# USING OBJECT PROCESS METHODOLOGY TO DEVELOP INTERFACES AND SMART ELECTRONIC PROCEDURES FOR SIMULATED TELEROBOTIC OPERATIONS

By

Yongkai Eugene Yang

B.Eng. Mechanical Engineering, Nanyang Technological University

SUBMITTED TO THE SYSTEM DESIGN AND MANAGEMENT PROGRAM IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN ENGINEERING AND MANAGEMENT

AT THE

MASSACHUSETTS INSITITUTE OF TECHNOLOGY

FEBRUARY 2017

Signature of Author Signature redacted

Yongkai Eugene Yang
System Design and Management Program
January 09, 2017

Certified by Signature redacted

Dr. Charles M. Oman, Ph.D.
Senior Lecturer, Department of Aeronautics and Astronautics, Man Vehicle Laboratory
Thesis Supervisor

Certified by Signature redacted

Prof. Dov Dori, Ph.D.
Information and Systems Engineering Professor
Thesis Supervisor

Accepted by Signature redacted

Prof. Warren Seering, Ph.D.
Weber-Shaughness Professor of Mechanical Engineering

This page is intentionally left bank.

# USING OBJECT PROCESS METHODOLOGY TO DEVELOP INTERFACES AND SMART ELECTRONIC PROCEDURES FOR SIMULATED TELEROBOTIC OPERATIONS

By

Yongkai Eugene Yang

Submitted to the System Design and Management Program
on January 09, 2017 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in Engineering and Management

## ABSTRACT

This thesis addresses two questions: 1) How should an existing space telerobotic simulator be extended to incorporate Smart Electronic Procedures (SEP)? 2) Are there benefits to using a single Object-Process Methodology (OPM; ISO 19540) from the field of system engineering as an alternative to the Task Analysis (TA) methods traditionally used in human factors engineering (HFE)? A NASA sponsored EP development project provided the opportunity to investigate.

HFE traditionally supports analysis and design by using multiple Task Analysis (TA) methods, including Hierarchical TA (HTA), Tabular TA (TTA), and Abstraction Hierarchy (AH). But the three techniques combined neither defines all the necessary preconditions for each task to succeed nor produces an executable model of the entire system that can be simulated and tested for logical correctness, and results are presented in a form that can be difficult for others to comprehend. To evaluate OPM usefulness, a space telerobotic operation was analyzed using successive HTA, TTA and AH techniques, and compared with a corresponding OPM analysis. A single OPM model precisely specified the preconditions and post-conditions for all the processes and described the relationships between system objects—both human and non-human—and the processes in its hierarchy of Object-process Diagrams that translate on the fly to Object-Process Language – a subset of natural English. Advantages of OPM include its holistic approach, bimodal presentation, simplicity, computability, and logical correctness testing capability via animated simulation.

The OPM model also defined the architecture and logic of the SEP and the Control Panel(CP) – two essential parts incorporated into the existing telerobotic simulator. Simulated subsystems were introduced to enable simulation of setup, shutdown and off-nominal scenarios as defined by the OPM model. The SEP has several automation options, catches erroneous actions and ensures preconditions for each step are satisfied. The CP interfaces has several functions, including automating failure recovery and showing automated customize procedures to restore system to pre-failure configuration.

This thesis considered only one application and further applications are needed to demonstrate the utility of OPM in the broader HFE domain. Nonetheless, the advantages of OPM over traditional TA methods demostrated OPM as a viable alternative to current HFE practices.

Thesis Supervisor 1: Dr. Charles M. Oman, Ph.D.
Title: Senior Lecturer, Department of Aeronautics and Astronautics, Man Vehicle Laboratory

Thesis Supervisor 2: Prof. Dov Dori, Ph.D.
Title: Information and Systems Engineering Professor

## ACKNOWLEDGEMENTS

First and foremost, I would like to thank my sponsor, Defence Science and Technology Agency (DSTA) for giving me the opportunity to pursue a postgraduate education at Massachusetts Institute of Technology (MIT). This would not be possible without the recommendation and support of Mr. Pek Beng Tit, Mr. Chong Ban Kiat, Mr. V Shana Babu and Ms. Gayle Chan.

This research was supported by NASA under Contract NNX15AW35G. I would like to thank my thesis advisors, Dr. Charles Oman (Chuck) and Prof. Dov Dori. Chuck had provided insightful suggestions and valuable guidance throughout this research, galvanizing me to improve and refine my models and electronic procedure simulator design. I'm also very grateful to Chuck for his patient and thoroughness in reviewing my thesis. It was a great privilege to learn directly from Dov, who conceived the Object Process Methodology (OPM). Dov had always been responsive and helpful in guiding me through the development of the OPM model for this research and imparting me useful techniques towards improving the OPM model. My many thanks to the other members of the project team, Dr. Andrew Liu and Dr. Raquel Galvan. Both of whom had helped me greatly in the review of my thesis and the development of the electronic procedure simulator. I would also like to thank Ms. Heidi Jennings, (NASA-JSC) for her helpful discussion and input in the task and procedure development efforts, and Ms. Martina Stadler and Ms. Annie Abay for their contributions to the development of the simulator.

Next, I would also like to extend my appreciation to the System Design and Management (SDM) faculty and staff for their help throughout the program and my fellow classmates for making my education at MIT memorable and fruitful.

Most importantly, my deepest thanks to my wife, Huimin for her unwavering support and love. She had always been understanding towards my lack of time to be with her and continued to care for me. Her delicious homemade lunchboxes and foods never fail to keep me going when things got tough during the SDM program. Without the support of my loving wife and family back in Singapore, these would never be possible.

# TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

# 1 THESIS INTRODUCTION

## 1.1 Motivation

Space Exploration began during the Cold War Era, starting with the first man-made object, Sputnik 1 that orbit around the earth in 1957 [1]. The space race led to the inauguration of the Apollo program that landed the first human, Astronaut Neil Armstrong on the Moon on July 20, 1969[2]. Since then, manned space exploration continues but primarily in low-Earth orbit aboard the International Space Station (ISS). The vision of manned space exploration is to send humans into deep space, and the National Aeronautics and Space Administration (NASA) aims to send humans to Mars in the 2030's [3]. This goal requires the buildup of capabilities through research and development.

One area of research in support of the deep space exploration vision is in the domain of Human-Computer Interaction (HCI). To date, NASA's space vehicles systems – though automated at the subsystem level – have been largely crew-operated using physical or soft switches and knobs, simple electronic procedures, with ground controllers supporting the crew in real-time for time- and safety-critical tasks, including system configuration setup and troubleshooting. Future spacecraft will likely be controlled by the crew almost exclusively through software displays and employ electronic procedures that sense subsystem states and provide macro step execution capabilities. The long latency between ground control and future deep space missions will affect the efficacy of ground support, requiring the crew to operate autonomously. Hence, extensive use of automation is envisaged in future space missions to lighten the crew workload. Integrating automation into electronic procedure execution is a new operations concept to NASA and, as such, has created human factors concerns, such as overreliance, loss of situation awareness, and loss of currency or proficiency in performing procedures. These concerns warrant research to identify the best approaches to integrating electronic procedures and critical system information and developing pertinent HCI guidelines.

## 1.2 System Problem Statement

Research on guidelines for electronic procedure with automation features can be broken into two phases – (1) the development of a spacecraft simulator with electronic procedure imbedded and (2) the performance of human-in-the-loop studies to provide empirical evidence to support hypothesis and recommend HCI guidelines. This thesis will focus on the developmental efforts of a space telerobotic simulator incorporating some of these advanced automation features.

Traditional methods for human-machine interface design utilise Task Analysis, for example Hierarchical Task Analysis (HTA) [4], an HTA extension called Tabular Task Analysis (TTA) and a complementary technique called Abstraction Hierarchy (AH), that is part of Cognitive Work Analysis (CWA) [5], [6]. HTA, TTA and AH will be reviewed in Chapter 3. These techniques define the hierarchy of goals and tasks, and have proven useful in identifying gaps in formal knowledge and therefore have some face validity. Both methods, however, are human resource intensive, and have not proven reliable, as different analysists produce different models and are unable to predict design flaws, the sequence of states associated with each task, or to test whether the task description is logically complete. Also, the results are in graphical and tabular formats that are not easily comprehended by people who are not domain experts.

Consequently, two key system problem statements will be addressed in this thesis.

***System Problem Statement One:***

To evaluate the value of ISO19450 standard – Object-Process Methodology (OPM) for system designs, and experiment designs and studies pertaining to human factors with respect to traditional methods – Task Analysis(TA);

More specifically, using the space telerobotic electronic procedures project as an example, the thesis examines:

1. What are the important similarities and differences between OPM and TA?

2. Can human factors specialists use OPM as a method to model an entire human-machine system?

3. Can OPM produce an executable model, which can be tested for logical correctness?

4. Can the OPM analysis results be used to support human-machine interface design and also to confirm normal flow of operations and detect procedural errors and off-nominal system states in real-time.

**By** separately using Task Analysis and OPM to develop the systems architecture, procedures, interface requirements and designs, and scenarios insofar such that the advantages and disadvantages of OPM over Task Analysis can be comprehensively compared;

**Using** qualitative and where possible, quantitative assessments.

***System Problem Statement Two:***

**To** enable the investigation of the effect of integral and imbedded electronic procedures in future spacecraft systems on human performance;

**By** developing a simulator to emulate future spacecraft interfaces and designs with a wide range of capabilities, including (1) selectable automation options (e.g. macro procedure execution), (2) sensing the current system state and performing automatic verification, (3) preventing execution of a procedure if conditions are not satisfied, and (4) providing different display feedback that allows human subjects to perform a full set of integrated electronic procedures for simulated robotics task scenarios, including system setup, support Extra-Vehicular Activity (EVA) operations and system shut down, coupled with off-nominal and system failure scenarios;

**Using** Object-Process Methodology (OPM) – the ISO 19450 conceptual modeling methodology to create a computable model of the system and subsequently program the designs (e.g. control panel and switches), system logic (e.g. relationships between sub-systems) and automation options from the OPM into a telerobotics operations simulator.

## 1.3 System Design Approach

The simulation concept for the human-in-the-loop experiment is based on the robotic arm operations onboard the ISS as a proxy for a general automated system with imbedded electronic procedure in human supervisory control with specific interest in the future robotic arm or rover operations during deep space exploration. This simulator was built on the existing MIT Robotic Workstation Simulator (MIT-RWSS) that had been successfully used in several previous NASA and National Space Biomedical Research Institute (NSBRI) robotics projects [7]–[9]. None of the capabilities required for the experiment existed in the previous version. Therefore, these capabilities were developed and coded into the program.

The development of the simulator involved three primary tasks:

1) Creating the full-set of procedure for system setup, Extra-Vehicular Activity (EVA) operations, system shutdown and failure recovery that resembles the actual procedures used onboard ISS for the robotic arm. The procedures were developed and reviewed for validity by a current NASA robotics instructor and flight controller, Ms. Heidi Jennings.

2) Developing the architecture for the simulation that contains the logical system preconditions (e.g. system A needs to be on prior to system B) similar to a real system, providing information on the user interfaces, system state and automation options.

3) Integrating the electronic procedure and system architecture in the MIT-RWSS.

The first two tasks were accomplished using two methodologies – Task Analysis and OPM. This enabled the comparison between the two methods to address the second system problem statement. The third was achieved through software coding using Vizard, a library of virtual reality functions written in Python.

## 1.4 Thesis Organization

- Chapter 1, ***THESIS INTRODUCTION***: Provides the motivation, system problem statements and design approaches.
- Chapter 2, ***BACKGROUND***: Provides the background of the International Space Station robotic arm, MIT Robotic Workstation simulator, the needs of NASA and the requirements for the simulator.
- Chapter 3, ***TASK ANALYSIS***: Briefly discusses task analysis, introduces the Hierarchical Task Analysis (HTA), and explains the development of procedure using HTA and expansion of HTA to support system design and human factors considerations.
- Chapter 4, ***TELEROBOTIC ELECTRONIC PROCEDURES AND SIMULATOR DEVELOPMENT USING OBJECT PROCESS METHODOLOGY (OPM):*** Provides an introduction of OPM, highlights the development of procedures and simulator design using OPM and discusses electronic procedure automation options with OPM.
- Chapter 5, ***TASK ANALYSIS (TA) AND OBJECT PROCESS METHDOLOGY (OPM) DISCUSSION:*** Compares the similarities, benefits and disadvantages of the two methodologies – TA and OPM – with respect to their applications in this thesis.
- Chapter 6, ***MIT-RWSS SIMULATOR ENHANCEMENT AND RELATIONSHIPS TO OPM:*** Discusses the capabilities and features of the simulator developed, including those derived from the OPM model.
- Chapter 7, ***SUMMARY AND DISCUSSION***: Summaries the conclusions of the thesis and recommendations for future works.

# 2 BACKGROUND

## 2.1 The Robotic System

At the International Space Station (ISS), a robotic system known as the Mobile Servicing System (MSS) is used for assembly of the ISS, maintenance of the external system, movement of equipment and supplies around ISS and support of astronaut Extra-Vehicular Activities (EVA). The MSS comprises of five main systems, namely, the Space Station Remote Manipulator System (SSRMS), Special Purpose Dexterous Manipulator (SDPM), Mobile Transporter (MT), Mobile Remote Servicer Base System (MBS) and the Robotics Workstation (RWS). This system was jointly developed by the Canadian Space Agency (CSA) and National Aeronautics and Space Administration (NASA), and was commissioned in 2001.

### *2.1.1 Space Station Remote Manipulator System (SSRMS)*

The SSRMS is a three joint, robotic arm that is 17.6m long when fully extended and has seven degrees of freedom (DOF) as shown in Figure 1 below. It is frequently used to capture, move and then release attached devices or entire free floating spacecraft such as the JAXA H-II Transfer Vehicle (HTV), an unmanned ISS resupply spacecraft. On other occasions, space-suited astronauts ride the end of the arm to work sites, and use it for support when performing maintenance tasks. Both ends of the SSRMS arm have a Latching End Effector (LEE) equipped with power and data connections. This allows operators to "walk the arm" across a succession of Power Data Grapple Fixtures (PDGFs) on the ISS modules, or alternatively to a base system mounted on a mobile platform that runs along rails on the large ISS solar array truss. The PDGF has four receptacles that are mated to the LEE Latches for Power/Data/Video connection and structural/mechanical interface to the ISS. See Figure 2. In addition, the SSRMS has two functionally redundant strings (power, data, electronics), allowing continued operations in the event of a critical failure of a component [1]. Whichever LEE is attached, the other free end is used to move payloads around the ISS.

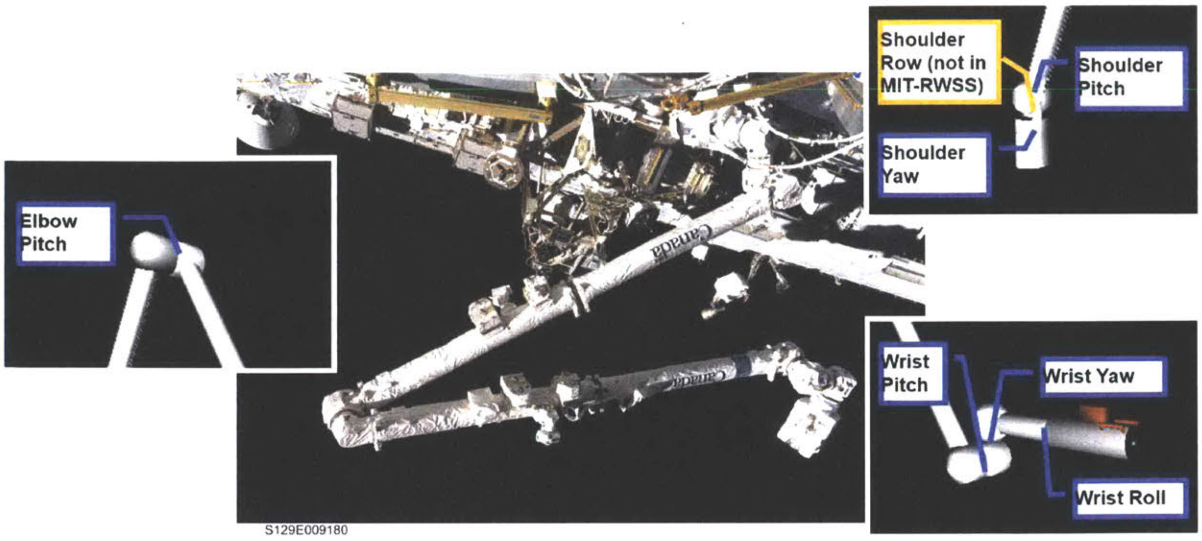


*Figure 1: SSRMS[10]*

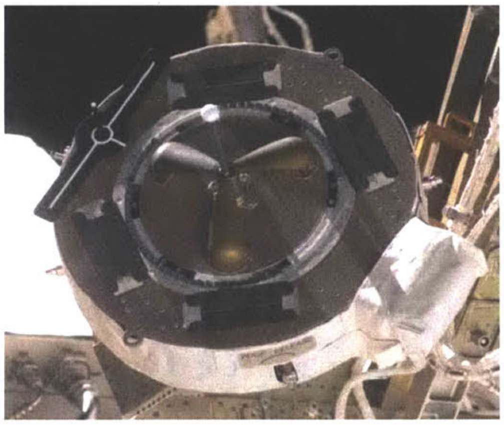


*Figure 2: Power Data Grapple Fixture[11]*

### 2.1.2 Special Purpose Dexterous Manipulator (SDPM)

The SDPM can be attached to the arm LEE to provide more dexterous remote manipulation capability in situations where it is not appropriate for a suited astronaut to ride the arm and/or to minimize EVA operations. The SDPM has two arms, each with seven DOF and connected to the LEE. It is dexterous to be able to manipulate small payloads (e.g. external Station batteries) located on the exterior of the ISS [12].

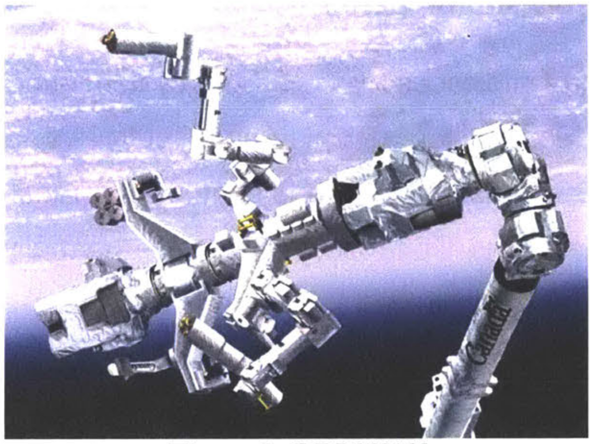
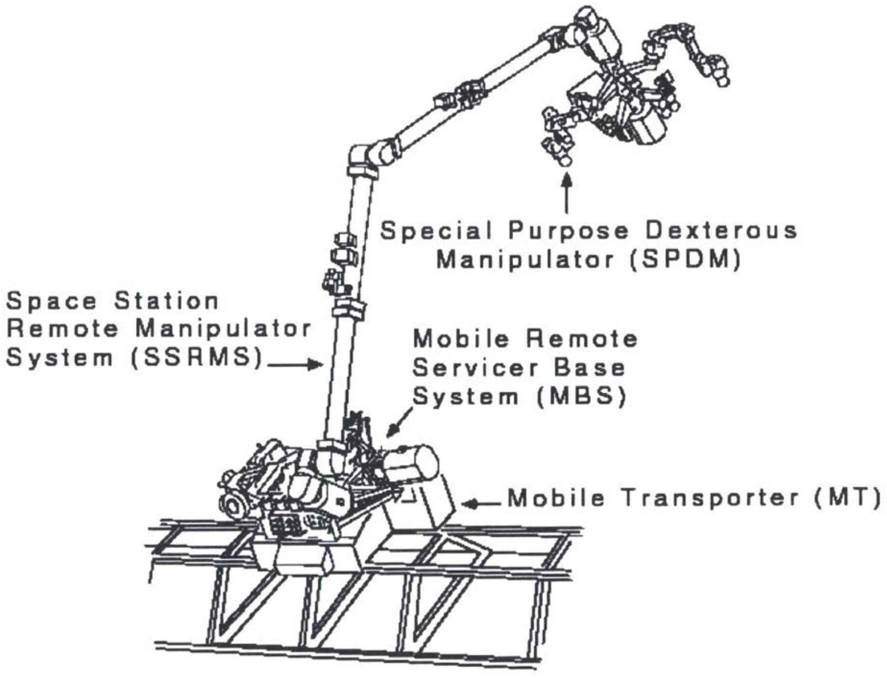


Figure 3: SPDM[13]

### 2.1.3 Mobile Base System (MBS) and Mobile Transporter (MT)

The MBS is attached to the MT installed on the exterior ISS truss and serve as an interface between the MT and the SSRMS. Together, the system provides a movable platform that enables the SSRMS to slide along rails on the ISS main truss structure to various part of the space station [14].


Special Purpose Dexterous Manipulator (SPDM)
Space Station Remote Manipulator System (SSRMS)
Mobile Remote Servicer Base System (MBS)
Mobile Transporter (MT)
Figure 4: MBS and MT[15]

### 2.1.4 *Robotic Workstation (RWS)*

The RWS is the control interfaces for the robotic system onboard ISS. It has two hand controllers, three video monitors, Display and Control Panel (DCP) and a Portable Computer System (PCS) laptop. The hand controllers, Translational Hand Controllers (THC) and Rotational Hand Controllers (RHC) are normally used to command the angular position and rotation of the LEE in three orthogonal axes respectively. Three video monitors are used to display camera views and overlays (e.g. brake status and joint angles). Switches and buttons to control cameras and the SSRMS and SPDM are located on the DCP. The PCS is used to display system status and control systems onboard the ISS.

There is also a second PCS (not shown in Figure 5) that generates a 3D perspective view of the arm known as the Dynamic Onboard Ubiquitous Graphics (DOUG). DOUG is a 3-D graphical viewing tool used for simulation scene displays, a situational awareness tool during robotic operations, an EVA planning tool for EVA task reviews, an SSRMS training simulator, and a simplified aid for EVA rescue flight simulator [16].

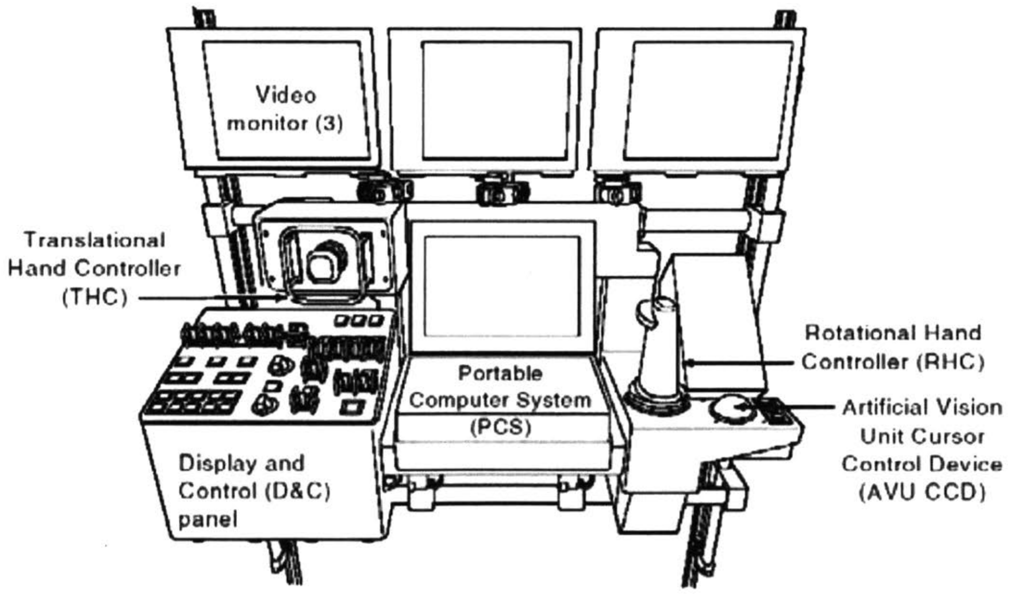

Video monitor (3)
Translational Hand Controller (THC)
Rotational Hand Controller (RHC)
Portable Computer System (PCS)
Artificial Vision Unit Cursor Control Device (AVU CCD)
Display and Control (D&C) panel

*Figure 5: RWS[15]*

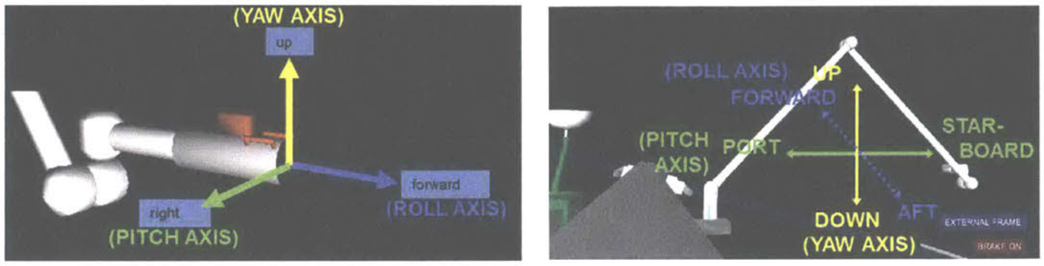
### *2.1.5 Operations and Safety of the SSRMS*

Typical SSRMS operations include repositioning the LEE to grapple a payload, dock and undock the payload, and/or move an EVA astronaut around the ISS to carry out maintenance tasks. It is important for the human operator controller to be able to maneuver the SSRMS and attached payloads around the ISS structure while maintaining a safe physical clearance since an injury to an astronaut or damages to the ISS, SSRMS or other systems could have cascading consequences because space operations are tightly coupled.

The SSRMS can be controlled from either robotics consoles in NASA's ISS Mission Control Center in Houston or by the astronauts onboard the ISS. RHC and THC inputs control three degrees of freedom rotation about and translation in two types of "command frames": internal and external. The internal command frame's axes are fixed with respect to the LEE i.e. the free end of the arm. When controlling using a camera attached to the end of the arm, use of this internal frame is relatively intuitive. In contrast, external command frames referenced to the ISS structure, with an origin and orientation chosen by the user for operational convenience, and are more intuitive when controlling using a camera fixed to the ISS, not the end of the arm. Figure 6 depicts the difference between internal and external command frame.

Although the SSRMS controller computes estimates of the arm location, these are subject to various errors, and therefore mission rules require that the operator to always verify the arm and payload clearance visually, either directly if controlling from the ISS cupola, or via video cameras, or both. Four video cameras are located on the SSRMS and fourteen others are located on ISS modules and the truss. The crew's views of the SSRMS movement is dependent on the location and orientation of camera views selected by the operator for display on video monitors. Currently, each monitor can only display one camera view. Similarly, the ground operators at Houston also depend on the same camera views to maintain situational awareness of the SSRMS' motions.

Nonetheless, there are limitations to the functions that ground operators can perform that primarily stems from flight safety rules. A conservative approach is justified due to the possibility of loss of communications between ground and the SSRMS during SSRMS operations [17].



Internal Command Frame External Command Frame

Figure 6: Internal and External Command Frame

All ISS crewmembers are qualified in basic telerobotic operations, but typically several are more highly trained and normally operate the system. Although the SSRMS in principle could be operated by one crewmember, in practice, NASA assigns two crew members. One serves as the primary operator, and a second manages the camera operations, handles communications with any EVA crew, monitors procedures, arm clearance and confirms the direction of motion. This approach was developed to improve crew situational awareness and improve the chances of detecting errors. An analogous "pilot flying/pilot monitoring" scheme has been used in aviation for many years for similar reasons. The second astronaut crew member can sometimes be replaced by the ground operator. But this approach, requires on real-time communication, which is not always possible [18].

The SSRMS control system also has automatic "fly to" modes where the operator specifies the destination configuration of the SSRMS (in joint space or Cartesian coordinates) and the SSRMS computes the trajectory required to reach it, and on command, executes it. This is referred as 'Autosequence' in the remaining sections. Though no input to the hand controllers is required from the operator in Autosequence, the operator is still required to maintain situational awareness of the SSRMS motion to ensure that clearances with surrounding structures and alignment with mating interfaces are maintained [19].

The velocity of the SSRMS is 37cm/s when unloaded and ranges from 1.2cm/s to 5.9 cm/s when loaded [20]. High mass/inertia, costly payloads, the great vulnerability of space vehicles to mechanical damage associated with space operations, and concerns to avoid overruns, reduce oscillations, and prevent collisions drove the requirement for such slow speeds. Nonetheless, the slow motion still requires significant levels of operator's vigilance [15].

#### ***2.1.5.1 Ground Support***

The SSRMS operations are heavily supported by the ground operators from ISS Mission Control Center (MCC) in Houston for safety considerations and to free up ISS crew resources for other tasks. System setup, shutdown and pre-position of the SSRMS are usually done by the ground crew. In the event of a failure during operations, the ISS crew hands over the MSS control to the ground operators who will troubleshoot and recover the system. The ISS crew resumes their operations after the ground crew rectified the problems. Such arrangement is feasible because of the proximity between the ISS and the MCC. But in deep space exploration, the latency between MCC and the spacecraft cause current task allocations arrangement between the ground and space crew tasks to be infeasible. Thus, future space crew has to rely on themselves for all system operations.

## 2.2 The MIT-RWSS

The MIT Robotics Workstation Simulator (MIT-RWSS) runs on Vizard, a program developed by WorldViz. Vizard is a virtual reality development software program originally developed at University of California and MIT that is an extension of the Python scripting language. This simulator primary models include the ISS structure and the Space Station Remote Manipulator System (SSRMS) (see para 2.1.1) as shown in Figure 7. As mentioned in para 1.3, this simulator had been used in numerous robotics studies at the Man Vehicle Laboratory (MVL) in MIT. Analogous to the RWS onboard ISS, the MIT-RWSS is a desktop computer that consists of three monitors, a Rotational Hand Controller (RHC) on the right, a Translational Hand Controller (THC) on the left, keyboard and a mouse (see Figure 8). It resembles the NASA's Dynamic Skills Trainers used for training astronauts in space robotics and proximity operations. As in the real SSRMS, a pair of three-axis THC and RHC joysticks are used to move the MIT-RWSS simulated arm (see Figure 9).

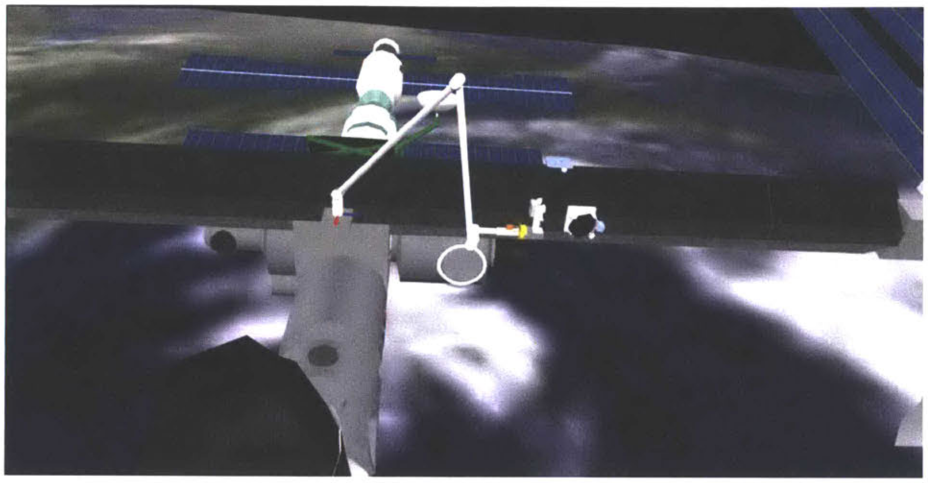


*Figure 7: Screenshot of MIT-RWSS Display*

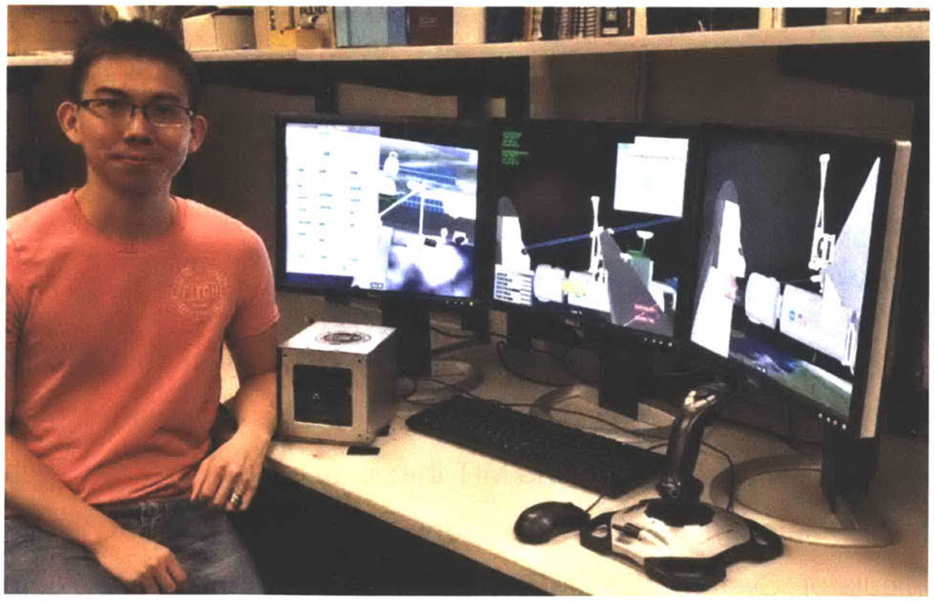
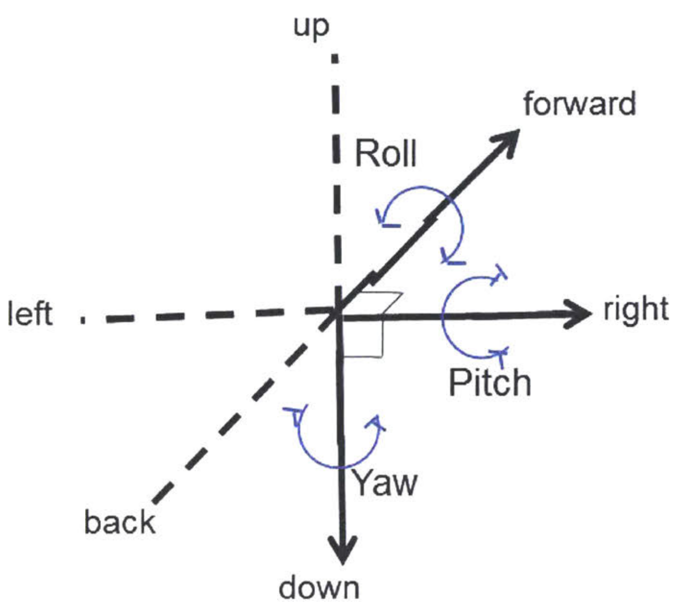


*Figure 8: MIT-RWSS*



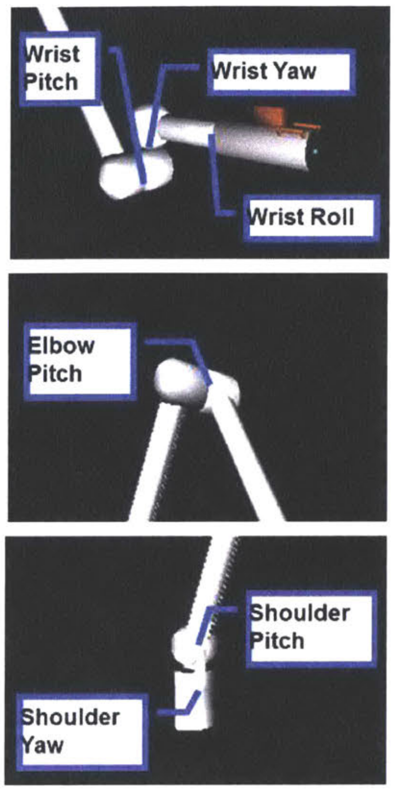


*Figure 9: MIT-RWSS SSRMS Six Degrees of Freedom and Joints*

### 2.2.1 *MIT-RWSS Existing Capability*

Past studies completed using the MIT-RWSS focused primarily on the operations of the SSRMS and how other factors (e.g. mental rotation ability and fatigue) affects the human performance in performing robotic tasks [7]–[9]. Some of the key tasks performed include the following:

1) Fly-to and Grapple – This is to maneuver the SSRMS using the THC and RHC from one location to another until the LEE is sufficiently close to the target to grapple. The task is to use the arm and LEE to pick up of objects, including astronauts and move them to another location.

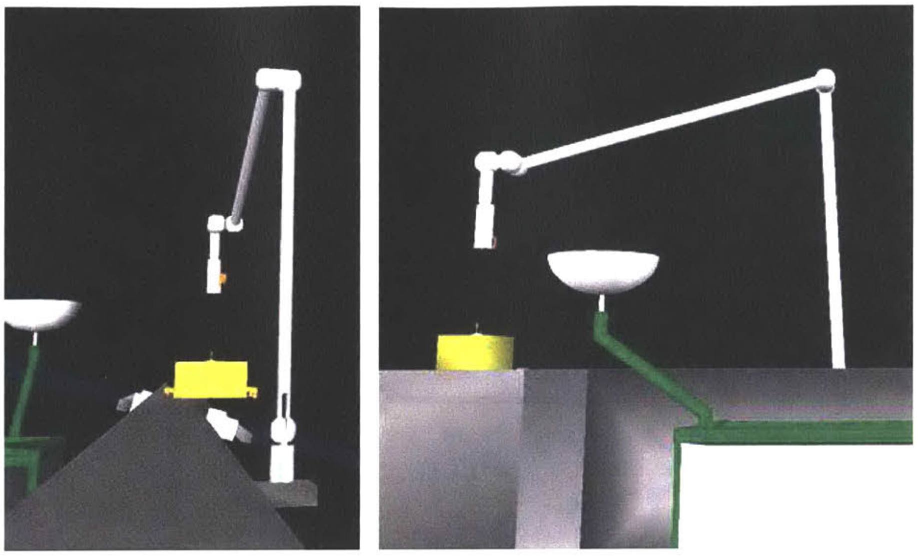


*Figure 10: Fly-to and Capture[21]*

2) Track and Capture – Subjects need to maneuver the SSRMS to capture a free drifting target, the HTV (see para 2.1.1) from a pre-grapple position within 90 seconds. This corresponds to the actual ISS operation where the crew needs to grapple the HTV within 90 seconds or abort the operation due to safety considerations – the HTV could collide with the ISS if it were allowed to continue to drift freely.

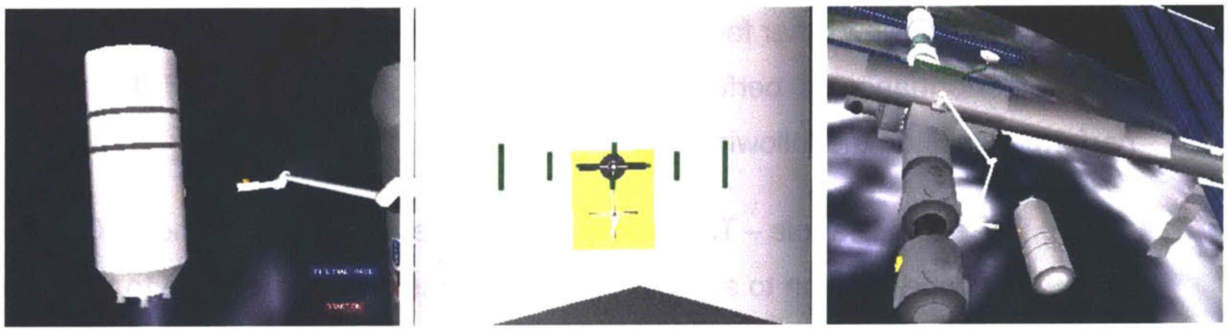


*Figure 11: Track and Capture View [22]*

3) Autosequence – After the operator configures the cameras and control systems, the operator uses this control mode to move the SSRMS automatically to a pre-selected destination on a pre-defined trajectory. The

operator must remain vigilant to ensure the SSRMS does collide with anything due a programming error or other failures.

In addition to the above tasks, the simulator has provisions for (1) Brake application, (2) Command Frame Selection – Internal Mode or External Mode, (3) Joint Angular Rate Limit Selection - Normal or Vernier and (4) selection of different camera views.

These current capabilities were insufficient to satisfy the System Problem Statement One. For purposes of this thesis, features and capabilities were added including simulated system setup and system shutdown, display of electronic procedure, and incorporation of automated electronic procedures. These features will be elaborated in later sections.

## 2.3 Significance of Imbedded Electronic Procedure

Today, manned space exploration takes place onboard the ISS located in low-Earth orbit such that the crew has access to large spacecraft volume and power budgets. Voice and data ISS-mission control communication delays are minimal, so mission controllers can effectively monitor ISS crew activities, and when necessary remotely operate – and occasionally troubleshoot – systems entirely from the ground, Thus, interfaces and procedures are designed to leverage on these advantages. Deep Space Exploration to Mars, however, will face greater constraints on available mass, volume, power, and impose communication delays of minutes, not seconds. This drives the design of the spacecraft towards consolidating the number of electronic displays and interfaces for controlling the spacecraft systems, and require operations, including troubleshooting, to be done much more autonomously.

One critical responsibility of the ground operator in mission control is to ensure that the astronauts execute all the required tasks according to standard procedure. Because of this arrangement, common errors when using procedures such as skipped items, deviation from procedure and misinterpretation [23], [24] will usually be caught by the ground cadre. Future operations that may involve a very small crew might require a single astronaut – who might not be in the best mental condition (e.g. fatigue) – to execute a robotic procedure alone, in the absence of support from Earth. Therefore, alternative precautionary systems and techniques need to be developed to prevent, detect and minimize the impact of procedural errors, and to assist the operator with troubleshooting. One proposed approach is for the robotic operator to utilize electronic procedures with imbedded automation. Nonetheless, there is as yet insufficient empirical evidence to support the level of imbedded automation [25] that should be implemented to reduce the mental workload, improve robotic arm's state awareness, prevent or detect procedural errors, and support diagnostic troubleshooting while not causing crew to become over-reliant on automation – loss of currency of basic skills and system knowledge that lowers workload during normal operations and also supports troubleshooting.

### *2.3.1 Use of Procedures and Checklists in Aviation[26][1]*

The history and rationale for written procedures and checklists in aviation and the human factors consideration have been well reviewed [23], [28]. Procedures are "do-lists" that dictate and specify a progression of sub-tasks and actions to ensure that the task will be carried out in a manner that is efficient, logical, and also error resistant. As aircraft became more complex, the procedures swelled, becoming impractically long. Pilots often skipped items, especially for normal procedures that are performed the same way several times per day, many days per week. Therefore, an alternative technique called "flows" was developed where during each phase of flight, pilots visually scan each control panel in succession and configure all switches from memory. Flows are fast and efficient, but pilots are fallible and flow errors are still occasionally made. To trap these, pilots backstop "flows" with a much shorter paper checklist consisting only "killer" items that would be disastrous if misconfigured. Except during very high workload periods, one pilot normally reads the checklist item aloud, and the other touches and visually verifies then speaks aloud the actual switch setting, e.g. responding "flaps 15" rather than just "set" or "check". Non-normal aviation procedures performed infrequently are different; they are of the step-by-step

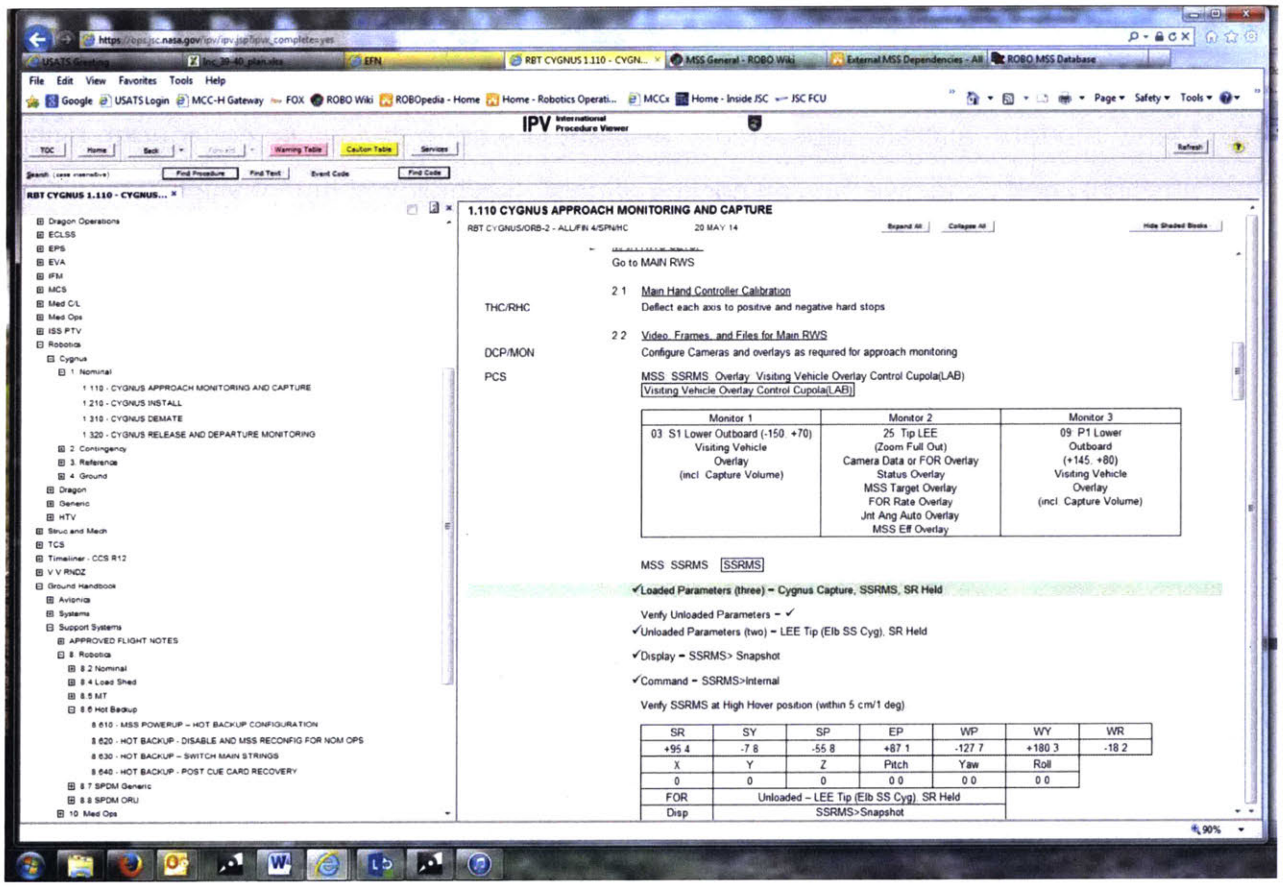
[1] The proposal in respond to NASA NRA [27] by A. Liu and C. Oman for Electronic Procedure was one of the main sources for this section.

"do-list". One pilot reads the item, and the other sets the switch and reads back the position. The most important or time critical items are at the top of the list.

Though flows and paper checklists enhance aviation safety, even using these techniques, crew still makes errors. Crew sometimes skip or deliberately defer checklist items, or become distracted and forget to return to them or forget to run the checklist entirely. Flow and paper checklist errors continue to be identified as causal factors of some aviation accidents. To further reduce checklist errors, Airbus and Boeing developed Electronic Checklists (ECL), and they are found in many newer model airliners today. The ECL lists the system states to check. The ECL can sense some system states, and those that are correctly configured are automatically checked off. Conversely, incomplete tasks are flagged, and deferred checklist steps automatically moved to a subsequent checklist. Any change in system state must be commanded manually by the flight crew using cockpit switches and other controls. Many modern aircraft subsystems are automated but always configured using physical knobs and switches on cockpit panels. It is believed that physically reaching up and configuring a switch or knob helps the crew maintain subsystem state awareness, and reduces the potential risk of crew deskilling and over-relying on ECLs [24]. For this reason, the crew cannot reconfigure aircraft subsystems using soft control buttons built into the ECL itself, even though in some cases it is technically possible, and might seem simpler.

### *2.3.2 ISS Robotics Procedures*

ISS robotics procedures currently are almost always in electronic rather than paper format and have a somewhat different characteristic than those used on airliners. They are often customized, and specific to particular types of robotic operations. Also, many weeks or months may have passed between crew training and procedure execution, and the skill level of the operator may vary from novice to expert. For these and other reasons, NASA robotic operations utilize detailed "do list" procedures, rather than "flows"/"killer item' checklists. (They are more like the non-normal procedures used in aviation.) Typically, procedures are organized into sequential steps and sub-steps, and often named, to make them functionally easier to remember and locate. A common standard for all types of ISS procedures was developed by ISS partners and used to standardize the format of the ISS Operations Data File (ODF). The ODF is the set of all procedures and supporting documentation used onboard the ISS[29]. All procedures are written in English to avoid the complexities of multiple editions. To reduce the paper procedures, electronic ODF procedures reside in laptops called Station Support Computers (SSC). These procedures are accessible to the crew using the software application – International Procedure Viewer (iPV) as shown in Figure 12. A procedure index is available on the left and the procedures are shown on the right in a tabbed window, with the current step highlighted as a placeholder. The iPV procedures can contain hyperlinks to other files, video and images, flow charts, and annotations.



- Dragon Operations
- ECLSS
- EPS
- EVA
- IFM
- MCS
- Med C/L
- Med Ops
- ISS PTV
- Robotics
  - Cygnus
    - 1 Nominal
      - 1 110 - CYGNUS APPROACH MONITORING AND CAPTURE
      - 1 210 - CYGNUS INSTALL
      - 1 310 - CYGNUS DEMATE
      - 1 320 - CYGNUS RELEASE AND DEPARTURE MONITORING
    - 2 Contingency
    - 3 Reference
    - 4 Ground
  - Dragon
  - Generic
  - HTV
- Struc and Mech
- TCS
- Timeliner - CCS R12
- V V RNDZ
- Ground Handbook
  - Avionics
  - Systems
  - Support Systems
    - APPROVED FLIGHT NOTES
    - 8 Robotics
      - 8 2 Nominal
      - 8 4 Load Shed
      - 8 5 MT
      - 8 6 Hot Backup
        - 8 610 - MSS POWERUP – HOT BACKUP CONFIGURATION
        - 8 620 - HOT BACKUP - DISABLE AND MSS RECONFIG FOR NOM OPS
        - 8 630 - HOT BACKUP – SWITCH MAIN STRINGS
        - 8 640 - HOT BACKUP - POST CUE CARD RECOVERY
      - 8 7 SPDM Generic
      - 8 8 SPDM ORU
    - 10 Med Ops

**1.110 CYGNUS APPROACH MONITORING AND CAPTURE**

RBT CYGNUS/ORB-2 - ALL/FIN 4/SPN/HC 20 MAY 14

Go to MAIN RWS

2 1 Main Hand Controller Calibration

THC/RHC Deflect each axis to positive and negative hard stops

2 2 Video, Frames, and Files for Main RWS

DCP/MON Configure Cameras and overlays as required for approach monitoring

PCS MSS SSRMS Overlay Visiting Vehicle Overlay Control Cupola(LAB)

Visiting Vehicle Overlay Control Cupola(LAB)

| Monitor 1 | Monitor 2 | Monitor 3 |
|---|---|---|
| 03 S1 Lower Outboard (-150, +70) Visiting Vehicle Overlay (incl Capture Volume) | 25 Tip LEE (Zoom Full Out) Camera Data or FOR Overlay Status Overlay MSS Target Overlay FOR Rate Overlay Jnt Ang Auto Overlay MSS Eff Overlay | 09 P1 Lower Outboard (+145, +80) Visiting Vehicle Overlay (incl Capture Volume) |

MSS SSRMS SSRMS

✓Loaded Parameters (three) – Cygnus Capture, SSRMS, SR Held

Verify Unloaded Parameters – ✓

✓Unloaded Parameters (two) – LEE Tip (Elb SS Cyg), SR Held

✓Display – SSRMS> Snapshot

✓Command – SSRMS>Internal

Verify SSRMS at High Hover position (within 5 cm/1 deg)

| SR | SY | SP | EP | WP | WY | WR |
|---|---|---|---|---|---|---|
| +95 4 | -7 8 | -55 8 | +87 1 | -127 7 | +180 3 | -18 2 |
| X | Y | Z | Pitch | Yaw | Roll | |
| 0 | 0 | 0 | 0 0 | 0 0 | 0 0 | |
| FOR | Unloaded – LEE Tip (Elb SS Cyg), SR Held | | | | | |
| Disp | SSRMS>Snapshot | | | | | |

Figure 12: Screenshot of the ISS iPV

For reasons related to ISS data and control security, the SSCs operate on a network separate from the ISS robotics control system. In contrast to aviation where some imbedded ECLs have access to the systems information via the aircraft data bus, the iPV cannot communicate with the ISS SSRMS, RWS, camera and intercom controls or with the DOUG running on a second PCS (refer to section 2.1.4). Therefore, it is unable to execute any tasks automatically, check the states of the system or validate that the crew performs the step as required.

