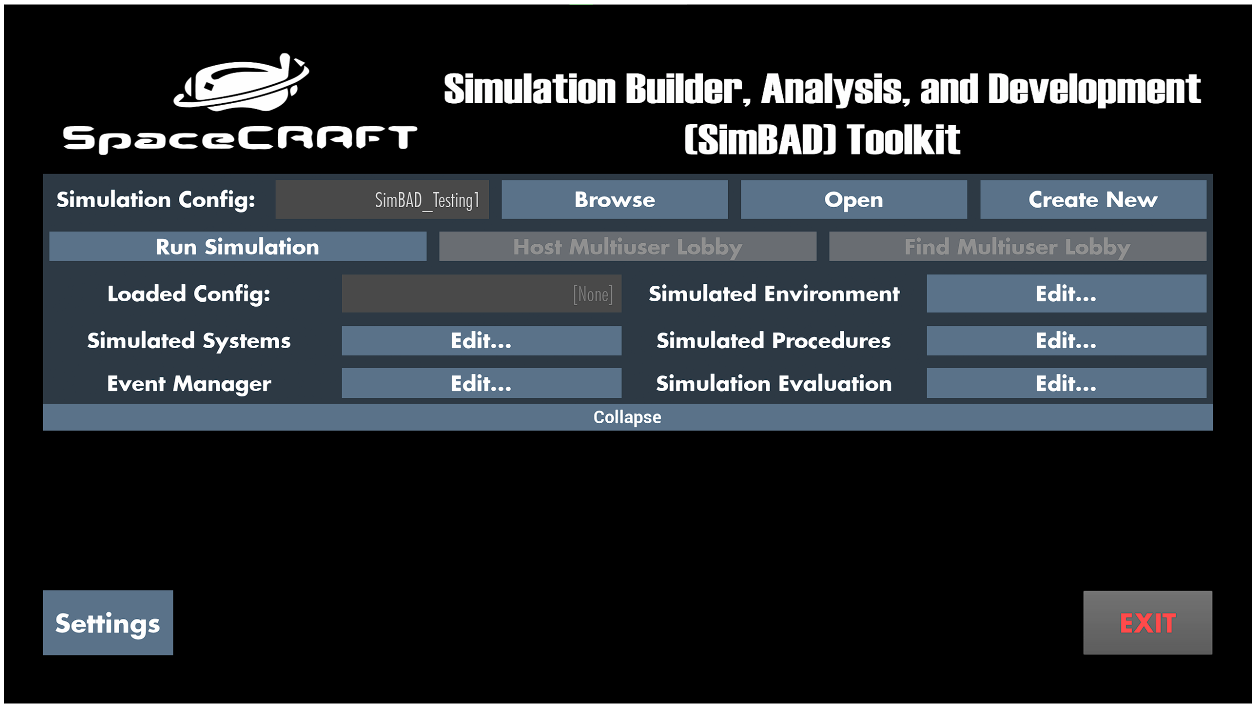
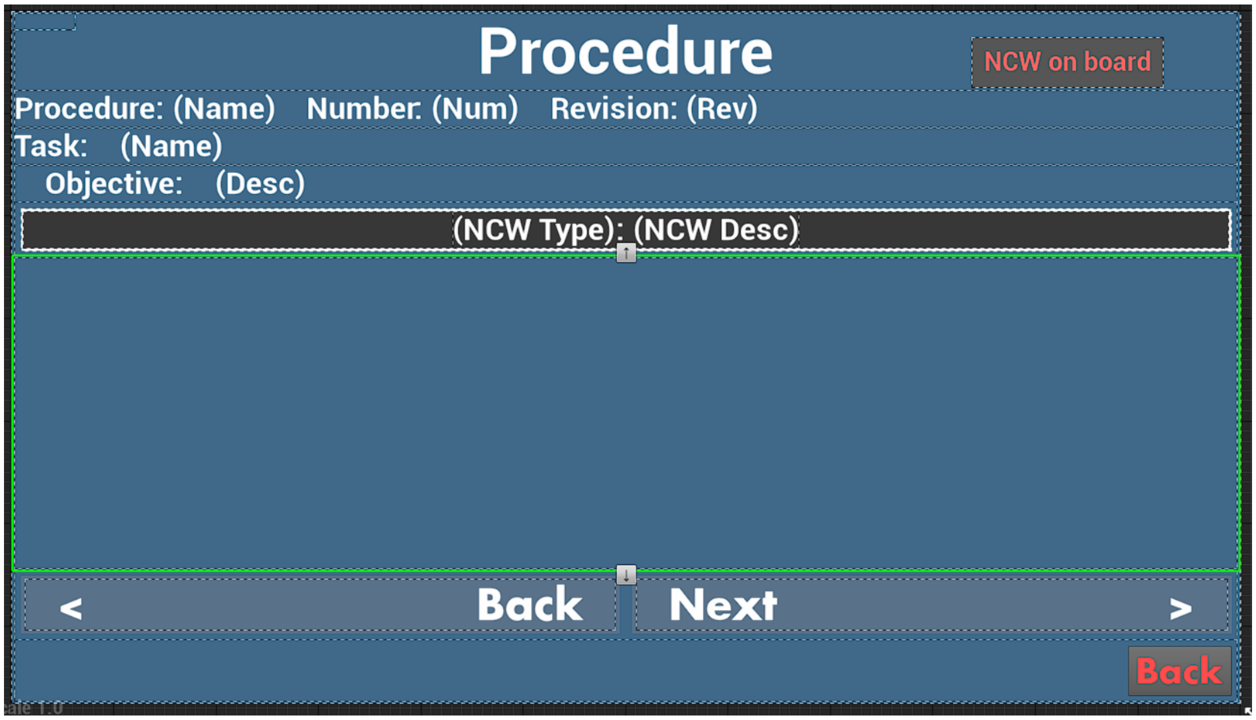