### *2.3.3 Prior Research on Electronic Procedures for Spacecraft Systems*

Billman et al. developed a prototype procedure viewer that could communicate with ISS subsystems, sense and control system states, and prompt the next action or actions, with status displays and control buttons embedded in the procedure steps for the carbon dioxide removal system (CDRS) [30]. Steps completed were highlighted in green. The focus of their study was on comparing manual execution between legacy (current) interfaces – electronic procedure is in one window and the procedure execution in another – and an integrated interface - electronic procedure in the same window as the controls and displays required for procedure execution. Integrated interface was shown to improve the operations accuracy and efficiency. Unlike legacy interface, no command errors were observed and completion time was reduced by approximately half.

Billman et al. used the Procedure Representation Language (PRL), an eXtensible Markup Language (XML) schema developed with NASA to build the electronic procedure. The PRL was developed for the NASA's Constellation program to support automation of procedure execution [31]. PRL has the capability to evaluate whether the preconditions are True before execution of the procedure, and have knowledge of the post-conditions. There are many types of instructions in PRL (see below) and each is controlled by Automation Data that is used to control execution in PRL.

1. Command Instruction – Issue a computer command (with parameters).

2. Verify Instruction – Compares a specific item to a target value.

3. Wait Instruction – Halts execution either for a specified period of time or until a Boolean expression becomes True.
4. Call Procedure Instruction – Calls another procedure.
5. Call Function Instruction – Calls a user-defined function running on the underlying system.
6. Manual Instruction – Commands that need to be performed by a human.
7. Input Instruction – Acquires data from a user (or other source).

In a follow-on study, they investigated the effects of different levels of automation – manual, partially automated and fully automated procedure – using the integrated interface [32]. The automation of a single or multiple procedural steps was done by selecting a breakpoint on the procedure to set where the automation should stop, in a manner analogous to a computer code debugger. Usability metrics focused on human-automation team performance. The evaluation, however, did not directly address the impact of procedure automation on operator workload or potentially decreased awareness of subsystems states when the automation rather than the human operator monitored sensors and activated effectors. The number of automation selection errors made by the subjects suggested that the implementation may need to be reconsidered.

## 2.4 Requirements of Experiment

### *2.4.1 NASA's Needs*

NASA is the sponsor for this project and its needs are as follows[27]:

1) Future spacecraft electronic procedures that are integral and imbedded to the spacecraft system displays must support crew in time- and safety-critical tasks, such as identifying and resolving spacecraft failures. Faults have to be communicated to the crew through health system displays and executed through

electronic procedure. The information display must enable the crew to diagnose the issues using the electronic procedure.

2) NASA needs empirical data to support HCI guidelines for new spacecraft interfaces and the optimum way to integrate electronic procedures and critical system information. The empirical data should consider automation options, interruption resilience, system display integration, and simultaneous use of procedures by multiple crewmembers.

3) The key focus is automation as it has great influence on the system design. Therefore, NASA needs to understand the extent to which procedure step execution should be automated and under which situations (e.g. nominal vs emergency, routine vs. novel).

4) The experiment should also consider the use of multiple electronic procedures to execute through a spacecraft's fault detection, isolation, and recovery tasks.

5) Human Factors concerns, in particularly overreliance and loss of situation awareness (SA) must be considered in the research. It remains unclear which are the best mitigations to inhibit SA loss and high workload, especially under time-critical events with automation. Therefore, empirical data are required to support the measures.

6) Human performance measures should include both objective and subjective metrics.

### *2.4.2 MIT-RWSS System Requirements*

To conduct the experiments that could meet the needs of NASA, the MIT-RWSS capabilities and features needed to be expanded. At the point when the MIT-RWSS requirements were defined, the experiment methodology had not yet been determined. Nevertheless, features and capabilities such as simulate system setup and system shutdown, display electronic procedure and automate procedure execution in single and multi-steps were envisaged. Therefore, the simulator requirements were developed to facilitate broad ranges of possible experiments with different independent variables such that not all features will be required in the initial experiment. The requirements envisaged are delineated below:

1) The Electronic Procedure shall include system setup, execution of EVA task, system shutdown and failure recovery.

2) Electronic Procedure (EP) shall be displayed on one of the three monitors. EP must not obscure the display to the extent that the operations cannot be conducted safety. Therefore, as a general guide, the EP should cover less than 20% of the display in one monitor.

3) There shall be a display to show the state of the system and it can be opened and closed at the preference of the crew.

4) The change of system state can be executed through electronic button/switch or using the keyboard, with the exception of critical functions such as ‘Safing’ and ‘Brake’ that will require hard switches.

5) There shall be a display to show the health status of the system and flag out any system failure.

6) The simulator shall be capable of performing various types of automation such as (a) verification of system state, (b) change of system state and (c) marking of a completed instruction.

7) The simulator shall allow the user to perform single-step, multi-steps or entire-procedure automation. Entire-procedure automation refers to executing the entire setup or shut down automatically.

8) The simulator shall have the capability to execute failure recovery procedure in the event of a system failure or error automatically if selected by user. It is expected that some steps in the failure recovery procedure require human intervention and those could be excluded from the automation.

9) The simulator shall have the feature that allows the users to select the procedure they wish to perform for both nominal and off-nominal situations.

10) The simulator shall have features to allow the experimenter to pause the simulation and blank out all screens.

11) A prompt is required for any changes in system state that need to be done manually by the operator.

12) The simulator shall have feedback to inform the user that the system is processing a change of state. The feedback should have a different level of information that is selectable by the experimenter.

13) Data including errors in executing a procedure, time a button is selected and time between two events (i.e. complete change in a system state to the trigger of a change in state) shall be recorded.

14) The EP and system state should be integrated such that check of state is based on the procedure instruction.

15) The automation should prevent the execution of a procedure or wrong step if it detects the system is in an inappropriate state. The human operator, however, remains responsible for detecting other kinds of errors that cannot be detected by the automation.

## 2.5 Development Methodology

In order to evaluate the similarities and differences between Task Analysis and OPM to address system problem statement Two in Section 1.2, both methodologies were used separately to develop the procedures and the architecture of the features to be coded in the MIT-RWSS. In the next two chapters, the author summarized how Task Analysis and OPM were used to address system problem statement One. In subsequent chapters, a comparison between the two techniques is described.

# 3 TASK ANALYSIS

## 3.1 Introduction

Task Analysis is a technique that involves collecting data, classifying the data and analyzing the data to understand human performing in work situation [33]. The origin of task analysis can be traced back to the "Scientific Movement" movement more than a century ago that focused on analysis of work with the intention to improve productivity [34], [35]. Since then, task analysis has gained traction and evolved. It has been extensively reviewed in literature and was said to cover over 100 variations of task analysis technique [36]. Kirwan et al. defined Task Analysis as the study of what an operator (or team of operators) is required to do in terms of actions and/or cognitive processes to achieve a system goal. Therefore, the use of Task Analysis would lead to more efficient and effective integration of the human element into system design and operations in three principal areas – safety, productivity and availability [37]. Task Analysis is often promoted as a valuable technique in the field of Human-Computer Interaction (HCI) [38] and is potentially useful at each stage of the system development [36].

Given the diversity of task analysis techniques, it is appropriate to select one that is well matched to the intent of this thesis. Crystal et al. studied some of the popular task analysis techniques for HCI and summarized their efficiency, effectiveness and empirical evidence as shown Figure 13 [38].

One of the objectives of this thesis was to design and develop a simulator platform to perform human-in-the-loop experiments. These experiments were meant to assess the implications of using a space telerobotic system with imbedded electronic procedures and built-in automation; both nominal and off-nominal scenarios had to be considered. In this process, it was necessary to define the operational procedures required, displays, controls and information needed to support the operations using a systematic approach that reduce the system complexity. Hierarchical Task Analysis (HTA) was useful for defining the goals and tasks of the new system design.

|  | TECHNIQUE | EFFICIENCY | EFFECTIVENESS | EVIDENCE |
|---|---|---|---|---|
| Technical | HTA | ▪ Decomposes complex tasks into subtasks<br>▪ Complex activities demand extensive hierarchy construction/charting | ▪ Improves problem diagnosis and useful for concurrent operations<br>▪ Does not account for system dynamics | MacLean et al., 1991<br>Annet and Stanton, 2000<br>Hollan et al., 2000<br>Shepherd 2001 |
|  | GOMS | ▪ Requires detailed analysis of keystroke level interaction | ▪ Improves productivity<br>▪ Not applicable to broader problems<br>▪ Ignores contextual factors | Card et al., 1983<br>Preece et al., 1994<br>John and Kieras, 1996 |
| Conceptual | CTA | ▪ Defines a coherent knowledge representation for the domain being studied<br>▪ Requires deep engagement with a particular knowledge domain | ▪ Increases the understanding of cognitive aspects of the task<br>▪ Captures task expertise<br>▪ Fails to fully incorporate learning, contextual and historical factors | Barnard and May, 2000<br>Chipman et al., 2000<br>Dubois and Shalin, 2000 |
| Work-Process | Activity Theory | ▪ Analyzes the activity, not the task, implying a potentially great increase in scope and complexity<br>▪ Requires near-ethnographic knowledge of culture | ▪ Accounts for learning effects<br>▪ Extends scope of technology<br>▪ Requires a high level of abstraction<br>▪ No disciplined set of methods<br>▪ Difficult to apply systematically | Kuutti, 1996<br>Hollan et al., 2000 |

*Figure 13: Efficiency, effectiveness and empirical evidence in task analysis research[38]*

## 3.2 Hierarchical Task Analysis (HTA)

HTA was introduced by Annett and Duncan in 1967 to evaluate an organization's training needs [39] and is probably be the most frequently used methodology by ergonomists in the UK [33], and probably worldwide. The key feature of HTA is that tasks – those things that the person is seeking to achieve – are essentially defined by goals rather than actions, and the complex tasks may be analyzed by decomposing a hierarchy of goals and sub-goals [33]. Decomposing goals and sub-goals into multiple level of hierarchy produces an extensive description of the activities required to fulfil the goals. These goals and sub-goals could be presented in a graphical format using hierarchical diagram as shown in Figure 14 or as a hierarchical list as presented in Figure 15. One of the problems of HTA was when to stop the analysis which was also discussed by Stanton [4]. There are several approaches. Piso suggested to stop the analysis when the task or operation was clearly described for both the operator and analyst [40]. Using Piso's suggested principle, the lowest level of the HTA would resemble a "do list".

In the graphical format, the rectangle box represents the goal, sub-goals or activities and the rounded rectangle represents the plan, which will be elaborated in the later section. Unfortunately, there are no formal standards governing HTA, causing the hierarchical diagram formats that different practitioners draw look different, although they use the same principles. It can be noted that the overall goal “Deal with chemical incident” is decomposed into sub-goals “receive notice from public about incident”, “gather information about incident”, “made decision about nature of incident” and “deal with chemical incident. These sub-goals are further decomposed into smaller sub-goals or activities. A line at the bottom of the rectangular box represents that further decomposition is not required, usually at a point where sub-goals dealt with the exchange of information (e.g. receiving, analyzing and sending information from one agent to another) [4]. The double forward slash “//” is used in the hierarchical list format (e.g. Figure 15).

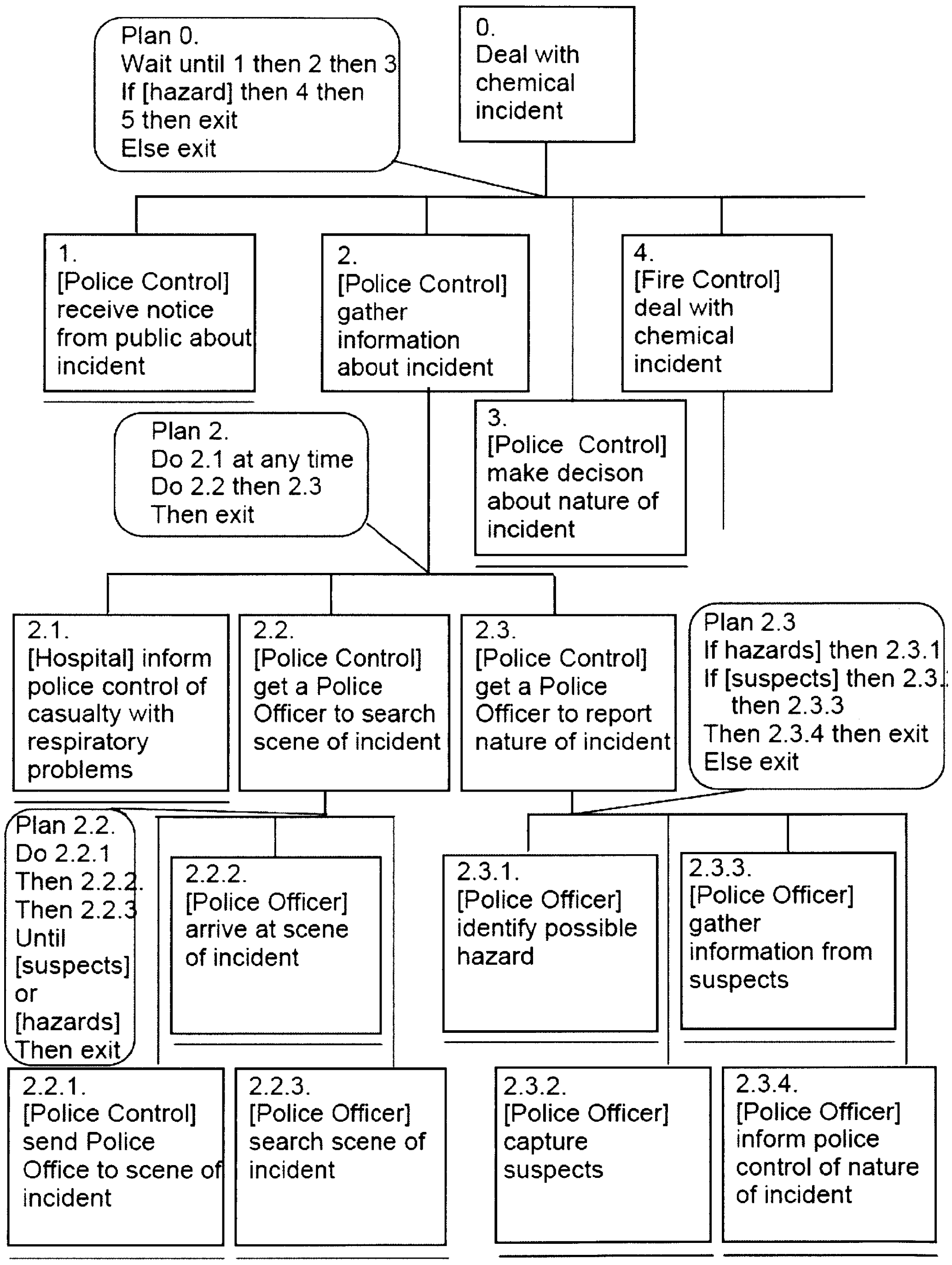

Plan 0.
Wait until 1 then 2 then 3
If [hazard] then 4 then
5 then exit
Else exit
0.
Deal with chemical incident
1.
[Police Control] receive notice from public about incident
2.
[Police Control] gather information about incident
4.
[Fire Control] deal with chemical incident
3.
[Police Control] make decison about nature of incident
Plan 2.
Do 2.1 at any time
Do 2.2 then 2.3
Then exit
2.1.
[Hospital] inform police control of casualty with respiratory problems
2.2.
[Police Control] get a Police Officer to search scene of incident
2.3.
[Police Control] get a Police Officer to report nature of incident
Plan 2.3
If hazards] then 2.3.1
If [suspects] then 2.3.2
then 2.3.3
Then 2.3.4 then exit
Else exit
Plan 2.2.
Do 2.2.1
Then 2.2.2.
Then 2.2.3
Until
[suspects]
or
[hazards]
Then exit
2.2.2.
[Police Officer] arrive at scene of incident
2.3.1.
[Police Officer] identify possible hazard
2.3.3.
[Police Officer] gather information from suspects
2.2.1.
[Police Control] send Police Office to scene of incident
2.2.3.
[Police Officer] search scene of incident
2.3.2.
[Police Officer] capture suspects
2.3.4.
[Police Officer] inform police control of nature of incident

Figure six. Part of the hierarchical diagram for the goal of "Deal with chemical incident"

*Figure 14: Example of a Hierarchical Diagram in HTA from Stanton 2006*

```
0. Deal with chemical incident
Plan 0: Wait until 1 then do 2 then 3 - If [hazard] then 4 then 5 then exit -
Else exit
            1. [Police control] receive notice from public about incident //
            2. [Police Control] gather information about incident
            Plan 2: Do 2.1 at any time if appropriate
            Do 2.2 then 2.3
            Then exit
                        2.1. [Hospital] inform police control of casualty with
                        respiratory problems//
                        2.2. [Police Control] get a Police Officer to search
                        scene of incident
                        Plan 2.2: Do 2.1.1 then 2.2.2 then 2.2.3
                        Until [suspects] or [hazards] then exit
                                    2.2.1. [Police Control]
                                    send Police Officer to
                                    scene of incident//
                                    2.2.2. [Police Officer]
                                    arrive at scene of
                                    incident//
                                    2.2.3. [Police Officer]
                                    search scene of incident//

                        2.3. [Police Control] get Police Officer to report
                        nature of incident
                        Plan 2.3: If [suspects] then 2.3.1
                        If[suspects] then 2.3.2. then 2.3.3
                        Then 2.3.4. then exit
                        Else exit
```

*Figure 15: Example of Hierarchical List from Stanton 2006*

Despite being in use for decades, HTA does not have a formal uniform standard adopted by all practitioners. Therefore, practitioners tend to customize HTAs to fit their needs. According to Annett et. al [41], HTA as a framework for task analysis, follows three main principles.

1) "At the highest level define the task as consisting of an operation and the operation is described in terms of its goal. The goal implies the objective of the system in some real terms of production units, quality or other criteria.

2) The operation is then broken down into sub-operations each defined by a sub-goal again measured in real terms by its contribution to overall system output or goal, and therefore measurable in terms of performance standards and criteria.

3) The important relationship between operations and sub-operations is one of inclusion; it is a hierarchical relationship. Although tasks are often proceduralised, that is the sub-goals have to be attained in a sequence, this is by no means always the case." [41]

Stanton[6] suggested the following steps in conducting HTA:

1) Define Task(s) under Analysis

2) Data Collection Process – Data regarding the task steps involved, the technology used, the interaction between man and machine and team members, decision-making and task constraints should be collected.

3) Determine the Overall Goal of the Task – the top of the hierarchy (e.g. Deal with Chemical Incident in Figure 14).

4) Determine Task Sub-goals – Breakdown the top goal into meaningful sub-goals.

5) Sub-goal Decomposition – Breakdown the sub-goals identified in step four above into further sub-goals and operations/activities.

6) Analysis of Plans – Determine how the goal are achieved (e.g. Plan 2. Do 2.1 at any time; Do 2.2 then 2.3; Then exit in Figure 14). The plan could be in one of the following categories:

    a. Linear – Do 1, then 2, then 3

    b. Non-linear – Do 1, 2 and 3 in any order

c. Simultaneous – Do 1, then 2 and 3 at the same time

d. Branching – Do 1; if X present, then do 2 then 3, but if X is not present, then EXIT

e. Cyclical – Do 1, then 2, then 3 and repeat until X

f. Selection – Do 1, then 2 or 3 [6].

The procedure in conducting HTA from step three to six can be summarized by the flowchart in Figure 16 [4].

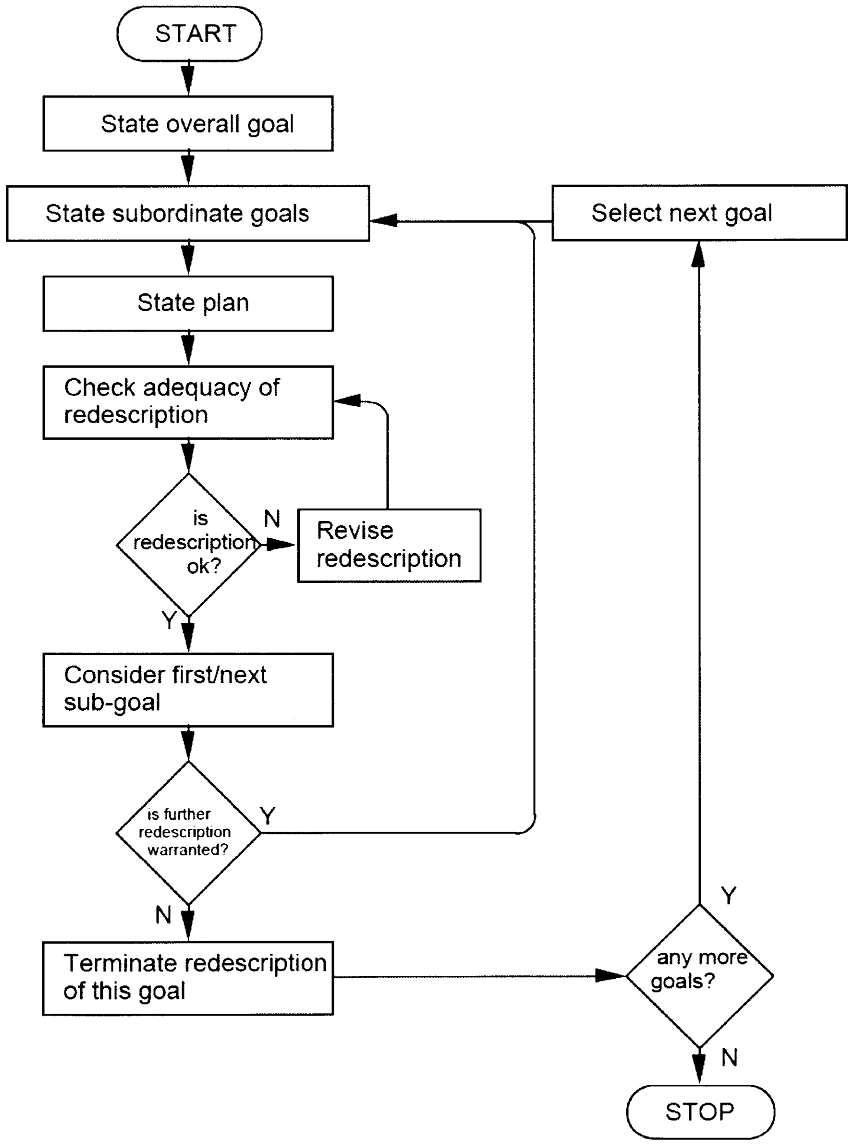
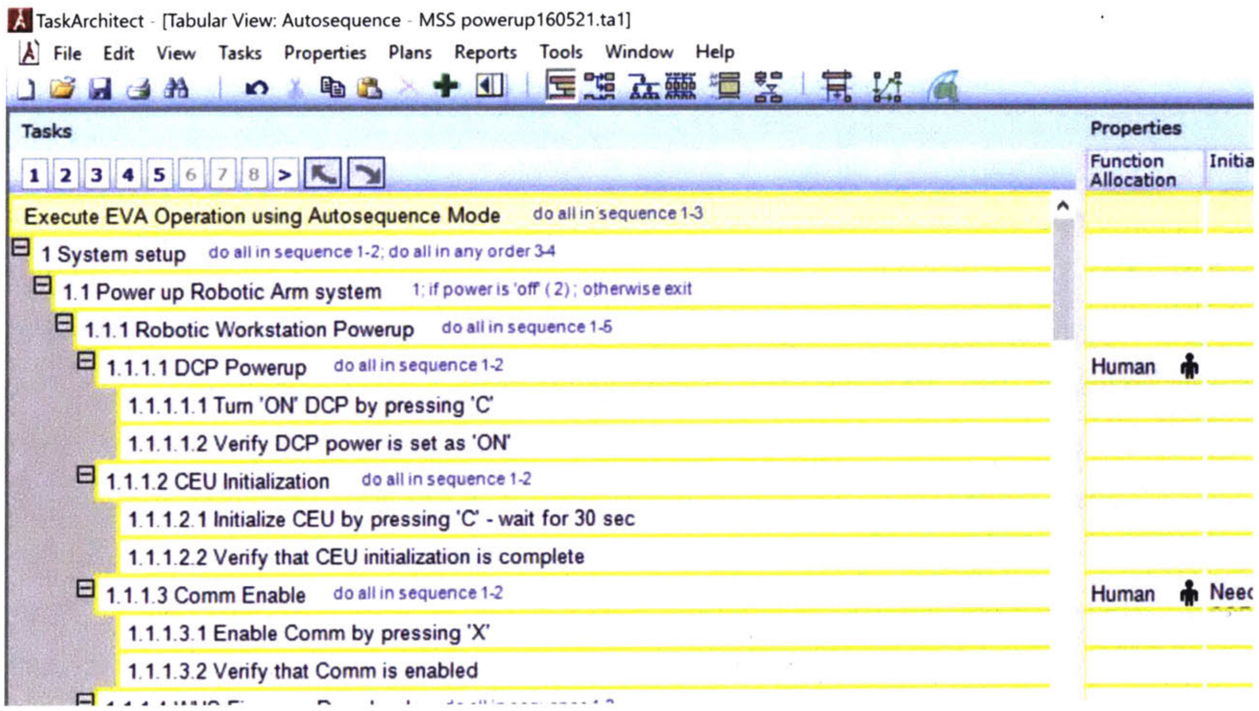


*Figure 16:Flowchart for Performing HTA*

### 3.2.1 *HTA Development Tools*

As reviewed by Salmon et al (2009) a variety of different commercial software tools have been developed for HTA, although conventional word processors and spreadsheets are often employed. For purposes of this thesis, the HTA of Robotic-EVA operations tasks as performed on the MIT-RWSS simulation was done using the TaskArchitect software developed by Kern Technology Group, Virginia Beach, VA. The environment facilitates the breakdown of goal and sub-goals hierarchically and displays the task analysis in different hierarchical diagrams or list format. See Figure 17 for example. **It also allows the analyst to include 'Properties' on the right so that additional information could be added to document the analysis.**


TaskArchitect - [Tabular View: Autosequence - MSS powerup160521.ta1]
File Edit View Tasks Properties Plans Reports Tools Window Help
Tasks
1 2 3 4 5 6 7 8 >
Properties
Function Allocation
Initia

| Tasks | Function Allocation | Initia |
|---|---|---|
| Execute EVA Operation using Autosequence Mode — do all in sequence 1-3 | | |
| 1 System setup — do all in sequence 1-2; do all in any order 3-4 | | |
| 1.1 Power up Robotic Arm system — 1; if power is 'off' (2); otherwise exit | | |
| 1.1.1 Robotic Workstation Powerup — do all in sequence 1-5 | | |
| 1.1.1.1 DCP Powerup — do all in sequence 1-2 | Human | |
| 1.1.1.1.1 Turn 'ON' DCP by pressing 'C' | | |
| 1.1.1.1.2 Verify DCP power is set as 'ON' | | |
| 1.1.1.2 CEU Initialization — do all in sequence 1-2 | | |
| 1.1.1.2.1 Initialize CEU by pressing 'C' - wait for 30 sec | | |
| 1.1.1.2.2 Verify that CEU initialization is complete | | |
| 1.1.1.3 Comm Enable — do all in sequence 1-2 | Human | Neec |
| 1.1.1.3.1 Enable Comm by pressing 'X' | | |
| 1.1.1.3.2 Verify that Comm is enabled | | |

*Figure 17:Screenshot of TaskArchitect*

## 3.3 Development of HTA for Robotics Simulation

The first step of HTA as suggested by Stanton is to define the task under analysis. For present purposes, a robotic operation involving moving an EVA astronaut located on the end of the arm using Autosequence control was analyzed. The procedure also incorporated powering up and shutting down the system. Also, the analysis considered procedures for troubleshooting and recovering from plausible system failures as they might occur in a real system. The output of the HTA should suggest the systems required and sequence of activities. This information could be used to support the development of the simulation i.e. the subsystems to be modeled in the simulator.

For Stanton's second step, data collection, a set of ISS robotics arm normal and troubleshooting procedures provided by a NASA expert was reviewed to develop the HTA. Replicating the full set of ISS procedures was inappropriate given that they include systems like the MBS and SPDM (see section 2.1) that are not modeled in the MIT-RWSS and new interfaces will be required for the automated electronic procedures capability. The primary objective was to understand the types of normal and troubleshooting tasks, goals and constraints the actual ISS crew used to perform in space.

### 3.3.1 Define Overall Goal and Sub-goals

For Stanton's step three, the EVA autosequence HTA was built, starting by defining the overall goal – execute normal EVA autosequence operations using the SSRMS. The purpose was to describe a scenario where the robotic operator moves an astronaut secured to the end of the SSRMS LEE to various locations around the ISS exterior. The larger goal of EVA operations was typically to replace, inspect or repair the systems onboard the ISS that were accessible only from the exterior. However, consistent with step three, the top level goal of the HTA itself was simply "execute EVA autosequence operations using the SSRMS". This is because, from the perspective of the robotic operator, that was the top level goal of the specific activities that he/she needed to perform. (The replace, inspect or repair of systems were the responsibility of the astronaut performing the EVA.)

With the completion of Stanton's step three, define the overall goal for the tasks, the next step is to decompose the goal into sub-goals. The sub-goals were (1) Setup up the robotic system, (2) Perform EVA operations and (3) Shutdown the system. These linear (performed in sequence) sub-goals described the end-to-end operation of the MIT-RWSS robotic system. Therefore, controls and displays for setup need to be included in the MIT-RWSS simulator. After the setup, the robotic operator moves the

EVA astronaut from one location to another using the Autosequence arm movement mode. The MIT-RWSS already had an existing capability to change the LEE position automatically instead of manually. Autosequence was selected for the experiment since it did not involve manual control, simplifying human error analysis and future systems have been postulated to depend heavily on some form of automation. The HTA analysis for autosequence involved the modelling of additional setup procedures not required for manual control. In most situations, the robotic operator would need to move the robotic arms to multiple locations. This can be represented as different sub-goals such as perform EVA operation 1, perform EVA operation 2, etc. In this HTA, two EVA operations were shown. The EVA operation task two and beyond, when broken down, follows the same activities/procedural steps. The differences between successive EVA operation tasks were minor e.g camera number and joint angles that are not important in the HTA description. The uniqueness about EVA operation task 1 is the camera setting for it is done as part of setting up because the crew should have an appropriate view of SSRMS before unsafing (refer section to 3.3.2). The tasks did not end after the EVA operation because the robotic system needs to be turned off/shutdown. From Stanton's step three and four, the first two-level of the HTA were created as presented in Figure 18. The plan aspect in Stanton's step six was as discussed in section 3.3.3

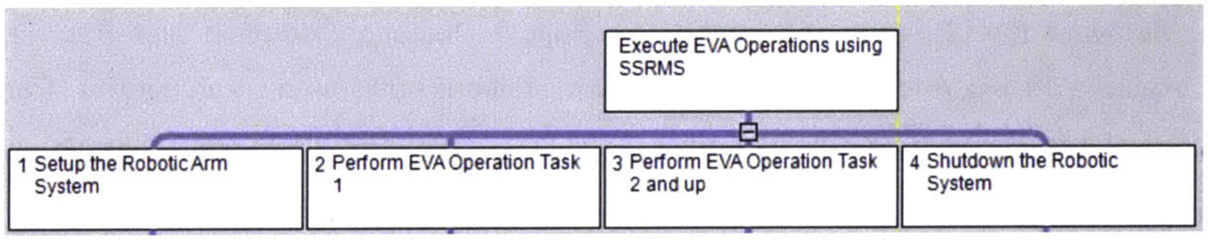

Execute EVA Operations using SSRMS
1 Setup the Robotic Arm System
2 Perform EVA Operation Task 1
3 Perform EVA Operation Task 2 and up
4 Shutdown the Robotic System

*Figure 18: Goal and Sub-goals*

### 3.3.2 Sub-goals and Operations of Setup the Robotic System

Stanton's step five is to decompose the sub-goals into further sub-goals and operations. Setup the Robotic System was decomposed further into sub-goals: (1) Power up Robotic Workstation, (2) Power up the SSRMS, (3) Power up the Video Components, (4) Setup display configuration and (5) Remove Safing. As part of the robotic system setup, the subsystems (1) to (3) are required to be powered up. Then, the robotic operator needs to configure the displays to the respective cameras to provide him/her with visual information of the SSRMS position. Once that is done and the operator is satisfied, before releasing the brakes on the arm joints, the operator needs to turn safing off. Safing is a safety mode of the robotic system that prevents inadvertent brake release on all the joints. Figure 19 illustrates the decomposition of Setup the Robotic System.

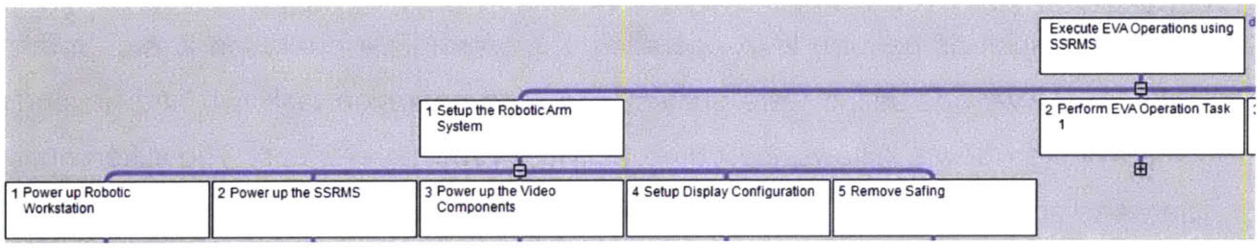

Execute EVA Operations using SSRMS
1 Setup the Robotic Arm System
2 Perform EVA Operation Task 1
1 Power up Robotic Workstation
2 Power up the SSRMS
3 Power up the Video Components
4 Setup Display Configuration
5 Remove Safing

*Figure 19: Sub-goals of Setup the Robotic System*

The Power-up Robotic Workstation can be further decomposed to its sub-goals as reflected in Figure 20, which is still a hierarchical diagram but viewed from left to right instead of top to bottom. Both views display the same information. From Figure 20, it shows that Power up Robotic Workstation is decomposed to (1) Power up the Display control Panel, (2) Initialize the Control Electronics Unit (CEU), (3) Enable the Communications (Comm)[2] with ISS System, (4) Download the Workstation Host Software (WHS) [3] and (5) Enable the Failure Detection Isolation and Recovery system (FDIR). A total of four level of decompositions have been created so far. Each level refines the sub-goals into more detail sub-goals. For the present purpose, the lowest level of the HTA needs to delineate the activities to be performed to achieve the sub-goals as shown in Figure 21. Take for example Power up the Display Control Panel (DCP). It requires the robotic operator to perform two tasks:

[2] This refers to the data communication link between the RWS and the ISS systems.
[3] This is to simulate the download of the operating software for the CEU. WHS is downloaded to the RWS laptop every time the RWS is powered up on the ISS. It is the interface between the other software that runs on the RWS.

(1) Turn the DCP on.

(2) Verify the DCP is on.

This two tasks form part of the procedure for '**Setup the Robotic Arm System**'. The completion of all level five activities would fulfil the level four sub-goals. Therefore, the lowest levels in the HTA model will define the procedural steps required to be executed by the operator. Detailed step by step procedure for telerobotic operations can be derived from this low level HTA. However, there is no way to validate the logical correctness of the procedure derived this way using HTA. The analysis also does not specify the components, information and pre-requisite system states needed to begin each procedural step.

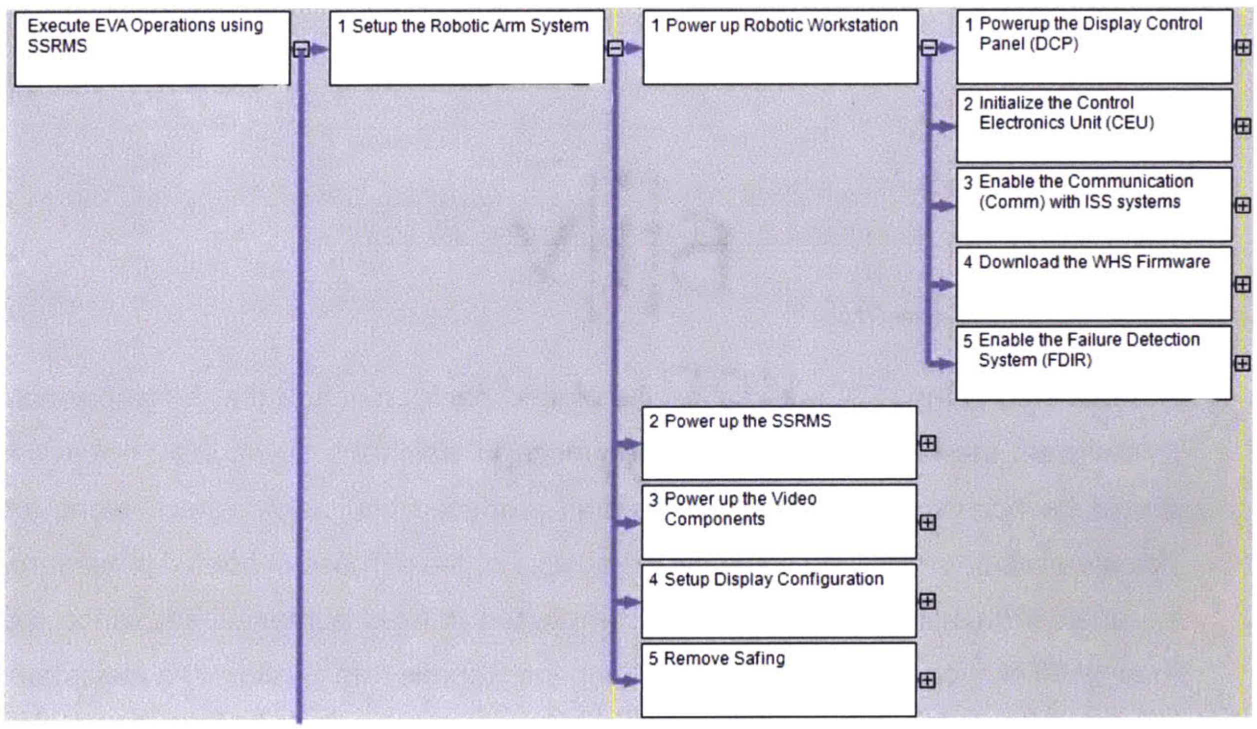
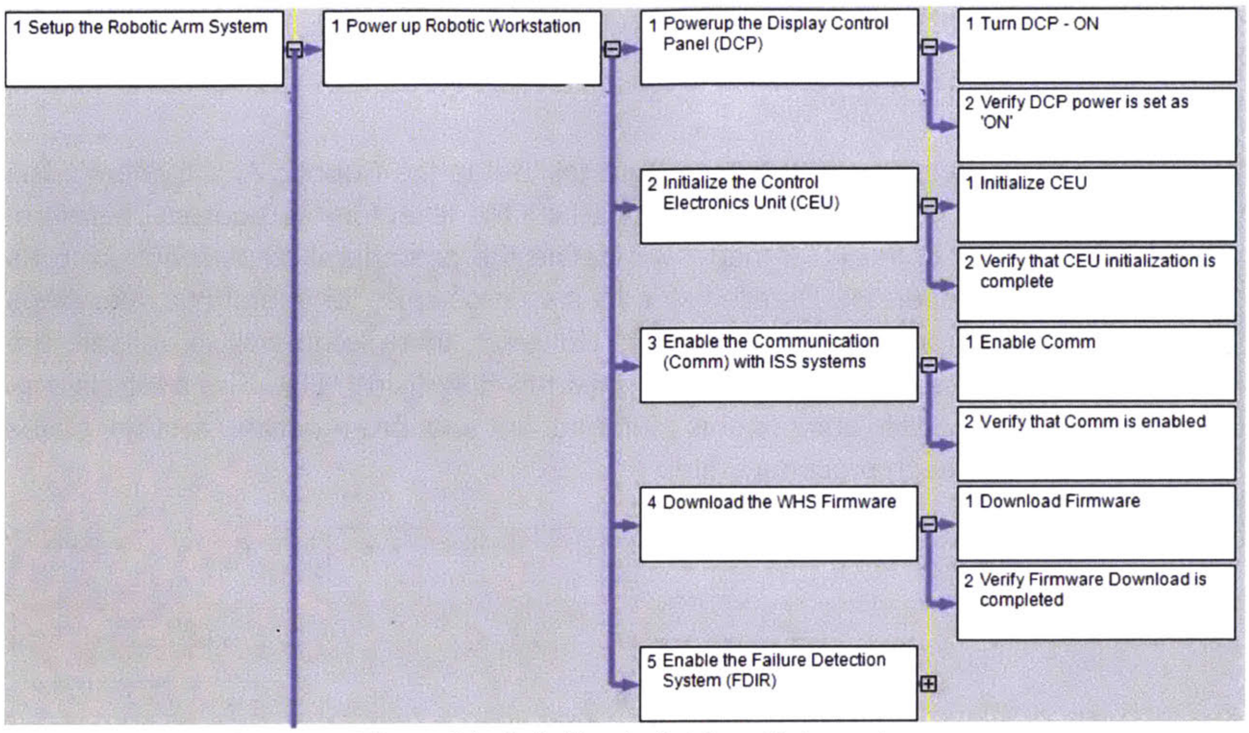

Execute EVA Operations using SSRMS
1 Setup the Robotic Arm System
1 Power up Robotic Workstation
1 Powerup the Display Control Panel (DCP)
2 Initialize the Control Electronics Unit (CEU)
3 Enable the Communication (Comm) with ISS systems
4 Download the WHS Firmware
5 Enable the Failure Detection System (FDIR)
2 Power up the SSRMS
3 Power up the Video Components
4 Setup Display Configuration
5 Remove Safing

*Figure 20: Further Decomposition of Sub-goals for Power Up Robotic System*



1 Setup the Robotic Arm System
1 Power up Robotic Workstation
1 Powerup the Display Control Panel (DCP)
1 Turn DCP - ON
2 Verify DCP power is set as 'ON'
2 Initialize the Control Electronics Unit (CEU)
1 Initialize CEU
2 Verify that CEU initialization is complete
3 Enable the Communication (Comm) with ISS systems
1 Enable Comm
2 Verify that Comm is enabled
4 Download the WHS Firmware
1 Download Firmware
2 Verify Firmware Download is completed
5 Enable the Failure Detection System (FDIR)

*Figure 21: Activities to Achieve Sub-goals*

### 3.3.3 Define the Plan

The last step in the HTA is to define the plan at each branch of the decomposition. This will set the requirement on the sequence in which the sub-goals or activities should be accomplished. In the diagrams produced by TaskArchitect, plans are indicated in blue text footnotes below the goals. For the purpose of normal procedures, the goals in this HTA are primarily accomplished in fixed numerical sequence (i.e. linearly). Take for example sub-goal, Setup the Robotic Arm System, the plan states do all in sequence 1 to 5 as reflected in Figure 22. That means, the robotic operator needs to 'Power up Robotic Workstation', then 'Power up the SSRMS', then 'Power up the Video Component', then 'Setup Display Configuration' before 'Remove Safing'.

Note that the plan for sub-goal, Perform EVA Operation Task 2 and up is cyclical. This indicates that the robotic operator would perform sequence 1 to 8 and repeat the sequence until the EVA operations are completed – at the level of detail in HTA, all the tasks appear identical. Thus, if the robotic operator needs to move the astronaut on EVA to three different locations, he/she would perform sequence 1 to 8 twice before moving on to the next sub-goal as shown in Figure 18, Shutdown the Robotic System.

The complete HTA for Execute EVA Operations using SSRMS is in **Appendix A**.

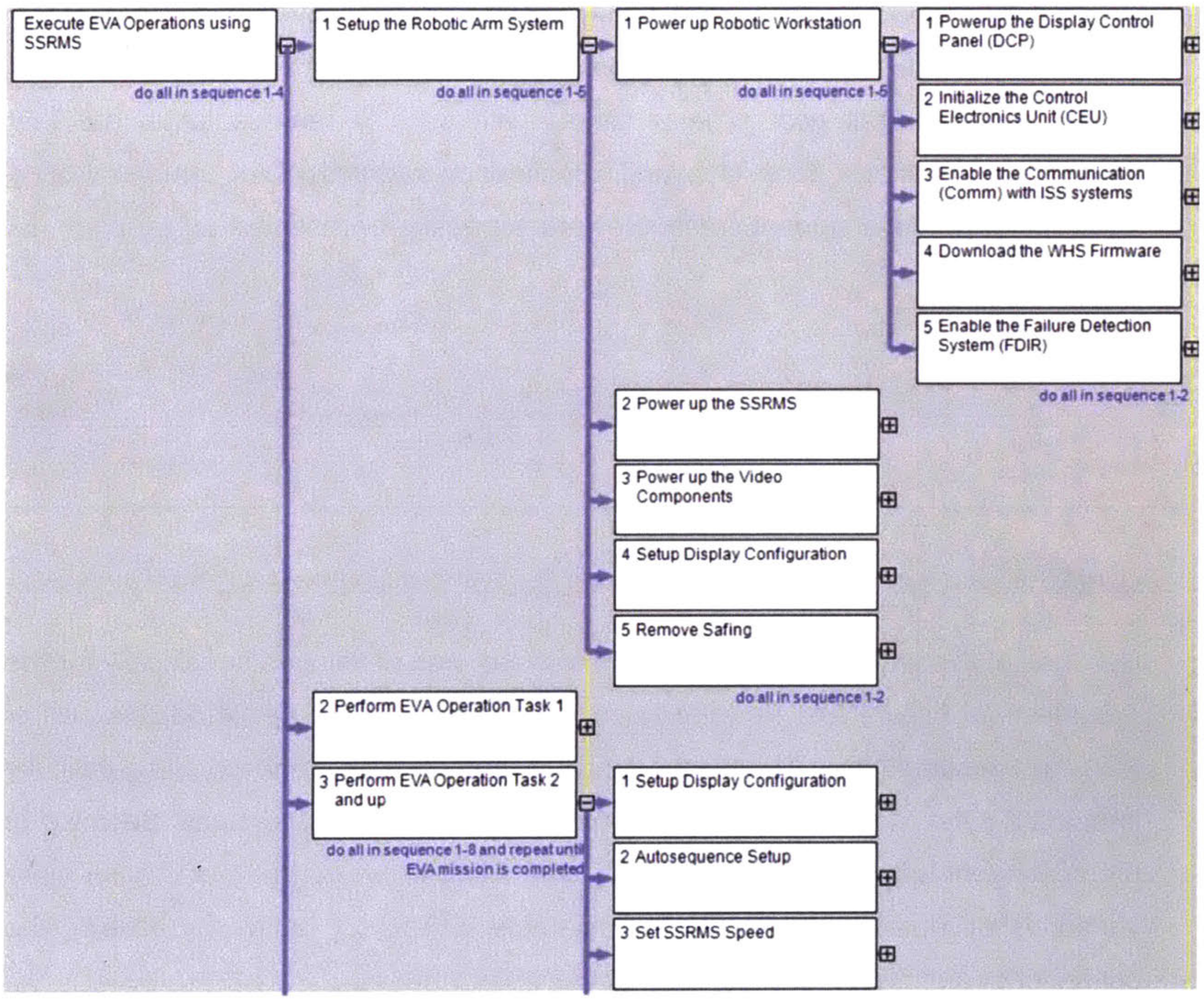

Execute EVA Operations using SSRMS
do all in sequence 1-4
1 Setup the Robotic Arm System
do all in sequence 1-5
1 Power up Robotic Workstation
do all in sequence 1-5
1 Powerup the Display Control Panel (DCP)
2 Initialize the Control Electronics Unit (CEU)
3 Enable the Communication (Comm) with ISS systems
4 Download the WHS Firmware
5 Enable the Failure Detection System (FDIR)
do all in sequence 1-2
2 Power up the SSRMS
3 Power up the Video Components
4 Setup Display Configuration
5 Remove Safing
do all in sequence 1-2
2 Perform EVA Operation Task 1
3 Perform EVA Operation Task 2 and up
do all in sequence 1-8 and repeat until EVA mission is completed
1 Setup Display Configuration
2 Autosequence Setup
3 Set SSRMS Speed

*Figure 22: First Four Levels of HTA with Plans*

### 3.3.4 Failure and Recovery

Up to this point, the HTA developed only covers the nominal procedure required to execute normal EVA operations using SSRMS. The simulation and task analysis, however, has to incorporate potential system failures where the robotic operator must perform failure detection, troubleshoot and recovery. Therefore, the procedure of troubleshooting and rectifying the failure have to be incorporated in HTA as well. It is not appropriate to incorporate the troubleshooting and recovery procedure goals and plans within the nominal procedure because failure can occur at any part of the normal procedure. The HTA methodology does not facilitate such representation where a separate set of procedures needs to be performed by interrupting a normal task to fulfil a new troubleshooting sub-goal. One method, illustrated in Figure 23, shows an example where a Troubleshoot Failure and Perform Recovery sub-goal is added to the overall goal. The problems with such a representation are that troubleshooting is not done last, and contingency interactions of troubleshooting steps with lower level normal procedures are not easily represented.

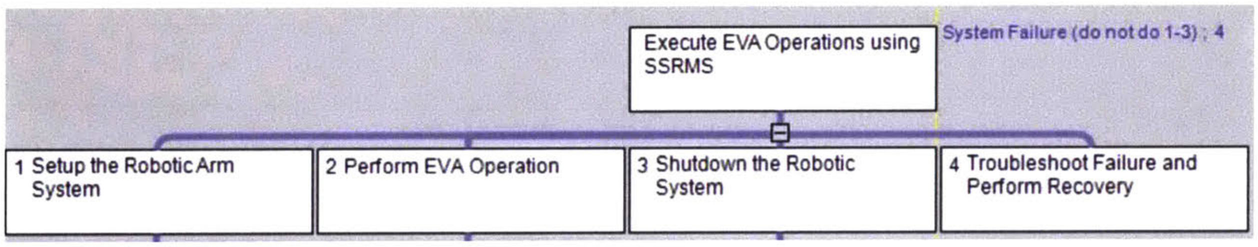

Execute EVA Operations using SSRMS
System Failure (do not do 1-3) ; 4
1 Setup the Robotic Arm System
2 Perform EVA Operation
3 Shutdown the Robotic System
4 Troubleshoot Failure and Perform Recovery

*Figure 23: Example of Troubleshoot Failure and Recovery add to Nominal Procedure HTA*

Because of this limitation, a separate HTA model was developed specifically for the Troubleshoot Failure and Perform Recovery procedures. Figure 24 depicts part of HTA for the overall goal Troubleshoot Failure and Perform Recovery. The difficulty interrelating the normal and troubleshooting HTAs relates to the plans. Because of the contingent nature of troubleshooting and failure recovery, the plans have more variations as illustrated in the simple example in Table 1 below, by building the contingency checks into the troubleshooting HTA.

The complete HTA for Troubleshoot Failure and Perform Recovery can be found in **Appendix A.**

*Table 1: Summary of Unique Plans in Troubleshoot Failure and Perform Recovery HTA*

| ***Goal/Sub-goals*** | ***Plan*** | ***Explanation*** |
|---|---|---|
| Troubleshoot Failure and Perform Recovery | Safing OFF(1); do all in sequence 2-3 | Perform (1) if safing is OFF. Otherwise, proceed straight to (2) and (3). |
| Recover from failure stated in Message Box | Error in Communication (1); Error in Videofeed (2) | If the error is communication, perform (1) and if the error is related to video, perform (2). |
| Recovery Communication Failure | do all in sequence 1-3; if Error Message Remains (4); 5 | Perform (1) to (3) sequentially. Then, perform (4) if the error message still exists. Lastly perform (5). |

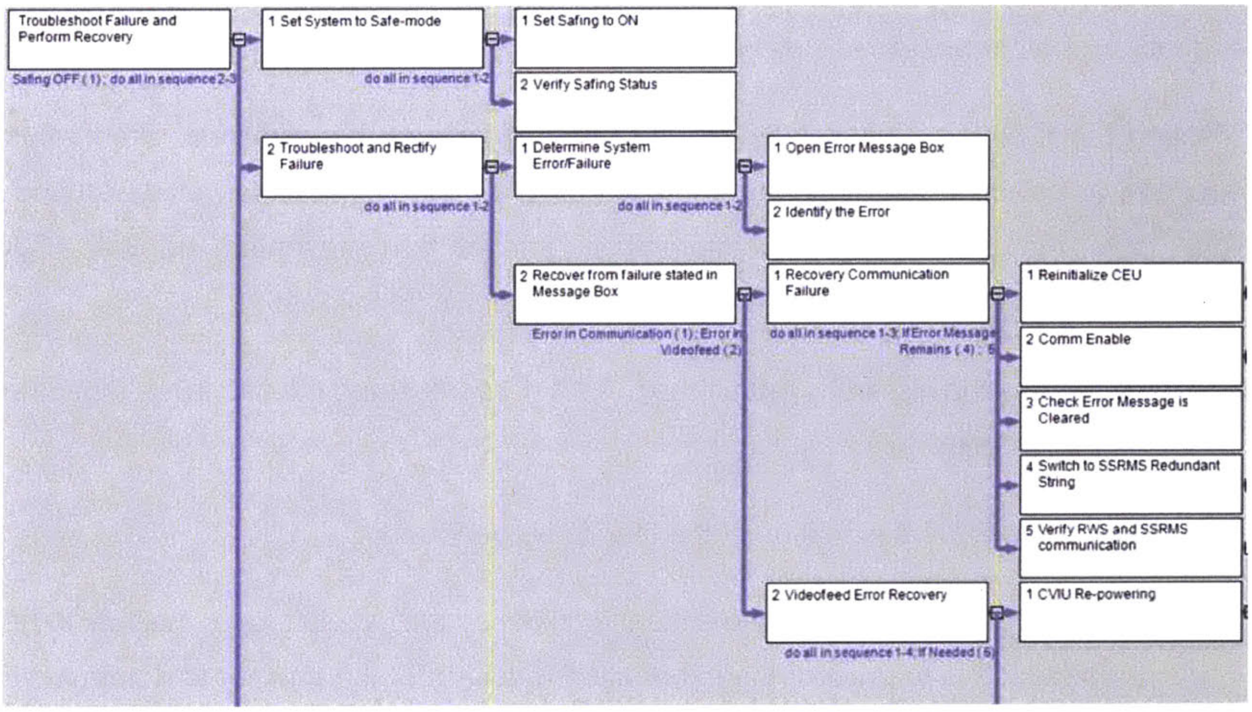

Troubleshoot Failure and Perform Recovery
Safing OFF ( 1 ); do all in sequence 2-3
1 Set System to Safe-mode
do all in sequence 1-2
1 Set Safing to ON
2 Verify Safing Status
2 Troubleshoot and Rectify Failure
do all in sequence 1-2
1 Determine System Error/Failure
do all in sequence 1-2
1 Open Error Message Box
2 Identify the Error
2 Recover from failure stated in Message Box
Error in Communication ( 1 ); Error in Videofeed ( 2 )
1 Recovery Communication Failure
do all in sequence 1-3; If Error Message Remains ( 4 ) ; 5
1 Reinitialize CEU
2 Comm Enable
3 Check Error Message is Cleared
4 Switch to SSRMS Redundant String
5 Verify RWS and SSRMS communication
2 Videofeed Error Recovery
do all in sequence 1-4; If Needed ( 6 )
1 CVIU Re-powering

*Figure 24: Part of HTA for Troubleshoot Failure and Perform Recovery*

## 3.4 Expansion of HTA using Tabular Task Analysis

The HTAs were developed to (1) perform EVA operations using the SSRMS and (2) troubleshoot failure and perform recovery cogently delineated the sequence of tasks and the specific do-list of the procedural instructions that need to be done to achieve the goal. However, the HTA does not explicitly define all the controls (interfaces), displays and information required for the activities. It also does not guarantee that the procedures are logically correct. These factors are also important in system design.

To address the issue of controls and displays requirements, an extension of HTA was sought. One approach is the Tabular Task Analysis (TTA) [42]. The TTA follows on from the HTA and takes each particular task-step or operation and considers specific aspects such as the enablers, display and control used, feedback and possible errors in a columnar format. Stanton [6] had used it to analyze part of the HTA to land aircraft X at New Orleans Airport using the auto land system. This methodology was applied to the lowest level tasks in the HTA developed in this thesis and properties such as control, feedback and other possible fields were used to present the information required.

Adapting the Kirwan's TTA concept to this thesis problem, the additional information required was populated by adding columns to the HTA hierarchical list format developed. Note that, the additional TTA information cannot be presented in the HTA graphical/diagram format. The definition for the headers is as follows:

1) ***No.***: This is the hierarchy list number. '1.1.1.1' represented the fifth level, including the overall goal.

2) ***Task***: This is the description of the task to be performed.

3) ***Initiating cue/event***: This is to state the condition that would trigger the task to be performed. It is to present the information required for the enabler to determine if the task should be performed.

4) ***Enabler***: It represents the thing – Human or Computer – responsible for the tasks. This could be set to 'Computer' to represent automation performing the task.

5) ***Required Actions***: This is to state what the enabler needs to perform to complete the task. Although the information appears to be the same as the task, it is used to state how the task can be done.

6) ***Control Used***: It states the control to be used for perform the required action. E.g. electronic switch is needed for the human to select the state of the system.

7) ***Display Info needed***: This is to present the information needed to be display. E.g. the state of the DCP needs to be displayed for the enables to perform the verification visually.

8) ***Display Type***: This is to define the display property required to present the information. E.g. It can be an on-screen indicator (electronically driven) or a light on the control (physical indicator).

9) ***Feedback***: This is the feedback required to be provided to the enabler in order to complete the task. E.g. the enabler needs DCP status to show it is in the state of 'ON' for the visual verification.

Table 2 described a few of the lowest level tasks from the EVA Operations using SSRMS HTA and Troubleshoot Failure and Perform Recovery HTA. The TTA concept is general as it allows any properties required for the assessment to be incorporated. Overall, the expanded HTA developed provides a somewhat more comprehensive analysis of the task. The TTA concept allows the systems or components required to support a task to be defined but unfortunately, the information cannot be easily presented graphically in the HTA diagram format. However, the TTA information enables the human and computer dependent aspects for the task to be identified, improving the efficacy of the task analysis significantly.

*Table 2: Tabular Task Analysis*

| No | Task | Initiating cue/event | Enabler | Required Action | Control Used | Feed-back | Display Type | Display Info needed |
|---|---|---|---|---|---|---|---|---|
| **EVA Operations using SSRMS HTA** | | | | | | | | |
| 1.1.1.1 | Turn DCP - ON | Procedure Instruction | Human | Change System State | Electronic Switch | | | |
| 1.1.1.1 | Verify DCP power is set as 'ON' | Procedure Instruction | Human | Verification | | ON | on-screen indicator | DCP Status |
| 1.1.2.1 | Initialize CEU | Procedure Instruction | Human | Change System State | Electronic Switch | | | |
| 1.1.2.2 | Verify that CEU initialization is complete | Procedure Instruction | Human | Verification | | Initialized | on-screen indicator | CEU status |
| 1.1.3.1 | Enable Comm | Procedure Instruction | Human | Change System State | Electronic Switch | | | |
| 1.1.3.2 | Verify that Comm is enabled | Procedure Instruction | Human | Verification | | Enabled | on-screen indicator | Comm Status |
| Troubleshoot Failure and Perform Recovery HTA | | | | | | | | |
| 1.1 | Set Safing to ON | System Error | Human | Safe the system | Physical Switch | | | |
| 1.2 | Verify Safing Status | Procedure Instruction or Standard Operating Practise | Human | Verfication | | Safing ON | On-screen Indicator | Safing Status |
| 2.1.1 | Open Error Message Box | System Error | Human | Open the health reporting system | Either Phyiscal or electronic switch | | | |
| 2.1.2 | Identify the Error | System Error | Human | Look for error flagged | | Video or Comm or etc. | On-screen Indicator | Error Type |

Nonetheless, as there is no restriction on the number of properties, the augmented HTA in a spreadsheet could have so many columns that it becomes incomprehensible. Hierarchical relationships are still represented only using the cumbersome HTA hierarchy list numbering scheme. Despite more details presented, the tabular format does not show the relationships of the systems and systems' state and the dependency between tasks (e.g. is Task A a prerequisite of task B (a dependency), or the designer simply prefers Task B has to happen after Task A in the procedure). These limitations constrain the HTA+TTA user's ability to fully comprehend the entire system. Moreover, the task analysis models do not validate the consistency or logical correctness of the design. For example, if a required task is inadvertently left out, it may not be apparent to the analyst and could go unnoticed, since HTA+TTA models cannot be exercised via computer simulations. As system complexity increases, the potential impacts of these representational limitations necessarily grows.

## 3.5 Abstraction Hierarchy

The HTA+TTA has provided a somewhat more complete model of the procedure, and now represents – albeit in a cumbersome way - human interactions and certain display and control requirements. However, the HTA+TTA does not clearly show how the overall system purposes and values are reflected in the design of the interfaces and procedures. To investigate, the author performed a Cognitive Work Analysis (CWA). CWA is an extension of traditional task analysis techniques to yield information about the knowledge, thought processes, and goal structures that underlie observable task performance [43]. CWA focuses on constraints rather than goals and plans. According to Salmon et al. [44], the first stage of CWA is Work Domain Analysis, which requires the description of the system as an Abstraction Hierarchy, described below. There are four other stages of CWA not attempted in this thesis.

Salmon et al. [44] noted that CWA presents a high level description of the system while HTA provides a complementary detailed description of activities at the minute level. An Abstraction Hierarchy (AH) [45] provides a context-independent description of the system [6]. AH comprises of five level of abstraction, starting with the overall function/purpose of the system down to the physical components. Means-ends links in an AH model show how individual components (means) influence the overall abstract objectives (ends) of the system. Means-ends links are thus a conceptual structure linking a parent and to its child, a means. The link reveals the resources or constraints at one level that must be used for satisfaction of resources or constraints at the preceding level. Usually, a set of resources or constraints at one level must be used to support any function, value or purpose at a higher level [46]. There are various names used for the five level of abstraction in the CWA literature and the nomenclature used by Xiao et al. [47] is used here. The description of the five Abstraction Hierarchy (AH) levels are explained by Stanton [6]:

1) Domain purpose – The purpose of the entire system. It is independent of any specific situation and is also independent of time.
2) Domain values – The key values that could be used to assess that the system is working well for its domain purpose.
3) Domain functions – The function that can be performed by the combined work system.
4) Physical Function – The physical function that objects can perform and are independent of the domain purpose.
5) Physical Objects – The key physical objects of the work system that is relevant.

For the purpose of this thesis, the domain purpose of the robotics procedure was assumed to be to perform EVA operations only. (It ignores the other functions of the MSS discussed in Chapter 2 related to the Mobile Transport System MT). The AH developed for the MSS is presented in Figure 25.

It is apparent that an AH analysis provides a different class of information than resulting from HTA+TTA. The AH model depicts how the physical components support the various physical functions that in turns support the domain function. Therefore, it contains information about the system that is not covered in the HTA+TTA model explicitly, especially at the second to the fourth level of abstraction.

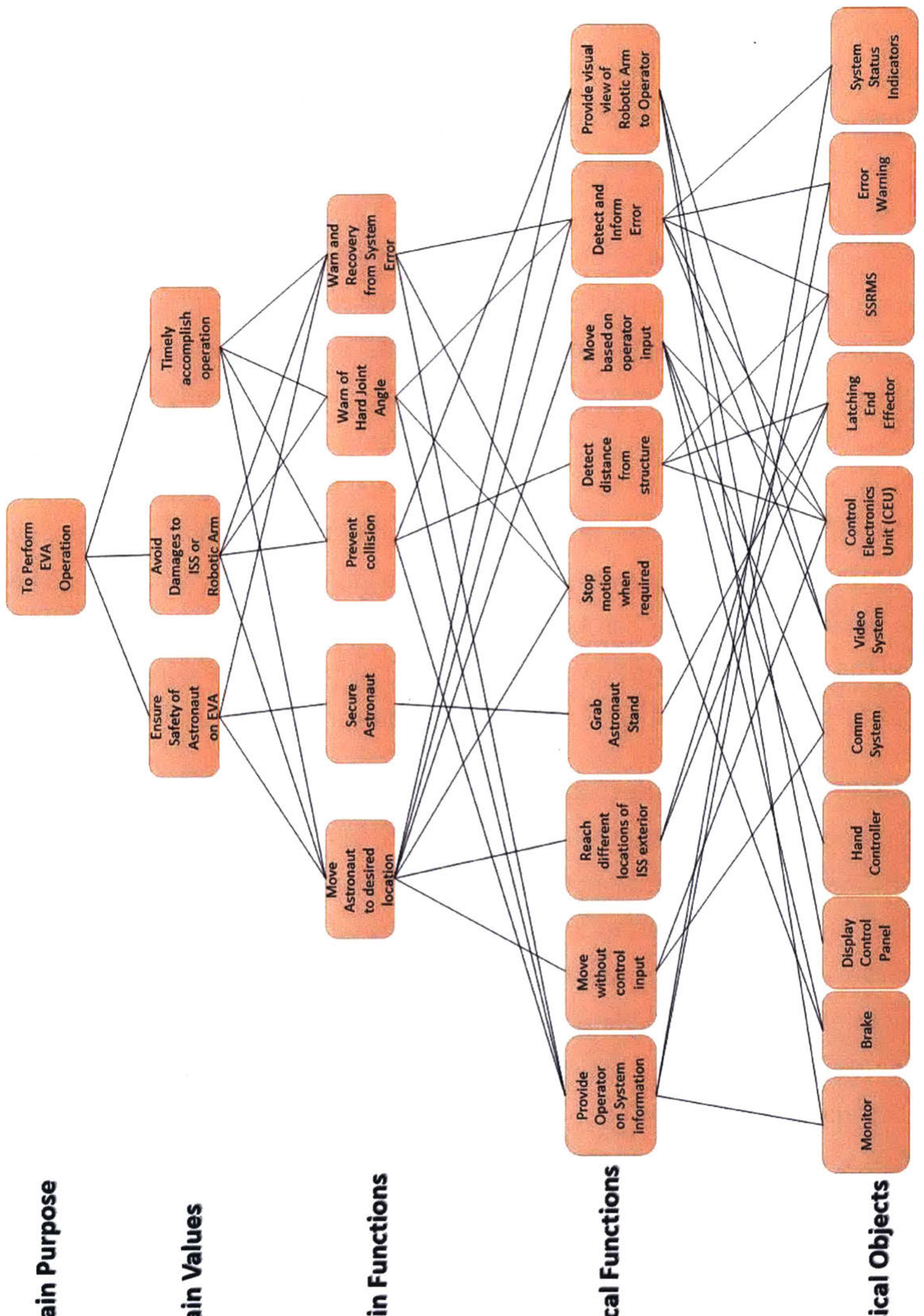

Domain Purpose
Domain Values
Domain Functions
Physical Functions
Physical Objects
To Perform EVA Operation
Ensure Safety of Astronaut on EVA
Avoid Damages to ISS or Robotic Arm
Timely accomplish operation
Move Astronaut to desired location
Secure Astronaut
Prevent collision
Warn of Hard Joint Angle
Warn and Recovery from System Error
Provide Operator on System information
Move without control input
Reach different locations of ISS exterior
Grab Astronaut Stand
Stop motion when required
Detect distance from structure
Move based on operator input
Detect and Inform Error
Provide visual view of Robotic Arm to Operator
Monitor
Brake
Display Control Panel
Hand Controller
Comm System
Video System
Control Electronics Unit (CEU)
Latching End Effector
SSRMS
Error Warning
System Status Indicators

Figure 25: Abstraction Hierarchy of Mobile Servicing System (simplified)

HTA focuses on how the purpose and sub-goals should be achieved through activities. Hence, values and physical functions and objects of the system are not as explicitly represented in the model. Understanding the values and functions of the system helps in the design of interfaces and control of the simulator. Take for example, the domain functions – "warn of collision". Tracing the means-ends links to the physical function abstraction level suggests a need to be able to detect clearance distance between the robotic arm and surrounding structures, and also provide the operator with visual feedback of imminent collision. The components of the system relevant to the said physical functions could be traced through the means-ends links, clearly showing the purpose of the components. Thus, the AH provides context to the components that might otherwise be ignored in the HTA+TTA model. These gaps were evident in the HTA developed. The HTA describes how the tasks should be done but does not explicitly incorporate system's values such as safety that appear in an AH representation. To some extent, some values (e.g. safety) can be implicitly incorporated in an HTA model. For instance, the HTA developed earlier could be modified to incorporate a concurrent sub-goal "Monitor SSRMS movements" in "Perform EVA Operation Task 1" (See Appendix A).

The AH framework provides a description of the systems that is useful in helping designers appreciate the purpose of the system and the physical function and objects required for the system. Like the TTA model, AH analysis does not show the relationships (e.g. structural) among the physical components that are paramount in building the simulator architecture. The AH framework specifics all the elements required at each level of abstraction (e.g. like a brake in the physical objects level and astronaut safety at the domain values level); for a complex system, the number of elements required to be listed, especially at the last two levels would be very large and difficult to present on a single page. Together with the means-ends links, the model would have many lines between each level making the model appear convoluted, which does not promote visualization and understanding. The diagram can be simplified by aggregating the elements into broader categories at each level so that the model becomes visually simple. The drawback is it dilutes the information, reducing the usefulness of the AH model. Therefore, the practitioners have to balance between comprehensibility and usefulness of the AH framework. The model in Figure 25 has been simplified. For example, it does not show the video system components like VDU and CVIU or the list the type of system status indicators like DCP status.

### *3.5.1 Does AH Value Add to HTA+TTA?*

Because the author included the TTA extension of HTA, missing elements like physical components were captured. Thus, some elements in AH model correspond to those in the HTA +TTA model. Therefore, from that perspective, the AH did not add much value to the task analysis effort. Nonetheless, because AH considers values, it helps identify tasks that are influence by values e.g. safety, that otherwise would be inadvertently left out. The author assessed that the AH model could help in the procedure and display designs development by deepening the designers' knowledge of the domain. This observation echoed Miller et al. conclusion that HTA and AH are complementary [48]. Thus, these models – HTA, TTA and AH – together deliver a more comprehensive picture of the domain/system.

## 3.6 Discussion

The HTA+TTA and AH improve the fidelity of the selected task analysis representations used for the purpose of this thesis. The procedures could be effectively developed, and top-level system information required to support the activities or decision-making are included.

The limitations of traditional task analysis (HTA+TTA and AH), however, remain evident. This chapter demonstrated that multiple techniques (i.e. HTA, TTA and AH) and models (e.g. separate HTA for nominal and failure recovery tasks) could be used collectively for system design and procedure development. It suggested appropriate displays and controls, and the steps needed for the procedure. But the HTA, TTA and AH analysis do not produce a computable model that can be run to demonstrate the logical correctness and completeness of the resulting procedure and what other system states are required for each procedural step to be successful. Note that the combination of HTA, TTA and AH was customized for this thesis on the author initiative and was not a standard approach. Without combining multiple task analysis techniques, it is not possible to develop sufficient detail to support both procedure development and engineering system design such as those required here. More information is still needed to complete the system design and a method must be found to check for logical correctness.

Also, although a copious amount of information can be included in the HTA+TTA format, the spreadsheet or computer database (e.g. TaskArchitect) presentation seems convoluted and difficult for others to grasp. This would significantly increase the likelihood of miscommunication between the user and the person who performed the HTA+TTA analysis. HTA+TTA and AH do not show the relationships of the systems and systems' state and the dependency between tasks, which are details required for the programmer to code the program. The system designer would not be able to validate the consistency or logical correctness of the procedure because of task analysis model limitations – HTA and all other task analysis representations are not computable model. Moreover, separate models have to be created for nominal and off-nominal scenarios and the method in HTA+TTA to represent their inter-relationships is extremely cumbersome.

The introduction of AH helps to put context to the physical components delineated in the HTA+TTA model. The AH explicitly shows the purposes and values of the domain and functions of those components that are not covered in the HTA+TTA model. The complexity of the system affects the presentation of the model. A complex system requires a large number of elements in each layer, in particularly the last two levels, can cause the model to become convoluted and hard to comprehend (e.g. the model could be spread across multiple pages).

HTA, TTA and AH collectively provide a broad description of the domain/system and can suggest necessary displays, controls and procedures, there is no way to validate the logical correctness of the result. The next chapter consider how system modelling techniques such as Object Process Methodology can be used to further develop and test the logical correctness of the procedures development and system design together with other advantages, thereby addressing the limitations of HTA, TTA and AH described in this chapter.

# 4 TELEROBOTIC ELECTRONIC PROCEDURES AND SIMULATOR DEVELOPMENT USING OBJECT PROCESS METHODOLOGY (OPM)

## 4.1 Introduction

The task analysis in the previous chapter illustrated how several traditional human factors engineering methods can be used to represent the entire EVA telerobotic operations tasks. Nonetheless, to include the nominal and off-nominal procedures to the existing simulator, the designers also need to 1) understand the relations between the procedure and the physical and informational elements, 2) know the inter-relationships between these physical and informational elements and 3) have a mechanism to check for logical correctness/completeness so that inappropriate system behavioral modes resulting from design or procedures developed could be discovered. Therefore, besides HTA+TTA, an additional tool would be needed to lay out the subsystems, processes, conditions and display of the simulation, preferably in graphical format.

Conceptual system modeling is important, particularly in the early stage of engineering design. In the present case, conceptual modeling allowed the designers of the telerobotic electronic procedure system to study the logical structure of the combined human/procedure/robotics system before coding the system and human-in-the-loop testing. System modeling at the conceptual level also helps users to more fully understand the entire system and provides everyone in the project team with a common intellectual approach using visual symbology and terminology. There are several system modeling languages available from the field of Systems Engineering – System Modeling Language (SysML), Unified Modeling Language (UML) and Object-Process Methodology (OPM). Table 3 [49] shows the high-level comparison of some possible modeling languages considered for the present purpose and some of their features. There are other programs such as Matlab (Simulink) that can be used for multi-domain dynamic system and state machine simulations. However, these programs require technical expertise and in-depth engineering knowledge of the systems being modeled that is often not available at the initial stage of the design process.

OPM, on the other hand, does not require the users to specify the subsystems and their inter-relationship in complete detail; OPM users can decide what level of details is needed to adequately specify the important elements of the system. [Note: It is possible to incorporate more technical details (e.g. equations) into an OPM model, but this was not required for the present purposes.] OPM development is done using a publicly available Object-Process Computer Aided System Engineering tool, OPCAT, described in detail in later section.

*Table 3: Comparison Between Three Modeling Languages*

| | UML | SysML | OPM |
|---|---|---|---|
| Number of diagram types | 13 | 9 | 1 |
| Number of symbols | ~120 | ~120 | ~20 |
| Graphics-text bimodality | No | No | Yes |
| Simulation/Animation ability | No | Partial (in some tool for some diagram types) | Yes |
| Built-in complexity management (in-zoom, unfold) | No | No | Yes |

The OPM was selected for this project to conceptualize the design of electronic procedures for the telerobotic simulator and to understand the human and machine interaction. Compared to UML and SysML, OPM is easier to learn relative to the other modeling tools due to its very simple object-process syntax (with only 18 symbol types) and built in animation capability of the whole model – an important feature in debugging conceptual modeling. Moreover, OPM has been applied to wide variety of fields such as System Architecting, Operational Analysis, communication protocol and decision automation [49]–[55]. But to the best of the author's knowledge, OPM has not yet been applied in the domain of Human Factors Engineering. Fortunately, OPM is not a domain specific tool, underscoring it as a potential and promising conceptual modeling tool for the intent of this thesis.

## 4.2 Object-Process Methodology (OPM)

### *4.2.1 What is OPM?*

OPM [56], [57] is a holistic conceptual modeling approach to the design and development of systems. OPM is a formal yet intuitive paradigm for systems architecting, engineering, development, life cycle support, communication, and evolution. OPM simplifies the complexity of the system being modeled in a format that is simple and intuitive. OPM is ISO 19450 [58], enabling practitioners to have a normative reference document that is common to all OPM users, so anyone who knows OPM should be able to understand any OPM model. Applications of OPM range from satellite control software [59] to large, complex socio-technical systems.

OPM depicts a conceptual model in two semantically equivalent modalities – graphical and textual. The graphical model consists of a set of Object-Process Diagrams (OPDs), and the textual model is a natural set of English sentences written in Object-Process Language (OPL) that are translated on the fly automatically based on the graphical representation. OPD is graphical representations of a system using a relatively small set of symbols (e.g. arrows, rectangles, ellipses, etc.) and annotations whose meanings are easily learned by non-domain expert. From the OPDs, OPCAT (Object-Process CASE Tool) generates OPL text corresponding to the graphical representations. If the users are uncertain about their choice of the graphical elements, they can crosscheck the corresponding OPL description to ascertain that their choice reflects their intention.

### *4.2.2 Building Blocks of OPM*

The elements of any systems model described in OPM requires 1) Things and 2) Link. Object and Process are OPM things, and Link expresses the relations between things graphically.

#### *4.2.2.1 OPM Things: Objects and Processes*

An Object is defined as a thing that exists or might exist physically or informatically. The Object could exist in one of their permissible states that represent the system's structure, and is then known as a stateful object. Objects are described by nouns e.g. computer, tree, house and the first letter of the name of any object is always capitalized. In an OPD, an object is graphically represented by a rectangle. The states of an object are represented by rounded rectangles ("rountangle") drawn within the rectangle, and the names of these states are never capitalized. Figure 26 illustrates an object – **Computer** (a physical object), and the computer may have two possible states - **on** or **off**.

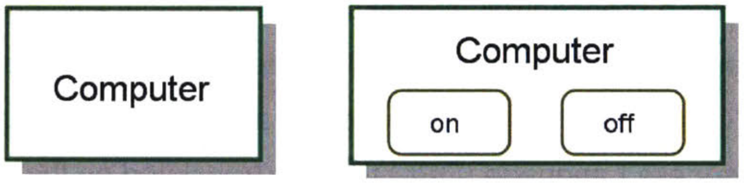


*Figure 26: Examples of Object (left) and Stateful Object (right) Represented in OPD*

A process is a thing that transforms one or more objects, representing the system's behavior. Processes transform objects by creating them, consuming them, or changing their state. Unlike an object, a process is an action e.g. Baking and Burning, so processes are typically given descriptive names in the gerund form, ending in "ing". An ellipse in OPD graphically represents processes.

*Figure 27: An Example of Process Represented in OPD*

Essence and affiliation can further categorize objects and processes. Essence refers to the nature of the object or process i.e. physical or informatical. Affiliation of an object or process has two properties – systemic or environmental. Systemic means that the object or process is part of the system and environmental means that the object is external to the system. The graphical representations of these properties are reflected in Figure 28.

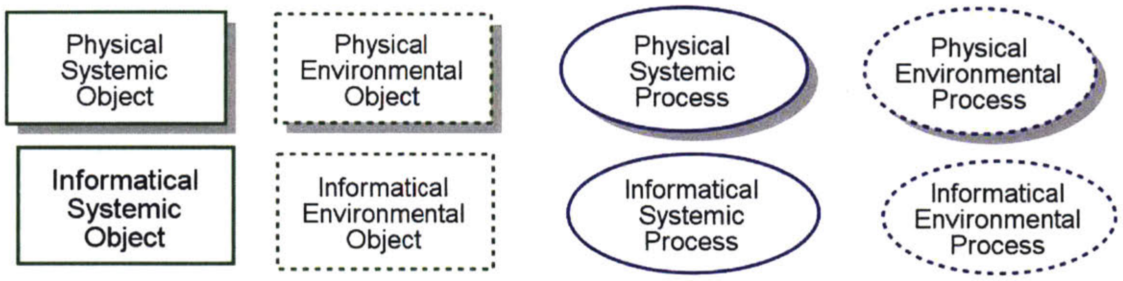

Physical Systemic Object
Physical Environmental Object
Physical Systemic Process
Physical Environmental Process
Informatical Systemic Object
Informatical Environmental Object
Informatical Systemic Process
Informatical Environmental Process

*Figure 28: Graphical Representation of OPM Things Properties*

### 4.2.2.2 OPM Links

OPM links describe the relationships between OPM things. The two types of links in OPM are procedural links and structural links. A procedural link denotes how the system operates to attain its function, designating time-dependent or conditional triggering of processes, which transform objects. A structural link specifies an association that persists in the system for at least some interval of time, i.e. an aspect of the system that is not contingent upon conditions that are time-dependent [58]. Table 4 shows the OPM procedural links and the OPL generated when OPM things i.e. object and process are connected and the explanations for the various connections.

Table 5 illustrates the OPM structural links and the explanations for these links. See ***Appendix B*** for an OPM example.

<u>*Table 4:* OPM Procedural Links and Explanations</u>

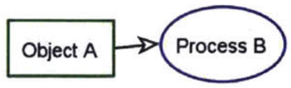
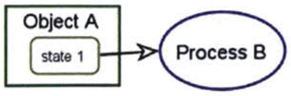
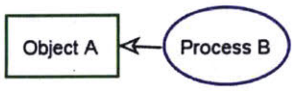
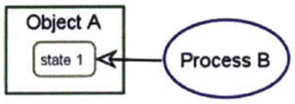
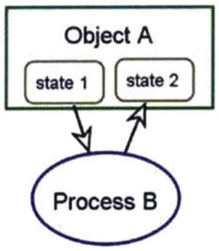
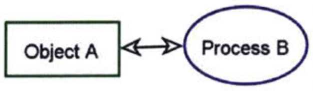
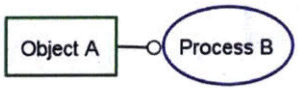
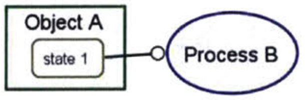
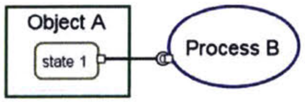
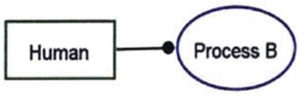
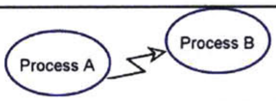
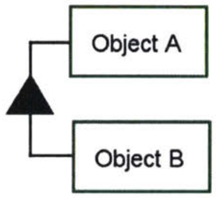
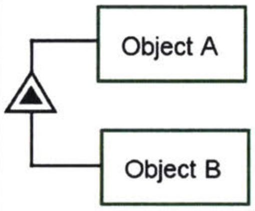
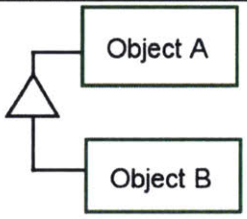
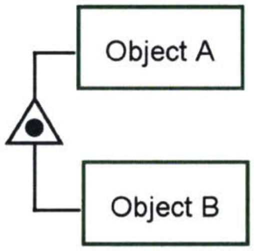
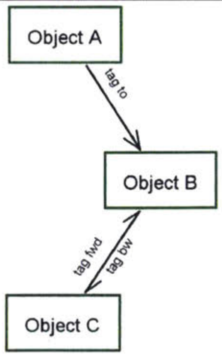
| Link Name | Symbol | OPL | Semantics |
|---|---|---|---|
| Consumption link |  | Process B consumes Object A. | The link denotes Process B consume Object A. Object A existence is a precondition for process B. |
| |  | Process B consumes state 1 Object A. | The link denotes Process B consume Object A at state 1. Object A at state 1 existence is a precondition for process B. |
| Creation link |  | Process B yields Object A. | The link denotes Process B create Object A. Object A existence is a postcondition of process B. |
| |  | Process B yields state 1 Object A. | The link denotes Process B create Object A at state 1. Object A at state 1 existence is a postcondition of process B. |
| Input and Output link |  | Process B changes Object A from state 1 to state 2. | The link denotes that Process B changes Object A at state 1 to state 2. The existence of Object A at state 1 is a precondition. Object A at state 2 is a postcondition of process B. |
| Effect link |  | Process B affects Object A. | The link denotes Process B changes the state of Object A |
| Instrument/ Condition link |  | Process B requires Object A. | The links denote that the existence of Object A or Object A at state 1 is required for Process B execution. If condition is not met, the system halt. If, however, 'c' is in the lollipop, process B is skipped and the next process (if any) tries to execute. |
| |  | Process B requires state 1 Object A. | |
| |  | Process B occurs if Object A is state 1. | |
| Agent link |  | Human handles Process B. | The link denotes a human operator is required to for Process B happen. |
| Invoke link |  | Process A invokes Process B. | The link denotes Process A triggers Process B |

*Table 5: OPM Structure Links and Explanations*

| Link Name | Symbol | OPL | Semantics |
|---|---|---|---|
| Aggregation-Participation |  | Object A consists of Object B. | The link denotes that Object B is part of Object A |
| Exhibition-Characterization |  | Object A exhibits Object B. | The link denotes that Object A has a feature (Object B) |
| Generalization-Specialization |  | Object B is an Object A. | The link denotes that Object B is a form of Object A |
| Classification-Instantiation |  | Object B is instance of an Object A. | The link denotes that Object B is an instance of the source pattern/class (Object A). |
| Tagged structural links Unidirectional Bidirectional |  Object A<br>tag to<br>Object B<br>tag fwd<br>tag bw<br>Object C | Object A tag Object B.<br>Object B tag back Object C.<br>Object C tag fwd Object B. | The link is for user to add text to describe the relationship or association between objects-object or process-process |

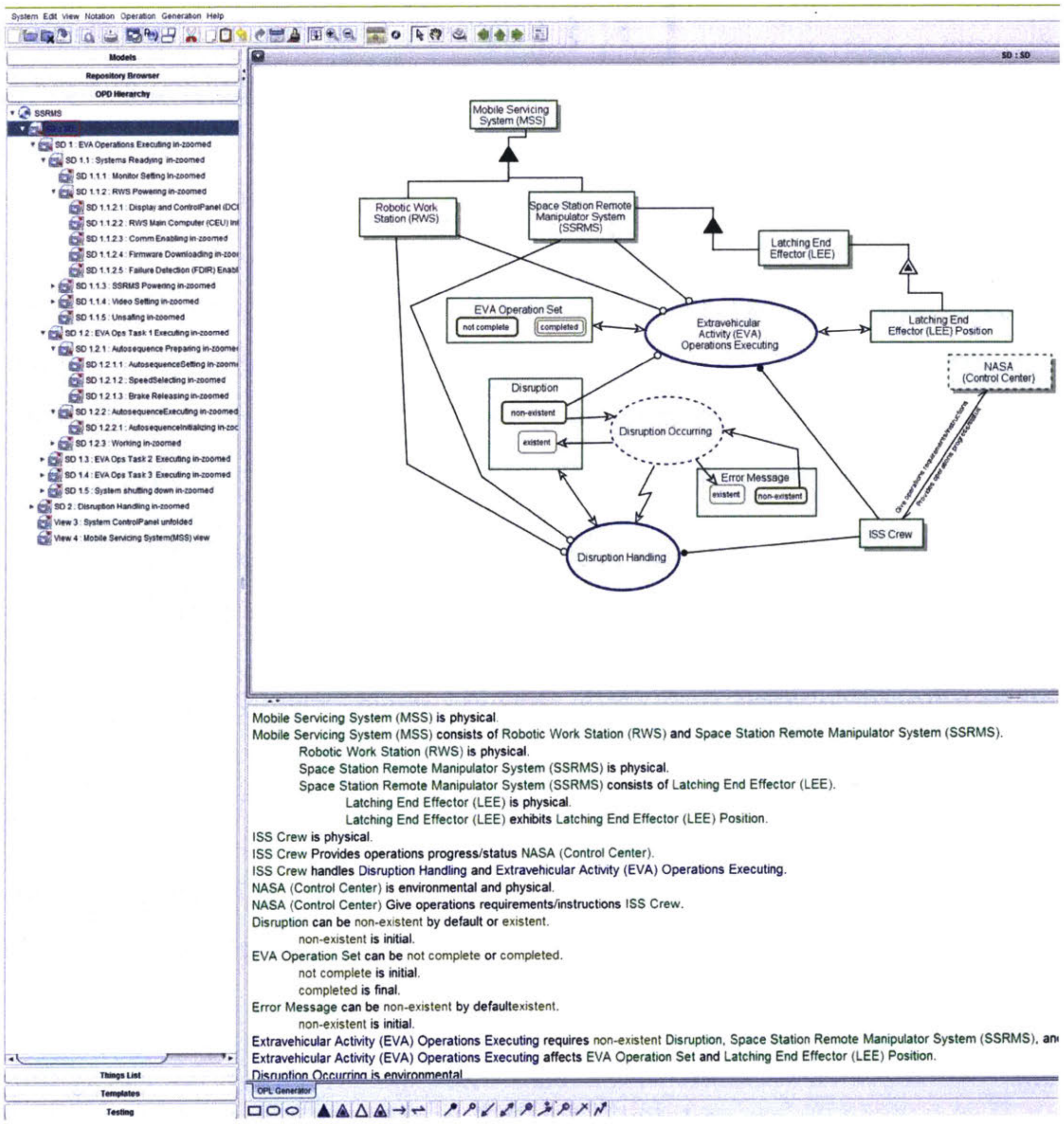
### *4.2.3 OPM Development Platform.*

The conceptual modeling of a system in OPM consists of sets of OPDs arranged hierarchically. This arragement has some correspondence to the HTA+TTA hierarchy numbering methodology but makes the relationships more graphically explicit and intuitive, and provides more details. The first step of OPM is to define the function of the system in the system diagram (SD). Function is defined as the activities, operations and transformations that cause, create or contribute to performance [60]. Function emerges when a process transforms (create, destroy or change) one or more objects in the system. An object that is acted upon i.e. create, destroy or change, is known as an operand or transformee. Conversely, an object that is required to support a process is known as an instrument. The same object can be an operand in one process and an instrument in another. Each OPD may be examined by "in-zooming" an OPM thing to show a new diagram depicting the next lower level in the hierarchy. (Note that "in-zooming" is, therefore, different than conventional graphical "zooming in" which is mere magnification.) An ellipse with thicker border means that the process has been further detailed by in-zooming so that a subordinate OPD exists that shows more details. OPD can also be created by "unfolding" OPM things to show their structural relationships. This hierarchy of OPDs enables system designers to reduce the visual complexity and apparent complexity of a system by developing the conceptual model at a high-level initially and refining it by in-zooming or unfolding.

OPCAT version 4.0 software[4] [61] has a multi-paned, user-friendly interface that allows users to build their OPDs using all the OPM symbols and run animation to validate the consistency of their model. All 18 graphical symbols in the OPM lexicon appear at the bottom of the interface as graphical tools as shown in Figure 29, and tooltips are provided to remind the user of the definitions. OPCAT will flag an error if a link created by the user violates the syntax of OPM. (However, it is up to the user to ensure that the graphical representation appropriately decribe the intended system.) As the model is built, it is checked "on the fly" for logical correctness. A "test system" function in OPCAT allows the user to execute the model, and show the

[4] OPCAT is can be downloaded for free from http://esml.iem.technion.ac.il/ .

succession of active objects and processes via graphical animation by changing the colors of active things. This model animation capability is detailed later. A model that is inconsistent or logically incorrect will halt unexpectedly during the animation; highlighting faulty design logic (logical flow of the system) that might otherwise go unnoticed. This allows the system designer – in the present case the designer of the robotics electronic procedures - to explore and validate their design logic and operating concepts. The animation feature is an indispensable tool in procedure development for this thesis because it helps to identify missing elements and relationships.



System Edit View Notation Operation Generation Help
Models
Repository Browser
OPD Hierarchy
SSRMS
SD 1 : EVA Operations Executing in-zoomed
SD 1.1 : Systems Readying in-zoomed
SD 1.1.1 : Monitor Setting In-zoomed
SD 1.1.2 : RWS Powering in-zoomed
SD 1.1.2.1 : Display and ControlPanel (DC
SD 1.1.2.2 : RWS Main Computer (CEU) in
SD 1.1.2.3 : Comm Enabling in-zoomed
SD 1.1.2.4 : Firmware Downloading in-zoo
SD 1.1.2.5 : Failure Detection (FDIR) Enab
SD 1.1.3 : SSRMS Powering in-zoomed
SD 1.1.4 : Video Setting in-zoomed
SD 1.1.5 : Unsafing in-zoomed
SD 1.2 : EVA Ops Task 1 Executing in-zoomed
SD 1.2.1 : Autosequence Preparing in-zoome
SD 1.2.1.1 : AutosequenceSetting in-zoom
SD 1.2.1.2 : SpeedSelecting in-zoomed
SD 1.2.1.3 : Brake Releasing in-zoomed
SD 1.2.2 : AutosequenceExecuting in-zoomed
SD 1.2.2.1 : AutosequenceInitializing in-zoc
SD 1.2.3 : Working in-zoomed
SD 1.3 : EVA Ops Task 2 Executing in-zoomed
SD 1.4 : EVA Ops Task 3 Executing in-zoomed
SD 1.5 : System shutting down in-zoomed
SD 2 : Disruption Handling in-zoomed
View 3 : System ControlPanel unfolded
View 4 : Mobile Servicing System(MSS) view
Things List
Templates
Testing
SD : SD
Mobile Servicing System (MSS)
Robotic Work Station (RWS)
Space Station Remote Manipulator System (SSRMS)
Latching End Effector (LEE)
EVA Operation Set
not complete
completed
Extravehicular Activity (EVA) Operations Executing
Latching End Effector (LEE) Position
NASA (Control Center)
Disruption
non-existent
existent
Disruption Occurring
Error Message
existent
non-existent
Give operations requirements/instructions
Provides operations progress/status
ISS Crew
Disruption Handling
OPL Generator

```
Mobile Servicing System (MSS) is physical.
Mobile Servicing System (MSS) consists of Robotic Work Station (RWS) and Space Station Remote Manipulator System (SSRMS).
        Robotic Work Station (RWS) is physical.
        Space Station Remote Manipulator System (SSRMS) is physical.
        Space Station Remote Manipulator System (SSRMS) consists of Latching End Effector (LEE).
                Latching End Effector (LEE) is physical.
                Latching End Effector (LEE) exhibits Latching End Effector (LEE) Position.
ISS Crew is physical.
ISS Crew Provides operations progress/status NASA (Control Center).
ISS Crew handles Disruption Handling and Extravehicular Activity (EVA) Operations Executing.
NASA (Control Center) is environmental and physical.
NASA (Control Center) Give operations requirements/instructions ISS Crew.
Disruption can be non-existent by default or existent.
        non-existent is initial.
EVA Operation Set can be not complete or completed.
        not complete is initial.
        completed is final.
Error Message can be non-existent by defaultexistent.
        non-existent is initial.
Extravehicular Activity (EVA) Operations Executing requires non-existent Disruption, Space Station Remote Manipulator System (SSRMS), an
Extravehicular Activity (EVA) Operations Executing affects EVA Operation Set and Latching End Effector (LEE) Position.
Disruption Occurring is environmental.
```

*Figure 29: OPCAT User Interfaces*

In OPCAT, the System Diagram (SD) is the topmost diagram in a model that presents the most abstract view of the system. The SD typically - but not always - shows a single process as the main function of the system, along with the most significant objects that enable it and the ones that are transformed by it [62]. The conceptual modeling can then be refined through in-zooming and unfolding, with the multiple hierarchical OPDs that is listed in the left column of the OPCAT interface shown in of Figure 29.

### *4.2.4 Framework for Human Factors Engineering Analysis and Design Using OPM.*

Analogous to Stanton's framework for HTA described earlier, the author developed an OPM-based framework for Human Factor Engineering analysis and design of a planned task. OPM provides for including both the human and the technical system she or he operates in a single model, producing a holistic representation of the entire system's architecture—the combination at all levels of its structure and behavior that enables its function, which is performing the task. The OPM model enables simulating the system behavior in nominal and off-nominal conditions. Following are the major stages of our OPM human factors engineering analysis and design framework. The descriptions below assumed that users know OPM.

1. ***Define the task.*** Determine what is the task to be performed and what is considered to be a successful termination of the human-machine task.

2. ***Determine the system boundary.*** Decide what is included in the human-machine system, what is the system boundary, and what are the objects outside of the system (e.g. Sun) with which the system interacts.

3. ***Collect Data***– below is a non-exhaustive guide to the information required.
   a. The high-level processes to be performed, their order and dependencies, their inputs and outputs.
   b. The systems required to support those processes.
   c. The conditions for performing the task.

d. The subprocesses (at the appropriate level of depth) involved in those high-level processes.
e. The systems and activities outside the system boundary that might affect the activity outcome.
f. The distribution of work and responsibilities between the human or human team members and the machine.
g. The human and machine decision processes.
h. The preconditions and post-conditions of the activities.
i. Principles, guidelines, and best practices in the domain being analyzed.

4. ***Create the OPM system diagram (SD)***. SD is the bird-eye view of the system, aimed at quickly providing understanding of the human-machine task and the objects involved in performing it.
   a. Start with modeling the task as the top-level process—the system's function
   b. Add the main objects involved as enablers (agents – humans, and instruments – non human objects), key systems and operands (transformees), i.e., objects that the task transforms (creates, consumes or changes their states).
   c. Connect the objects to the top-level process using the appropriate procedural link: agent, instrument, consumption, result or effect links.
   d. Connect objects to objects using the appropriate structural link: aggregation-participation, exhibition-characterization, or generalization-specialization.
   e. Add the condition and event control modifiers to ensure correct operational semantics.
   f. Check the newly created or edited OPL sentence to verify that the graphical edit of the model is correctly reflected by the sentence. If not, correct the graphical model until the text reflects your modeling intent.

5. ***Perform animated simulation.*** The simulation ensures that the model executes correctly, so it has to be performed after each significant graphic edit operation of the OPM model, otherwise it becomes difficult to track the logical error introduced since the last correct animated simulation.

6. ***Zoom into the process.*** In-zoom into the process in order to model the next level of detail.
   a. Arrange the subprocesses within the in-zoomed process context according to their order of execution in a top-to-bottom order, taking the top-most ellipse point of each subprocess as the reference point.
   b. Locate parallel processes (if any) at the same height.
   c. Keep the number of processes in each in-zoom OPD to no more than five. If there are more than five processes, review the processes and assess whether some could be clustered together and in-zoomed in the next level of detail.

7. ***Connect existing objects to processes***. The objects in the predecessor OPD are automatically depicted in the in-zoomed OPD and are connected to the outer ellipse of the in-zoomed process.
   a. If an object should be linked to all the subprocesses in the in-zoomed process with the same procedural link, leave it connected to the outer process ellipse Otherwise connect the object to specific relevant subprocesses using the appropriate procedural links.

8. ***Add lower-level objects***.
   a. Determine if new objects related to the subprocesses need to be added.
   b. If so, create each such object, including its states if relevant, and connect the object or its appropriate state to the subprocess using the correct procedural link
   c. If needed, add the correct control modifier (event or condition) to the procedural link.
   d. Connect the newly objects to their ancestors as parts, attributes, or specializations using the appropriate structural link.

9. ***Ensure consistency***.
   a. Check that every process in the OPD has at least one operand, i.e., that the process transforms (creates, consumes, or changes the state of) at least one object.

b. Wherever applicable, assess the enabling links – instrument link for a non-human object that the process requires in order to execute, and agent link if the process is performed by a human (the agent for that process).
c. Check the newly created or edited OPL sentence to verify that the graphical edit of the model is correctly reflected by the sentence. If not, correct the graphical model until the text reflects your modeling intent.

10. ***Verify pre-and post-conditions***.
    a. Check the correctness and completeness of the objects and their states required for performing each subprocess and of the links from these objects or their states.
    b. (If any) For each combination of improper precondition set for each subprocess prepare a contingency subprocess to take care of this off-nominal situation or at least issue an informative message specifying what prevents that subprocess from starting its execution.
    c. (If any) For each combination of improper post-condition set for each subprocess issue an informative message specifying what prevented that subprocess from properly terminating its execution and what postcondition was violated.

11. ***Continue model refinement.***
    a. Recursively performs refinement operations mainly of process in-zooming and parallel object unfolding by repeating step 5 to 11 until no further details are deemed appropriate.
    b. Stop the refinement when the level of detail is sufficient to fully specify the system's structure and behavior such that it can be implemented with minimal need for further explanations or interpretations.

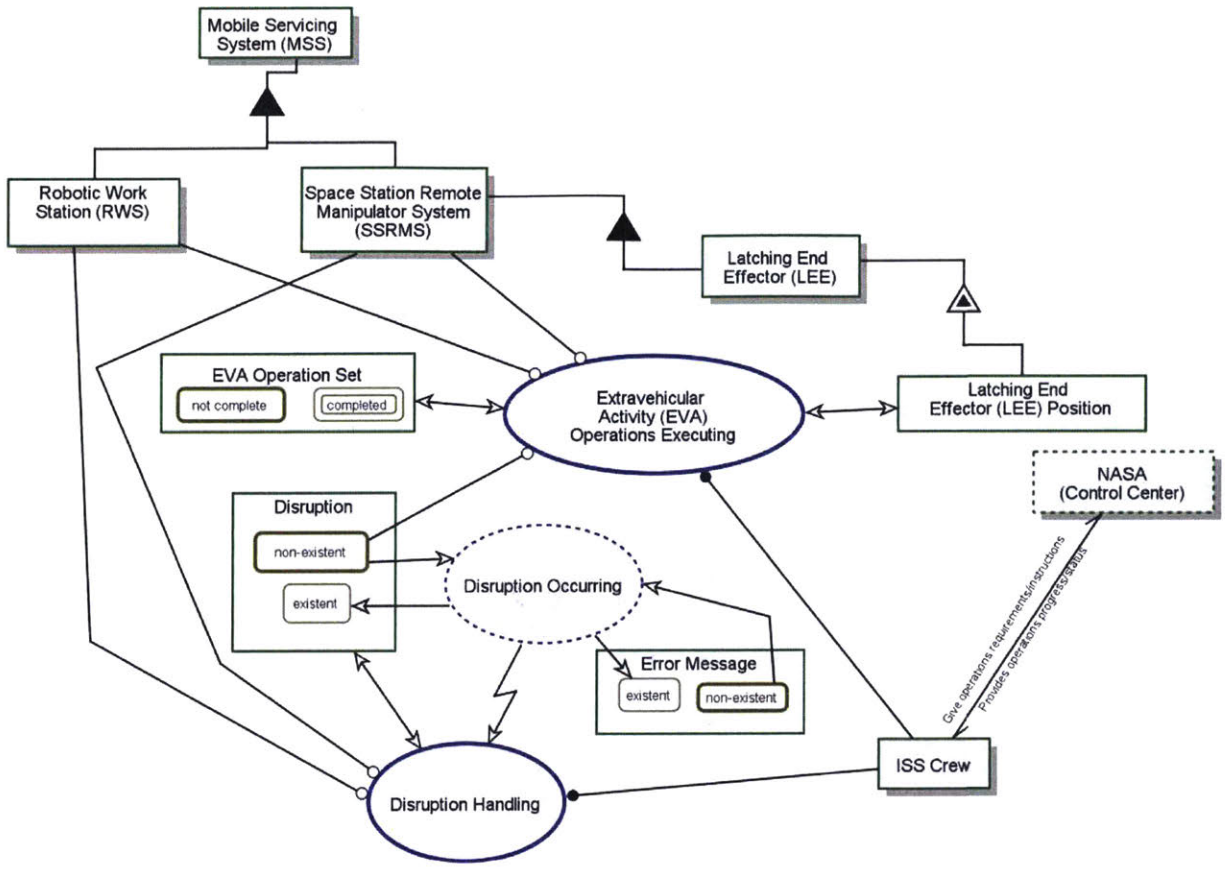
## 4.3 OPM for Electronic Procedure Development and Simulator Architecture

The OPM human factors engineering analysis and design framework described in the previous section as an alternative to traditional TA was applied. Using this framework, we constructed a single formal and executable OPM model of a space telerobotic operation that includes both nominal and off-nominal operations. The fidelity of the model was sufficient to use it as the basis for deriving the necessary displays and controls, the appropriate procedures, the relationships between the displays and controls, and the preconditions for each process to succeed. The following sections explain how OPM was utilized to (1) develop the procedure to be performed, (2) determine which subsystems and components had to be added to the existing telerobotic simulator, and (3) define the necessary preconditions for specific procedural steps to be successfully executed. The resulting OPM defines the system architecture of the improved MIT-RWSS and the dependency of successive procedural steps.

The resulting OPM consists of a top-level system diagram (SD) that is abstract and three to five subordinate levels built downward through in-zooming, refining each to a more concrete level. Altogether there are 73 OPDs. It is not surprising that the in-zooming levels defined in the OPM model correspond roughly to those in the HTA model.

Important and illustrative aspects of the robotics procedure OPM are detailed in the sections that follow. For the interested reader, the entire model is presented in Appendix B. In these sections, words in **green and bold** refer to an object, and those in **blue and bold** refer to a process and those in **bold** refer to a state.