Figure E.1: SimBAD Front End


Procedure

NCW on board

Procedure: (Name) Number: (Num) Revision: (Rev)

Task: (Name)

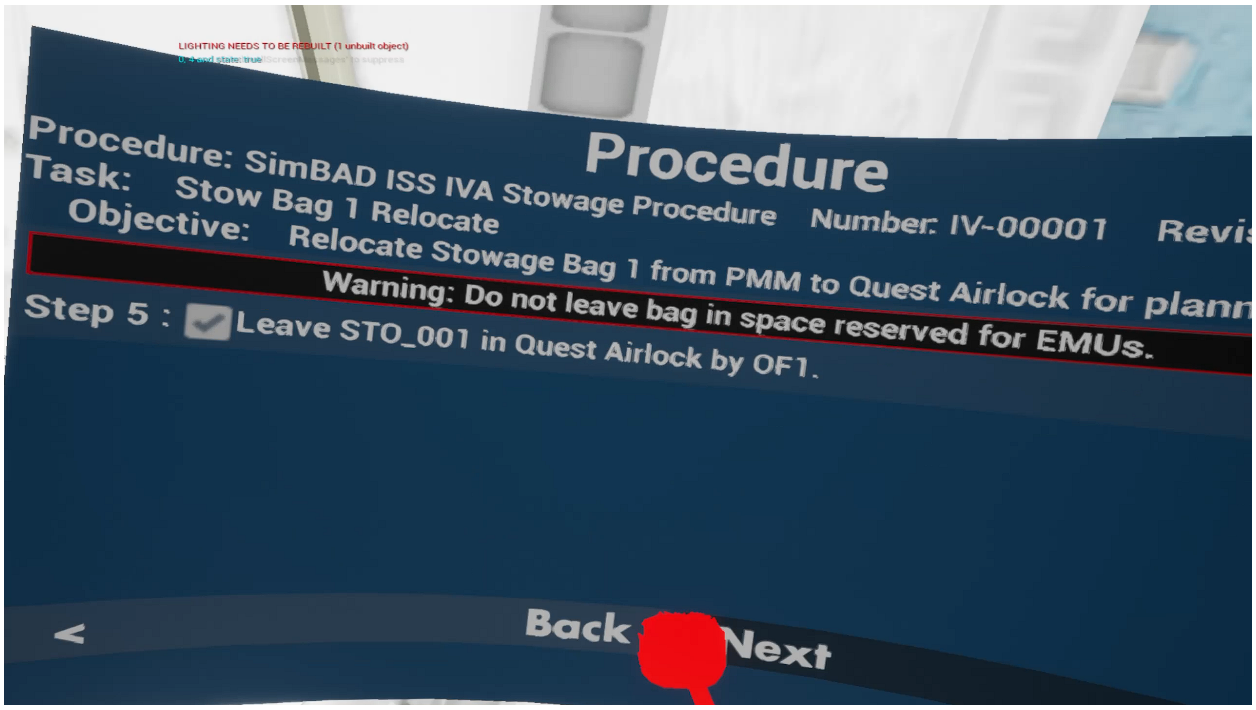
Objective: (Desc)

(NCW Type): (NCW Desc)

< Back Next >

Back

Figure E.2: In-Sim Procedure UI Design


LIGHTING NEEDS TO BE REBUILT (1 unbuilt object)

Procedure

Procedure: SimBAD ISS IVA Stowage Procedure Number: IV-00001 Revi

Task: Stow Bag 1 Relocate

Objective: Relocate Stowage Bag 1 from PMM to Quest Airlock for plann

Warning: Do not leave bag in space reserved for EMUs.

Step 5 : ✓ Leave STO_001 in Quest Airlock by OF1.

< Back Next

Figure E.3: User interacting with In-Sim Procedure UI