Figure 30 presents the OPD (SD level) and the corresponding OPL text description summarizing the robotics EVA objects and processes at the topmost abstract level. The SD is useful for explaining the intention of the model to others. The SD describes the function of the system – to execute EVA operations – and resembles the top level domain purpose in the previously described AH. The **Latching End Effector (LEE)** is an object at the end of the **SSRMS** and it has attributes of **LEE Position** – locations of LEE in 3D space that would change during the **Extravehicular Activity (EVA) Operations Executing** process. At this high level, failures are represented generically as a "**Disruption**" object that at any moment is either "**existent**" or "**non-existent**"


Mobile Servicing System (MSS)
Robotic Work Station (RWS)
Space Station Remote Manipulator System (SSRMS)
Latching End Effector (LEE)
EVA Operation Set
not complete
completed
Extravehicular Activity (EVA) Operations Executing
Latching End Effector (LEE) Position
NASA (Control Center)
Disruption
non-existent
existent
Disruption Occurring
Error Message
existent
non-existent
Give operations requirements/instructions
Provides operations progress/status
ISS Crew
Disruption Handling

Mobile Servicing System (MSS) is physical.
Mobile Servicing System (MSS) consists of Robotic Work Station (RWS) and Space Station Remote Manipulator System (SSRMS).
Robotic Work Station (RWS) is physical.
Space Station Remote Manipulator System (SSRMS) is physical.
Space Station Remote Manipulator System (SSRMS) consists of Latching End Effector (LEE).
Latching End Effector (LEE) is physical.
Latching End Effector (LEE) exhibits Latching End Effector (LEE) Position.
ISS Crew is physical.
ISS Crew Provides operations progress/status NASA (Control Center).
ISS Crew handles Disruption Handling and Extravehicular Activity (EVA) Operations Executing.
NASA (Control Center) is environmental and physical.
NASA (Control Center) Give operations requirements/instructions ISS Crew.
Disruption can be non-existent by default or existent.
non-existent is initial.
EVA Operation Set can be not complete or completed.
not complete is initial.
completed is final.
Error Message can be non-existent by defaultexistent.
non-existent is initial.
Extravehicular Activity (EVA) Operations Executing requires non-existent Disruption, Space Station Remote Manipulator System (SSRMS), and Robotic Work Station (RWS).
Extravehicular Activity (EVA) Operations Executing affects EVA Operation Set and Latching End Effector (LEE) Position.
Disruption Occurring is environmental.
Disruption Occurring changes Error Message from non-existent to existent and Disruption from non-existent to existent.
Disruption Occurring invokes Disruption Handling.
Disruption Handling requires Robotic Work Station (RWS) and Space Station Remote Manipulator System (SSRMS).
Disruption Handling affects Disruption.

*Figure 30: OPM of the top level SD conceptual model of simulation with its OPD (top) and OPL (bottom) automatically generated in OPCAT*

The **EVA Operations Executing** process requires the **ISS Crew** (who are the agents – human enablers in the OPM terminology), the **RWS**, **SSRMS** and **Disruption** in **non-existent** state, and the process changes the state of **EVA Operation Set**, which represents the entire MIT-RWSS operation from **System Setup** to **System Shutdown**. The initial state of **EVA Operation Set** is **not complete** and the final state is **completed** i.e. all the intended **EVA operation Set** is performed. **NASA (Control Center)** is not part of the system but they provide operations instructions and receive status updates from the **ISS Crew**. This is because the system boundary is the ISS.

The **Disruption**'s **non-existent** state would change to **existent** due to a **Disruption Occurring** that is an external event. **Disruption Occurring** also generates an **Error Message**. A disruption will halt the normal **EVA Operations Executing** process and trigger the **Disruption Handling** process. The **ISS Crew** will need to perform the **Disruption Handling** that affects the state of **Disruption** by changing **Disruption** back to **non-existent**. The **EVA Operations Executing** will resume when **Disruption** is eliminated i.e. **non-existent**.

The user can run the model from the top-level SD (Figure 30) by selecting the test system button (), to go into the animation window. Next, select the **Extravehicular Activity (EVA) Operations Executing** process and then click on the play button (). It will show the objects that are active and the state which they are in. Use the test setting button () to select the animation parameters such as number of steps, automatic move between OPDs, show life-span diagram, etc. Then, click the activate button () to activate the selected process. If in single step mode, use the forward button () to move to the next step. At each step, the life span table beneath the animation window document the successive system states. Right clicking the table allow the user to export the system state trajectory as a table. Logical errors halt the animation. 'Testing problem logs' beneath the animation window specifies the cause and location of the error.

### 4.3.1 EVA Operations Executing In-zoomed

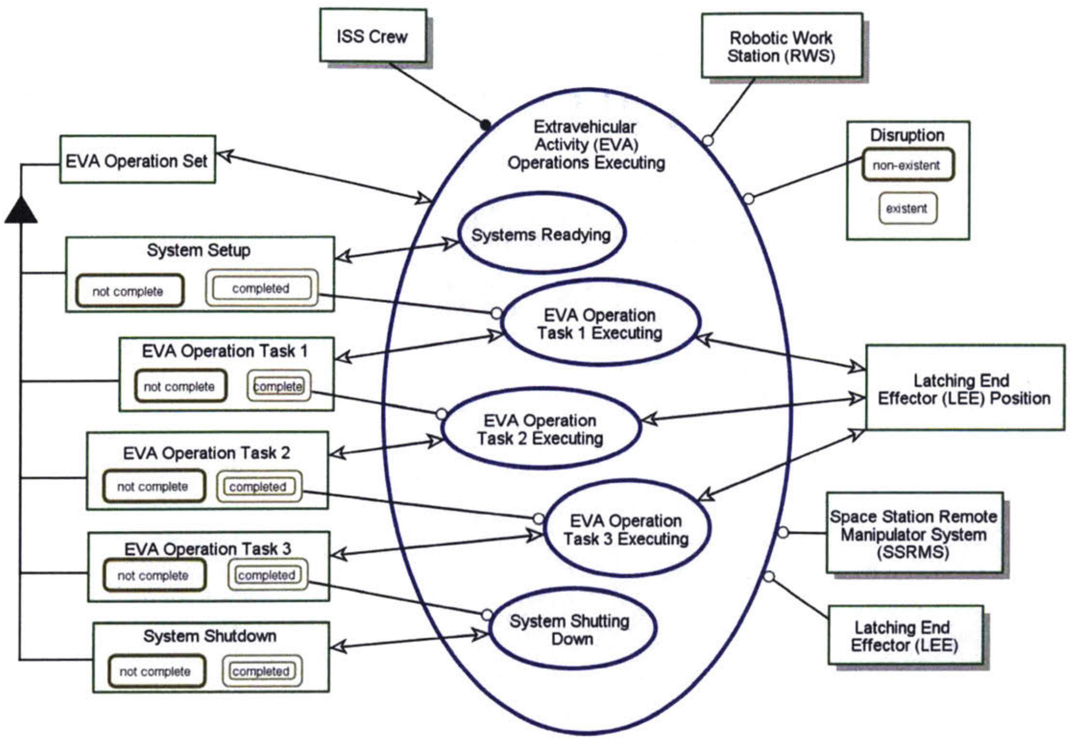


Figure 31: OPD of EVA Operations Executing In-Zoomed

The **EVA Operations Executing** process is in-zoomed in Figure 31 to show its subprocesses and the breakdown of **EVA Operation Set.** Five subprocesses (ellipses), namely **Systems Readying**, **EVA Operation Task 1 Executing, EVA Operation Task 2 Executing, EVA Operation Task 3 Executing and System Shutting down** are revealed. Each of the process affects (change the state of) a corresponding object that is a subset of the **EVA Operation Set**. **System Readying** affects the state of **System Setup,** and **System Setup** in **completed** state is a prerequisite for **EVA Operation Task 1 Executing**. **EVA Operation Task 1 Executing** affects the state of **EVA Operation Task 1. EVA Operation Task 1** needs to be **completed** before the crew can proceed to the next subprocess – **EVA Operation Task 2 Executing**. A similar pattern can be seen from the OPD until **System Shutting Down**, the last process of the **Extravehicular Activity (EVA) Operations Executing**.

The EVA operations task executing subprocesses move the astronaut attached to the end of the **LEE** to different locations around the exterior of the ISS. Thus, the OPD clearly represented this concept by linking **LEE Position** to three processes – **EVA Operation Task 1/2/3 Executing**. It also meant that the LEE is stationary during **System Readying** and **System Shutting Down**.

#### *4.3.1.1 System Readying In-zoomed*

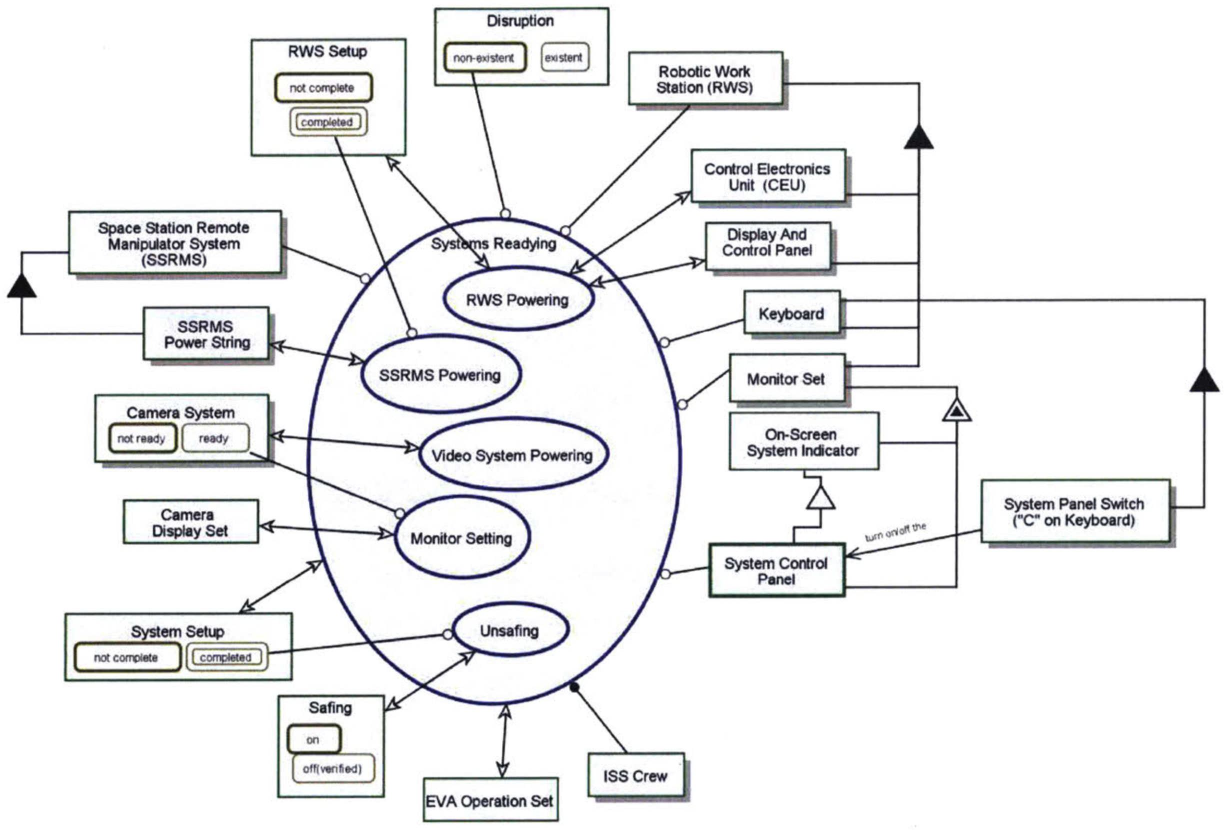


*Figure 32: OPD of System Readying In-Zoomed*

Figure 32 shows the in-zoomed OPD of **System Readying.** The subprocesses that need to be performed are **RWS Powering**, **SSRMS Powering, Video System Powering, Monitor Setting** and **Unsafing**. There are a couple of new, lower-level physical and informatical objects visible in this view that were created for the simulator because the OPD analysis (and the HTA+TTA) showed they were necessary. **Keyboard, Monitor Set** and **System Control Panel** are linked to the **System Readying** ellipse which indicates that these three objects apply to all the in-zoomed subprocesses. In the robotics simulation, the **Keyboard** is used to emulate switches. The key 'C' is the **System Panel Switch** to display the **System Control Panel**. The **System Control Panel** is a feature of the **Monitor Set**. It is also a form of **On-Screen System Indicator** that is displayed on the **Monitor Set**. **On-Screen System Indicator** is display that show the state or mode of the MSS's systems. In simple terms, **System Control Panel** is a window/panel that pops up on one of the monitors that show all the information required for **System Readying** and **System Shutting Down**.

As indicated by the double headed "effect" link arrows, **RWS Powering** changes the state of **RWS Setup**, **Control Electronics Unit (CEU)** and **Display And Control Panel** (DCP). These systems are part of the **RWS** as illustrated by the solid black triangle structural aggregation link. **CEU** is the actual computer system on the real **RWS** that processes all the commands sent to the **SSRMS**. The Display and Control panel is described in section 2.1.4.

**SSRMS Powering** requires **RWS Setup** to be **completed** and it affects the state of **SSRMS Power String**, which is a subsystem of the **SSRMS**. **SSRMS Power String** are power and data cables between **SSRMS** and the ISS, including RWS.

**Video System Powering** affects the state of **Camera System**, which needs to be **ready** for **Monitor Setting. Monitor Setting** affects the **Camera Display Set**.

The last process in **System Readying** is **Unsafing**. It requires **System Setup** state to be **completed** and changes the state of **Safing.**

#### 4.3.1.1.1 RWS Powering In-Zoomed

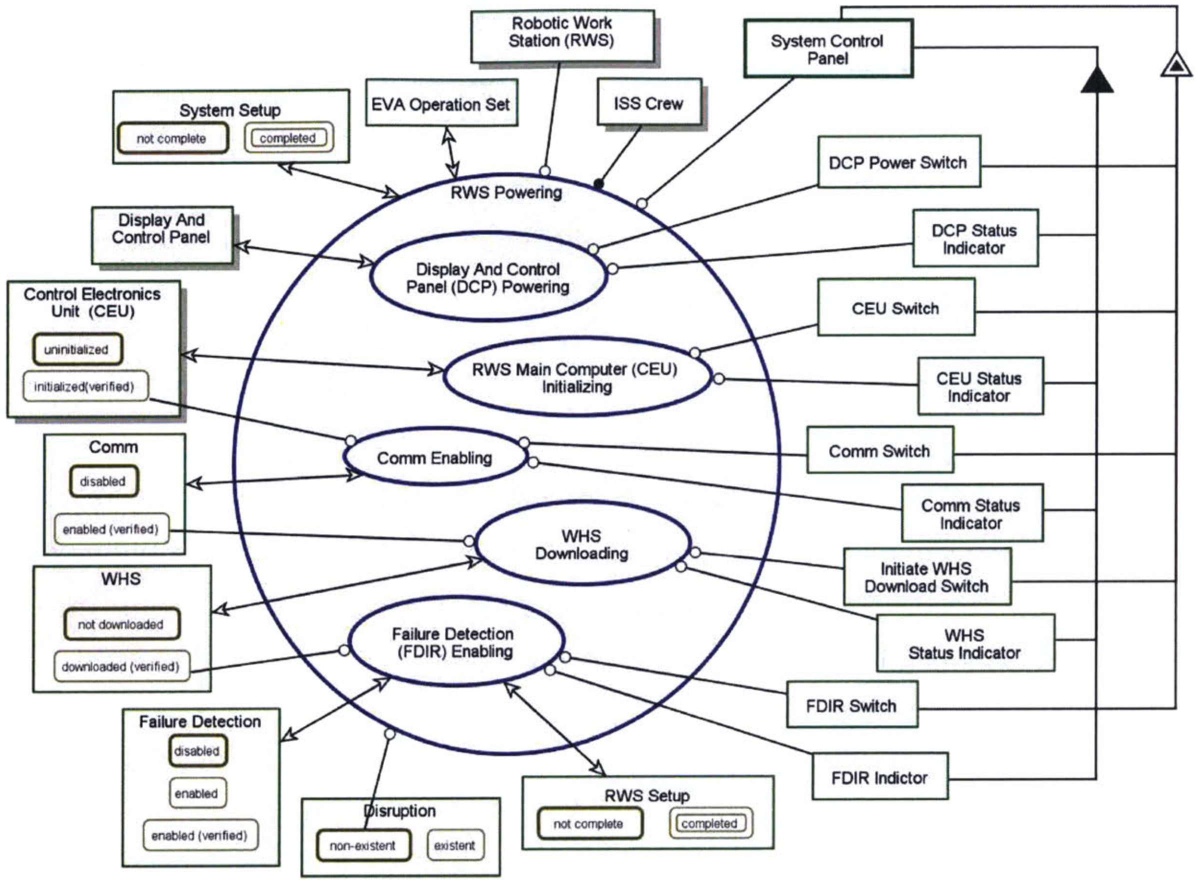


*Figure 33: OPD of RWS Powering In-Zoomed*

Figure 33 **RWS Powering** in-zoom shows all the RWS systems' states that need to be changed during power up. The OPD shows that **Display And Control Panel (DCP) Powering** and **RWS Main Computer (CEU) Initializing** are independent, but the former needs to be performed first based on the arrangement of the ellipse. This concept cannot be represented in HTA – that means the **Display And Control Panel (DCP) Powering** is not a precondition for **RWS Main Computer (CEU) Initializing** but the author chose to use this sequence based on physical switch placement. In contrast the two earlier subprocesses, **Comm Enabling, WHS** and **Failure Detection (FDIR) Enabling** are dependent on the state of the predecessor system e.g. **Comm Enabling** requires **CEU** to be **initialized(verified)**.

These prerequisites need to be coded in the simulation program to prevent the process from happening if the conditions are not satisfied.

The **DCP Power Switch, CEU Switch, Comm Switch, Initiate Firmware Download Switch** and **FDIR Switch** are features incorporated to the **System Control Panel**. These are soft switches in the **System Control Panel**.

The **DCP Status Indicator**, **CEU Status Indicator, Comm Status Indicator**, **WHS Indicator** and **FDIR Indicator** are part of the **System Control Panel**.

4.3.1.1.1.1 In-Zoomed of Processes in DCP Powering

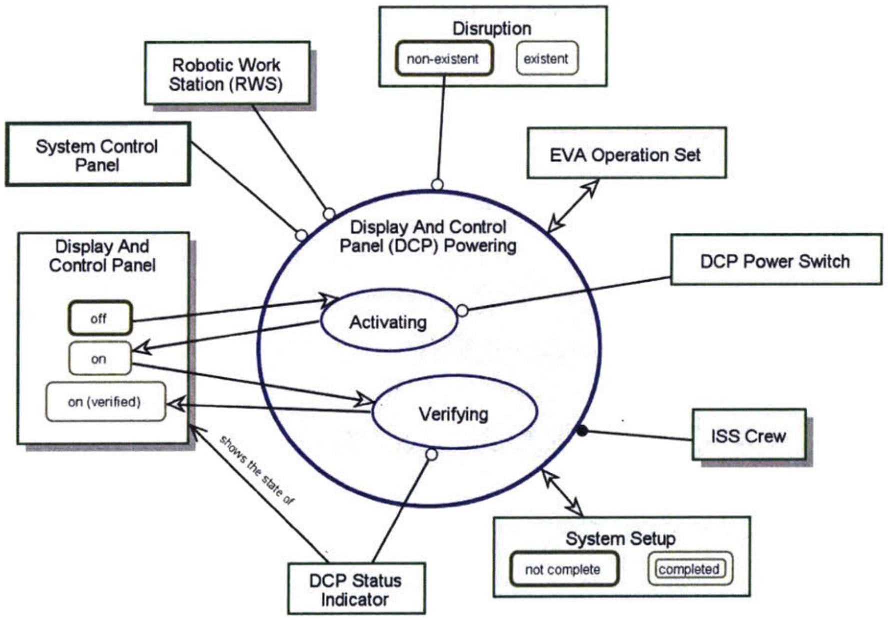


*Figure 34: OPD of Display And Control Panel Powering In-Zoomed*

The in-zoomed OPD of **DCP Powering** in Figure 34 illustrates the lowest level in the hierarchy for **Systems Readying**. **Activating** is the subprocess that changes the **Display And Control Panel** from **off** to **on** and requires the **DCP Power Switch**. **Verifying** is the subprocess that confirms the correct state is set and changes the state of **Display And Control Panel** to **on(verified)**. This representation is required because a process should always change or affect the state of object(s). The verification of the **Display And Control Panel** state requires the DCP Status Indicator to be visible to the ISS Crew. Both subprocesses require a human agent.

In OPCAT, the user can set the duration of any processes using the "Activation Time" in the process properties dialog box. There is a minimum and maximum activation time to allow user to specify a range if needed. This method was used to input the system delays in the OPM model.

This OPD defines two procedural steps for System Readying by Activating followed by Verifying. The union of the OPDs lowest in the OPD hierarchy together, called the system map, forms and explicitly expressed the entire detailed procedure needed to be performed. Thus, this OPD defines the procedure for DCP Powering as follows:

[1] Set DCP Power – On
[2] Verify DCP Power – On

For the interested reader, the remaining four OPM procedures within System Readying are detailed in ***Appendix B***.

#### *4.3.1.2 EVA Operation Task 1 Executing*

Figure 35 presents the OPD of EVA Operation Task 1 Executing in zoomed. EVA Operation Task 1 Executing can proceed only if System Setup is **completed**.

There are eight subprocesses in this in-zoomed OPD. All the subprocesses require ISS Crew except for SSRMS Relocating. (The relocation subprocess is entirely automated and move the LEE from the Latching End Effector (LEE) Position **initial** state to the **task 1 position** state.) At the completion of SSRMS Relocating, it (a) removes the Autosequence Display (created in Autosequence Executing in-zoomed discussed in section 4.3.1.3.1), (b) resets SSRMS Mode to **manual** and (c) Speed Mode to **normal**. These automated charges are required to reset the system for the subsequent EVA operations and need to be implemented in the simulator. (Note that such important details cannot be readily graphically represented in HTA, TTA and AH diagrams but are necessarily incorporated in the OPM model.)

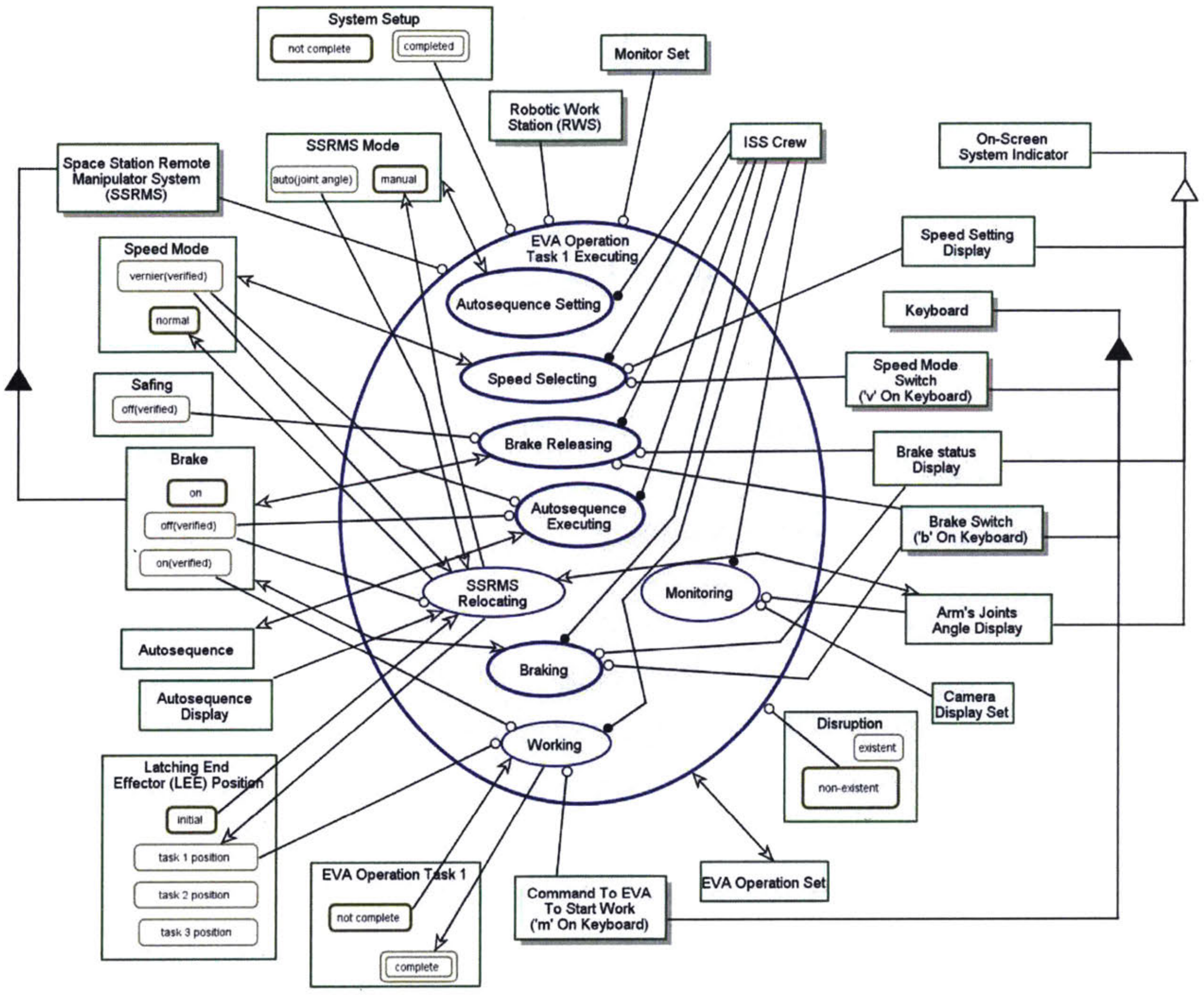


*Figure 35: EVA Operation Task 1 Executing In-Zoomed*

Beside the **SSRMS Relocating** is **Monitoring**. The top of the **SSRMS Relocating** and **Monitoring** ellipses are at the same level, showing that these two subprocesses will happen concurrently. The **ISS Crew** needs to monitor the movement of the **SSRMS** for safety violations (e.g. (a)clearance violation - the SSRMS is too close to a structure, and (b) joint angle limit – the joints reach its maximum range of motion) using **Camera Display Set** and **Arm's Joints Angle Display**. **SSRMS Relocating** affects the **Arm's Joints Angle Display**. The OPM methodology compelled the user to think further about the human role and recognized that there was an additional concurrent **Monitoring** task that had been omitted from the HTA + TTA model. Though the **Monitoring** task was included in the AH analysis, the OPM analysis was performed first. However, the AH analysis alone could have potentially identified the need.

The solid black triangle shows that **Speed Mode Switch, Brake Switch** and **Command To EVA To Start Work** (to emulate verbal command given to astronaut on EVA) are on the **Keyboard**. The **Speed Setting Display, Brake Status Display** and **Arm's Joints Angle Display** are forms of **On-Screen System Indicator.**

**Working** involves the **ISS Crew** performing repairs on the ISS exterior. In the simulation, the 'm' key on the **Keyboard** is used to trigger this activity. **Working** also requires **Brake** to be **on(verified)** to ensure that the **SSRMS** does not move inadvertently.

Although Figure 35 shows all the intended information, the OPD is cluttered and impairs communicating the concepts to others. Therefore, the OPD was simplified by grouping some of the subprocesses together and move some objects to the in-zoomed view instead. Through this effort, the author re-categorized the eight subprocesses in Figure 35 into three subprocesses: 1) **Autosequence Preparing**, 2) **Autosequence Executing** and 3) **EVA Task 1 Proceeding**. The simplified OPD is presented in Figure 36. For example, the simplified version only shows **Autosequence Preparing** and **Autosequence Executing** affects **Speed Mode**. It is not required to indicate the change in states in this OPD; these details should be shown in the in-zoomed.

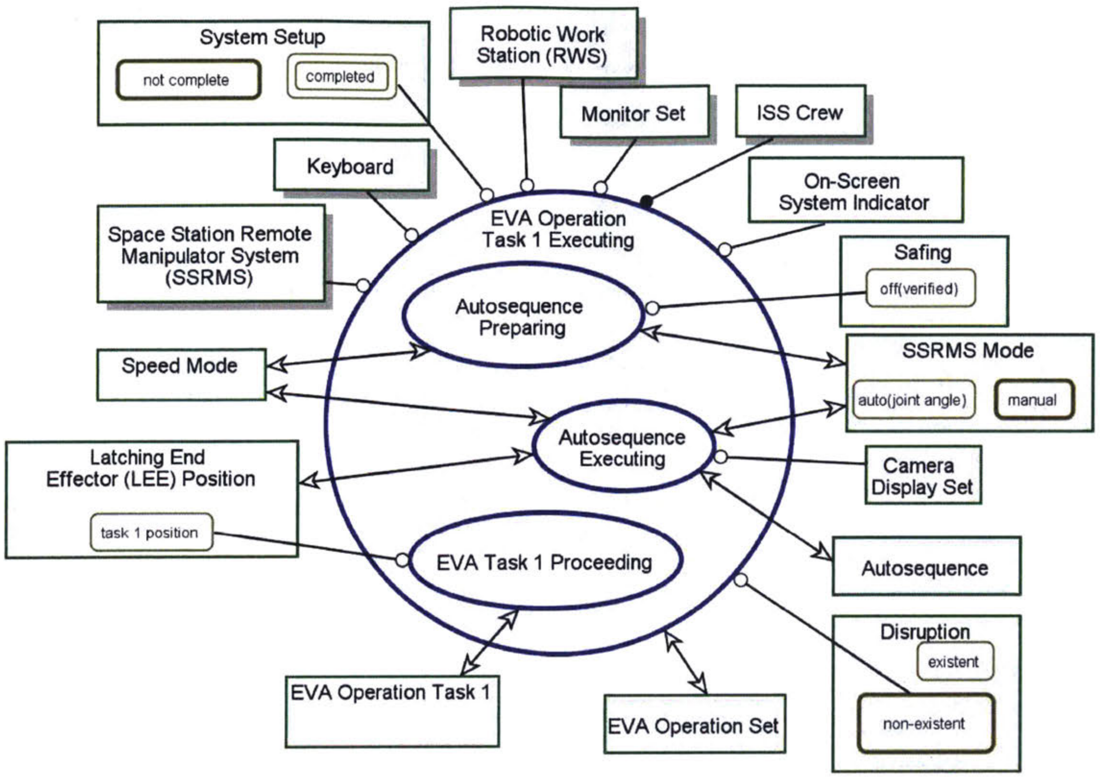


*Figure 36: EVA Operation Task 1 Executing In-zoomed (Improved)*

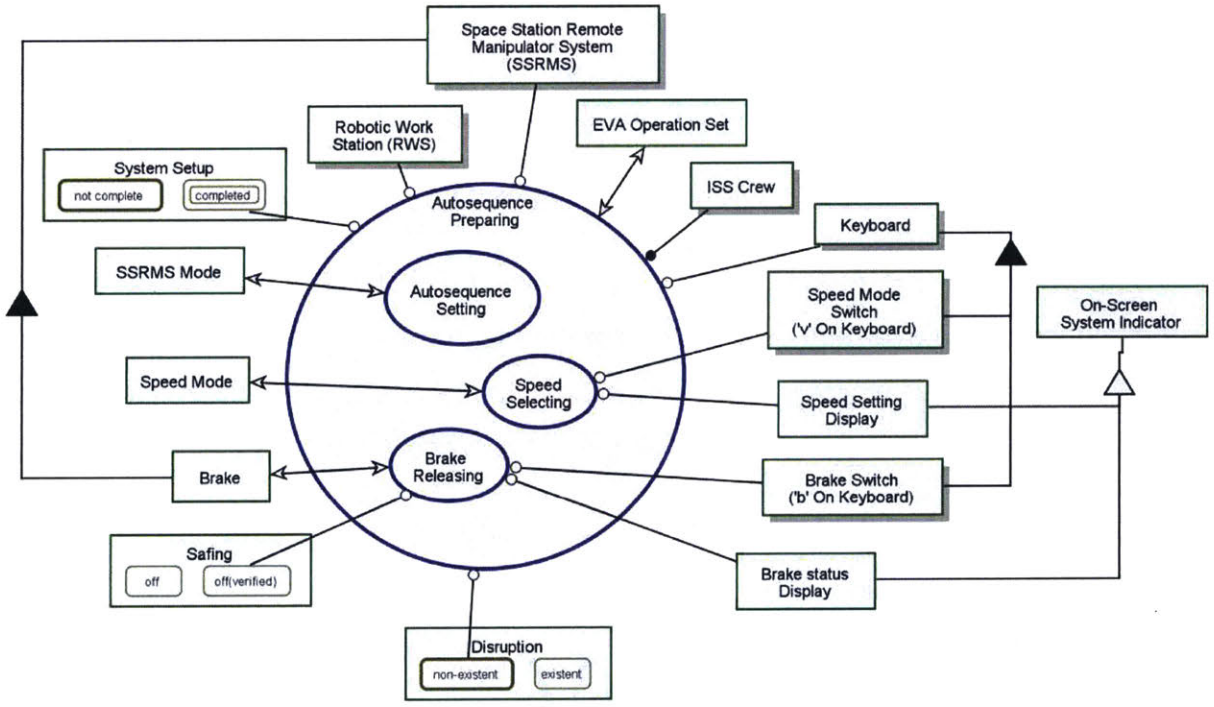


*Figure 37: Autosequence Preparing In-zoomed*

**Autosequence Preparing** grouped subprocesses associating with setting up the automated arm movement as shown in Figure 37. In this in-zoomed, information on the switches and display were added but still keeping the OPD simple and neat

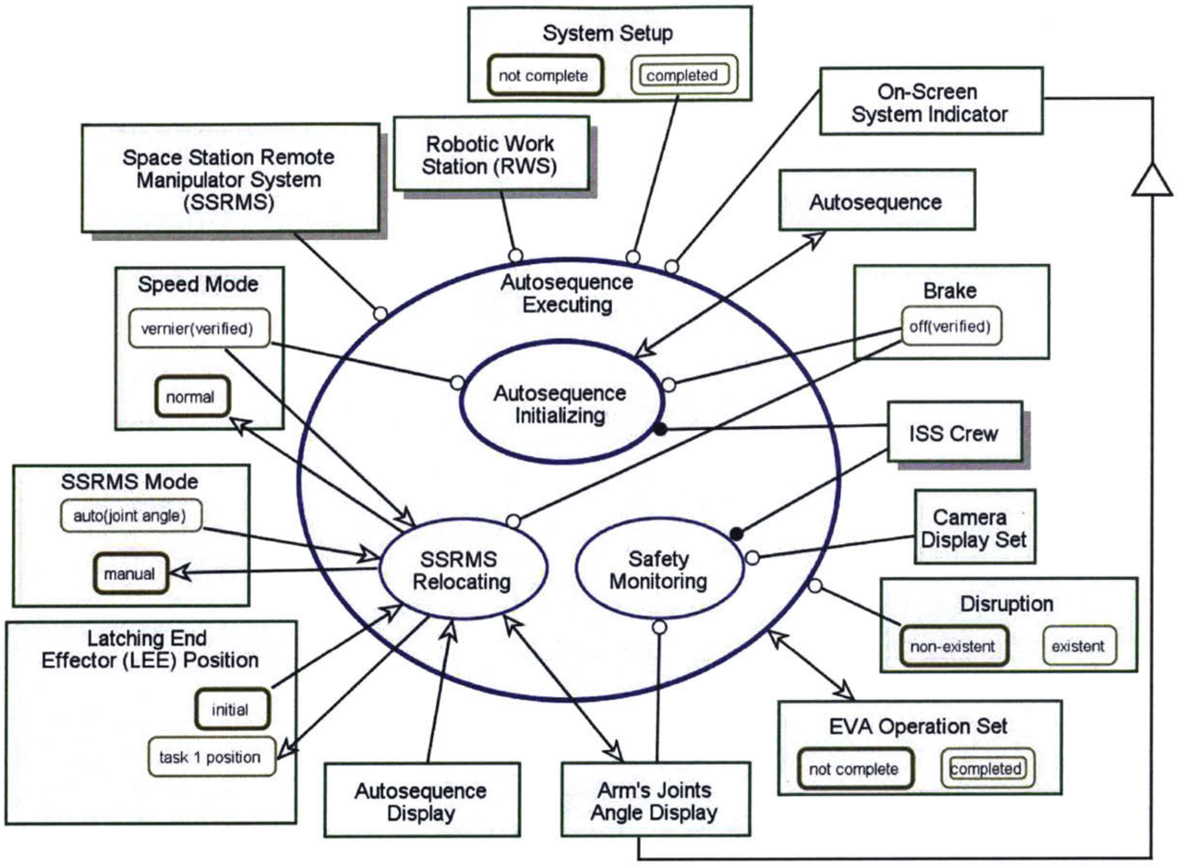


*Figure 38: Autosequence Executing In-zoomed*

Figure 38 shows the **Autosequence Executing** OPD that depicts the execution of the robotic arm movement. The state changes to **Speed Mode**, **SSRMS Mode** and **LEE Position** are explicitly shown in this OPD instead to illustrated how SSRMS Relocating affects the operands (as described earlier paragraphs).

Similarly, Figure 39 shows the subprocesses in **EVA Task 1 Proceeding** and the objects required to support those two subprocesses. All other objects that were not required were excluded.

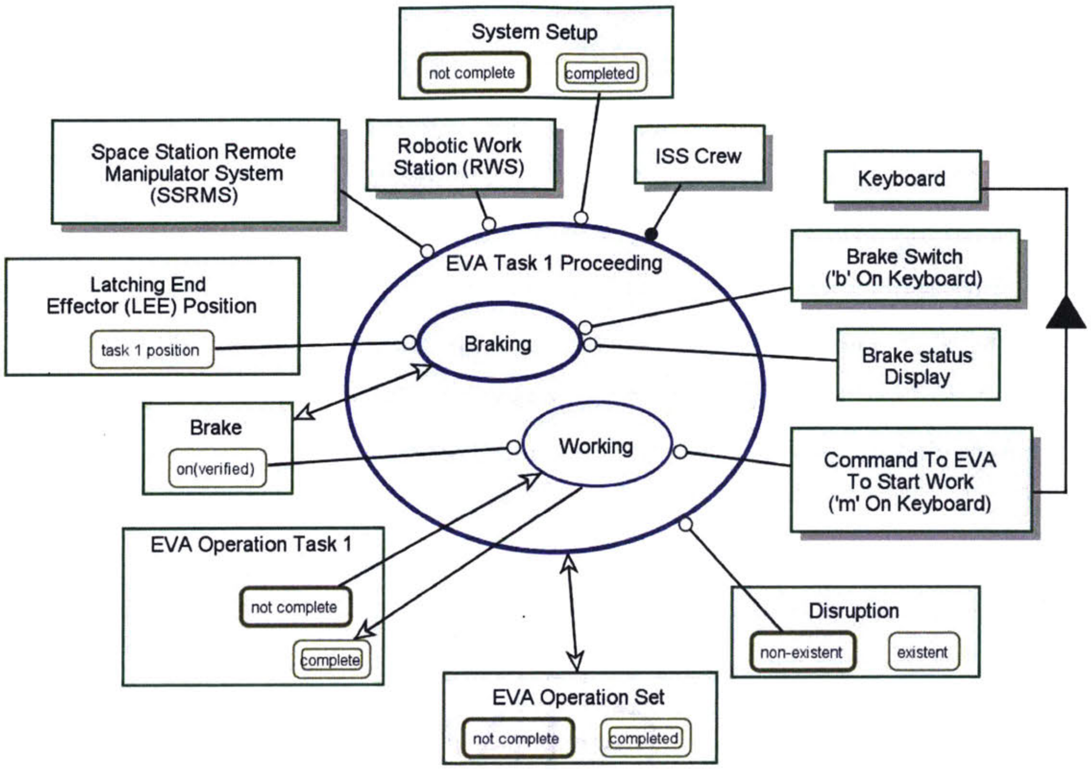


*Figure 39: EVA Task 1 Proceeding In-zoomed*

Figure 36 to Figure 39 covers the same level of information as Figure 35. But by breaking Figure 35 into four separate OPDs, the visual complexity was removed, and a clearer picture of the processes emerged. Figure 38 may still appear slightly busy but is still a significant improvement from Figure 35. Because this OPM considers both human and systems, the number of OPM things can be large. Therefore, it is worthwhile to reiterate the OPDs and simplify the diagram through further groupings to keep the OPD uncluttered and readable as describes in the framework.

#### *4.3.1.3 Autosequence Setting In-Zoomed*

Figure 40 is the in-zoomed of **Autosequence Setting**. **Automode Selecting** is triggered by **Automode Button** which is key 'A' on the **Keyboard** and changes the **SSRMS Mode** from **manual** to **auto**. It also created the **Autosequence Panel.** The **Autosequence Panel** enables the selection of 'joint angle', one of the two

**SSRMS Mode** in auto. The other auto mode is the Frame of Resolution (FOR). **Selecting Joint Angle** produces the **Joint Angle Panel** that is part of the **Autosequence Panel** and changes **SSRMS Mode** to **auto(joint angle)** state. In **auto(joint angle)** state, **ISS Crew** can input the joint angles using the **Keyboard** that changes **Joint Angle Panel** from **blank** to **provided**. Next, the **ISS Crew** needs to verify the **Joint Angle Panel** before **Loading** it to the system. A **Load Button** is available on the **Autosequence Panel** to trigger the **Loading** that creates the **Autosequence** in **hold** state. The **Autosequence Panel** also has a **Confirm Button** that will be used in a later process.

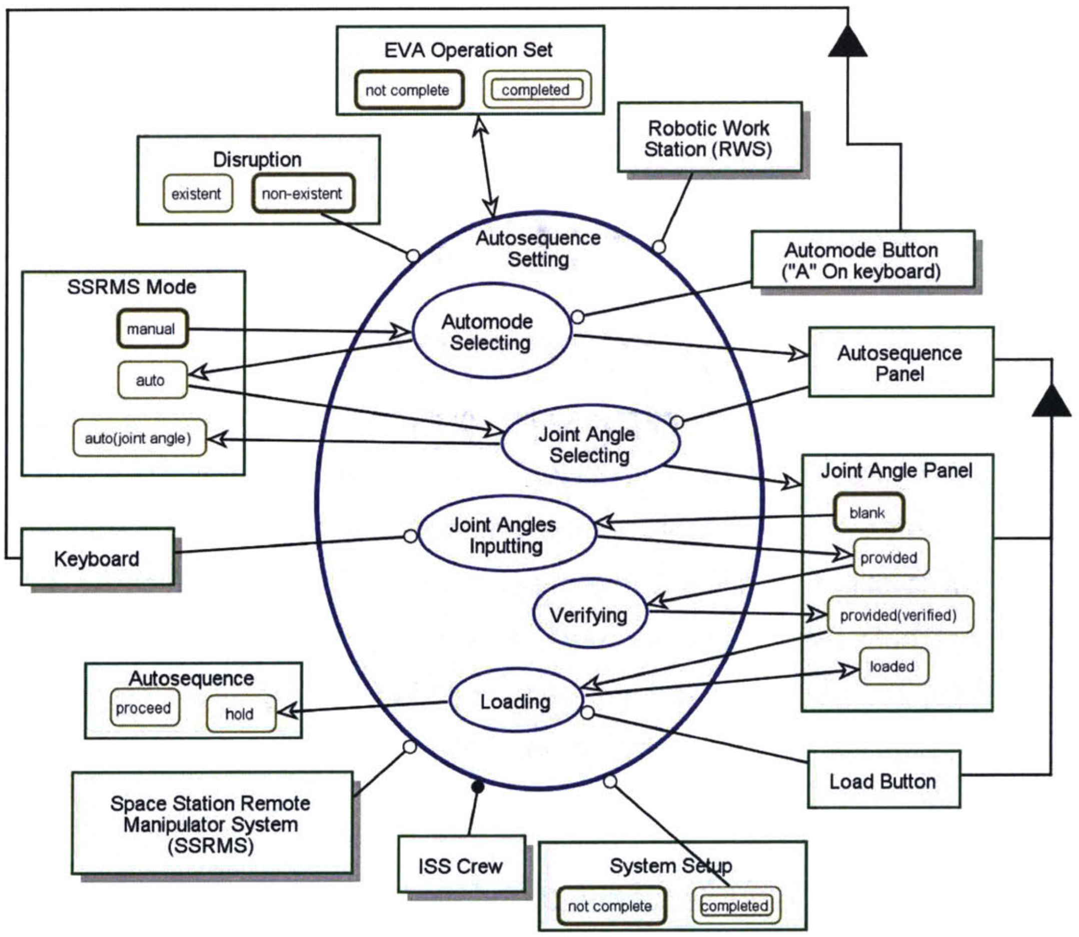


*Figure 40: Autosequence Setting In-Zoomed*

The OPM structure links explicitly define design requirements – in this case, a need to code an **Autosequence Panel** that allows selection of **SSRMS mode**, fields to input joint angles if 'joint angle' mode is selected and button interfaces – **Load Button** and **Confirm Button.** The **Autosequence Panel** is also not part of the **Control Panel.** In contrast, the HTA+TTA framework does not define design requirements. In conceptual design, it is a good practice to reduce ambiguity as much as possible, even if the requirements are implied.

The OPD defines the procedure for **Autosequence Setting** as follows:

[1] Select – Autosequence
[2] Select (SSRMS Mode) – Joint Angle
[3] Input Joint Angles
[4] Verify – Joint Angles
[5] Load Joint Angles

### 4.3.1.3.1 Autosequence Initializing In-Zoomed

After all the **Autosequence Preparing** is completed, the next subprocess is **Autosequence Executing** (see Figure 38). Figure 41 is the in-zoomed of **Autosequence Initializing**. The **ISS Crew** can proceed to confirm 'Autosequence' using the **Confirm Button** and **Autosequence** is changed from **hold** to **proceed**. **Confirming** also requires the following pre-conditions: a) **Safing – off(verified),** b) **Brake -off(verified),** c) **Speed Mode – vernier(verified)**, d) **Joint Angles – loaded**, e) **CEU – initialized(verified),** f) **Display And Control Panel – on(verified)**, g) **Comms – enable(verified),** h) **Failure Detection – enabled(verified),** i) **WHS – download(verified)** and j) either **SSRMS Prime String** or **SSRMS Redundant String** in **operational(verified).**

Note that the graphical representation here reflects an OR relationship between **SSRMS Prime String operational(verified)** and **SSRMS Redundant String operational(verified)** for the **Confirming** subprocess. That means, **SSRMS Prime String** and/or **SSRMS Redundant String** must be **operational(verified)** for the **Confirming** subprocess to occur. One limitation of OPCAT animation is that although "XOR" and "OR" logic can be graphically drawn in OPCAT, OPCAT animation simulate them as if they are "AND" **logic.** So, this type of graphical construct poses a problem when animating this part of the model. In this case, the animation will halt because normally only one of the two SSRMS strings will be **operational(verified)**. Thus, users have to manually intervene and properly set the state of the **SSRMS Redundant String** to be **operational(verified)** to continue the animation.

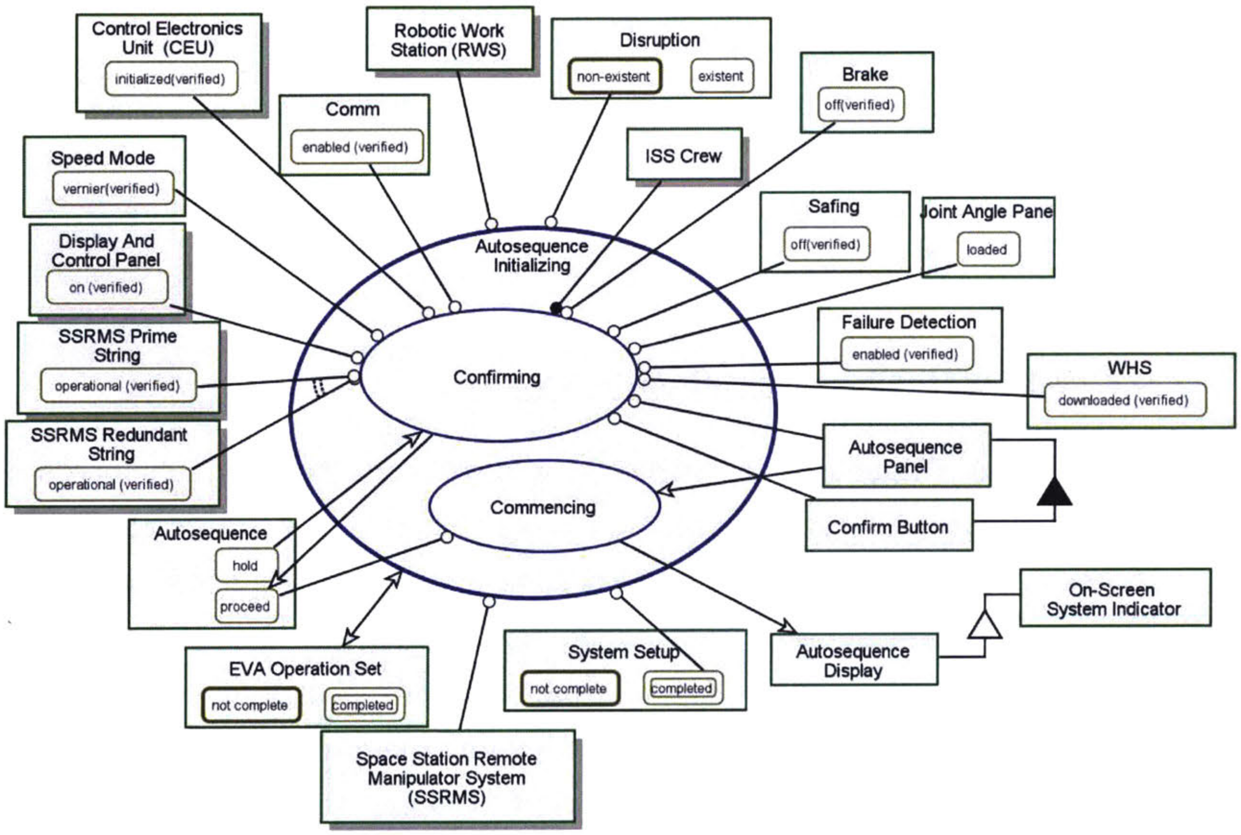


*Figure 41: OPD of Autosequence Executing In-Zoomed*

**Commencing** closes the **Autosequence Panel** and creates an **Autosequence Display** to inform the crew that 'Autosequence' is in-progress.

This OPD reminds the programmer to code the simulation to check the above-mentioned objects states as part of **Initializing**. Such capability is important because we do not want the simulation to allow **Autosequence** to **proceed** when preconditions are not met. This is a closed-loop safety feature to prevent mishaps. It also sets the requirements that the **Autosequence Panel** needs to be removed.

Also, an **Autosequence Display**, a form of **On-Screen System Indicator**, must appear to provide visual cues that the system is performing 'Autosequence'. If this requirement had been ignored, the **Autosequence Panel** would remain active in the animation, reminding the system designer of the need to remove it. (This is an example of the usefulness of OPM animation capability.)

The OPD defines the procedure for Autosequence Executing as follows:

[1] Confirm Autosequence
[2] Verify Autosequence – Commencing

### 4.3.2 *Disruption Handling In-Zoomed*

This section covers the OPD for the **Disruption Handling** process in the model (see Figure 30). In addition to a nominal procedure discussed in section 4.3.1, the OPM includes scenarios for handling system failures analogous to those used today on ISS, but in a fashion completely autonomous from mission control support, as would be typical on an exploration mission far from Earth. System failure requires the ISS Crew to stop the on-going operations and transition to troubleshooting and recovery. The operations can resume after the failure has been rectified and the operator is confident that all systems state have been properly set. Having this capability in the human-in-the-loop robotics simulator enables the study of human performance in non-routine procedures and re-entry to resume operations.

Though many failures are possible, for purposes of the present human in the loop experiments, two failures (**Disruption Kind**) – **communications Loss** and **video error** – were chosen and modelled using OPM. The management of these failures assumed that these particular failures occur during **SSRMS Relocating** i.e. the **SSRMS** is moving under automation and the **ISS Crew** is required to (see Figure 35 or Figure 38) monitor the system and arm movement. Though failures can conceivably occur at anytime, limiting the **Disruption** to a specific portion of the experiment for all subjects was desired for reasons of human experimental design.

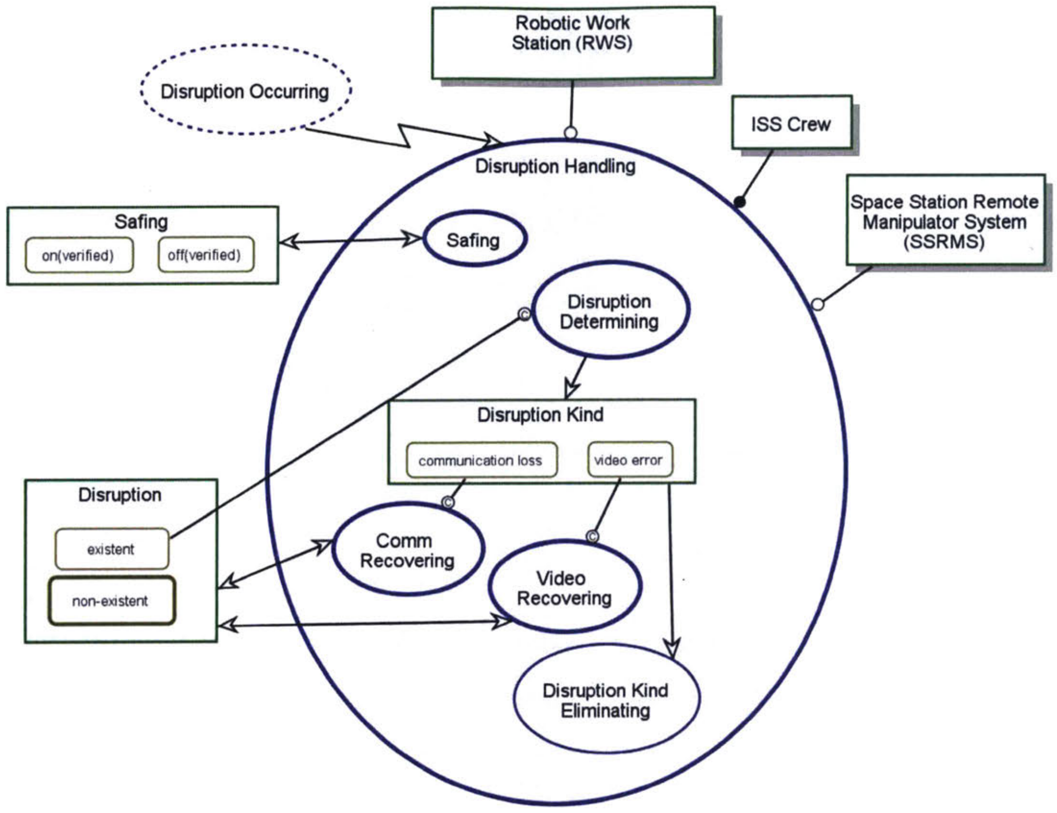


*Figure 42: OPD of Disruption Handling In-Zoomed*

Figure 42 depicts the in-zoomed OPD for **Disruption Handling**. This process is triggered after **Disruption Occurring**. The ISS Crew is expected to handle the **Disruption Handling** process. This OPD shows that ISS Crew needs to 'safe' the system, determine the **Disruption Kind,** and perform the correct recovery procedure. The last process **Disruption Kind Eliminating** consumes (eliminates) the **Disruption Kind** object after either **Comm Recovering** or **Video Recovering** is executed.

#### *4.3.2.1 Safing In-Zoomed in Disruption Handling*

The **Safing** in-zoomed OPD in **Disruption Handling** is presented in Figure 43. **Activating** is triggered by the **Safing Switch** and it changes the state of **Safing**, **Speed Mode**, **SSRMS Mode** and **Brake**. In this subprocess, **Safing** changes from **off(verified)** to **on** and simultaneously, **Speed Mode** is reset to **normal**, **SSRMS Mode** is reset to **manual** and **Brake** is set to **on**. This is part of the safety feature of the 'safing' capability. The OPD explicitly informs the programmers about the relationships between **Safing** and the other objects (**Speed Mode, SSRMS Mode, Brake**).

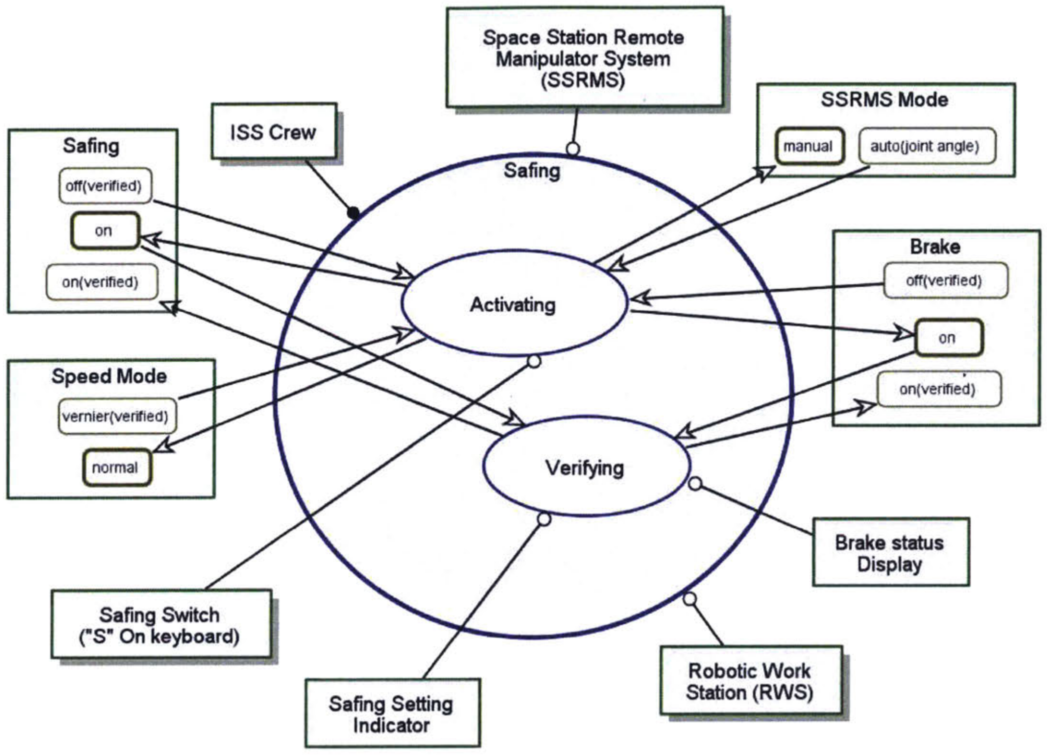


*Figure 43: OPD of Safing In-Zoomed in Disruption Handling*

The OPD defines the procedure for **Safing** as follows:

[1] Set Safing – On
[2] Verify Safing – On
[3] Verify Brake – On

### *4.3.2.2 Disruption Determining In-Zoomed*

**Disruption Determining** is the subprocess where the **ISS Crew** identifies the type of failure i.e. **Disruption Kind**. The **ISS Crew** needs to access the **System Control Panel** that has an **Error Message** section using the **System Panel Switch**. The **existent** of **Error Message** is necessary for **Error Message**

**Checking** so that the **ISS Crew** can diagnose the **Disruption Kind**. See Figure 44 for the OPD. The **Disruption Kind** state will determine which recovery subprocess will be executed. See ***Appendix B*** for recovery subprocesses details.

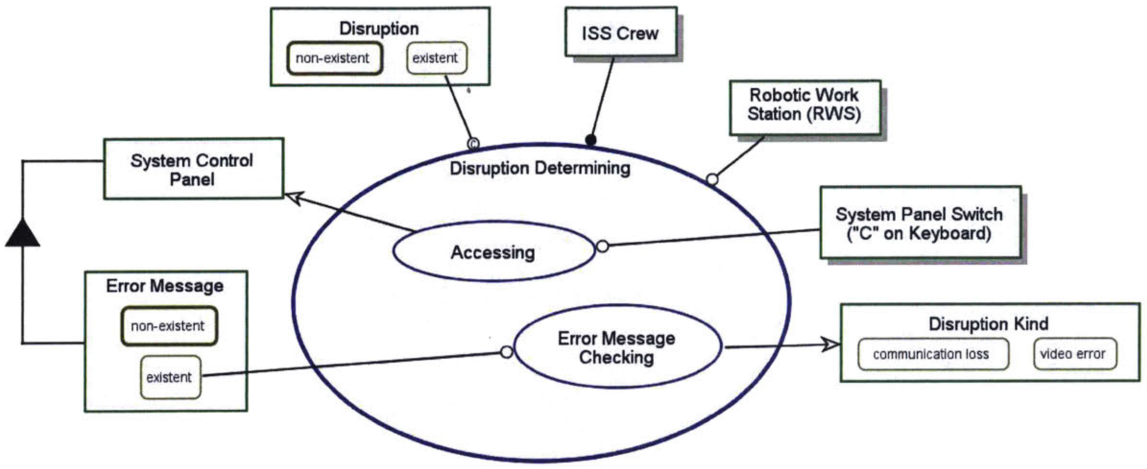


*Figure 44: Disruption Determining In-Zoomed*

### *4.3.3 OPD of Objects Structural Relationship*

So far, the OPDs do not represent the Electronic Procedure (EP) that the crew will refer to during the robotic operations and the integration of the EP with the system. These will be covered in the next section. The OPDs described from para 4.3.1 to 4.3.2.2 explained (1) the procedures for robotic EVA, (2) the logical flow of the electronic procedure and displays, (3) the preconditions and post-conditions of each process and (4) the structural relationships amongst the objects that will be programmed into the MIT-RWSS. Although the objects' structural relationships are dispersed over various OPDs, OPCAT allows the creation of separate views to delineate these relationships.

Figure 45 presents an unfolded view of the structural relationships of the **Mobile Servicing System (MSS).** From this diagram, it clearly shows that the electronic procedure system programmer needs to implement six switches that are triggered using the **Keyboard** (shaded in green).

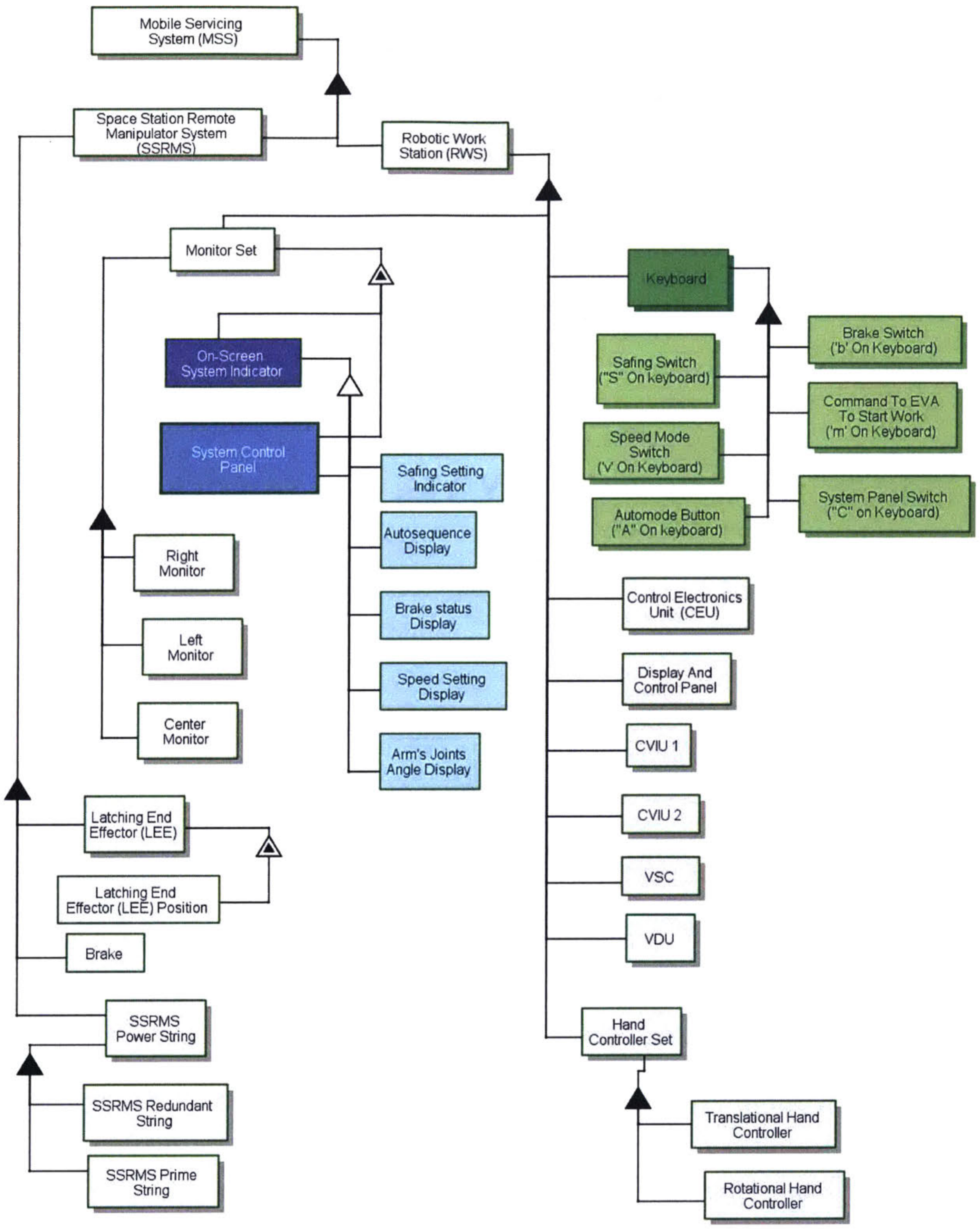

Mobile Servicing System (MSS)
Space Station Remote Manipulator System (SSRMS)
Robotic Work Station (RWS)
Monitor Set
Keyboard
On-Screen System Indicator
System Control Panel
Brake Switch ('b' On Keyboard)
Safing Switch ("S" On keyboard)
Command To EVA To Start Work ('m' On Keyboard)
Speed Mode Switch ('V' On Keyboard)
Automode Button ("A" On keyboard)
System Panel Switch ("C" on Keyboard)
Safing Setting Indicator
Autosequence Display
Brake status Display
Speed Setting Display
Arm's Joints Angle Display
Right Monitor
Left Monitor
Center Monitor
Control Electronics Unit (CEU)
Display And Control Panel
CVIU 1
CVIU 2
VSC
VDU
Latching End Effector (LEE)
Latching End Effector (LEE) Position
Brake
SSRMS Power String
SSRMS Redundant String
SSRMS Prime String
Hand Controller Set
Translational Hand Controller
Rotational Hand Controller

*Figure 45: OPD of MSS Structural Relationship*

The other notable information is the **On-Screen System Indicators** (shaded in blue) that comes in various forms. This reminds the programmer to implement indicators on the monitor to show the 1) **Safing** state, 2) **Autosequence** state, 3) **Brake** state, 4) **Speed Mode** and 5) **Arm's Joint Angles**. Another specialization of **On-Screen Indicators** shown in the OPD is the **System Control Panel.**

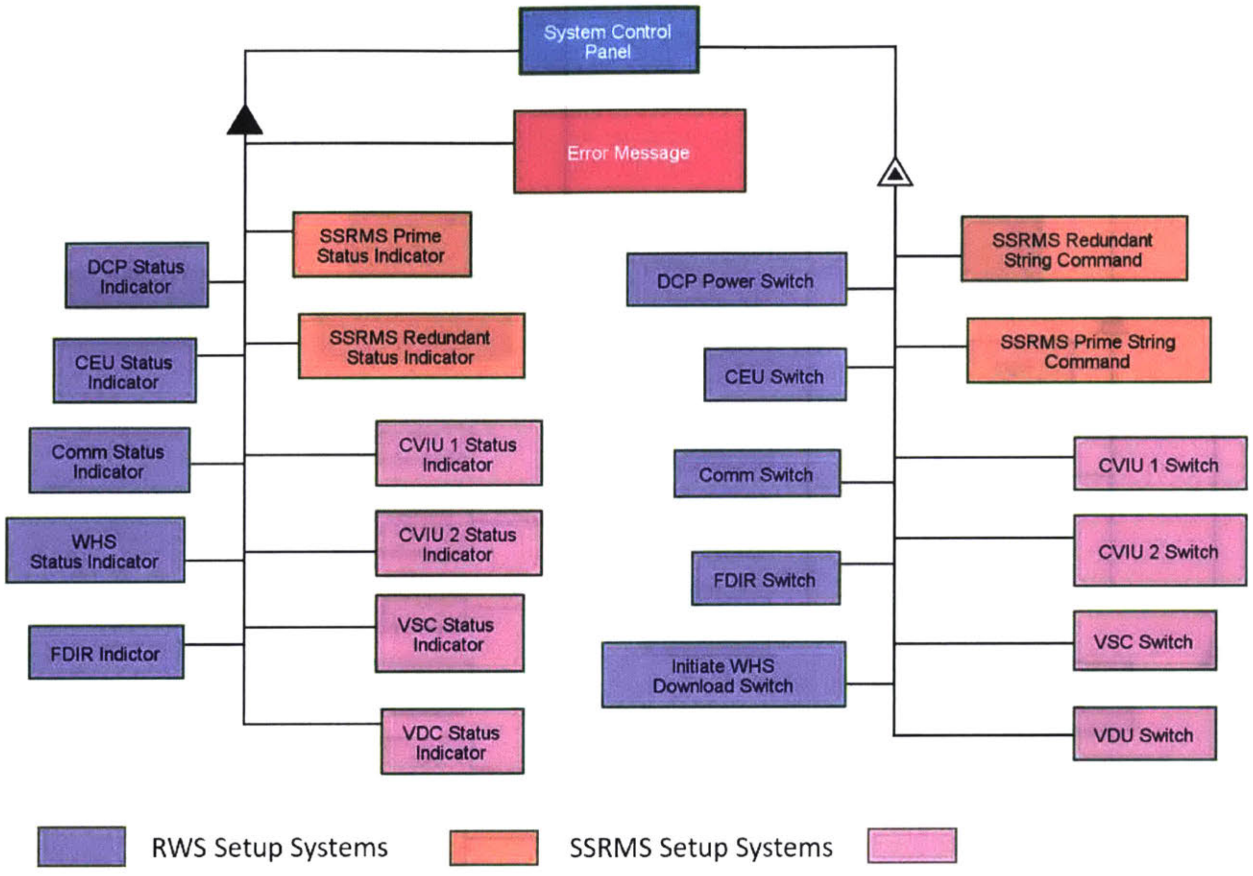


*Figure 46: OPD of System Control Panel Structural Relationships*

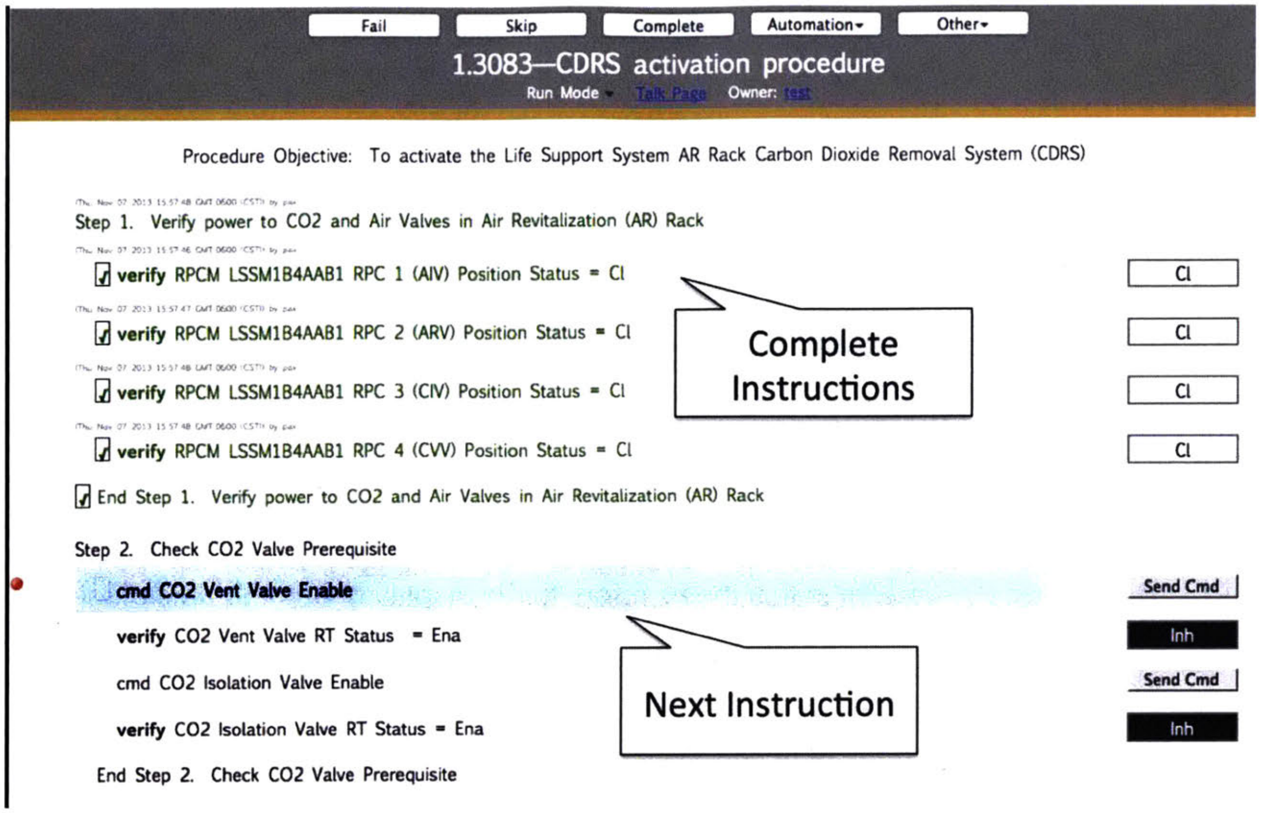
The **System Control Panel** is unfolded to show its structural relationships with other objects as illustrated in Figure 46. It informed that the **System Control Panel** needs to provide subsystems status (as illustrated by the aggregation-participation symbol ▲) and features that allow the operator to change the state of these systems (as shown by the exhibition-characterization symbol ◬). OPCAT allows the users to change the color of OPM objects to distinguish it from the other. Thus, color code is used to provide another dimension of information in OPM. Indicators and switches are categorized to support RWS Setup, SSRMS Setup, Video Setup using purple, orange and pink shading respectively. It implicitly instructs to the programmer to group them accordingly in the **System Control Panel**. The **Error Message** does not belong to any of the category and should be placed by itself within the System Control Panel.

## 4.4 Electronic Procedure and System Integration

After the development of the OPM representation of the robotic EVA electronic procedures in section 4.3, the next step is to consider the integration of Electronic Procedures display into the MIT-RWSS simulation, and how it will be represented in the OPM. **Electronic Procedure** in this thesis refers to the display of automatable electronic procedures on the monitor; so it is a feature of the **Monitor Set**. It is assumed that – unlike on ISS – the electronic procedure system can fully sense and control elements of the MSS, so it can keep track of all the MSS states and disruptions, know what has been done, and prompt the user what to do next.

### *4.4.1 Electronic Procedure Usage Concept*

The fundamental usage principles of automated electronic procedures are similar to those of traditional procedures – in electronic or paper format – that are not integrated. In traditional procedures, the users typically read the instruction in the procedure, execute the task/activity and mark on the procedure or remember that they have accomplished that particular step. The key difference in an automated procedure is that these efforts could be assigned to the computer. In a detailed procedure i.e. "to-do" list, an instruction will be called out to change a system configuration or setting and a separate instruction will ask the user to verify that particular system or setting is in the correct state. This concept was demonstrated by Schreckenghost et al. [32]. An example is shown in Figure 47 – Step 2. Every step has a checkbox that is ticked when that step is completed. The ticking of the checkboxes can manual or automated.



1.3083—CDRS activation procedure

Run Mode ▾ Talk Page Owner: test

Procedure Objective: To activate the Life Support System AR Rack Carbon Dioxide Removal System (CDRS)

(Thu Nov 07 2013 15:57:48 GMT 0600 (CST)) by pax

Step 1. Verify power to CO2 and Air Valves in Air Revitalization (AR) Rack

(Thu Nov 07 2013 15:57:46 GMT 0600 (CST)) by pax

✓ **verify** RPCM LSSM1B4AAB1 RPC 1 (AIV) Position Status = Cl — Cl

(Thu Nov 07 2013 15:57:47 GMT 0600 (CST)) by pax

✓ **verify** RPCM LSSM1B4AAB1 RPC 2 (ARV) Position Status = Cl — Cl

(Thu Nov 07 2013 15:57:48 GMT 0600 (CST)) by pax

✓ **verify** RPCM LSSM1B4AAB1 RPC 3 (CIV) Position Status = Cl — Cl

(Thu Nov 07 2013 15:57:48 GMT 0600 (CST)) by pax

✓ **verify** RPCM LSSM1B4AAB1 RPC 4 (CVV) Position Status = Cl — Cl

✓ End Step 1. Verify power to CO2 and Air Valves in Air Revitalization (AR) Rack

Step 2. Check CO2 Valve Prerequisite

**cmd CO2 Vent Valve Enable** — Send Cmd

**verify** CO2 Vent Valve RT Status = Ena — Inh

cmd CO2 Isolation Valve Enable — Send Cmd

**verify** CO2 Isolation Valve RT Status = Ena — Inh

End Step 2. Check CO2 Valve Prerequisite

Figure 47: Electronic Procedure Example from Schreckenghost et al.

Based on the previous discussion, each steps in any procedure can be catergorized as the follow proceses (independent of whether the human or automation accomplishes it):

1. **Configuration Changing**: Read the instruction to change a system configuration or setting (e.g. Set System A – On), and then send the command to initiate that change (e.g. flipping switch or pushing buttons). (No execution verification is required in this category)

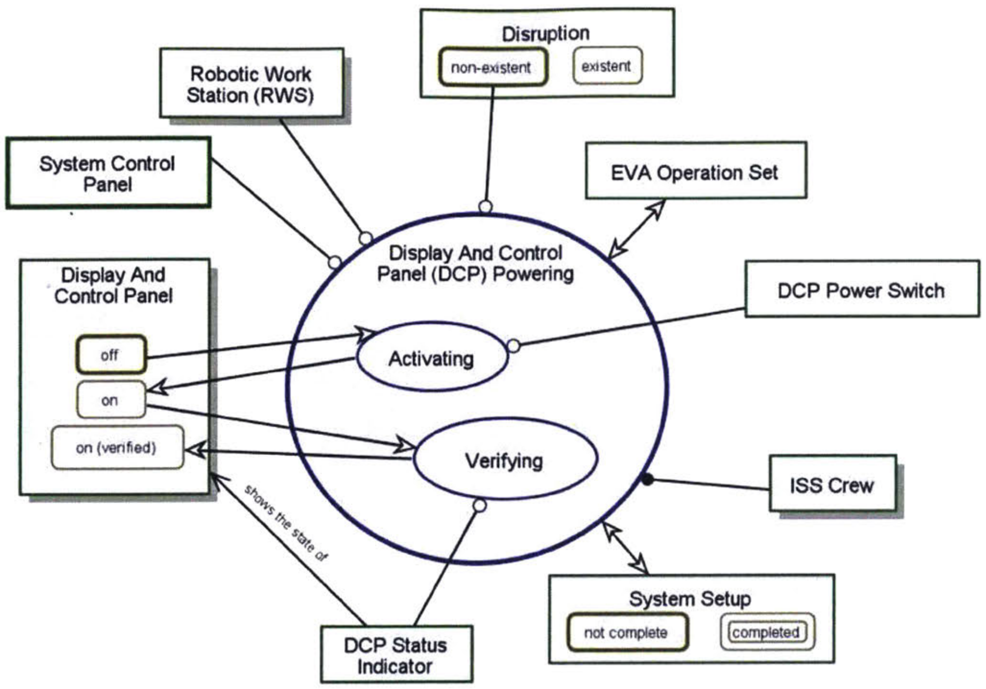
2. **Electronic Procedure Configuration Change Instruction Marking**: After a configuration change (1), mark the procedure instruction by putting a tick in a checkbox beside to indicate the configuration command has been given. This helps the user to keep track of procedure progress. An electronic procedure always requires this step.

3. Configuration Changed Verifying: If the instruction requires it (e.g. Verify System A – On), verify that the system actually achieved the correct state after the configuration change in (1) i.e. compare the actual system's state with the instructed state. Usually, when a configuration change is made (1), verification is also appropriate.

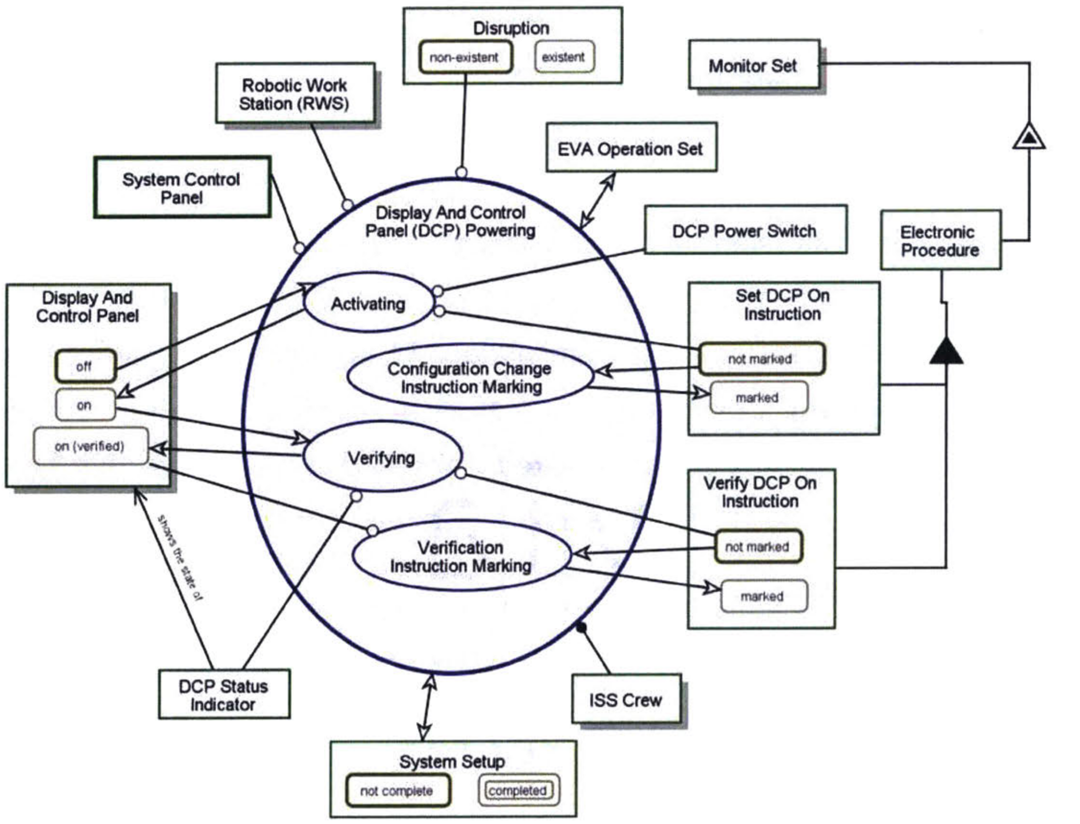
4. Electronic Procedure Verification Instruction Marking**:** After the verification, mark the procedure instruction in (3) by putting a tick in the checkbox beside the verification instruction. An electronic procedure always requires this step.

The leaf OPDs – the OPDs at the lowest level in the OPD hierarchy – in the original OPM were modified to reflect these four subprocess categories as shown in Figure 48. Figure 48(b) depicts the modification made to the Display And Control Panel Powering OPD with respect to the original (Figure 48(a)). Two important subprocesses to illustrate categories (2) and (4) discussed above were added – Electronic Procedure Configuration Change Instruction Marking and Electronic Procedure Verification Instruction Marking. These subprocesses describe marking (put a tick) the checkbox beside each instruction on the Electronic Procedure after that step is accomplished. Therefore, to support these two subprocesses, Set DCP On Instruction and Verify DCP On Instruction with states (a) **not marked** and (b) **marked** were added. These instructions are part of the Electronic Procedure. Configuration Change Instruction Marking changes Set DCP On Instruction state from **not marked** to **marked** and Verification Instruction Marking changes Verify DCP On Instruction from **not marked** to **marked.**

In the original OPD (Figure 48(a)), only the execution of the steps are presented. Based on steps (1) and (3) discussed above, an instruction is also required for the respective subprocesses, Activating and Verifying to occur. This is represented in the OPD by indicating Set DCP On Instruction in **not marked** is required for Activating and Verify DCP On Instruction in **not marked** is required for Verifying.



(a) DCP Powering (Original)



(b) DCP Powering with Electronic Procedure

*Figure 48: DCP Powering Comparison with Electronic Procedure Integrated*

By modifying all the leaf OPDs, a complete OPM conceptual model representing the architecture and logic of how the simulator with electronic procedure should operate was produced. This could not have been done using the three task analysis techniques – HTA, TTA and AH. When the **Electronic Procedure** unfolds, it contains all the procedural steps developed, each with **marked** and **not marked** states. The only object included in the upper level OPDs is the **Electronic Procedure** connected to the outer ellipse with an effect link.

### *4.4.2 Automation of Electronic Procedures*

For research purposes, the author wanted to allow the researchers to select which of the four subprocess categories would be automated in any particular experiment; those subprocesses not automated would be performed by human. The automated version of the four subprocesses were defined as follows:

1. *Auto-cmd (command)* – This is automated **Configuration Changing.** The automation reads the instructions to determine which object (e.g. DCP) and which state to be changed (e.g. On or Off), and sends a command to change the object states.

2. *Auto-mark(cmd)* – This is automated **Electronic Procedure Configuration Change Instruction Marking.** The automation recognizes that a command has been sent to change the state of an object and marks that instruction checkbox in the **Electronic Procedure.** (Note: It does not verify the state of the object has actually changed.)

3. *Auto-verify* – This is automated **Configuration Changed Verifying.** The automation reads the instruction to determine the object (e.g. DCP) and correct object's **state** (e.g. On) and verifies the object has achieved the correct state. The automation informs the operator visually if the verification reveals a discrepancy.

4. *Auto-mark(verify)* – This is automated **Electronic Procedure Verification Instruction Marking.** The automation marks the instruction checkbox in (3) if there is no discrepancy.

*Table 6: Automation Options*

| | Auto-cmd | Auto-mark(cmd) | Auto-verify | Auto-mark(verify) |
|---|---|---|---|---|
| ***Option 0*** | No | No | No | No |
| ***Option 1*** | No | Yes | No | No |
| ***Option 2(0)***[5] | No | No | Yes | No |
| ***Option 2(1)***[5] | No | No | Yes | Yes |
| ***Option 3(0)*** [5] | No | Yes | Yes | No |
| ***Option 3 (1)*** [5] | No | Yes | Yes | Yes |
| ***Option 4(0)*** [5] | Yes | No | Yes | No |
| ***Option 4(1)*** [5] | Yes | No | Yes | Yes |
| ***Option 5*** | Yes | No | No | No |
| ***Option 6*** | Yes | Yes | No | No |
| ***Option 7(0)*** [5] | Yes | Yes | Yes | No |
| ***Option 7(1)*** [5] | Yes | Yes | Yes | Yes |

In the automated electronic procedure system developed by Schreckenghost et al. [32], marking was always automated. However, as noted the author provided the capability to activate automation of the four subprocesses separately for experimental purposes. For example, in Option 0 (see Table 6), the human performs all the subprocesses, without any automation. In Option 1, the human does the

[5] 0 represents the automation will not mark the verification instruction and 1 represents the automation will mark the verification instruction. This way of representation is because Auto-mark(verify) could only happen if Auto-verify is 'Yes'.

Configuration Changing but the automation does the Electronic Procedure Configuration Change Instruction Marking. The human performs the Configuration Changed Verifying and Electronic Procedure Verification Instruction Marking if required. In Option 2(0), the human performs Configuration Changing and Electronic Procedure Configuration Change Instruction Marking, but the automation performs Configuration Changed Verifying. The human does the Electronic Procedure Verification Instruction Marking (as designed by (0) in Option 2(0)). All twelve possible options are shown in Table 6.

#### *4.4.2.1 Generic OPM Describing Automation Options*

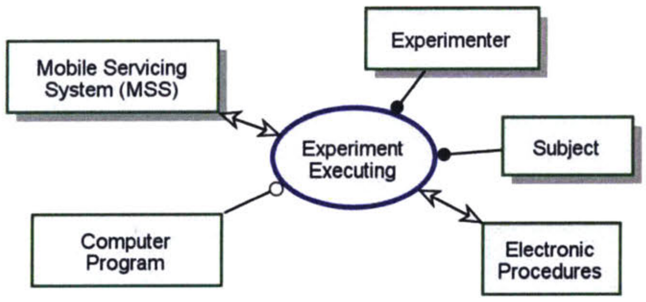


*Figure 49: System Diagram of Automation Options OPM*

To illustrate the different automation options, they were described in a generic OPM model – separate from the OPM discussed earlier. In Figure 49, Experiment Executing is a top-level system diagram describing the entire experiment. Conceptually, a dozen OPDs, each describing EVA Operations Executing using a different automation option would be within it. The System Diagram that shows the Experiment Executing affects MSS and Electronic Procedure and the enablers are Experimenter, Subject (human) and Computer Program (automation).

An in-zoom OPD of Experiment Executing (see Figure 50) shows the generic automation concept. The Experimenter perform Automation Selecting to set Automate Option in one of the twelve states. The selected state determines how the Subject will execute the procedure. Other than Option 0 Procedure

**Executing** and **Option 7(1) Procedure Executing**, all remaining options and subprocesses require both the **Subject** and **Computer Program** (automation). Not all the options from Table 6 are included for simplicity.

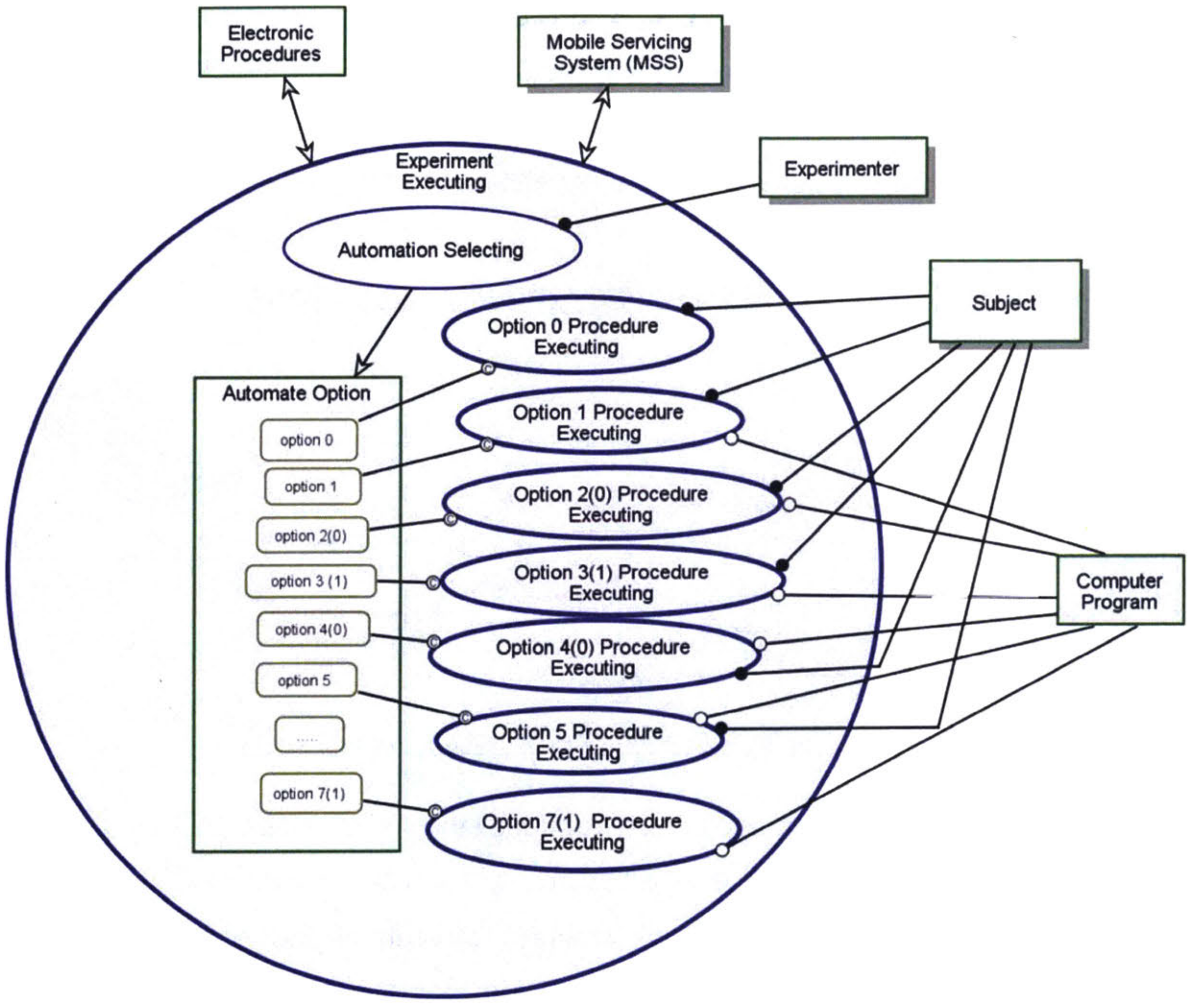

Electronic Procedures
Mobile Servicing System (MSS)
Experiment Executing
Experimenter
Automation Selecting
Subject
Option 0 Procedure Executing
Automate Option
option 0
Option 1 Procedure Executing
option 1
Option 2(0) Procedure Executing
option 2(0)
option 3 (1)
Option 3(1) Procedure Executing
Computer Program
option 4(0)
Option 4(0) Procedure Executing
option 5
Option 5 Procedure Executing
.....
option 7(1)
Option 7(1) Procedure Executing

*Figure 50: OPD of Executing Experiment In-Zoomed*

Figure 51 is the in-zoomed of **Option 2(0) Procedure Executing** and the subprocess is triggered if **Automate Option** is **option 2(0)**. This OPD provides a pictorial view of which enabler i.e. **Subject** (human) or **Computer Program** (automation) is responsible for each of the subprocess.

This **Experiment Executing** OPM model demonstrates how OPM can specify an engineering concept in a concrete but generic way; clearly prescribing how each automation option should be implemented in the simulation.

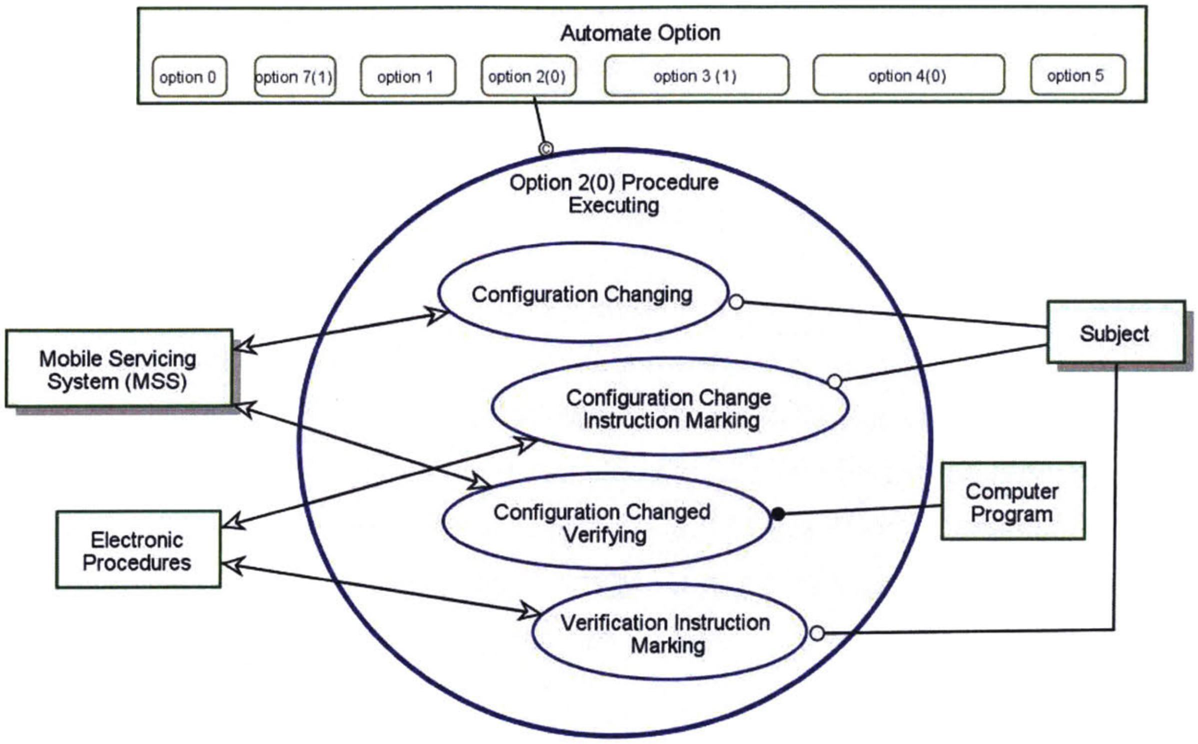

Automate Option
option 0
option 7(1)
option 1
option 2(0)
option 3 (1)
option 4(0)
option 5
Option 2(0) Procedure Executing
Configuration Changing
Configuration Change Instruction Marking
Configuration Changed Verifying
Verification Instruction Marking
Mobile Servicing System (MSS)
Electronic Procedures
Subject
Computer Program

*Figure 51: OPD of Performing Procedure Option 2*

## 4.5 Discussion

In this chapter, it was shown that OPM can represent the information resulting from traditional task analysis and incorporates other important information related to relationships between objects and processes in ways that make the OPM model logically consistent and computable. OPM analysis also provided a method for representing the conceptual framework required that programmers could refer to when developing the code. Though OPM has been around for several years, the use of this methodology for the purpose of hierarchical task analysis and human-machine interface design is novel. The advantages and disadvantages of OPM as compared to task analysis are discussed in the next chapter.

# 5 TASK ANALYSIS (TA) AND OBJECT PROCESS METHDOLOGY (OPM) DISCUSSION

## 5.1 Introduction

Chapters 3 and 4 described several types of task analysis and OPM modelling respectively and how they can be applied. In this chapter, the similarities of these two techniques, and the advantages and disadvantages of OPM relative to HTA+TTA and AH will be discussed.

## 5.2 Similarities Between HTA + TTA and OPM

Both HTA and OPM analysis take a hierarchical approach: HTA starts with the overall goal and decomposes it into sub-goals and tasks. Similarly, OPM begins at an abstract level that describe the overall system function or objective and refining it through in-zooming into the subprocesses. The hierarchical arrangement enables the designers to reduce the apparent complexity of the system or tasks. It was noted that hierarchical arrangements of in-zoomed OPDs in OPM and decomposition of sub-goals in HTA are relatively similar as shown in the Figure 52 example.

The initial guidelines in the application of both techniques are congruent: (1) Define the functions/tasks to be modeled and (2) collect data to understand the goals/tasks/processes involved, and the human, physical and informatical objects. This is, however, not unanticipated as it provides the basis of a systematic analysis in any field of study. The next three steps of HTA, (3) Determine the Overall Goal of Task, (4) Determine Task Sub-goals and (5) Sub-goal Decomposition resemble the refinement of processes in OPM through in-zooming. Though the concept is similar, the details within each level differ significantly; OPM represents much more information.

While TTA and OPM describe the events/cues, interfaces and feedback displays, there is a subtle difference. In the TTA model, the level of detail is a user's choice – the TTA framework does not compel the user to fully define system interfaces and/or events. In comparison, in OPM, the user is compelled to define the objects (operands and instruments) and the changes to the objects' states associated to the processes, and link them in a logically consistent manner.

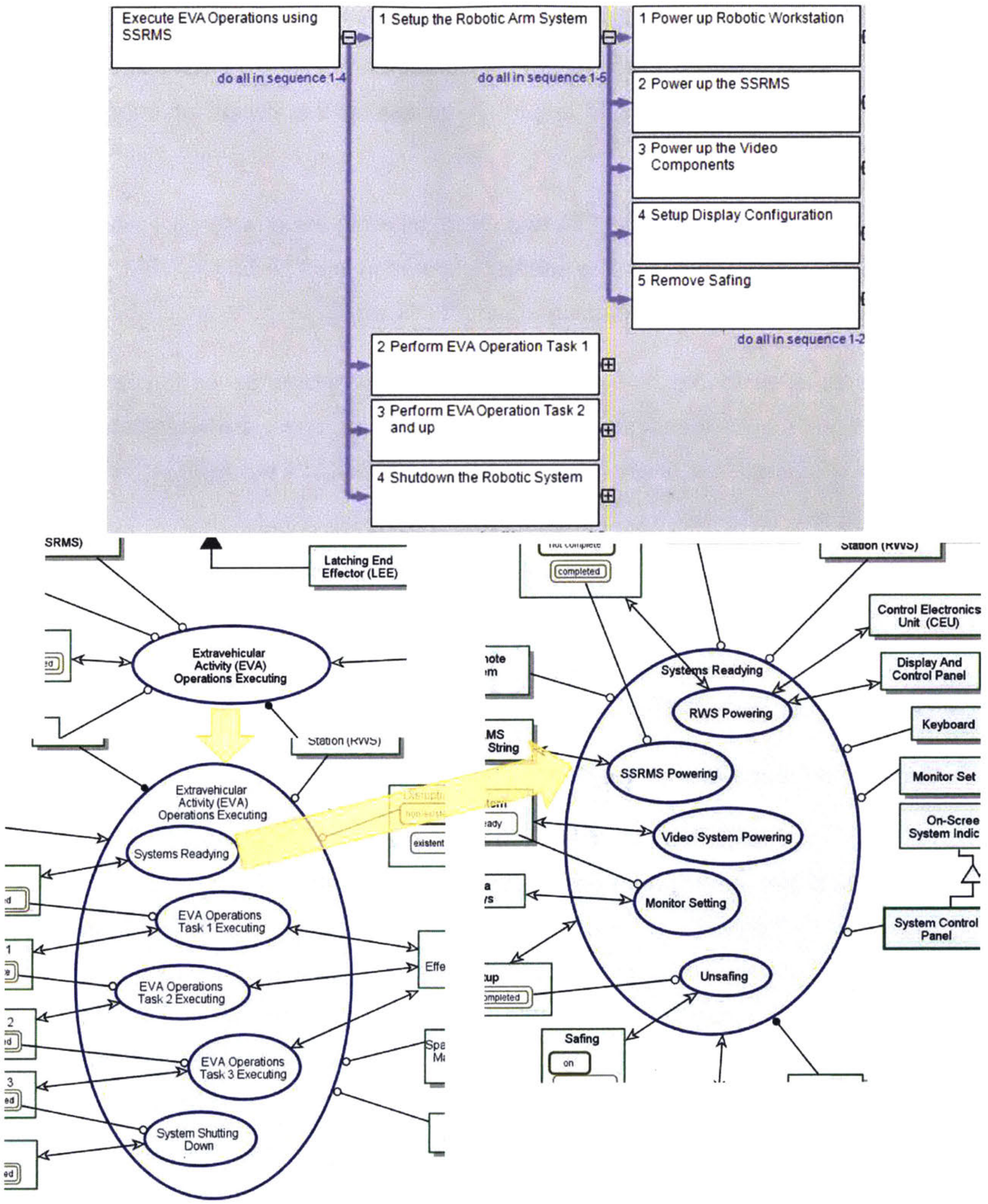

Execute EVA Operations using SSRMS
do all in sequence 1-4
1 Setup the Robotic Arm System
do all in sequence 1-5
1 Power up Robotic Workstation
2 Power up the SSRMS
3 Power up the Video Components
4 Setup Display Configuration
5 Remove Safing
do all in sequence 1-2
2 Perform EVA Operation Task 1
3 Perform EVA Operation Task 2 and up
4 Shutdown the Robotic System
Latching End Effector (LEE)
Extravehicular Activity (EVA) Operations Executing
Systems Readying
EVA Operations Task 1 Executing
EVA Operations Task 2 Executing
EVA Operations Task 3 Executing
System Shutting Down
completed
Control Electronics Unit (CEU)
Display And Control Panel
Keyboard
Monitor Set
On-Scree System Indic
System Control Panel
RWS Powering
SSRMS Powering
Video System Powering
Monitor Setting
Unsafing
Safing
on
existent

Figure 52: HTA and OPM Hierarchical Comparison Example

Both TTA and OPM model allow the user to determine what processes (activities) are associated with any objects (e.g. switch or indicator). This is particularly easily done in OPM by unfolding the object of interest and then using the option 'Complete Links\Add Missing Things'. In TTA, the process is more cumbersome since the user has to search the database using Excel. This is helpful in assessing the design of interfaces and feedback displays.

Human is represented in both the TTA and OPM, allowing the enabler(s) – either human or computer – to be determined. The enabler is stated in the TTA and in OPM, processes requiring human are represented using an agent link.

The events or cues to the human that inform a task should be performed are also represented with both techniques. In OPM, events or cues are represented using instruments i.e. conditional or event links that join an object to a process. In TTA, they are presented as text in a column.

The remaining sections will cover the advantages and disadvantages of OPM over HTA+TTA.

## 5.3 Advantages of OPM over TA (HTA and TTA)

### *5.3.1 Presentation of the OPM vs HTA and TTA*

HTA has two presentation options (refer to section 3.2). Graphical format i.e. hierarchical diagram is preferred for visualization. But the HTA graphical format (see Figure 18 to Figure 21 or Figure 53) does not contain the information found in the TTA. For a more complete picture of the entire system, the HTA+TTA model is preferred. As illustrated in the example in Table 7, the presentation of HTA+TTA is in a hierarchical list format similar to a spreadsheet. Also, tasks descriptions in HTA combine the object (system affected e.g. DCP) and process (e.g. turn on) of the OPM, and the supporting systems (e.g. switch) require to completed the task/activity and details are spread across multiple columns. This affects the user's ability to visualize the system structure and understand the logical flow of the activities.

The OPM contains a more logically complete set of information (i.e. processes, objects, states, plan, systems' relationships, pre-conditions and post-conditions) than the HTA hierarchical diagram and/or HTA + TTA at all levels. The OPM provides two semantically equivalent presentations – graphical and textual. This information is explicitly represented by symbols in the OPD, clearly depicting the OPM user's concept. Although at first sight, OPDs may appear complex, they can be easily understood once the reader learns the symbols definitions. For those not familiar with the symbols or if the users are uncertain about their choice of the graphical elements, they can crosscheck the corresponding OPL to ascertain that their choice reflects their intention as shown in Figure 55. Though it is helpful for OPM practitioners to check whether the symbols drawn make sense, using OPL to present a complex model is inappropriate.

Figure 53, Table 7 and Figure 54 represent DCP Powering in HTA (hierarchical diagram), HTA+TTA and OPD format respectively. The contrast between the three models is apparent. OPM allows the reader to visualize (1) the changes in state of objects after each process, (2) all the supporting instruments (objects) required, (3) processes and objects relationships (structural and procedural), and (4) how the activity (e.g. DCP Powering) affects the overall goal. On the contrary, both the HTA and TTA format shows less information. When there are many columns, the TTA becomes difficult for the reader to comprehend.

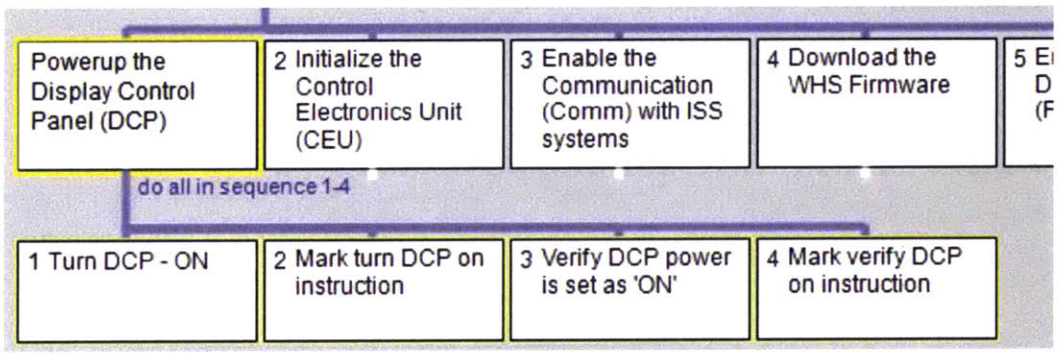

Powerup the Display Control Panel (DCP)
2 Initialize the Control Electronics Unit (CEU)
3 Enable the Communication (Comm) with ISS systems
4 Download the WHS Firmware
5 E D (F
do all in sequence 1-4
1 Turn DCP - ON
2 Mark turn DCP on instruction
3 Verify DCP power is set as 'ON'
4 Mark verify DCP on instruction

*Figure 53: HTA (Hierarchical Diagram) – DCP Powering*

*Table 7: HTA+TTA of DCP Powering*

| No | Task | Initiating cue/event | Enabler | Required Action | Control Used | Feed back | Display Type | Display Info needed |
|---|---|---|---|---|---|---|---|---|
| ***EVA Operations using SSRMS HTA*** | | | | | | | | |
| 1.1.1.1 | Turn DCP - ON | Procedure Instruction | Human | Change System State | Electronic Switch | | | |
| 1.1.1.2 | Mark Change Instruction | Previous Task done | Human | Check Instruction | | | | |
| 1.1.1.3 | Verify DCP power is set as 'ON' | Procedure Instruction | Human | Verification | | ON | on-screen indicator | DCP Status |
| 1.1.1.4 | Mark Verification Instruction | Previous Task done | Human | Check Instruction | | | | |

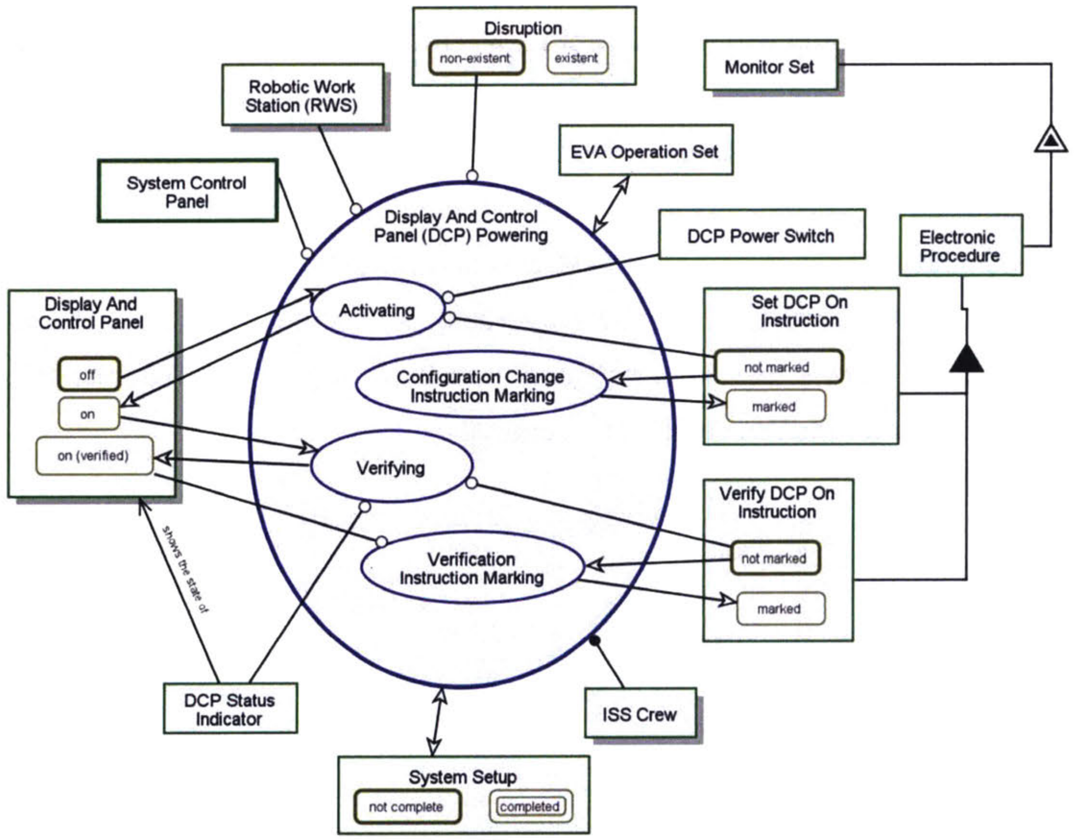


*Figure 54: OPD of DCP Powering*

ISS Crew is physical.
ISS Crew handles Display And Control Panel (DCP) Powering.
Robotic Work Station (RWS) is physical.
System Setup can be not complete or completed.
not complete is initial.
completed is final.
Display And Control Panel is physical.
Display And Control Panel can be off by defaulton, or on (verified).
off is initial.
Monitor Set is physical.
Monitor Set exhibits Electronic Procedure.
Electronic Procedure consists of Set DCP On Instruction and Verify DCP On Instruction.
Set DCP On Instruction can be not marked or marked.
not marked is initial.
Verify DCP On Instruction can be not marked or marked.
not marked is initial.
Disruption can be non-existent by default or existent.
non-existent is initial.
DCP Status Indicator shows the state of Display And Control Panel.
Display And Control Panel (DCP) Powering consists of Activating, Verifying, Configuration Change Instruction Marking, and Verification Instruction Marking.
Display And Control Panel (DCP) Powering requires Robotic Work Station (RWS), non-existent Disruption, and System Control Panel.
Display And Control Panel (DCP) Powering affects EVA Operation Set and System Setup.
Display And Control Panel (DCP) Powering zooms into Activating, Configuration Change Instruction Marking, Verifying, and Verification Instruction Marking.
Activating requires not marked Set DCP On Instruction and DCP Power Switch.
Activating changes Display And Control Panel from off to on.
Configuration Change Instruction Marking changes Set DCP On Instruction from not marked to marked.
Verifying requires not marked Verify DCP On Instruction and DCP Status Indicator.
Verifying changes Display And Control Panel from on to on (verified).
Verification Instruction Marking requires on (verified) Display And Control Panel.
Verification Instruction Marking changes Verify DCP On Instruction from not marked to marked.

Figure 55: OPL of DCP Powering

### 5.3.2 Animation (Simulation) Capability

The OPM animation capability was described in section 4.3 and is one of the most significant advantages of OPM compared to traditional task analysis. Using the animated simulation, the OPM system model can be "debugged" similar to a computer program, and it can be tested for logical consistency. An unexpected halt in the simulation highlights potential logical errors or fallacies that need to be addressed in order to rectify the control logic and the flow of the system's behavior. This combined OPM-based modeling and simulation activities minimize the risk of performing inaccurate analysis and consequently developing a prototype based on an incorrect design that will adversely affect project cost and schedule. The earlier an error is detected, the less costly it is to correct it, and the cost increases exponentially with the system development stage. Worse yet, a faulty design, or one that overlooks potential off-nominal system behavior might lead to hazardous situations in the real system if it goes undetected. Animation allows the system analyst to activate specific states and run the model. The result provides an extremely important check for model consistency.

The animated simulation can also be used to help all the stakeholders involved to visualize the system in action with the various processes involved, their sequence, the personnel and instrumentation, as well as the preconditions and post-conditions for each process. This feature helped the author explained his design to the project team and programmers helping in the simulator coding. When presenting the model to an audience, the author suggests to first explain the overall model concept using the SD and explain the meaning of some of the symbols. Then systematically in-zoom to several lower level so that the audience can appreciate the model concept. Once the audience understand the model concept, run the animations. Therefore, by comparison, OPM is potentially a more powerful human factors engineering design tool.

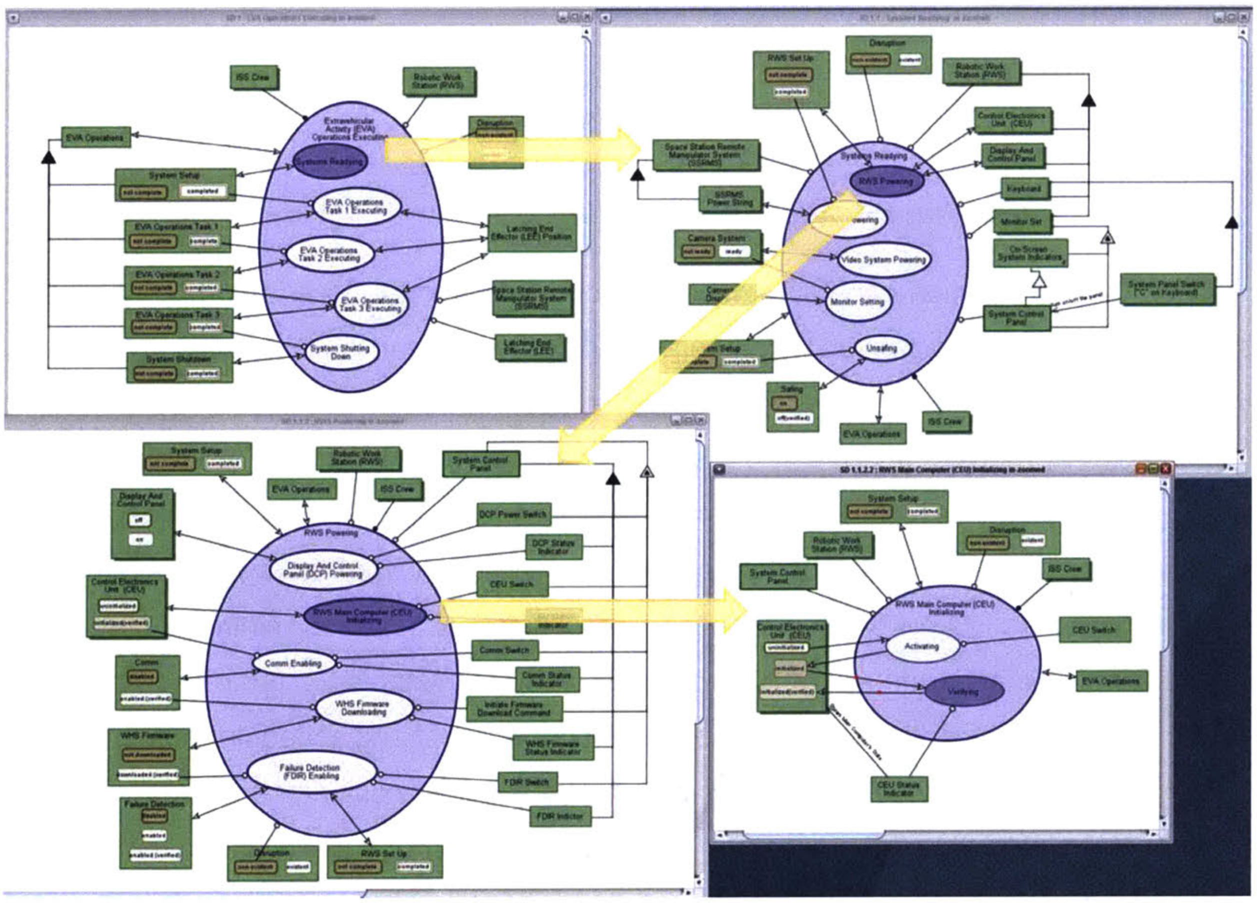

ISS Crew
Robotic Work Station (RWS)
EVA Operations
Extravehicular Activity (EVA) Operations Executing
Systems Readying
EVA Operations Task 1 Executing
EVA Operations Task 2 Executing
EVA Operations Task 3 Executing
System Shutting Down
Latching End Effector (LEE) Position
Space Station Remote Manipulator System (SSRMS)
Latching End Effector (LEE)
RWS Powering
Video System Powering
Monitor Setting
Unsafing
Camera System
Keyboard
Monitor Set
System Control Panel
System Panel Switch
Display And Control Panel (DCP) Powering
RWS Main Computer (CEU) Initializing
Comm Enabling
WHS Firmware Downloading
Failure Detection (FDIR) Enabling
Control Electronics Unit (CEU)
Activating
Verifying
CEU Status Indicator
CEU Switch
Disruption
RWS Set Up

*Figure 56: OPM Simulation View*

Figure 56 animated simulation of the OPM model for **EVA Operations Executing** system. The OPD at the top-left is SD1 – the OPD at level 1, in which the function of the system – the main process **EVA Operations Executing**, at the root of the OPD tree – the System Diagram (SD; level 0) was in-zoomed. To the right of SD1 is SD1.1 – the OPD in the next detail level, in which **System Readying** was in-zoomed. Similarly, the next level down appears at the bottom left of Figure 56, and the most detailed level is at the bottom right – this is the same OPM shown larger in Figure 34. At this level, the simulation is currently performing the **Verifying** leaf process, marked as dark (purple). There is a (red) dot running from the state **on** of **CEU** and another dot running from the **Verifying** subprocess to the state **on (verified)** of the same of **CEU** object. In general, the dark purple shaded ellipse shows the processes that are currently active. Active procedural links representing interactions between objects and processes are show with red dots running along them. A shaded state represents the current state of an object. The OPD tree is traversed in a depth-first manner. When a process is completed, the simulation automatically moves to the next process based on the top-to-bottom graphical arrangement of the processes.

### *5.3.3 One Technique vs Multiple Techniques*

OPM allows the user to employ a single methodology for analysis and design. Task Analysis users have to combine multiple techniques to achieve a somewhat similar level of description albeit without guarantee that the information is logically complete and the result is not a computable model.

### *5.3.4 One OPM Model vs Multiple TA Representations*

The user can easily modify an OPM model to incorporate additional scenarios. Take for example, an off-nominal disruption was added to the original model that describe the nominal EVA operations. Together with the simulation capability, the user can test the troubleshooting and recovery procedure design for a specific disruption as noted in section 4.3.2. In comparison, the troubleshooting and recovery procedure could not be incorporated into the nominal HTA and/or HTA+TTA procedure because the disruption can occur at any part of the experiment as discussed in section 3.3.4. Thus, HTA and HTA+TTA requires different sets of analysis.

### *5.3.5 System Design and Simulator Development*

HTA clearly shows the tasks that need to be done to achieve the goal but does not include the information (i.e. objects, states, systems' relationships and preconditions) required for designers and programmers team to understand how the simulator should be coded. The HTA+TTA could help minimize the gap as it allows physical and informatics information to be included. Also, the HTA+TTA model does not show the relationships between the objects and the dependency between the objects' state and processes. Conceivably, an analyst or designer could add columns to the TTA to record the required information. However, the TTA representation would become hard to comprehend and navigate. Also, nothing in the HTA+TTA process compels the analyst to add this information, whereas it is inherently required when constructing an OPM model.

OPM is able to circumvent the limitations of HTA+TTA and present the necessary details in the OPD. It can concisely show the dependency between systems' states and processes using the instrument link (refer to Table 4 for details). The objects supporting the processes are also depicted in the OPD. Furthermore, OPM allows analyst to describe how and when an object is created and destroyed. Take for example the **Autosequence Panel** box discussed in section 4.3.1.3 and 4.3.1.3.1; it is created in the **Automate Selecting** process in **Autosequence Setting** in-zoomed and destroyed at the **Commencing** process in the **Autosequence Executing** in-zoom. The OPDs also illustrate the objects that are in the **Autosequence Panel** and how the processes affect the states of those objects e.g. **Joint Angles**.

In system design and development, changes are inevitable but it is critical to understand the repercussions associated with the changes. OPM simulation allows the analysts or designers to understand how changes impact the other part of the design and make appropriate amendments. This feature is especially beneficial in complex systems design that involves multiple engineers in different domains.

Working on a common model potentially improves cross-domain communications and highlights issues early in the program rather than later, which will be costly.

### *5.3.6 Usefulness of Unfolding Structural Relationships in OPM*

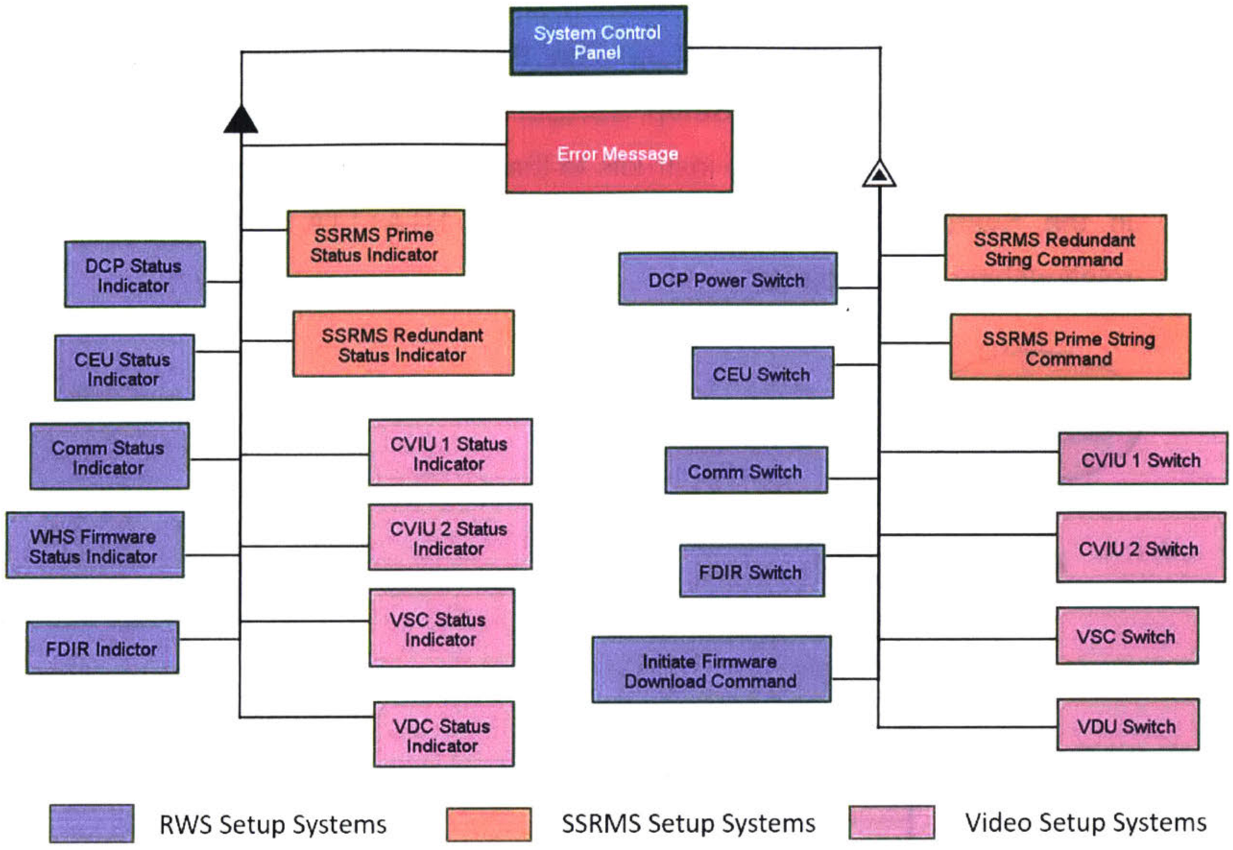


*Figure 57: Structural Relationship of System Control Panel*

As part of building the OPM model, structural relations between objects, and occasionally also between processes, are defined. Unfolding a complex object produces a diagram showing all the structural relations, including parts, attributes, and specializations, among selected objects in a single view. This is useful for designing the interface and developing the simulator. Indeed, in the space telerobotic simulator enhancement, this capability of the OPM model was key in designing the underlying structure and logic of the **System Control Panel**. Section 4.3.3 discussed this feature extensively. The unfolded **System Control Panel** in Figure 57 cogently shows that the **System Control Panel** needs to provide systems status and features that allow users to change the state of these systems.

OPCAT allows the users to change the color of objects and processes in OPD. (The colors are not show during animation.) Color code could be used (if needed) to provide another dimension of information. For example, indicators and switches are categorized to support RWS Setup, SSRSM Setup, Video Setup using purple, orange and pink shading. It implicitly instructs to the programmer to group them accordingly in the **System Control Panel**. In contrast, HTA+TTA does not consider the relationships of the objects.

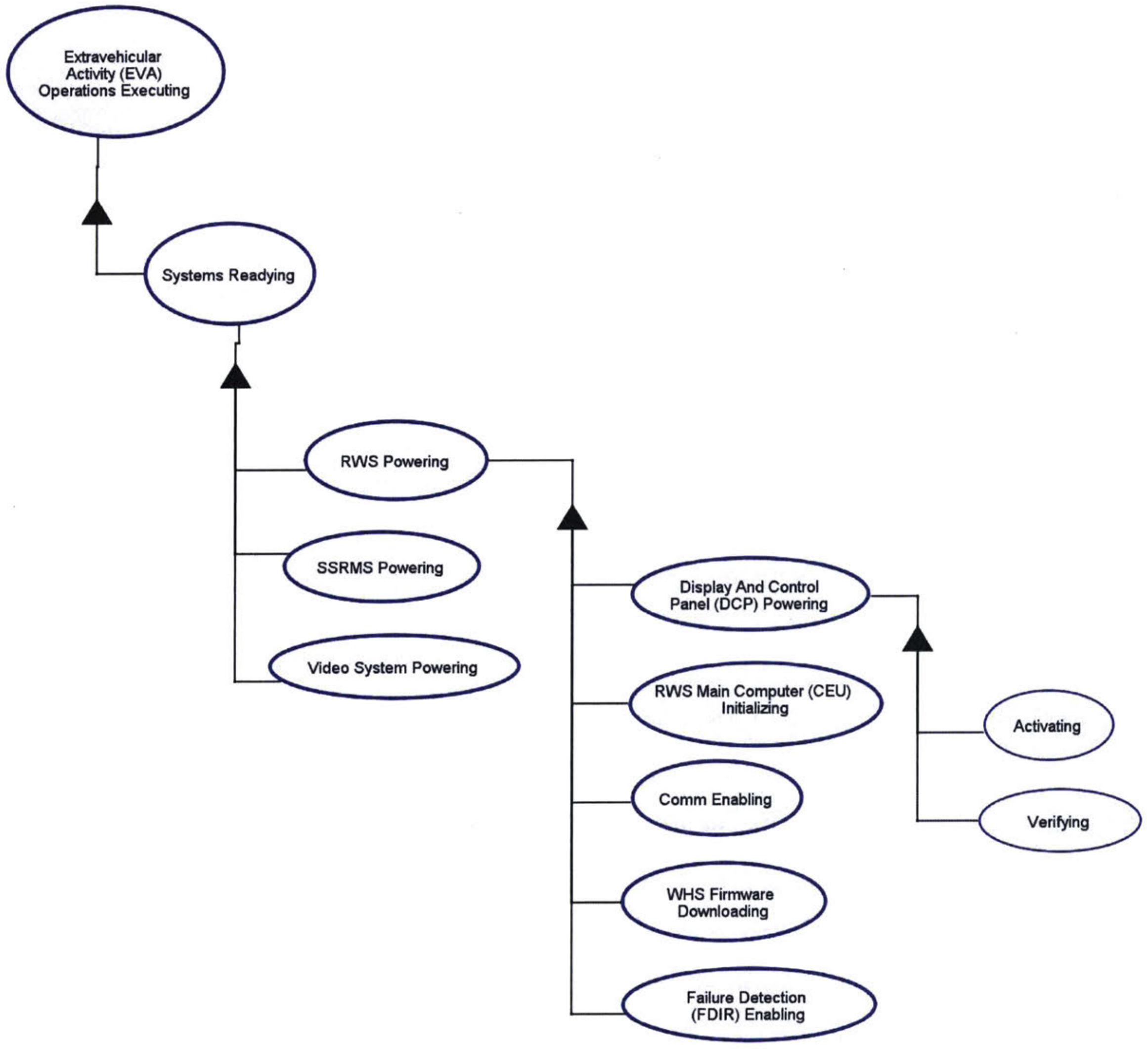

Extravehicular Activity (EVA) Operations Executing
Systems Readying
RWS Powering
SSRMS Powering
Video System Powering
Display And Control Panel (DCP) Powering
RWS Main Computer (CEU) Initializing
Comm Enabling
WHS Firmware Downloading
Failure Detection (FDIR) Enabling
Activating
Verifying

*Figure 58: Structure Relationship Example of Processes*

Similarly, OPCAT allows structural relationship of processes to be unfolded and the output resembles the HTA hierarchical diagram as shown in Figure 20 and Figure 21. It is useful for understanding the structure of the processes involved.

Furthermore, the spatial and connectivity relationships between objects can be described in OPM using tagged links as shown in Figure 59. For example, the system designer can use it to state the location of DCP Indicator is above CEU Status Indicator and vice versa (Figure 59a). Also, connectivity relationship can be clearly expressed; The tagged link in Figure 59b informs that LEE is connected to the ISS by grapple not bolts.

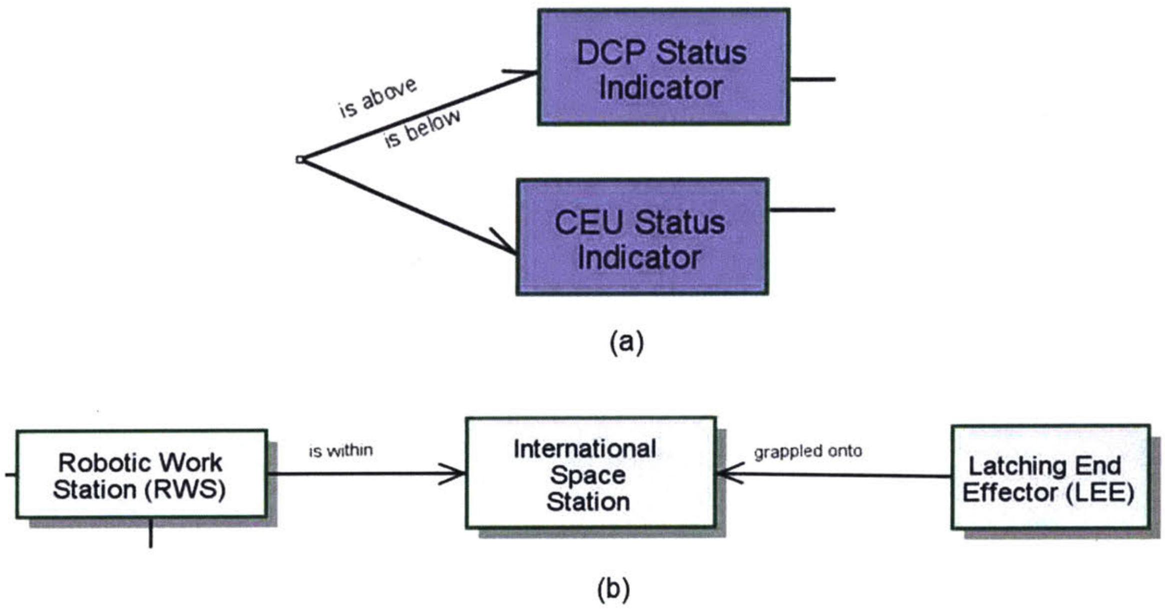


Figure 59:(a) Spatial Relationship of Informatical objects; (b) Spatial and Connectivity Relationships of Physical Objects

### 5.3.7 Other Potential Advantages of OPM

The next few sub-sections will discuss some of the potential advantages of OPM that has not been explored in this thesis. The author opines that it is valuable to highlight them for the benefits of those interested in OPM and to show that the advantages of OPM is not limited to those demonstrated in this thesis.

#### 5.3.7.1 Usefulness for Workload Prediction

OPM model could potentially be used to derive metrics that could correlate with mental and physical workload. Because the OPD shows the instruments and preconditions for a process, the number of preconditions the human needs to verify prior to performing the tasks (e.g. **Autosequence Initializing**) or objects to monitor (e.g. **Camera Display and Arm's Joint Angles Display**) as part of a task could be used as a proxy for workload prediction in specific tasks. Also, since the duration – minimum and maximum time – for each process can be modelled, user could determine the number of tasks the human need to do over a defined period using the animation or run pilot experiments using the animation (refer to section 5.3.7.3). Thereby, collecting data to predict the workload for that period. Using OPM for a proxy workload prediction has not been proven but those who are interested could investigate further.

#### 5.3.7.2 Usefulness in Modelling Human Decisions Processes

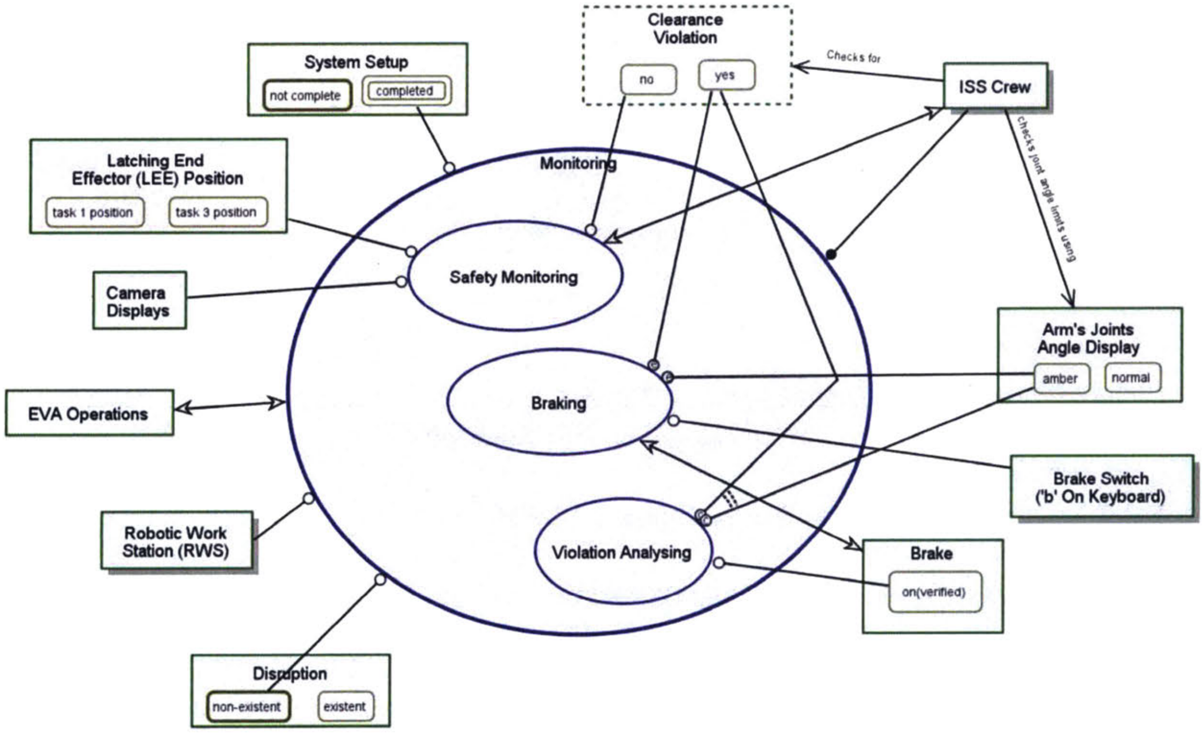


Figure 60: Monitoring

The modelling human decision making in human factors applications can be considered analogous to modelling decision making in a system engineering context [52] in OPM. Accepting that human decision making is not always rational – e.g. subject to misperceptions, confirmation biases and loss aversion, rational decision making processes can be represented in OPM. The OPM models human decision making through graphical representations by defining the pre-conditions or objects states required for the process and how the state of an object will trigger certain processes as illustrated in Figure 60. **Monitoring** is the process that happens concurrently with SSRMS relocating as reflected in Figure 38. The OPD depicts that the **ISS Crew** needs to check for **Clearance Violation** and joint angle limits violation i.e. the **Arm's Joints Angle Display** will turn **amber**. (The present example assumes clearance judgement is perfectly accurate.) If **Clearance Violation** is **yes** or **Arm's Joints Angle Display** is **amber**, the **ISS Crew** needs to trigger the **Brake**. Then, the ISS Crew needs to analyze the reason of the violation and **Violation Analyzing** requires bake to be **on(verified)**. Using OPM to represent human decision process is not explored in this thesis. Nonetheless, this simple example shows that human factors engineers could use OPM to model and analyze human decision making processes.

#### *5.3.7.3 Usefulness as a Training, Practice and/or Evaluation Tool*

Another interesting aspect of an OPM is, if the model is sufficiently detailed, it could potentially be used as a training, practice and/or evaluation tool. Trainees could step through each task in OPM and see how each subsystem state is affected. It could give the trainees a clearer and broader perspective of the system operations concept (e.g. how system states change and why those changes are needed) – beyond a step by step procedure execution. The trainee would need to be familiarized with OPM first.

Since OPCAT can be run on a Windows laptop, the OPM model could be used as a procedure refresher training tool if a realistic simulator is not available. Going through each tasks in OPM – referencing the written procedure if available – could help the operator refresh their memory of the steps and reinforce their mental model.

The OPM can also be used as concept evaluation tool. Procedural operations can be simulated using OPM to get the end-user feedback about the system structure and logic flow before actual implementation. For example, different operationt options could be tried out with the end users for preliminary assessments of the concept using the OPCAT animation. This approach is much faster as it is significantly easier and less time-consuming to develop than an actual simulator, and does not dilute the logical flow of the procedural steps.

One possible way to implement these concepts is using the OPCAT animation capability. Processes that required an agent (human) can be set to stop. Therefore, a human (e.g. trainee) would need to activate those processes so that the simulation can continue. Figure 61(a) is the decomposition of the processes that focus on RWS powering and Figure 61(b) shows the system states. The two bottom diagrams in Figure 61 are the in-zoom of **System Readying** and **RWS Powering** to provide an overall view of the simulation current position. The simulation is currently at **RWS Main Computer (CEU) Initializing**. To continue, the human needs to click the **Activating** process and then the activate button (💡). This action is the analogous to changing the state of the system in the simulator.

By the way, Figure 61(b) could also be used to inform the programmer the layout of the control panel. (The states with "**verified**" will not be shown to the human in the simulator to avoid confusion i.e. '**on**' and '**on(verified)**' will be displayed as On in the simulator.)

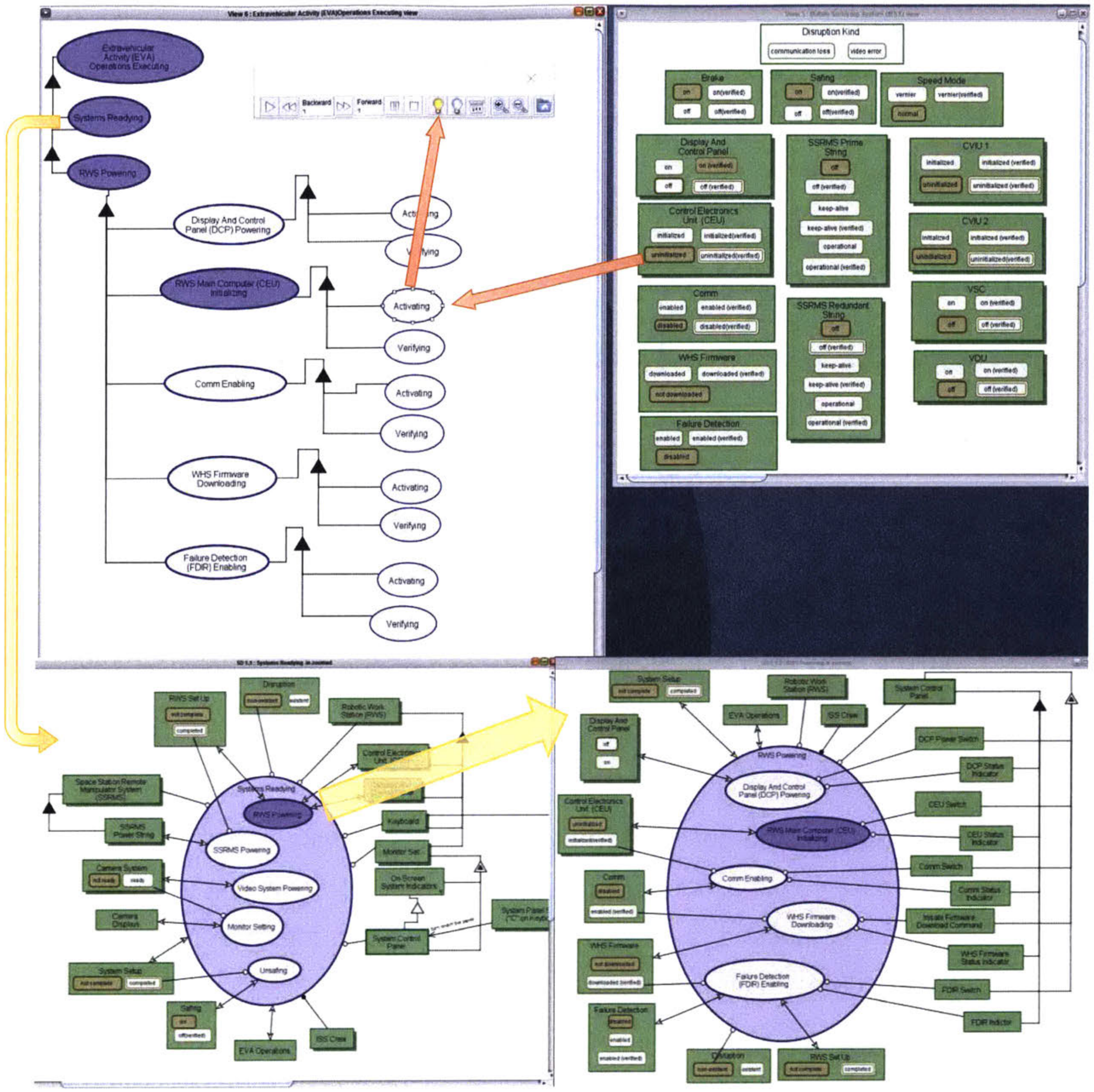


Figure 61: (a) Process Decomposition (Top-Left), (b) System State View (mimic Control Panel) (Top-Right), (c) System Readying In-zoomed (Bottom-Left) and (d) RWS Powering In-Zoomed (Bottom-Right)

## 5.4 Disadvantages of OPM over TA (HTA and TTA)

OPM does have some disadvantages with respect to HTA+TTA. The advantages discussed above do come at a price.

### 5.4.1 Learning Effort

HTA and TTA are relatively easier to learn than OPM. It is expected that with short training, a novice could develop a reasonable HTA + TTA analysis. For OPM, a novice needs to learn and understand OPM syntax before using it to build a model of the system, hence, requiring additional effort.

### 5.4.2 Development Time

The development of an OPM model is somewhat more time-consuming than HTA+TTA, even excluding Stanton's initial two data collection steps mentioned in Section 3.2. For the purpose of this thesis, both models were developed to a roughly comparable level of detail. The HTA + TTA model developed first and described in section 3.3 and 3.4 took about 15 hours. The OPM model described in section 4.3 took about 50 hours. Because the OPM model was developed after HTA, the development time was accelerated since the general hierarchical structure was already understood.

Making changes in HTA + TTA representations such as adding, promoting, demoting and relocating any sub-goals and tasks can be done with a few mouse clicks in TaskArchitect or Excel and no validation is needed. On the contrary, because an OPM model is composed of multiple hierarchical OPDs, affected OPDs have to be individually updated. If there is an error in the grouping or hierarchical arrangement, all the in-zoomed OPDs have to be completely redrawn. OPCAT automatically asks if the user wants to add the new object to the related OPDs, but it is still prudent to go through the OPDs and check for logical consistency by running the animation. Based on the author's experience, adding another procedure step to the lowest hierarchy in a complex OPM model took 5-10x longer than the corresponding change in HTA+TAA. Of course, less information is incorporated in the HTA + TTA representation.

## 5.5 Comparison between OPM and AH

OPM represents functions using objects and processes. The domain purpose, domain functions and physical functions used in the AH model are described in OPM but in another way. The description of the function may differ significantly but the intent remains unchanged. Take for example the physical function – "move without control input" – in AH. In OPM, it is represented as "Relocate SSRMS with auto SSRMS mode (see Figure 38). It is evident that the physical objects reflected in AH are the instruments that support the processes in the OPM.

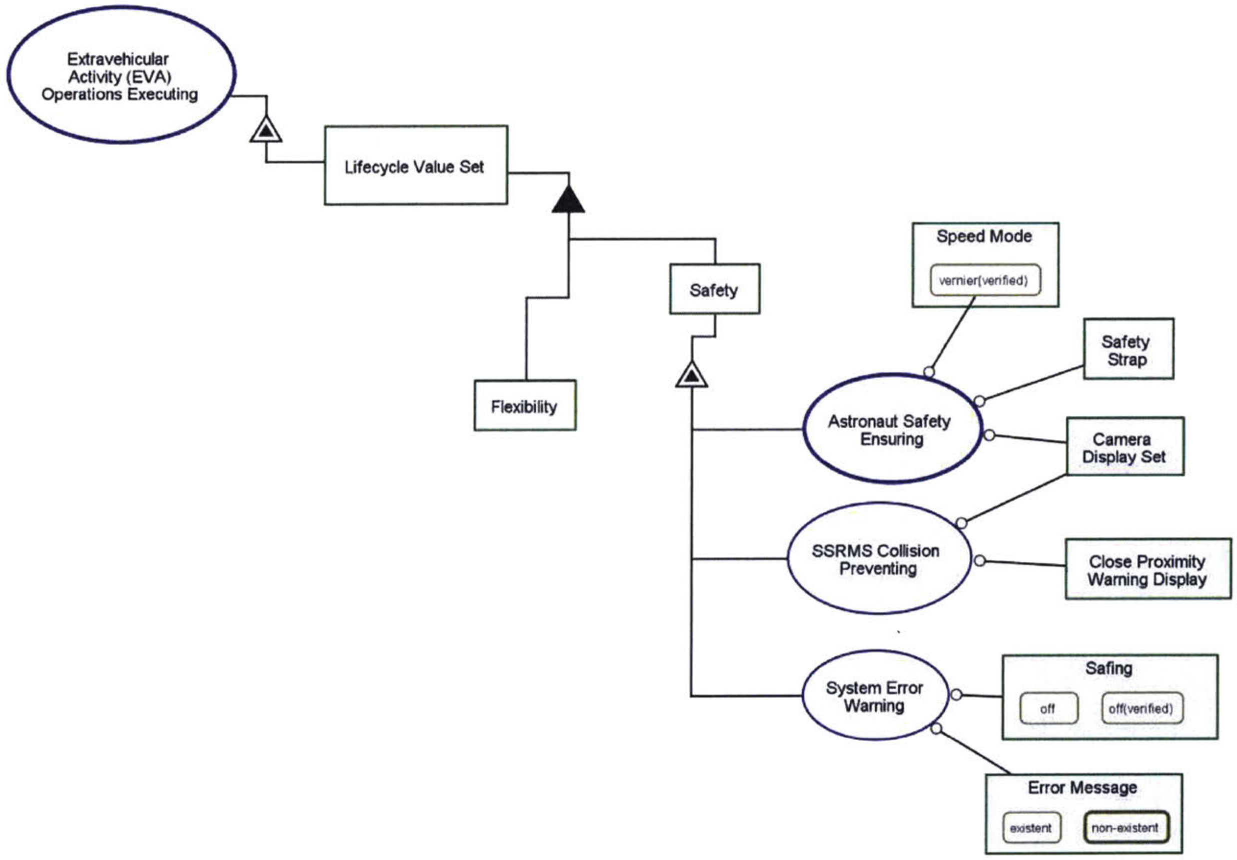


*Figure 62: OPM Domain Values Example*

Although not mandated by OPM, domain values used in AH analysis, such as **Safety** and **Flexibility**, (also known as non-functional requirements or "ilities"), can be naturally incorporated into the OPM model as shown in Figure 62. **EVA Operations Executing** has the attribute **Lifecycle Value Set** with **Safety** and **Flexibility** as members of this set.

**Astronaut Safety Ensuring** is a feature of **Safety** and this process requires instruments such as a **Safety Strap** to secure the astronaut on the platform and **Camera Display Set** for viewing the astronaut.

In-zooming into **Astronaut Safety Ensuring** (Figure 63) presents details into the processes design to ensure the safety of the astronaut. It shows that the **EVA Astronaut**, who is part of the **ISS Crew** has attributes of **Secure to LEE** and **Stability**, and are affected by the two processes – **Astronaut Securing** and **Astronaut Well-being Ensuring**. This OPD also depicts the instruments required to support the respective process. Interestingly, this simple example presents more details than the domain values in AH. The in-zoom required the user to consider the processes needed fulfil the function to ensure the safety of the astronaut and the objects affected by the processes. This exercise also demonstrated the versatility of OPM. Hence, this can be considered as an advantage of OPM rather than a problem.

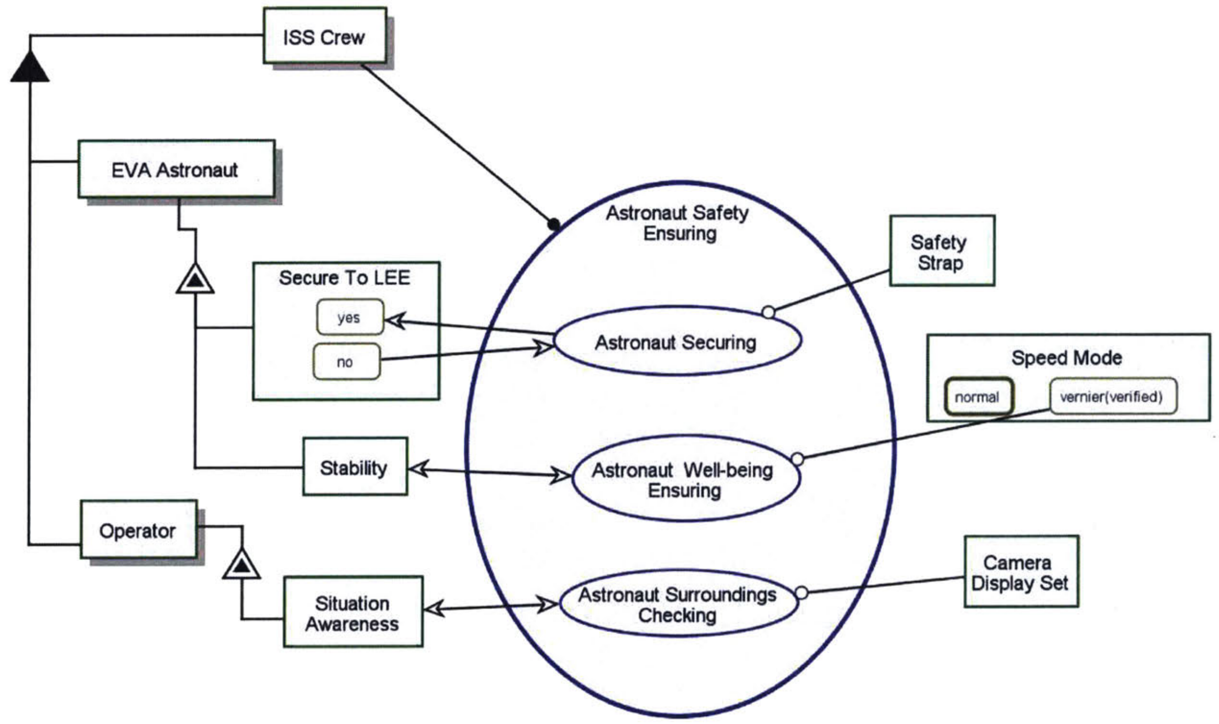


*Figure 63: In-zoomed of Astronaut Safety Ensuring*

One disadvantage of AH graphical representation is that each abstraction level cannot be further decomposed. A complex system AH model can spread across multiple pages with many means-ends links forming a complex web-like structure that can be confusing and hard to comprehend. Also, AH shows the constraints but does not depict the operations concept and provide a complete picture. OPM hierarchical (in-zoomed) graphical representation reduces the complexity and provide a complete picture of the operations concept, including constraints, and has animation capability (AH does not) that improves the visualization of the domain. In addition, OPCAT allows user to create a view that show all the processes that affect a specific component rather than tracing lines in the AH model.

## 5.6 Discussion

A primary goal of this was to compare OPM with HTA, TTA and AH methodologies as applied to a human factors engineering design problem. As there are many task analysis approaches, one cannot conclude that OPM is superior to all task analysis methods in human factors engineering.

Nonetheless, at least for the application considered in this thesis, OPM was unequivocally superior to HTA, TTA and AH together. OPM provides much the same information as a combination of HTA, TTA and AH analyses, and has additional advantages; OPM reduces apparent complexity using a hierarchical structure like HTA, contains details of subsystem/components, includes dependencies between objects and processes, and has the potential usefulness for workload prediction, to model human decision making, and/or as a training/practice tool. Most importantly, OPM produces a computable model that can be checked for logical correctness and incorporate multiple scenarios in one model. The main drawbacks are that the creation and modification of the model are time consuming and requires proficiency with the OPM syntax. But considering the loss in time and cost to rework a prototype because the system design concept has not been verified through modeling, and the potential cost of undetected design errors, the time and cost committed to develop an OPM model is likely justifiable. Therefore, this thesis demonstrated OPM as a potentially valuable tool in system analysis and design for human factors engineering.

# 6 MIT-RWSS SIMULATOR ENHANCEMENT AND RELATIONSHIPS TO OPM

## 6.1 Introduction

The enhancement of the MIT-RWSS involves adding subsystems simulations to improve procedure realism and creating a smart automated electronic procedure capability, and was based on the OPM model developed in this thesis. The enhancements had to meet the requirements defined in Section 2.4.2 and included some other features to improve the realism and improve usability.

The new subsystem simulations required creation of a new system control panel as an overlay on the left monitor as shown in Figure 64. The operator uses it to configure the subsystems, check system health status and specify which procedure is being displayed. After the operator complete troubleshooting and rectifying a system failure, a wizard (if enabled) provides appropriate procedures that guide the operator how to reset the system and rejoin the normal procedural flow. The smart automated electronic procedure is shown on a new procedure viewer that appears as an overlay on the top right corner of the center monitor. An **Autosequence Panel** (if active – refer to Figure 40) will appear as an overlay on the right monitor.

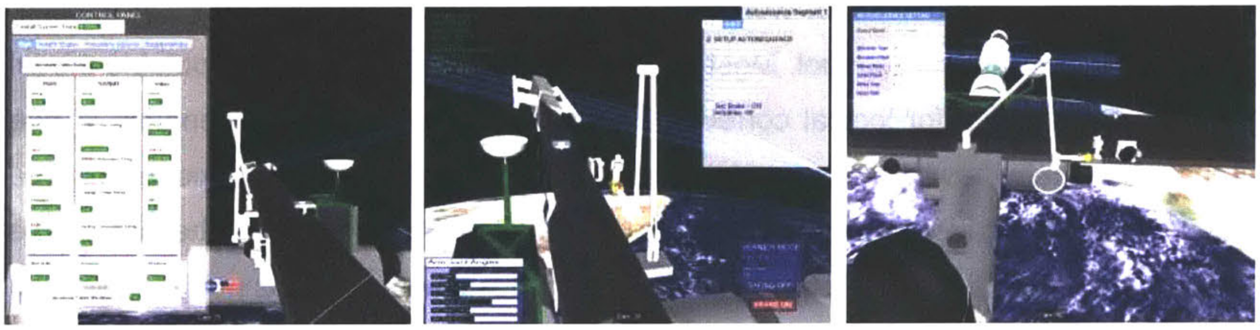


*Figure 64: View of Monitors. Left Monitor – Control Panel and "clearance" view, Center Monitor – Electronic Procedure, Joint Angle Display and "task" view, Right Monitor – Autoseqence Panel and "big picture" view.*

NASA astronaut are trained to use three camera views: a "task" view – often shown on the center monitor, an arm "clearance" view – frequently show on the left monitor, and a "big picture" view – on the right monitor. The format of user displays and controls for ISS robotics must conform to ISS international display and graphics commonality standards (e.g. SSP 50313, JSC 26976). However, since our objective was to prototype a robotic interface for exploration missions further in the future, we set aside these formal constraints and attempted a clean sheet design. We did rely on conventional human factors display concepts, such as Neilson's 10 usability heuristics [63].

## 6.2 OPM Role in the MIT-RWSS Enhancement

The conceptual OPM model developed in Sections 4.3 and 4.4 outlines the architecture of the simulator's new subsystems and automated electronic procedures capabilities. The OPM model was not embedded in the simulation in anyway but the concepts and logic derived from the model were implemented in the MIT-RWSS.

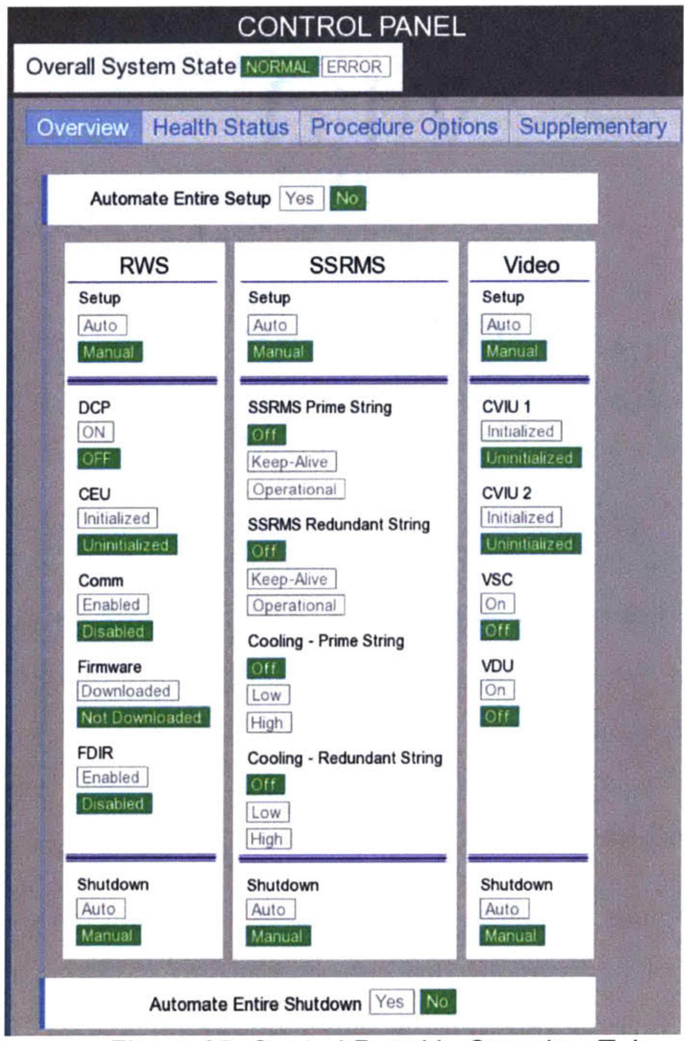
One important value of OPM model – it was use in determining the necessary system preconditions and post-conditions for each procedural step. Pre-conditions define necessary states of supporting systems and the tool that must exist before a process can execute. For example, during the setup process, the Communication system ('**Comm**') must be 'Enabled' prior to initialization of the Control Electronics Unit ('**CEU**') (see Figure 33). Failure to fulfill these pre-conditions constitutes a procedure error. The importance of trapping these errors is discussed in Section 6.5.1.3. Post-conditions show the new state of the system affected by the process. The procedure system will always check if a change in one subsystem state affects another subsystem's state. Take for example in a scenario where initially the **CEU** is **initialized** and the **Comm** is **enabled**. If the **CEU** is later **uninitialized**, the **Comm** status will be changed to **disabled** because of its dependence on the **CEU** state (Figure 43).

## 6.3 System Setup and Shutdown Features

Two distinct additional tasks – system setup and system shutdown – were added to the existing MIT-RWSS and appropriate procedures defined. System setup requires the operator to power up several subsystems when preparing the SSRMS for operation. Similarly, the subjects are required to shutdown the subsystems after operations are completed. Like the real subsystems, the new simulated subsystems were assumed to require some time to power up. Generally, three seconds was used, but this choice can be adjusted by the experimenter.

## 6.4 MIT-RWSS Control Panel (CP)

The Control Panel content was derived in part from the OPD analysis (Figure 46) and is the primary interface used for the following CP Functions: (1) display system state, (2) change system state, (3) select Auto-Cmd automation, (4) shows system error(s), (5) optionally Automate Recovery (6) Select Procedure and (7) display reference schematics and diagrams.



Overall System State [✓ NORMAL] ERROR

Automate Entire Setup Yes [✓ No]

RWS
Setup: Auto [✓ Manual]
DCP: ON [✓ OFF]
CEU: Initialized [✓ Uninitialized]
Comm: Enabled [✓ Disabled]
Firmware: Downloaded [✓ Not Downloaded]
FDIR: Enabled [✓ Disabled]
Shutdown: Auto [✓ Manual]

SSRMS
Setup: Auto [✓ Manual]
SSRMS Prime String: [✓ Off] Keep-Alive Operational
SSRMS Redundant String: [✓ Off] Keep-Alive Operational
Cooling - Prime String: [✓ Off] Low High
Cooling - Redundant String: [✓ Off] Low High
Shutdown: Auto [✓ Manual]

Video
Setup: Auto [✓ Manual]
CVIU 1: Initialized [✓ Uninitialized]
CVIU 2: Initialized [✓ Uninitialized]
VSC: On [✓ Off]
VDU: On [✓ Off]
Shutdown: Auto [✓ Manual]

Automate Entire Shutdown Yes [✓ No]

*Figure 65: Control Panel in Overview Tab*

The Control Panel is shown in Figure 65. Tabs are used to select different display information (e.g. Overview vs Health Status). Use of tabs improves usability, allows adding more interface elements without creating visual complexity, and minimizes the display of unnecessary information.

Information on the Overall System State is always available in the Control Panel regardless which tab the operator selects because of its importance to safety. This is the **Error Message** referenced in the OPD shown in Figure 30. When an error occurs, the state changes to 'Error' (Figure 66) and a warning message appears on the center monitor.

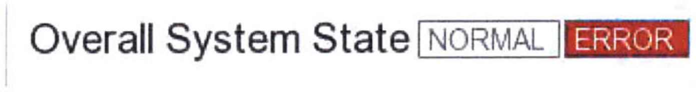


*Figure 66: Overall System State - Error*

The control panel can be shown or hidden by the operator and should not be accessed if the SSRMS arm is actually moving. Thus, the simulator is coded to turn off the Control Panel when the SSRMS arm brake is released and requires the **Brake** to be '**on**' to appear due to safety considerations.

### *6.4.1 Display System State and Change System State (CP Functions 1 and 2)*

The Control Panel system states display and button design were derived from the structural OPD in Figure 46. The soft button is a system indicator (as illustrated by the aggregation-participation symbol ▲ symbol) and a soft switch (as shown by the exhibition-characterization ◬ symbol) at the same time. Active buttons are colored (either green or red). This layout was also derived from this OPD. The color code used in this OPD designates sub-categories like RWS, SSRMS and Video. In the control panel, this correspond to the individual column. Considering usability, this layout was arranged based on the sequence in the system setup procedure such that the operator works from top to bottom and then left to right. E.g. **DCP** to **FDIR** (top to bottom), then proceed to **SSRMS Prime String** (left to right).

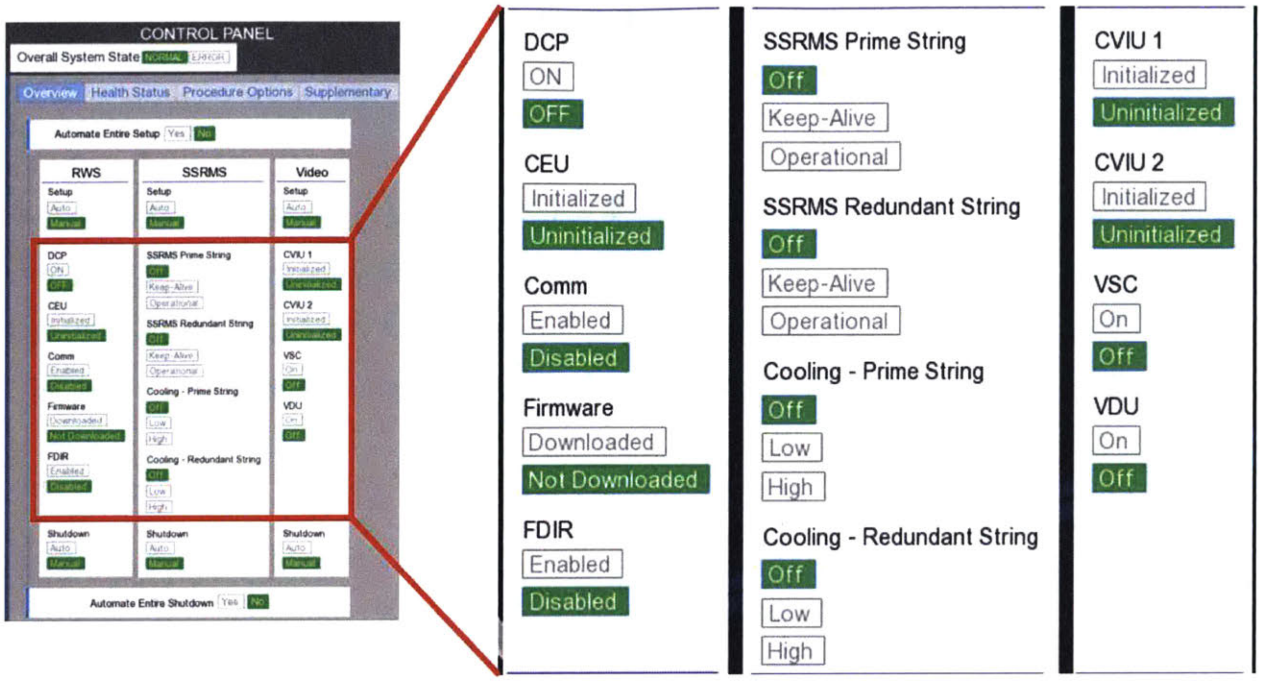


*Figure 67: Control Panel – System State Display and Switches*

### 6.4.2 Automation Selection (CP Function 3)

At the start of an experiment, the control panel will always appear configured for full manual operation as shown in Figure 68 with all 'Manual' buttons colored green. As discussed in section 4.4.2, the experimenter has a choice of twelve possible automation options. The experimenter defines the automation mode to be used in the experiment by editing an experiment configuration file (see section 6.8). If experimenter wants the entire setup to be done in 'Auto-cmd' mode, the subject nonetheless needs to manually select Automate Entire Setup 'Yes' to begin the automated system setup. This is to ensure that the subject knows when the automation begins.

There is two ways to automate the setup and shutdown. If the operator selects Automate Entire Setup 'Yes', the Setup buttons of RWS, SSRMS and Video will change to 'Auto' and the system setup will proceed automatically (same for shutdown). Alternatively, if the operator leaves Automate Entire Setup as 'No', they could choose to enable automation of RWS, SSRMS and Video separately by selecting the respective 'Auto' buttons. (To trap certain operator errors, not all combinations are possible: 1) RWS(Manual), SSRMS (Manual), Video (Auto) and 2) RWS(Manual), SSRMS (Auto), Video (Auto) are not allowed.)

For safety considerations, selecting any change in automation state triggers a warning message (an 'Are you sure?' caution) like the one shown in Figure 81.

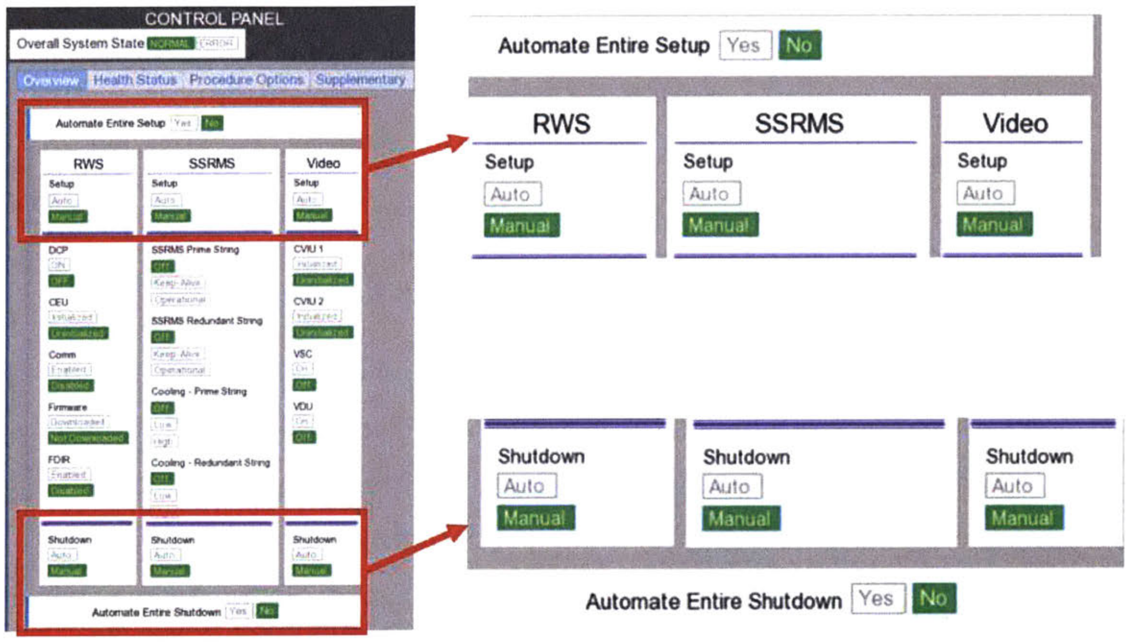


Figure 68: Control Panel – Automation Selection (Auto-cmd)

### 6.4.3 System Health Status (CP Function 4)

The Overall System State at the top of the Control Panel only indicate whether a failure (i.e. **Disruption**) is **existent** or **non-existent**. If the operators selects the Health Status tab, they will be able to see the type of failure (i.e. **Disruption Kind** refer to Figure 42). The type of failure (e.g. Video) will be highlighted in red, analogous to the warning displays on aircraft.

Also, there is a Select Joint to Move dropdown panel in the Health Status tab. The panel is a capability added to enable operator to perform single joint arm movement of the SSRMS to troubleshoot joint failure

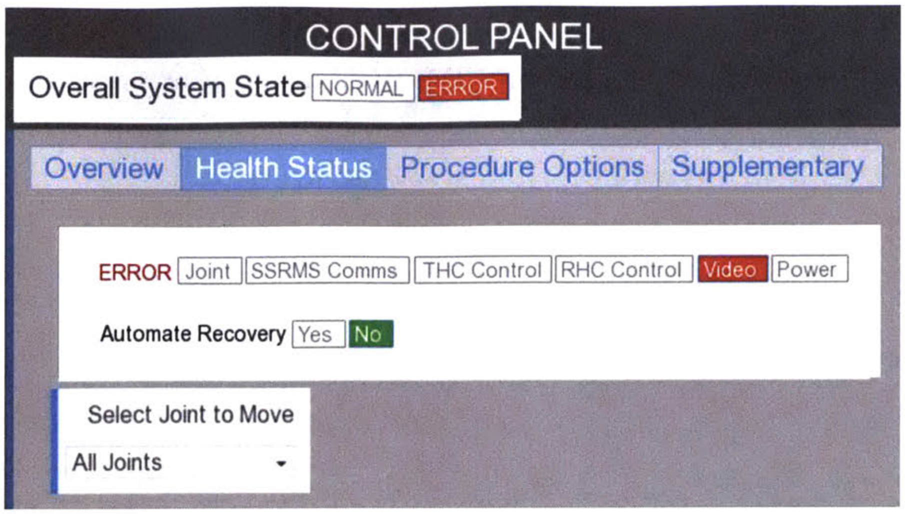


Figure 69: Control Panel – System Health Status

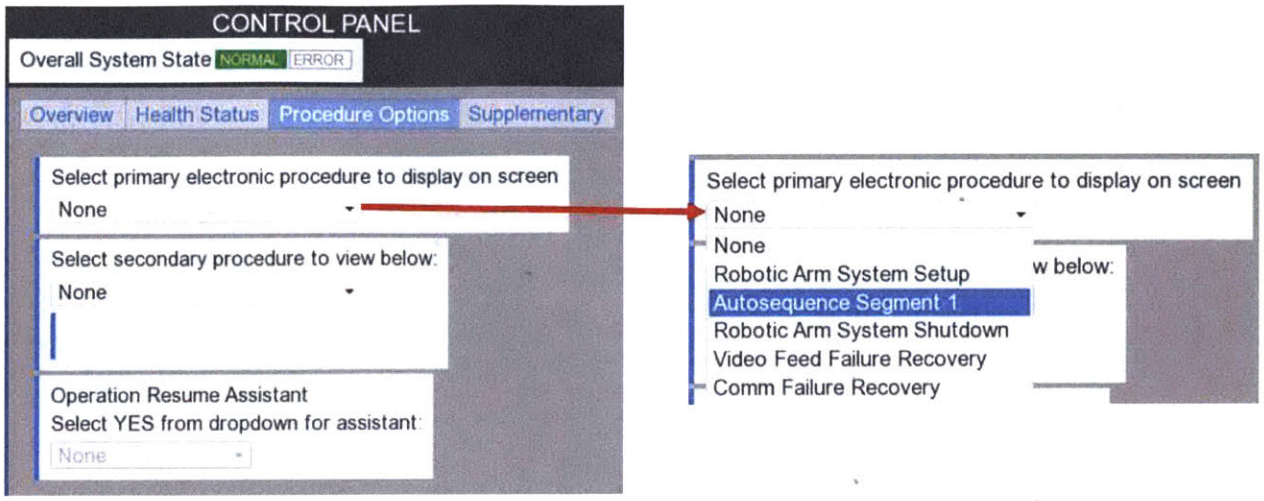
### 6.4.4 Optionally Automate Recovery (CP Function 6)

An interesting and novel capability in the Control Panel is the automation of failure recovery. The automate procedures were designed using the OPM. Currently, two recovery procedures, **communication loss** and **video error** (see Figure 42) were implemented in the simulator as a proof of concept. The Automate Recovery (see Figure 69) functions similarly to the Automate Entire Setup feature discussed previously. If 'Yes' is selected, the automation will execute the recovery procedure selected. Note that each recovery procedure was designed only to resolve the specific failure. The restoration of the entire system to the states that existed before the failure is also necessary to rejoin the operation and is a separate process that will be discussed in section 6.4.5.1.

### 6.4.5 Procedure Options (CP Function 6)

The Procedure Options tab contains a few sub-panels with dropdown menus as shown in Figure 70. The first (top) dropdown menu allows the subjects to change the electronic procedure displayed on the center monitor. The second dropdown enables operator to display another procedure (if needed) in the Control Panel as shown in Figure 71. The second procedure is intended just for reference and has no automation capability. Dropdown menu design is suitable given the small number of procedures (between 5 to 7) required for the experiment. The last dropdown is the

Operation Resume Assistant procedure described in the next section (6.4.5.1). It is disabled by default and only enabled after a failure.

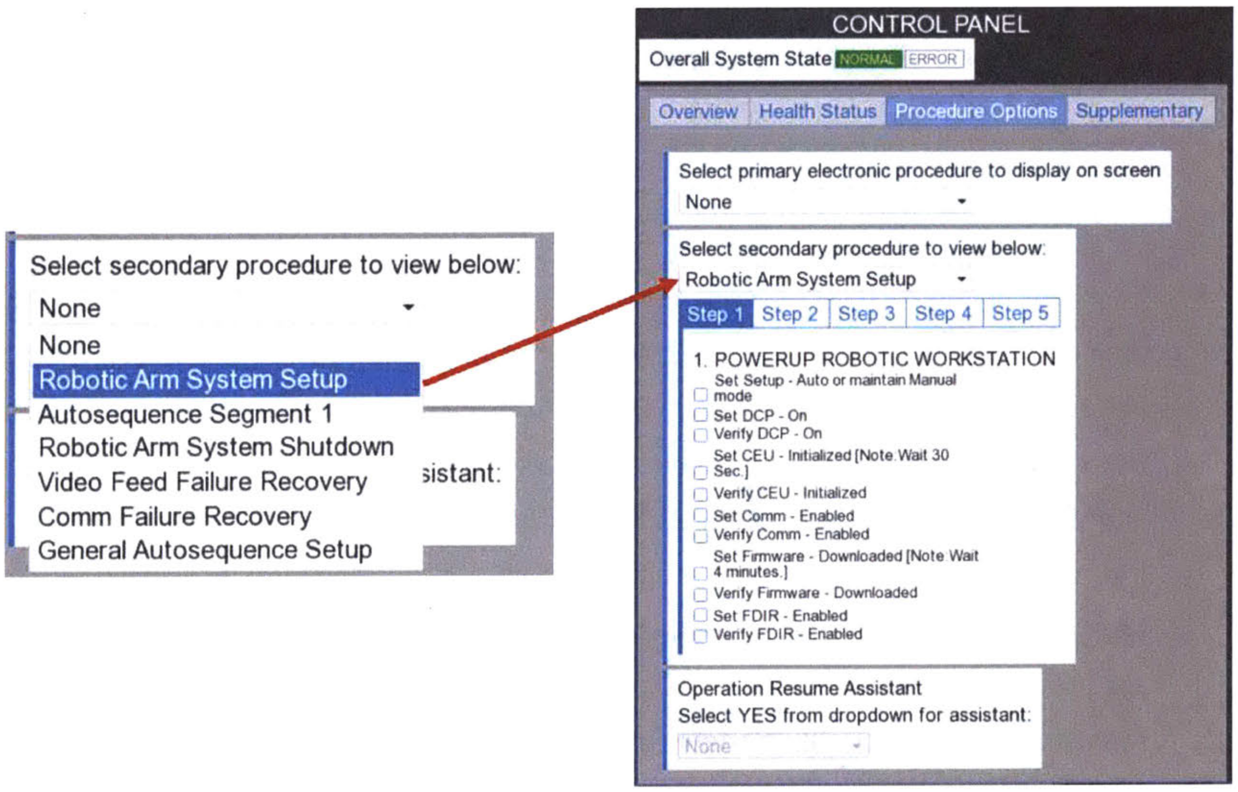


Select primary electronic procedure to display on screen: None

Select secondary procedure to view below: None

Operation Resume Assistant
Select YES from dropdown for assistant: None

Figure 70: Control Panel – Procedure Options



Select primary electronic procedure to display on screen: None

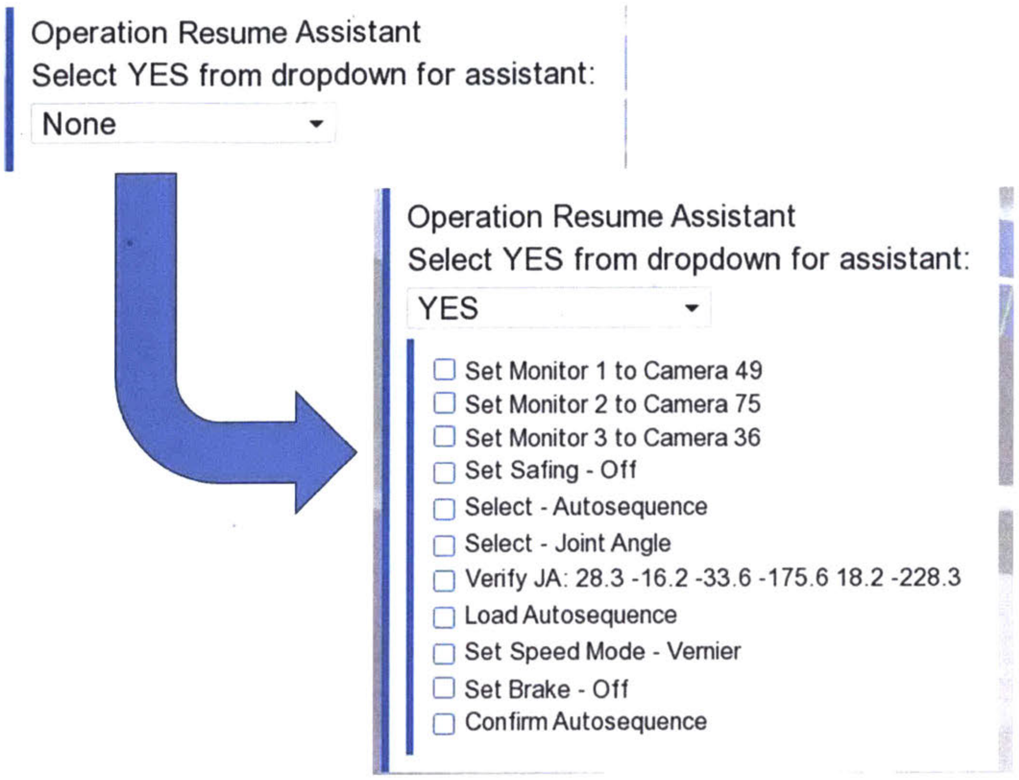
Select secondary procedure to view below: Robotic Arm System Setup

1. POWERUP ROBOTIC WORKSTATION
- ☐ Set Setup - Auto or maintain Manual mode
- ☐ Set DCP - On
- ☐ Verify DCP - On
- ☐ Set CEU - Initialized [Note: Wait 30 Sec.]
- ☐ Verify CEU - Initialized
- ☐ Set Comm - Enabled
- ☐ Verify Comm - Enabled
- ☐ Set Firmware - Downloaded [Note: Wait 4 minutes.]
- ☐ Verify Firmware - Downloaded
- ☐ Set FDIR - Enabled
- ☐ Verify FDIR - Enabled

Operation Resume Assistant
Select YES from dropdown for assistant: None

Figure 71: Dropdown Menu

#### 6.4.5.1 ***Operation Resume Assistant***


Operation Resume Assistant
Select YES from dropdown for assistant:
None

Operation Resume Assistant
Select YES from dropdown for assistant:
YES

- ☐ Set Monitor 1 to Camera 49
- ☐ Set Monitor 2 to Camera 75
- ☐ Set Monitor 3 to Camera 36
- ☐ Set Safing - Off
- ☐ Select - Autosequence
- ☐ Select - Joint Angle
- ☐ Verify JA: 28.3 -16.2 -33.6 -175.6 18.2 -228.3
- ☐ Load Autosequence
- ☐ Set Speed Mode - Vernier
- ☐ Set Brake - Off
- ☐ Confirm Autosequence

*Figure 72: Operation Resume Assistant Procedure*

In a real system, failures can potentially occur at any time. However, in this simulation, failures were assumed to happen only during Autoseqence. When a failure occurs, the operator should use the pre-defined failure recovery procedure. But executing the failure recovery procedure typically changes other the system states from value existing just prior to the failure (e.g. **Camera**, **Speed Mode**, etc.). If these states are not reset, the operation should not continue and the operator needs to re-configure the system to the correct states (e.g. change camera, set speed mode, release brake, etc). Normally, this would require them to systematically check all system states – for example by redoing the entire procedure from the beginning – which would be unnecessarily laborious. To avoid the need to begin the entire procedure again, a novel "Operation Resume Assistant" capability was developed. When activated, it displays the procedure instructions needed to reset the system states altered by the failure recovery procedure to the configuration existing before the failure in a logical flow. The execution of the procedure furnished by the resume assistant must be done manually; there is no automation. This enhances operator state awareness.

#### 6.4.5.1.1 Operation Resume Assistant Development Concept and Logic

This section elaborates on the development concept and logic of the Operation Resume Assistant guided by the OPM model. Comparing the system states before the failure and after recovery will reveal all the discrepancies. The simulator records the entire systems' state when failure occurs. The OPM logic, programmed into the simulator, defines the preconditions, post-conditions and instructions for every configuration change, since in OPM any change can only be made by a process. When the Operation Resume Assistant is activated, it will check the current system states against the stored states and identifies any differences, and presents to the operator the appropriate procedure instructions in the proper logical flow. For example, assumed that a failure happened in the midst of an Autoseqence. Then the operator performed the failure recovery to rectify the problem. The operator then needs to setup the Autosequence again but the procedure is not included in the recovery procedure. The Operation Resume Assistant will display procedures guiding the operator to setup the Autosequence in a logically correct order e.g. 'Select – Autoseqence' must be before 'Select – Joint Angle' as defined in Figure 40.

### 6.4.6 Display Reference Schematics and Diagrams (CP Function 6)

The Supplementary tab allows schematics or diagrams required for the experiment to be displayed in the Control Panel as shown in Figure 73. This is to ensure all the reference diagrams or schematics the subject needs are available.

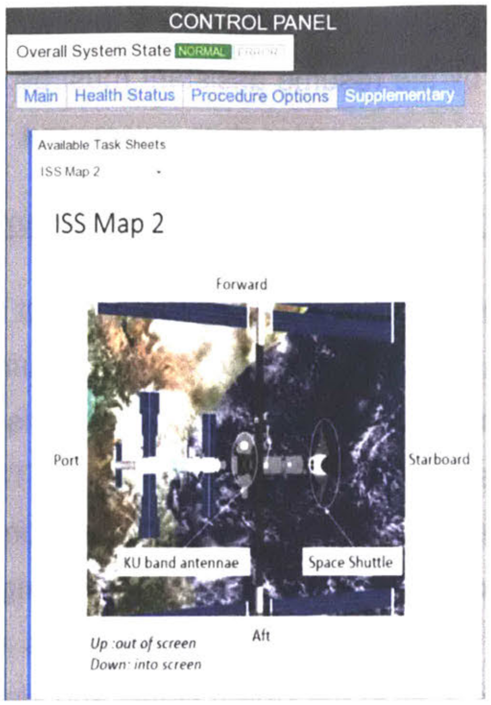


Figure 73: Supplementary Tab

## 6.5 Electronic Procedure

The Electronic Procedure viewer was integrated into the simulator and display as an overlay in the center monitor as shown in Figure 74. It incorporates new automation features that are described below. Figure 75 shows the electronic procedure used for system setup. The title of the procedure is displayed at the top and each procedure is broken down into sub-sections displayed in separate tabs. The OPM hierarchy was used as a guide to determine the sub-sections. For example, in Figure 75, the Powerup Robotic Workstation (Sub 1) corresponds to the processes (including in-zooms) described in the Figure 33 OPD. Sub-sections help improve usability and reduce the display space. Each tab has a header that describes the function/purpose for the sub-steps. For example, in Figure 75, the sub-steps (e.g. Set DCP – On, etc.) fulfil the function of Powerup RWS.

Operator can select any tabs to view the upcoming procedure instructions but cannot execute any instructions until all preceding instructions are completed. To improve visibility and awareness, we adopted the convention that the current instruction is always displayed in blue with enlarged font while completed instructions are shown in white font with a tick in the checkbox besides each procedure. Procedure instructions that have not been executed are in black font. The simulator is normally shown the procedure in the top right corner of the center monitor.

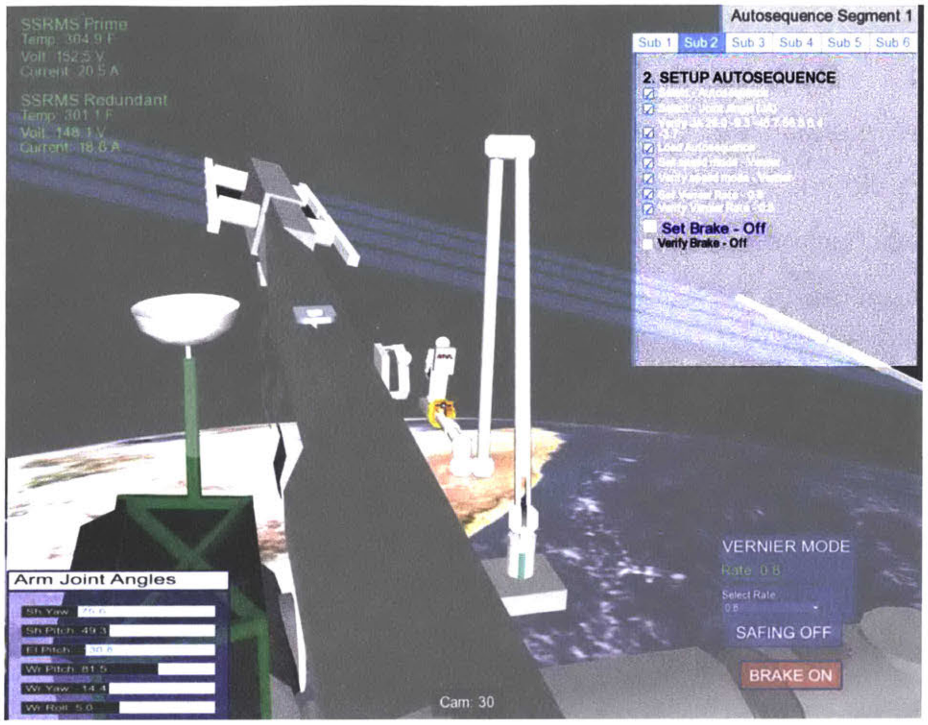
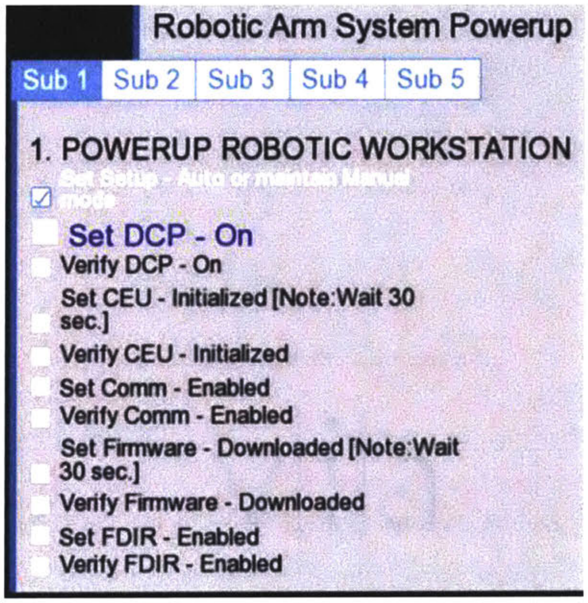


Figure 74: Overview of Center Monitor with Electronic Procedure



Robotic Arm System Powerup

Sub 1 | Sub 2 | Sub 3 | Sub 4 | Sub 5

1. POWERUP ROBOTIC WORKSTATION

- [✓] Set Setup - Auto or maintain Manual mode
- Set DCP - On
- Verify DCP - On
- Set CEU - Initialized [Note:Wait 30 sec.]
- Verify CEU - Initialized
- Set Comm - Enabled
- Verify Comm - Enabled
- Set Firmware - Downloaded [Note:Wait 30 sec.]
- Verify Firmware - Downloaded
- Set FDIR - Enabled
- Verify FDIR - Enabled

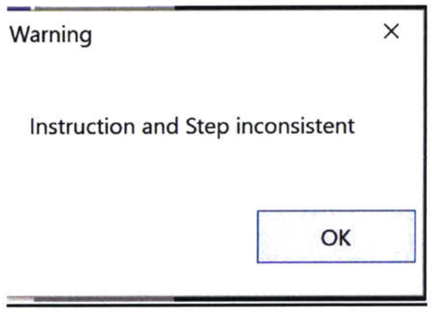
Figure 75: Screenshot of Electronic Procedure

### 6.5.1 *Smart Electronic Procedures*

The electronic procedure includes several OPM model derived automation capabilities: 1) Trapping of inappropriate configuration changes, 2) Verification of automated configuration, 3) Automated preconditions validation, 4) Electronic procedure marking error prevention and 5) Warm up requirement protection. The Smart Electronic Procedures determines the current instruction by find the next unmarked checkbox. It determines the subsystem referred to (e.g DCP), the type of actions required (e.g. Set or Verify) and the final subsystem state required (e.g. On or Off).

#### 6.5.1.1 *Trapping Inappropriate Configuration Change Actions*

Because of the Smart Electronic Procedures capability, subjects can only perform a configuration change that is called out in the procedure for the current instruction. For example, if the current instruction is 'Set DCP – On' and the operator click CEU 'initialized' instead, an error message as shown in Figure 76 will appear on the center monitor. This feature helps to ensure procedure adherence and prevents the subject from performing the wrong procedural instruction.


Warning

Instruction and Step inconsistent

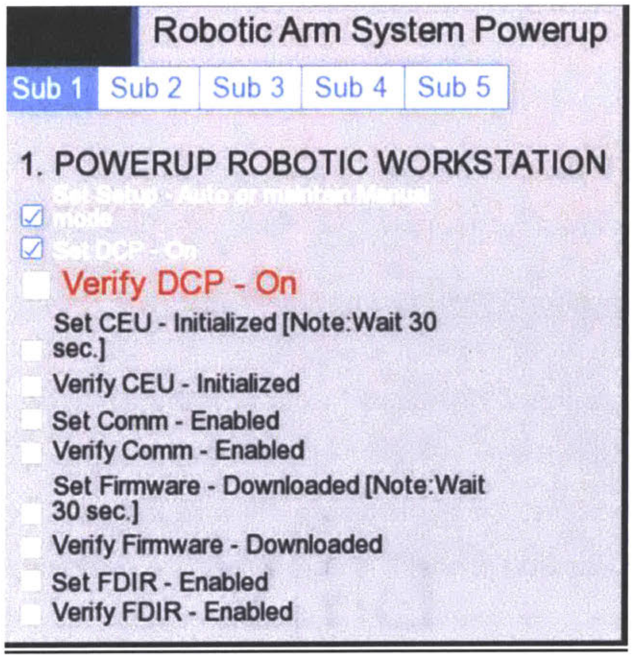
*Figure 76: Instruction and Step Inconsistent*

Although this may seem overly restrictive, it is an important feature that enhances safety. However, there are exceptional situations where out of sequence configuration changes are needed. For example, after recovering from a system failure, the subject needs to reset the system back to where it was before the failure without using the electronic procedure. In such situations, the Smart Electronic

Procedures waives the rule e.g. operator can Select – Autosequence without a corresponding procedure instruction. Also, the operator can always engage Brake and Safing.

### 6.5.1.2 <u>*Automated Configuration Change Verification*</u>

This section will be discussed in the context of the Auto-verify automation type explained in section 4.4.2. When the procedure involves a verification instruction, the Smart Electronic Procedures automatically cross checks the required system's state and actual system's state. If they do not match, the instruction text automatically turns 'Red' (see Figure 77) and the operator will not be able to mark that instruction as completed. The text will revert back to 'Blue' when the system is set to the correct state by the operator.



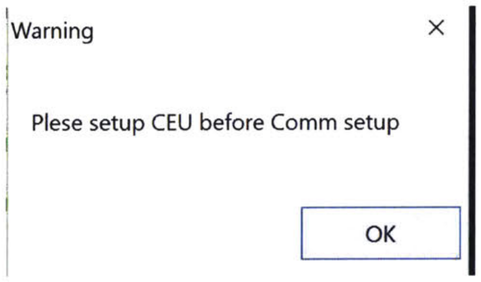
1. POWERUP ROBOTIC WORKSTATION

- ✓ Set Setup - Auto or maintain Manual mode
- ✓ Set DCP - On
- ☐ Verify DCP - On
- ☐ Set CEU - Initialized [Note:Wait 30 sec.]
- ☐ Verify CEU - Initialized
- ☐ Set Comm - Enabled
- ☐ Verify Comm - Enabled
- ☐ Set Firmware - Downloaded [Note:Wait 30 sec.]
- ☐ Verify Firmware - Downloaded
- ☐ Set FDIR - Enabled
- ☐ Verify FDIR - Enabled

<u>*Figure 77: Electronic Procedure – Verification Error*</u>

### 6.5.1.3 <u>*Automated Preconditions Validation*</u>

There might be unforeseen circumstances where an operator changes a system configuration before the preconditions are met and the result could have safety implications. For example, a software bug automatically switches a system state without informing the operator. Since it is not reasonable to plan a solution for unknown failures, appropriate "**safety** gate" must be in place to detect inappropriate system states. The OPM model specifies the required preconditions for each process. These preconditions were manually coded into the Smart Electronic Procedures program, creating an Automated Preconditions Validation feature. When any change configuration step take place, the required preconditions are checked. If the preconditions are not satisfied, an error message as reflected in Figure 78 will appear. In this example, the operator tried to set **Comm** to **enable** prior to **initializing** **CEU**.

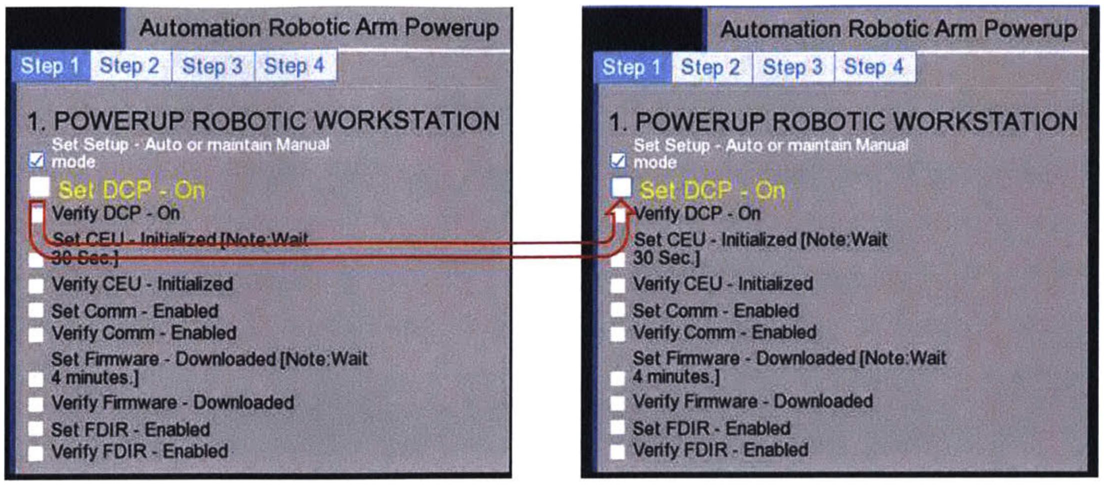

Warning

Plese setup CEU before Comm setup

*Figure 78:Warning Message for Wrong Selection*

## 6.5.1.4 Electronic Procedure Marking Error Prevention

The OPM requires that **Configuration Change Instruction Marking** should only proceed after **Activating** is completed. Nonetheless, the operator could potentially mark the checkbox (i.e. **Configuration Change Instruction Marking** (See Figure 48) before the preceding process is completed. To prevent premature marking, the configuration change instruction checkbox is disabled (i.e. cannot be marked and shown in grey) initially and only enabled (shown in blue outline) after the configuration change for the current process is completed, as reflected in Figure 79. Note that this feature only applies to configuration change instruction and not to configuration change verification instructions because the simulator cannot track human visual verification. In auto-verify mode, the checkbox will be disabled if the system state is incorrect as elaborated in 6.5.1.2.



Automation Robotic Arm Powerup

Step 1 | Step 2 | Step 3 | Step 4

1. POWERUP ROBOTIC WORKSTATION

- ✓ Set Setup - Auto or maintain Manual mode
- ☐ Set DCP - On
- ☐ Verify DCP - On
- ☐ Set CEU - Initialized [Note:Wait 30 Sec.]
- ☐ Verify CEU - Initialized
- ☐ Set Comm - Enabled
- ☐ Verify Comm - Enabled
- ☐ Set Firmware - Downloaded [Note:Wait 4 minutes.]
- ☐ Verify Firmware - Downloaded
- ☐ Set FDIR - Enabled
- ☐ Verify FDIR - Enabled

Automation Robotic Arm Powerup

Step 1 | Step 2 | Step 3 | Step 4

1. POWERUP ROBOTIC WORKSTATION

- ✓ Set Setup - Auto or maintain Manual mode
- ☐ Set DCP - On
- ☐ Verify DCP - On
- ☐ Set CEU - Initialized [Note:Wait 30 Sec.]
- ☐ Verify CEU - Initialized
- ☐ Set Comm - Enabled
- ☐ Verify Comm - Enabled
- ☐ Set Firmware - Downloaded [Note:Wait 4 minutes.]
- ☐ Verify Firmware - Downloaded
- ☐ Set FDIR - Enabled
- ☐ Verify FDIR - Enabled

Figure 79: Electronic Procedure Marking Error Prevention. (The checkbox is grey when disabled and outline in blue when enabled.)

### 6.5.1.5 SSRMS Warm Up Requirement Protection

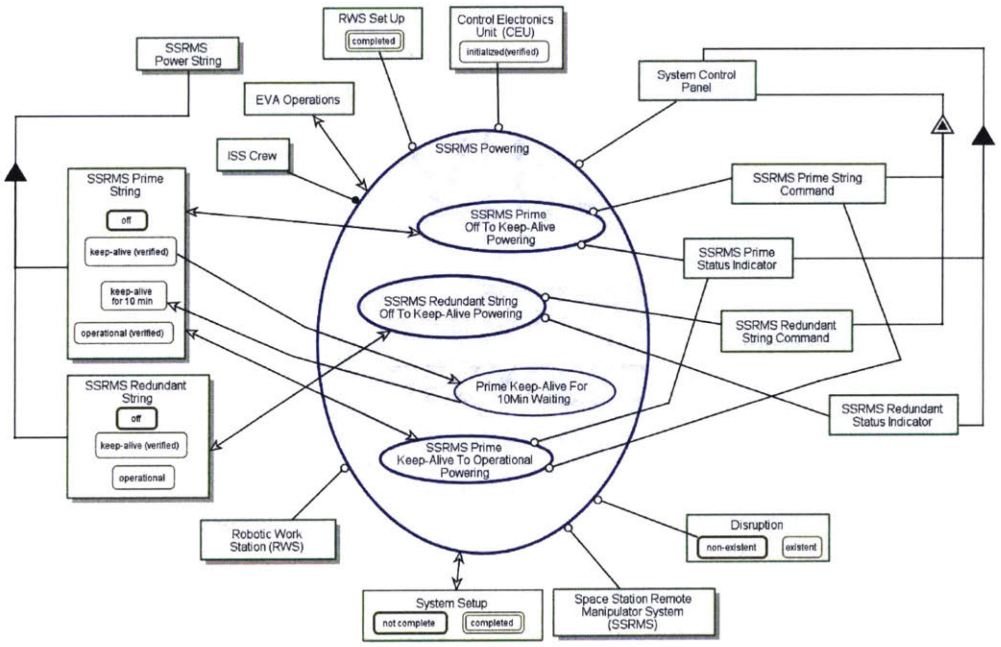
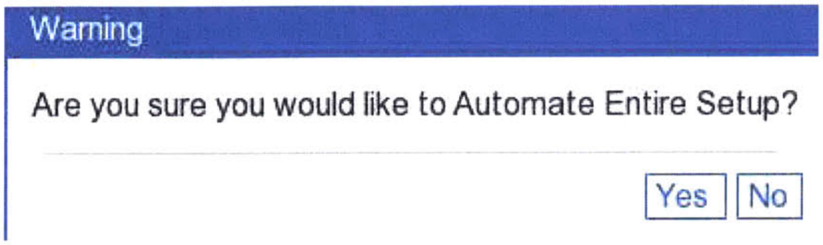

RWS Set Up
completed
Control Electronics Unit (CEU)
initialized(verified)
SSRMS Power String
EVA Operations
System Control Panel
ISS Crew
SSRMS Powering
SSRMS Prime String
off
keep-alive (verified)
keep-alive for 10 min
operational (verified)
SSRMS Prime Off To Keep-Alive Powering
SSRMS Prime String Command
SSRMS Prime Status Indicator
SSRMS Redundant String Off To Keep-Alive Powering
SSRMS Redundant String Command
Prime Keep-Alive For 10Min Waiting
SSRMS Redundant String
off
keep-alive (verified)
operational
SSRMS Redundant Status Indicator
SSRMS Prime Keep-Alive To Operational Powering
Robotic Work Station (RWS)
Disruption
non-existent
existent
System Setup
not complete
completed
Space Station Remote Manipulator System (SSRMS)

Figure 80: OPD of SSRMS Powering In-Zoomed

During system setup, the **SSRMS Prime String** must be put in **keep-alive** state for a minimum of 10 min before it can be set to **operational** (see OPD in Figure 80). The Warm Up Requirement Protection feature prevents the operator from switching **SSRMS Prime String** to **operational** prematurely. When such an attempt is made, a warning message will appear and the command is ignored.

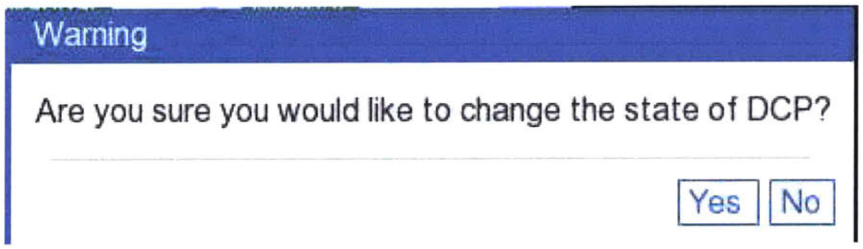
## 6.6 Preventing Inadvertent Commands

One principle of user interface design is to ask the operator to verify any step that is difficult to reverse or otherwise consequential. For important system state changes (e.g. turning on and off a subsystem or releasing the brake), a warning message appears requiring the operator to verify their command. It allows the operator to cancel the command if it is selected erroneously as shown in the Figure 81 and Figure 82 examples.


Warning

Are you sure you would like to Automate Entire Setup?

Figure 81: Warning Message when Automate Entire Setup is selected

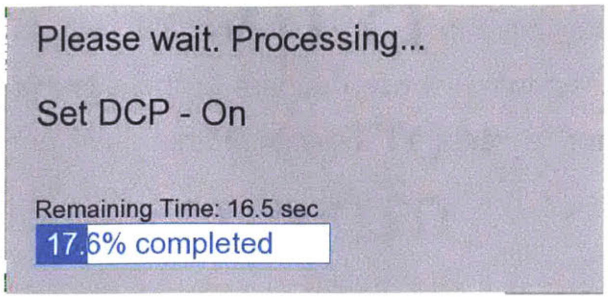

Warning

Are you sure you would like to change the state of DCP?

Figure 82: Warning Message when Change of State is selected

## 6.7 Processing Feedback to Operator

Another principle of user interface design is to provide feedback to the operator on the completion state of an ongoing process. The simulator displays a message to indicate that it is processing a system state change. It shows the instruction being processed, the remaining time and the percentage of completion as illustrated in Figure 83.



Please wait. Processing...

Set DCP - On

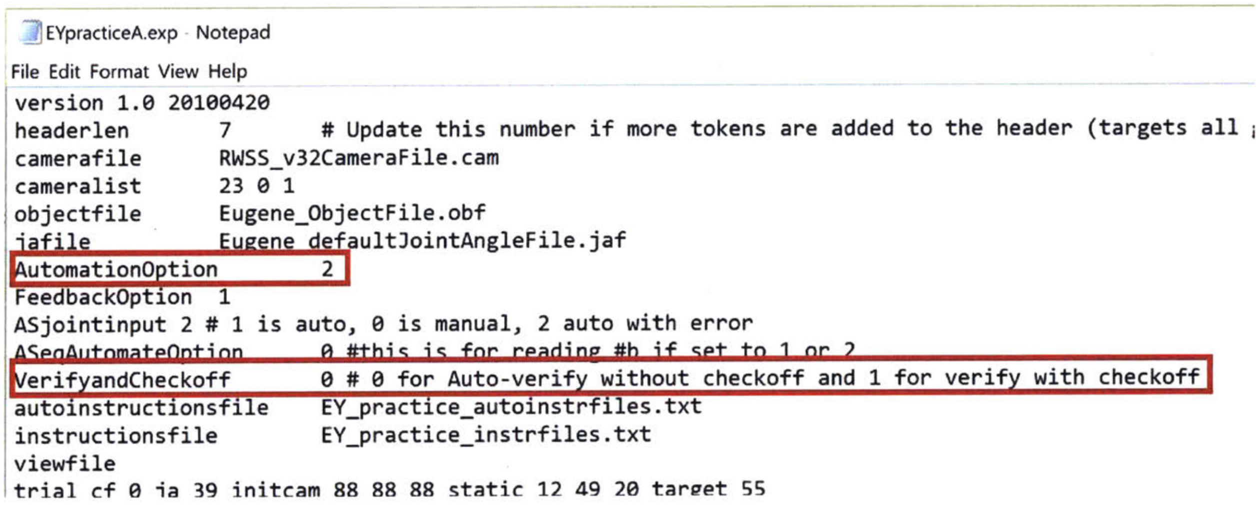
Remaining Time: 16.5 sec

17.6% completed

*Figure 83: Processing Message*

## 6.8 Experiment Automation Options Selection

Provisions that allow the experimenter to tailor the Automation Options as discussed in section 4.4.2 were made. The experimenter does this by editing an experiment setup file. The experimenter specifies which option to use by changing the 'AutomationOption' and 'VerifyandCheckoff' numbers in the experiment configuration file as reflected in Figure 84 (boxed red). For example, reference to Table 6, if the experimenter wants 'Option 2(0)', he/she inputs '2' in the AutomationOption row and '0' in the VerifyandCheckoff row.

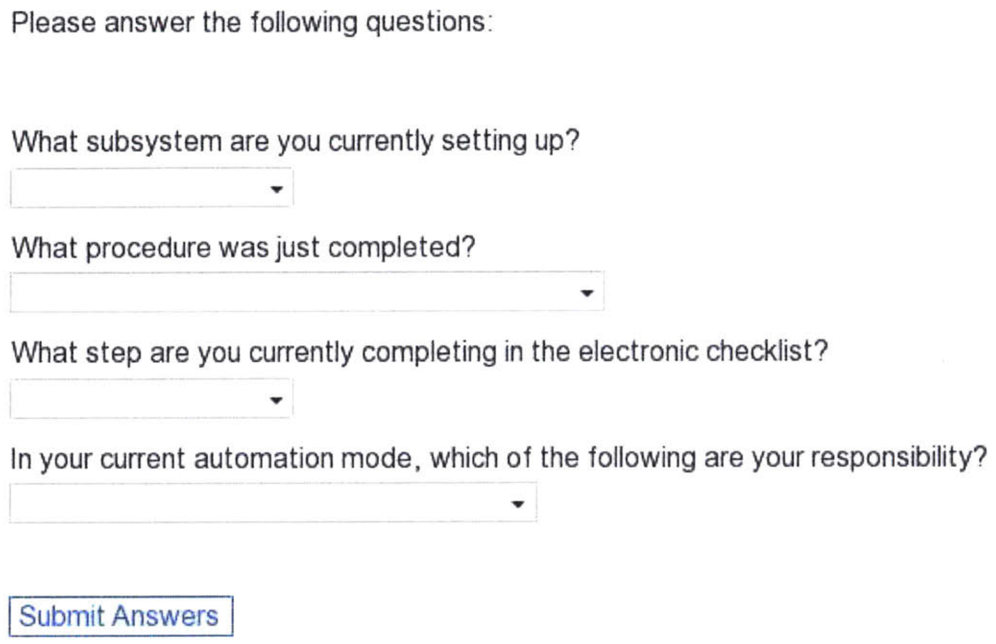


```
version 1.0 20100420
headerlen       7       # Update this number if more tokens are added to the header (targets all
camerafile      RWSS_v32CameraFile.cam
cameralist      23 0 1
objectfile      Eugene_ObjectFile.obf
jafile          Eugene_defaultJointAngleFile.jaf
AutomationOption            2
FeedbackOption  1
ASjointinput 2 # 1 is auto, 0 is manual, 2 auto with error
ASegAutomateOption          0 #this is for reading #h if set to 1 or 2
VerifyandCheckoff           0 # 0 for Auto-verify without checkoff and 1 for verify with checkoff
autoinstructionsfile        EY_practice_autoinstrfiles.txt
instructionsfile            EY_practice_instrfiles.txt
viewfile
trial cf 0 ia 39 initcam 88 88 88 static 12 49 20 target 55
```

*Figure 84: Experiment Configuration File*

## 6.9 Situation Awareness Questionnaire Display

Determination of the operator situation awareness (SA) was required for the experiment. To determine the SA of the subject, the Situation Awareness Global Assessment Technique (SAGAT) [64] was adopted. The experimenter provides a list of SA questions in a file. At the appropriate time during the experiment, the simulation pauses, the displays are blanked out and a randomly chosen subset of the questions appear as shown in Figure 85. The operator is required to answer the questions before resuming the experiment.


Please answer the following questions:

What subsystem are you currently setting up?

What procedure was just completed?

What step are you currently completing in the electronic checklist?

In your current automation mode, which of the following are your responsibility?

*Figure 85: SAGAT Popup*

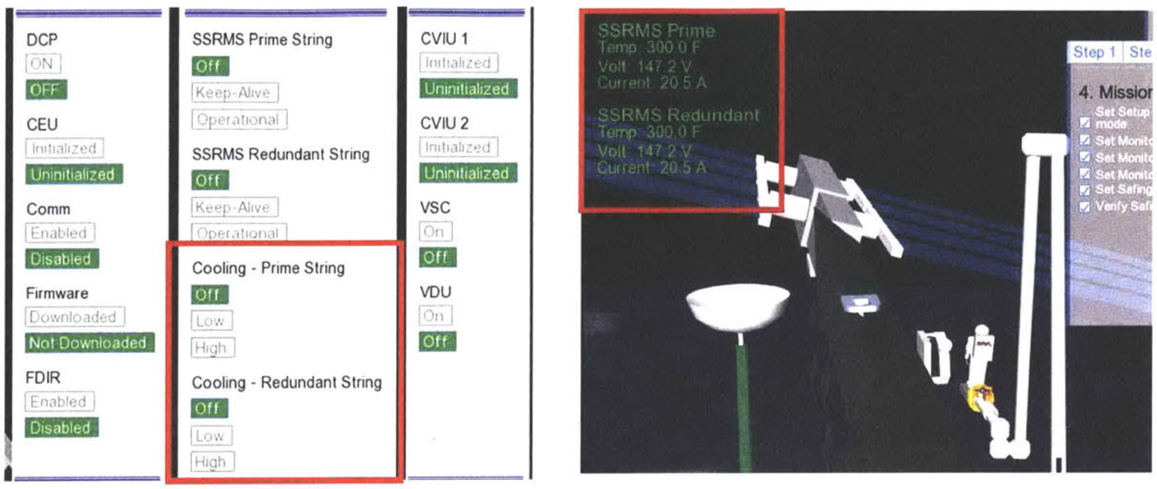
### 6.10 Additional Features to Increase Complexity

Two addition of two features – 1) SSRMS utilities, and cooling displays and controls and 2) Selectable Vernier rates – were added. These features were added after the OPM model development to increase the monitoring workload and task complexity of the operator.

#### 6.10.1 SSRMS Utilities and Cooling Displays and Controls

An overlay was created to show the temperature, current and voltage of the SSRMS on the center monitor and cooling system displays and controls were added to the control panel as presented in Figure 86. The subjects are expected to monitor the temperature, current and voltage of the SSRMS. A simple logic was coded in the simulator that triggers the temperature to rise gradually after the **SSRMS**

**Prime/Redundant String** is **operational** for a duration (e.g 5 minutes). The operator is required to set the cooling **system** to "High" to maintain the temperature in the safe range.



DCP
ON
OFF

CEU
Initialized
Uninitialized

Comm
Enabled
Disabled

Firmware
Downloaded
Not Downloaded

FDIR
Enabled
Disabled

SSRMS Prime String
Off
Keep-Alive
Operational

SSRMS Redundant String
Off
Keep-Alive
Operational

Cooling - Prime String
Off
Low
High

Cooling - Redundant String
Off
Low
High

CVIU 1
Initialized
Uninitialized

CVIU 2
Initialized
Uninitialized

VSC
On
Off

VDU
On
Off

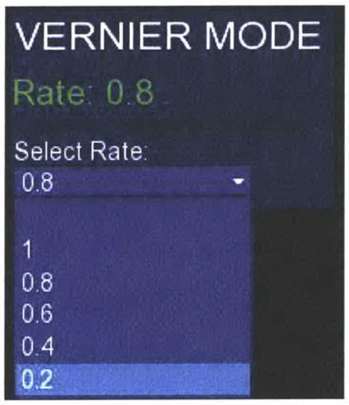
SSRMS Prime
Temp: 300.0 F
Volt: 147.2 V
Current: 20.5 A

SSRMS Redundant
Temp: 300.0 F
Volt: 147.2 V
Current: 20.5 A

Figure 86: Control Panel with Cooling (Left) and SSRMS Utilities Information Overlay (Right)

### 6.10.2 Selectable Vernier Rate

The operator has to set the arm **Speed Mode** from **normal** to **vernier** (see Figure 104 in Appendix B) as part of the Autosequence setup. To increase the task complexity, when in **vernier**, the operator must now select one of the five speed rates as displayed in Figure 87. The Vernier Rate changes the speed of the SSRMS movement and the procedure calls for several changes during the experiment.



VERNIER MODE
Rate: 0.8
Select Rate: 0.8

Figure 87: Vernier Model with Vernier Rate Selection

## 6.11 Summary

Many enhancements were added to the MIT-RWSS as part of this thesis. The importance of the OPM model and how it guides the development of the simulator were emphasized in this chapter, in particularly, the Control Panel design including the Operation Resume Assistant and the Smart Electronic Procedures. This chapter also covers other enhancements such as feedback to operator and SA questionnaire. The simulator has also met the system requirements stated in Section 2.4.2.

# 7 SUMMARY AND DISCUSSION

## 7.1 Conclusions

This thesis had two objectives: One was to evaluate the advantages and disadvantages of Object Process Methodology (OPM) as compared to conventional human factors task analysis techniques. The second was to integrate an electronic procedure system into an existing space telerobotic simulator and to incorporate several novel automation features.

### *7.1.1 Comparison of OPM with Task Analysis*

OPM [56] is a technique recently developed in the field of system engineering to graphically describe and mathematically model complex systems in a simple, intuitive and computable way. However, it has not previously been applied to human factors engineering analysis and design problems. In this thesis, two methods – traditional Task Analysis and OPM – were used separately to describe the procedural steps required to operate a complex space telerobotic system resembling that on the International Space Station, including new subsystems, displays and controls required to implement a new type of electronic procedures system. Applying both traditional task analysis methods and OPM to the same human factors engineering analysis and design problem allowed the author to compare the pros and cons of these methods.

#### *7.1.1.1 Task Analysis and its Limitation*

HFE practitioners often use multiple methodologies e.g. Hierarchical Task Analysis (HTA), Tabular Task Analysis (TTA) and Abstraction Hierarchy (AH) to support analysis and design. As reviewed in section 3.2 (pg. 41), when applied to a complex space telerobotic task, HTA yielded a hierarchical description of activities needed to achieve a system goal in a graphical or list formats spanning many pages. However, since the HTA methodology does not require the description of criteria or conditions, hardware components (e.g., switches), and displays needed for each task (see section 3.3, pg. 47), the outcomes of the HTA were insufficient for detailed system analysis and design. Moreover, even for the same operation, different off-nominal scenarios, such as recovery from failure, have to be represented separately from the nominal, "sunny day" operation. TTA was used to augment the HTA analysis by defining additional information, such as the displays and controls required, the information feedback, task enablers, and other details needed in a columnar format (see section 3.4, pg. 56). Although a copious amount of information can be incorporated into tables, such tables could span many pages, making it difficult to navigate and comprehend the entire system and specifically how humans and machines in the system interact. The purpose of each subsystem and component are also not explicitly stated in the HTA and TTA analysis. Furthermore, even the combined detailed HTA and TTA analysis did not enable to fully enumerate the relationships between the subsystem states and the dependencies between tasks and could not answer questions like "Is subsystem A being set to 'On' in Task A a prerequisite of Task B, or is it the case that the designer simply preferred Task B to happen after Task A in the procedure with no apparent reason?" The AH technique described in section 3.5 (pg. 59) complemented the HTA+TTA analysis by providing a "means-ends" rationale between the top level objectives (e.g. perform EVA robotics) and values (maintain EVA astronaut safety) and the physical components. Applied successively, the combined HTA, TTA and AH representations as a whole were relatively comprehensive, but several limitations and deficiencies remained evident. For example, different off-nominal scenarios, such as recovery from failure, had to be represented separately from the nominal, "sunny day" operation and required separate analysis (section 3.3.4, pg. 53). The HTA, TTA and AH models did not fully represent the relationships and interactions among subsystem and components, nor did they specify the conditions for each activity to proceed and terminate successfully. Consequently, the combined HTA, TTA and AH models could not produce an executable or computable description that could be coded and simulated to prove and certify the logical correctness and completeness of the procedures. Detection of logical errors is important, since if they go unnoticed in the design stage, and only are detected in prototype testing, they can strongly impact project cost and schedule Moreover, these TA models could not produce information that would be sufficient to be applied to the detailed design stage.

### 7.1.1.2 *OPM and its Advantages over Task Analysis*

OPM is a single technique for conceptual modelling of complex systems by decomposing the overall system goals into a hierarchical set of diagrams. At each level, system elements are described in terms of basic "object" and "process" building blocks and their interrelationships by "links" (section 4.2, pg. 67). Objects, processes, and links are graphically described using a syntax with a total of only 18 elements, making it relatively easy to learn. Objects and processes are represented together in a single framework, as opposed to other modeling languages often used in software-development-centric applications such as Matlab/Simulink, UML and SysML. Since many human factors practitioners have little formal background in computer science, the simplicity of OPM is a significant advantage. Unlike task analysis, OPM methodology allows different scenarios to be incorporated in a single model (section 5.3.4, pg. 117). OPM models are created interactively using the OPCAT[6] software and are therefore "computable" in the sense that the analyst can run an "animation" of the graphical model, and watch as various processes in the task hierarchy diagrams become active, and change the states of associated objects. As models are built and animated, OPCAT flags various logical errors.

Both HTA (including TTA) and OPM employ hierarchical representations in an effort to reduce apparent system complexity (Figure 52, pg. 111) and can be used to derive the full set of procedure instructions needed for the EVA operations. However, OPM formally represents a larger set of instructions with more complete information about (1) the human agents who are involved in each step, (2) the robotic subsystems and components involved in each step and how their states change by each subprocess, (3) supporting information, such as displays and messages they present for each situation, (4) required preconditions, such as

[6] Object-Process Computer Aided System Engineering tool.

“Turning **System A** to on requires that **System X** be **operational** and (5) post-conditions of processes, such as the creation of a new window panel for performing the subsequent step), (6) the structural relations between objects and the procedural relations between processes and objects, and (7) What to do in cases of malfunction for any step and under various conditions. These relationships are explicitly defined graphically when the analyst creates a series of hierarchical Object Process Diagrams (OPDs) (see Figure 32, pg. 82 for an example). The last item is the most critical, since the system must be ready for all possible failures and precise prescription of how to mitigate each problematic situation is critical in space systems in particular. The model developed prescribed the architecture and logic of the simulator, which was essential for the prototyping effort, including the contingencies and how they are handled.

As the OPDs are created, OPCAT automatically generates Object Process Language (OPL, see Figure 30, pg. 79) – an English language text descriptions of the same logic that the analyst can use to check whether the graphical symbols were chosen appropriately. One could argue that TTA analysis could incorporate much of the same information. However, the format of TTA is relatively unstructured, so the tabular descriptions inevitably would be idiosyncratic, and relationships between resulting system elements become relatively difficult visualize and understand. Notably, in TTA the level of detail represented is ultimately the user’s choice – the TTA framework does not compel the analyst to define system interfaces and/or events in a logically consistent way. In comparison, since OPM models must be computable, the user is forced to define the objects and the changes to the objects’ states associated to the processes, and link them in a logically consistent manner.

The most important advantage of OPM is therefore the structured syntax that enables models developed to be tested (animated) for logical consistency. An unexpected halt highlights fallacies early in the analysis process, rather than later in the design or prototype testing stage. The animation can also be used to help an audience of other analysts, engineers or trainees visualize all the various processes involved. See section 5.3.2 (pg. 115) for additional detail. The OPCAT software also includes an "unfolding" option, producing a diagram showing structural relationships among objects in a single view that is useful in the interface design process, as demonstrated in this thesis. See section 5.3.6 (pg. 119) for examples. In the telerobotic simulator development, the OPM analysis and model structure influenced important portions of the Control Panel design (section 6.4, pg. 132) including the layout, displays and interfaces needed and the Operation Resume Assistant feature (section 6.4.5.1, pg. 138). Also, as detailed in section 6.5.1 (pg. 142), the OPM model was used to derive the Smart Electronic Procedures system, an important enhancement added to the simulator. OPM does not compel the analyst to formally consider domain values the way AH does, but they can nonetheless be represented in OPM if necessary as shown in section 5.5 (pg. 127), and are sometimes included in other OPM applications in system engineering. In addition to the advantages specifically demonstrated in this thesis, as discussed in section 5.3.7 (pg.121), OPM application in a human factors task analyses could potentially be employed for operator training or practice, workload prediction, or more complex human decision making modeling than those considered here.

The main drawback of OPM for human factors task analysis is that the creation and modification of the OPM model might be time consuming, as they require some proficiency with OPM modelling syntax. As noted in Section 5.4.2 (pg.126), the OPM modeling process required at least 3-4 times as much time as the HTA, TTA and AH analysis. However, considering the potential cost of undetected design or operating procedure errors that are prevented thanks to the OPM model and the other advantages of OPM specified above, the time and cost committed to develop an OPM model is arguably justified. Moreover, once a designer is trained in OPM modeling and "thinking OPM", modeling becomes agile and the benefits increase. For the procedurally complex application considered in this thesis, the OPM was unequivocally superior to HTA, TTA and AH individually or taken together. OPM provides much the same information as a combination of HTA, TTA and AH analyses, yields a computable system simulation, and graphical views of the system that are demonstrably useful in making engineering design decisions. In this respect OPM models represent a much stronger engineering design tool than traditional human factors task analyses.

### *7.1.2 Telerobotic Simulator Enhancements*

Enhancements were made to an existing space telerobotic system. New more detailed subsystem simulations were added to improve procedural realism. A new electronic Control Panel and a Smart Electronic Procedures System were created, and procedures for both nominal and off-nominal situations. Notably, the system guides the operator how to identify specific failures, and then reset appropriate system states to resume normal operations. A variety of automation options in the electronic procedures functions were provided. Experimental evaluation of the latter is the next step in the research.

The operation Control Panel was an important new feature added to the telerobotic simulator, and is detailed in section 6.4 (pg. 132). It is the primary interface used for seven functions: (1) display of system states, (2) change specific system states, (3) selection of command automation, (4) display of system error(s), (5) (optional) automated recovery from failures, (6) procedure selection and (7) (optional) display of reference schematics and diagrams. As detailed in section 6.4.5.1, (pg. 138), after a failure is rectified, the Operation Resume Assistant function provides the operator with a set of logically correct, step by step procedures that will properly restore the pre-failure system configuration.

The "Smart Electronic Procedure" was a second important enhancement, and is described in section 6.5.1 (pg. 142). The OPM model enabled the Smart Electronic Procedure to (1) identify and prevent inadequate actions that are not in line with the procedure instruction, (2) verify if the preconditions for each procedure instruction are met, and prevent its execution even if that was what the procedure instructions told the operator to perform, and (3) automatically execute processes as reflected in the OPM model if all the preconditions are met. It appropriately constrains the actions that the operator can take to ensure procedure adherence and checks the necessary preconditions for every procedural step, reducing the likelihood of operator errors and improving safety. Although it constrains operations, the system also recognizes when to waive the constraints so that operations can proceed seamlessly. For any truly consequential system state change, a warning message appears asking the operator to confirm the action to prevent inadvertent selection from proceeding (section 6.6, pg. 146). The user is also provided feedback on the completion state of an ongoing process (section 6.7, pg. 146). Altogether, twelve different combinations of automation options (section 4.4.2, pg. 105) were incorporated and are available for experimental evaluation.

## 7.2 Future Work

### *7.2.1 OPM and Task Analysis*

This thesis introduces OPM as new tool for human factors engineering practitioners. The thesis focusses on analysis of a procedurally intensive space telerobotic task, and in that context, OPM was evaluated only in comparison to HTA, TTA, and AH analysis methods. OPM has demonstrated advantages, summarized above. However, one certainly should not over-generalize and conclude that OPM can or should replace all traditional task analysis techniques. The early stages of the analysis are quite similar in many respects, and in many ways are potentially complementary. It would be of value to apply OPM and task analyses to several other human factors problems of a different character than space telerobotic, and to include other traditional techniques including other steps of Cognitive Work Analysis (CWA) besides the AH analysis included here. It is also important to demonstrate whether two analysts modelling the same OPM problem produce very different results. It has been recognized (e.g. Stanton et al [6]) that this often not the case for the traditional task analysis methods. The author posits that OPM would not entirely eliminate this issue but the formality and standardized OPM approach likely will reduce the variability of results. (This is a study that could be done in a class room setting). Methods comparison exercises could also be extended to compare OPM analysis in human factors applications with a similar approach using another formal system engineering modeling language, such as UML or Matlab.

Beyond this, experiments could also be carried out to evaluate the potential value of OPM models by engineers for design concept evaluation and by operators as training tools. Conceivably practice with an OPM model of a system and associated procedures could improve training effectiveness because it gives the trainee a broad perspective on the relationship between system goals and procedures, and the interdependencies between various tasks. OPM models could be made available early before a fully functional hardware simulation was built. Also, there may be circumstances where a building a fully functional simulation to train users is impractical. For example, NASA has concerns about how to best train astronaut crews on entirely new tasks long after they have departed from earth. A study could be done to evaluate if adding OPM simulation capability as discussed in section 5.3.7.3 improves training effectiveness.

### *7.2.2 MIT-RWSS Electronic Procedure*

With OPM, the display required for each task is apparent, hence, the simulator could be improved to provide customize display. Customize display refers to highlighting system state displays associated with the task or showing only those required. Experiment can be conducted to determine if customize display that shows only required information would help human performance.

### *7.2.3 Imbed OPM Model in Simulation*

In this thesis, all the important concepts derived from the OPM model were coded into the enhanced space telerobotic simulator manually, but the OPM model itself was not directly integrated into the simulator. However, the OPM model can potentially be embedded within the simulator by including the OPM model as the underlying part of the system software. Thus, the OPM model could be used to monitor the system operations in real time, informing exactly what procedural step is being performed or was just completed, whether all the necessary preconditions were satisfied to begin the next procedural step, what the result of the next procedural step should. It can stop the nominal process if errors were detected and either decide or advise what corrective actions should be performed to resume normal operation. In this sense the model could act as a surrogate for (or in parallel with) the human supervisor. The human would monitor the internal OPM model state estimates, and any other states the OPM model could not, and then choose either to allow the internal OPM model to execute a procedure in full automatic mode, or to intervene and execute the procedure one step at a time in either single step automatic or manual (button pushing) mode. In full "auto" mode, the logic functions as a "model reference" controller with the OPM model serving as an internal model used in the state estimation process. In single step mode, the OPM model could function in a manner analogous to an aircraft Flight Director, telling the human what it is appropriate to do next. Any changes in the logical behavior of the system can be incorporated simply by updating the internal OPM model, rather than having to recode software decision logic, such as that done in this thesis.

# APPENDIX A – EXECUTE EVA OPERATIONS HTA

Table 8 is HTA in hierarchical list format for Execute EVA Operations using SSRMS. Table 9 is the HTA in hierarchical list format for Troubleshoot Failure and Perform Recovery.

*Table 8: Hierarchical List of Execute EVA Operations using SSRMS*

| No. | Task | Plan |
|---|---|---|
| | Execute EVA Operations using SSRMS | do all in sequence 1-4 |
| 1 | Setup the Robotic Arm System | do all in sequence 1-5 |
| 1.1 | Power up Robotic Workstation | do all in sequence 1-5 |
| 1.1.1 | Powerup the Display Control Panel (DCP) | do all in sequence 1-2 |
| 1.1.1.1 | Turn DCP - ON | |
| 1.1.1.2 | Verify DCP power is set as 'ON' | |
| 1.1.2 | Initialize the Control Electronics Unit (CEU) | do all in sequence 1-2 |
| 1.1.2.1 | Initialize CEU | |
| 1.1.2.2 | Verify that CEU initialization is complete | |
| 1.1.3 | Enable the Communication (Comm) with ISS systems | do all in sequence 1-2 |
| 1.1.3.1 | Enable Comm | |
| 1.1.3.2 | Verify that Comm is enabled | |
| 1.1.4 | Download the WHS Firmware | do all in sequence 1-2 |
| 1.1.4.1 | Download Firmware | |
| 1.1.4.2 | Verify Firmware Download is completed | |
| 1.1.5 | Enable the Failure Detection System (FDIR) | do all in sequence 1-2 |
| 1.1.5.1 | Enable FDIR | |
| 1.1.5.2 | Verify FDIR is enabled | |
| 1.2 | Power up the SSRMS | do all in sequence 1-3 |
| 1.2.1 | SSRMS Prime String Off to Keep-Alive | do all in sequence 1-3 |
| 1.2.1.1 | Set SSRMS Prime String to Keep-Alive. | |
| 1.2.1.2 | Verify status as Keep-Alive | |
| 1.2.1.3 | Record start time of Prime String set as keep-alive | |
| 1.2.2 | SSRMS Redundant string Off to keep-Alive | do all in sequence 1-2 |
| 1.2.2.1 | Set SSRMS Redundant String to Keep-Alive | |
| 1.2.2.2 | Verify SSRMS Redundant Keep-Alive | |
| 1.2.3 | SSRMS Prime String to Operational | SSRMS is in keep-alive state for less than 10 mins [do not do 1-3] - SSRMS is in keep-alive state for at least 10 mins [do all in sequence 1-3] |

| No. | Task | Plan |
|---|---|---|
| 1.2.3.1 | Set SSRMS Prime String to Operational | |
| 1.2.3.2 | Verify SSRMS Prime Operational | |
| 1.3 | Power up the Video Components | do all in sequence 1-6 |
| 1.3.1 | CVIU 1 and 2 Initialization | do all in sequence 1-4 |
| 1.3.1.1 | Initialize CVIU 1 - wait 45 secs | |
| 1.3.1.2 | Verify CVIU 1 initialization complete | |
| 1.3.1.3 | Power CVIU 2 by pressing 'I' - wait 30 secs | |
| 1.3.1.4 | Verify CVIU 2 initialization complete | |
| 1.3.2 | PDGF VSC Powerup | do all in sequence 1-2 |
| 1.3.2.1 | Switch RPC Position from Open to Close | |
| 1.3.2.2 | Verify RPC Position is Closed | |
| 1.3.3 | Video Distribution Units Powerup | do all in sequence 1-2 |
| 1.3.3.1 | Power up VDU | |
| 1.3.3.2 | Verify VDU is on | |
| 1.3.4 | Verify Camera System is Ready | |
| 1.4 | Setup Display Configuration | do all in sequence 1-3 |
| 1.4.1 | Change Display 1 | do all in sequence 1-2 |
| 1.4.1.1 | Set to Camera 4 | 1; If Camera is not 4 (do all in sequence 2-3) ; otherwise exit |
| 1.4.1.1.1 | Check Current Camera Setting | |
| 1.4.1.1.2 | Press F1 | |
| 1.4.1.1.3 | Input 04 and press enter | |
| 1.4.1.2 | Adjust Camera Field of View | If SSRMS not in FOV (do concurrently 1-2) |
| 1.4.1.2.1 | Pan camera such that whole SSRM is in the field of view | |
| 1.4.1.2.2 | Zoom in/out such that whole arm is in the field of view | |
| 1.4.2 | Change Display 2 | do all in sequence 1-2 |
| 1.4.2.1 | Set to Camera 5 | 1; If Camera is not 5 (do all in sequence 2-3) |
| 1.4.2.1.1 | Check Current Camera Setting | |
| 1.4.2.1.2 | Press F2 | |
| 1.4.2.1.3 | Input 05 and press enter | |
| 1.4.2.2 | Adjust Camera Field of View | If SSRMS not in FOV (do concurrently 1-2) |
| 1.4.2.2.1 | Zoom in/out such that whole arm is in the field of view | |
| 1.4.2.2.2 | Pan camera such that whole arm is in the field of view | |
| 1.4.3 | Change Display 3 | 1; If Camera is not #6 (do all in sequence 2-3) |
| 1.4.3.1 | Check Current Camera Setting | |
| 1.4.3.2 | Set to Camera 6 | do all in sequence 1-2 |

| No. | Task | Plan |
|---|---|---|
| 1.4.3.2.1 | Click F3 | |
| 1.4.3.2.2 | Input 06 and press enter | |
| 1.4.3.3 | Adjust Camera Field of View | do concurrently 1-4 |
| 1.4.3.3.1 | Zoom in/out such that whole arm is in the field of view | |
| 1.4.3.3.2 | Pan camera such that whole arm is in the field of view | |
| 1.5 | Remove Safing | do all in sequence 1-2 |
| 1.5.1 | Press 'S' to remove Safing | |
| 1.5.2 | Verify Safing is removed | |
| 2 | Perform EVA Operation Task 1 | do all in sequence 1-7 |
| 2.1 | Autosequence Setup | do all in sequence 1-3 |
| 2.1.1 | Select Automode ('Shift + A') | |
| 2.1.2 | Enter the joint angles | |
| 2.1.3 | Verify Joint Angles (Confirmation Box) | |
| 2.2 | Set SSRMS Speed | do all in sequence 1-2 |
| 2.2.1 | Set to Vernier ('V button') | |
| 2.2.2 | Verify SSRMS on Vernier mode | |
| 2.3 | Release Brake | do all in sequence 1-2 |
| 2.3.1 | Press 'B' to release brake | |
| 2.3.2 | Verify brake is OFF | |
| 2.4 | Initialize Autosequence | do all in sequence 1-2 |
| 2.4.1 | Review joint angles - Pop up message with the joint angles entered | |
| 2.4.2 | Click 'Confirm' - 'proceed button to press' | |
| 2.5 | Activate Brake | Wait until LEE reached designated location [ 1] |
| 2.5.1 | Turn brake 'ON' ('B Button') | |
| 2.6 | Click 'M' for EVA to work | |
| 2.7 | Verify is EVA Mission Completed | |
| 3 | Perform EVA Operation Task 2 and up | do all in sequence 1-8 and repeat until EVA mission is completed |
| 3.1 | Setup Display Configuration | do all in sequence 1-3 |
| 3.1.1 | Change Display 1 | do all in sequence 1-2 |
| 3.1.1.1 | Set to Camera 4 | 1; If Camera is not 4 (do all in sequence 2-3) ; otherwise exit |
| 3.1.1.1.1 | Check Current Camera Setting | |
| 3.1.1.1.2 | Press F1 | |
| 3.1.1.1.3 | Input 04 and press enter | |
| 3.1.1.2 | Adjust Camera Field of View | If SSRMS not in FOV (do concurrently 1-2) |

| No. | Task | Plan |
|---|---|---|
| 3.1.1.2.1 | Pan camera such that whole SSRM is in the field of view | |
| 3.1.1.2.2 | Zoom in/out such that whole arm is in the field of view | |
| 3.1.2 | Change Display 2 | do all in sequence 1-2 |
| 3.1.2.1 | Set to Camera 5 | 1; If Camera is not 5 (do all in sequence 2-3) |
| 3.1.2.1.1 | Check Current Camera Setting | |
| 3.1.2.1.2 | Press F2 | |
| 3.1.2.1.3 | Input 05 and press enter | |
| 3.1.2.2 | Adjust Camera Field of View | If SSRMS not in FOV (do concurrently 1-2) |
| 3.1.2.2.1 | Zoom in/out such that whole arm is in the field of view | |
| 3.1.2.2.2 | Pan camera such that whole arm is in the field of view | |
| 3.1.3 | Change Display 3 | 1; If Camera is not #6 (do all in sequence 2-3) |
| 3.1.3.1 | Check Current Camera Setting | |
| 3.1.3.2 | Set to Camera 6 | do all in sequence 1-2 |
| 3.1.3.2.1 | Click F3 | |
| 3.1.3.2.2 | Input 06 and press enter | |
| 3.1.3.3 | Adjust Camera Field of View | do concurrently 1-4 |
| 3.1.3.3.1 | Zoom in/out such that whole arm is in the field of view | |
| 3.1.3.3.2 | Pan camera such that whole arm is in the field of view | |
| 3.2 | Autosequence Setup | do all in sequence 1-3 |
| 3.2.1 | Select Automode ('Shift + A') | |
| 3.2.2 | Enter the joint angles | |
| 3.2.3 | Verify Joint Angles (Confirmation Box) | |
| 3.3 | Set SSRMS Speed | do all in sequence 1-2 |
| 3.3.1 | Set to Vernier ('V button') | |
| 3.3.2 | Verify SSRMS on Vernier mode | |
| 3.4 | Release Brake | do all in sequence 1-2 |
| 3.4.1 | Press 'B' to release brake | |
| 3.4.2 | Verify brake is OFF | |
| 3.5 | Initialize Autosequence | do all in sequence 1-2 |
| 3.5.1 | Review joint angles - Pop up message with the joint angles entered | |
| 3.5.2 | Click 'Confirm' - 'proceed button to press' | |
| 3.6 | Activate Brake | Wait until LEE reached designated location [ 1] |
| 3.6.1 | Turn brake 'ON' ('B Button') | |
| 3.7 | Click 'M' for EVA to work | |

| No. | Task | Plan |
|---|---|---|
| 3.8 | Verify is EVA Mission Completed | |
| 4 | Shutdown the Robotic System | do all in sequence 1-3 |
| 4.1 | Video System Shut down | 1 |
| 4.1.1 | CVIU un-initialization | do all in sequence 1-3 |
| 4.1.1.1 | Turn off CVIU 1 (un-initialization) | |
| 4.1.1.2 | Verify CVIU 1 and 2 un-initialized | |
| 4.1.1.3 | Verify VSC and VDU are off | |
| 4.2 | SSRMS Shut down | do all in sequence 1-2 |
| 4.2.1 | SSRMS Prime String to Off | do all in sequence 1-2 |
| 4.2.1.1 | Set SSRMS Prime String to off | |
| 4.2.1.2 | Verify SSRMS Prime is off | |
| 4.2.2 | SSRMS Redundant to Off | do all in sequence 1-2 |
| 4.2.2.1 | Set SSRMS Redundant String to off | |
| 4.2.2.2 | Verify SSRMS Redundant is off | |
| 4.3 | RWS Shut Down | do all in sequence 1-2 |
| 4.3.1 | CEU un-initialization | do all in sequence 1-3 |
| 4.3.1.1 | Turn off CEU | |
| 4.3.1.2 | Verify CEU off | |
| 4.3.1.3 | Verify Comm disable | |
| 4.3.2 | DCP shut down | do all in sequence 1-2 |
| 4.3.2.1 | Turn off DCP | |

*Table 9: Hierarchical List for Troubleshoot Failure and Perform Recovery*

| No. | Task | Plan |
|---|---|---|
| | Troubleshoot Failure and Perform Recovery | Safing OFF ( 1) ; do all in sequence 2-3 |
| 1 | Set System to Safe-mode | do all in sequence 1-2 |
| 1.1 | Set Safing to ON | |
| 1.2 | Verify Safing Status | |
| 2 | Troubleshoot and Rectify Failure | do all in sequence 1-2 |
| 2.1 | Determine System Error/Failure | do all in sequence 1-2 |
| 2.1.1 | Open Error Message Box | |
| 2.1.2 | Identify the Error | |
| 2.2 | Recover from failure stated in Message Box | Error in Communication ( 1) ; Error in Videofeed ( 2) |
| 2.2.1 | Recovery Communication Failure | do all in sequence 1-3; If Error Message Remains (4) ; 5 |
| 2.2.1.1 | Reinitialize CEU | do all in sequence 1-3 |
| 2.2.1.1.1 | Uninitialize CEU | |
| 2.2.1.1.2 | Verify CEU Uninitialized | |
| 2.2.1.1.3 | Re-initilize CEU | |
| 2.2.1.1.4 | Verify CEU is initialized | |
| 2.2.1.2 | Comm Enable | do all in sequence 1-2 |
| 2.2.1.2.1 | Enable Comm | |
| 2.2.1.2.2 | Verify that Comm is Enable | |
| 2.2.1.3 | Check Error Message is Cleared | Error Message Remains [ 1] |
| 2.2.1.4 | Switch to SSRMS Redundant String | do all in sequence 1-2 |
| 2.2.1.4.1 | Change SSRMS Prime String to OFF | do all in sequence 1-2 |
| 2.2.1.4.1.1 | Set SSRMS Prim String from Operational to OFF | |
| 2.2.1.4.1.2 | Verify SSRMS Prime String is OFF | |
| 2.2.1.4.2 | Change SSRMS Redundant String from Keep-Alive to Operational | do all in sequence 1-2 |
| 2.2.1.4.2.1 | Set Redundant String from Keep-Alive to Operational | |
| 2.2.1.4.2.2 | Verify SSRMS Prime String is Operational | |
| 2.2.1.5 | Verify RWS and SSRMS communication | If Needed ( 1) ; do all in sequence 2-5 |
| 2.2.1.5.1 | Configure Cameras as Required | |
| 2.2.1.5.2 | Removing Safing | do all in sequence 1-2 |
| 2.2.1.5.2.1 | Press "S" to remove Safing | |
| 2.2.1.5.2.2 | Verify Safing is removed | |
| 2.2.1.5.3 | Release Brake | do all in sequence 1-2 |
| 2.2.1.5.3.1 | Press"B" to release brake | |
| 2.2.1.5.3.2 | Verify brake is OFF | |
| 2.2.1.5.4 | Deflect THC to ensure SSRMS move | |

| No. | Task | Plan |
|---|---|---|
| 2.2.1.5.5 | Engage Brake | do all in sequence 1-2 |
| 2.2.1.5.5.1 | Set Brake to ON | |
| 2.2.1.5.5.2 | Verify Brake is ON | |
| 2.2.2 | Videofeed Error Recovery | do all in sequence 1-4; If Needed ( 5) |
| 2.2.2.1 | CVIU Re-powering | |
| 2.2.2.1.1 | Uninitialize CVIU 1 | |
| 2.2.2.1.2 | Set CVIU 1 to initialize | |
| 2.2.2.1.3 | Verify CVIU 1 is initialized | |
| 2.2.2.1.4 | Set CVIU 2 to initialize | |
| 2.2.2.1.5 | Verify CVIU 2 is initialized | |
| 2.2.2.2 | VSC Re-Powerup | do all in sequence 1-2 |
| 2.2.2.2.1 | Set VSC to ON | |
| 2.2.2.2.2 | Verify VSC is ON | |
| 2.2.2.3 | Video Distribution Unit (VDU) Powerup | do all in sequence 1-2 |
| 2.2.2.3.1 | Power up VDU | |
| 2.2.2.3.2 | Verify VDU is powerup | |
| 2.2.2.4 | Verify Camera System Stauts is Ready | |
| 2.2.2.5 | Check Error Message is Cleared | |
| 2.2.2.6 | Reconfigure Camera Setting as required | |
| 2.2.2.7 | Removing Safing | do all in sequence 1-2 |
| 2.2.2.7.1 | Set Safing to OFF | |
| 2.2.2.7.2 | Verify Safing is OFF | |
| 3 | Return to Operations | do any one 1-2 |
| 3.1 | Continue with Manual Operations | |
| 3.2 | Re-Setup of JOCAS - Return to main procedure | |

# APPENDIX B - OPM MODEL DETAILS

1

## 1.1 Example of OPM Things and Links

Figure 88 is a simple OPM example to illustrate the OPM things and links discussed above.

Raw materials are consumed by Manufacturing to create a product – the Computer. The Manufacturing process requires a Factory. The Computer can be in either **off** or **on** state. Powering changes the computer from **off** to **on** state and Person A (a human enabler) is required for the Powering process to trigger i.e. someone needs to switch on the computer.

Desktop and Laptop are different forms of computer. A Desktop can be decomposed into parts such as Keyboard**,** Computer Case and Harddisk. The Harddisk is attached to the Computer Case and the Computer Case holds the Harddisk in place as reflected by the tags on the bi-directional arrow. The Laptop can have characteristic/features of Display and Operating Systems.

Thesis Writing is a process that requires the Computer to be in **on** state, a human – Person A needs to write the thesis using a Keyboard. Writing a thesis also requires the MS Words program that is a form of Writer, which is a feature of the Desktop and the process affects the Thesis**'s** states as shown by the double headed arrow.

The model also depicts the essence of Raw Material, Computer, Factory, Person A, Desktop, Laptop, Keyboard, Computer Case and Harddisk as physical, and remaining objects and processes as informatical. All the OPM things drawn are systemic besides Thesis Writing and Person A, which are environmental because they are not part of the computer system. Figure 89 shows the OPL automatically generated based on the OPD in Figure 89.

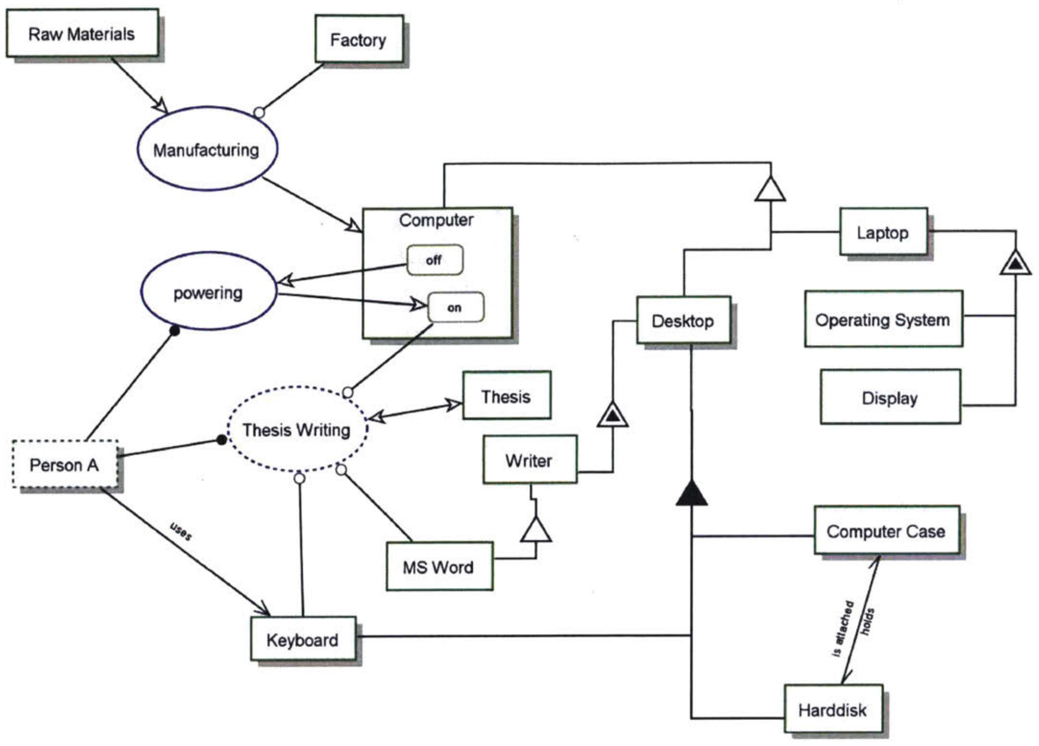


*Figure 88: OPM Example*

Raw Materials is physical.
Computer is physical.
Computer can be off or on.
Person A is environmental and physical.
Person A uses Keyboard.
Person A handles Thesis Writing and powering.
Factory is physical.
Laptop is physical.
Laptop is a Computer.
Laptop exhibits Operating System and Display.
Desktop is physical.
Desktop is a Computer.
Desktop exhibits Writer.
Desktop consists of Keyboard, Computer Case, and Harddisk.
    Keyboard is physical.
    Computer Case is physical.
    Computer Case holds Harddisk.
    Harddisk is physical.
    Harddisk is attached Computer Case.
MS Word is a Writer.
Manufacturing requires Factory.
Manufacturing consumes Raw Materials.
Manufacturing yields Computer.
powering changes Computer from off to on.
Thesis Writing is environmental.
Thesis Writing requires MS Word, Keyboard, and on Computer.
Thesis Writing affects Thesis.

*Figure 89: OPL for OPM Example*

## 1.2 System Setup

### *1.2.1 RWS Powering*

Figure 90 to Figure 93 are the in-zoomed of the remaining subprocesses in RWS Powering that are like **Display and Control Panel Powering** in-zoomed discussed above. One nuance is the **Verifying** subprocess in **FDIR Enabling** in-zoomed (Figure 93) changes the **RWS Setup** state from **not complete** to **complete**, which marks the completion of RWS setup.

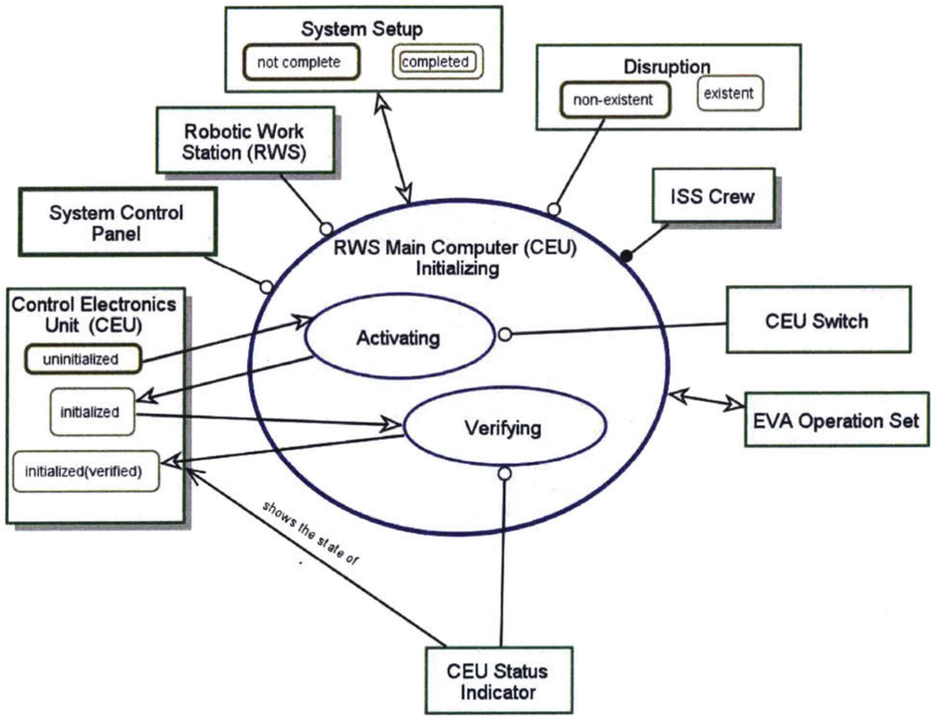


*Figure 90: OPD of RWS Main Computer (CEU) Initializing In-Zoomed*

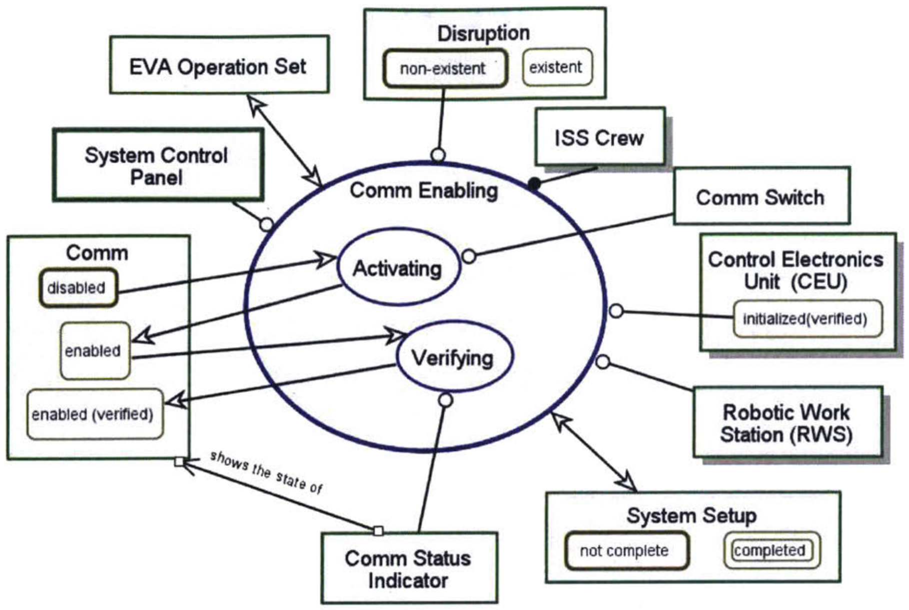


*Figure 91: OPD of Comm Enabling In-Zoomed*

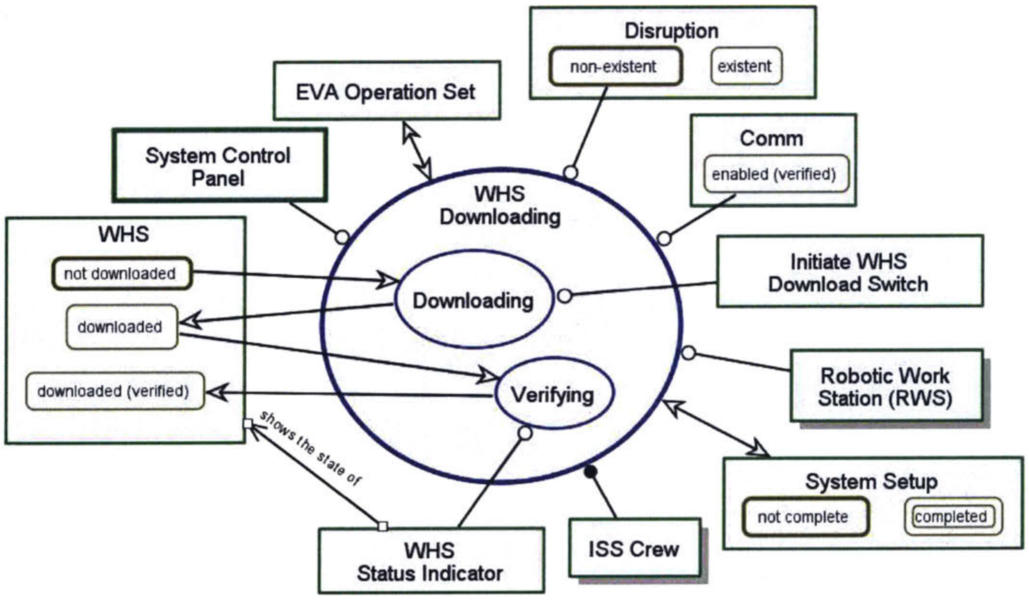


*Figure 92: OPD of WHS (Firmware) Downloading In-Zoomed*

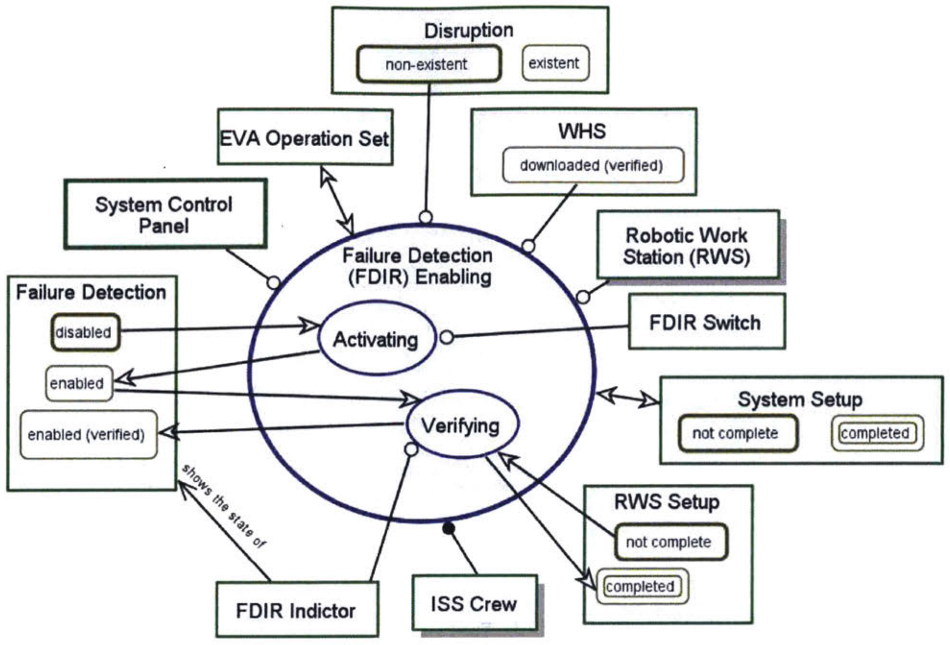


*Figure 93: OPD of Failure Detection (FDIR) Enabling In-Zoomed*

These OPDs define the procedure as follows:

[1] Set CEU – Initialized

[2] Verify CEU – Initialized

[3] Set Comm – Enable

[4] Verify Comm – Enabled

[5] Set WHS (Firmware) - Download

[6] Verify WHS (Firmware) – Downloaded

[7] Set FDIR – Enabled

[8] Verify FDIR – Enabled

### *1.2.2 SSRMS Powering In-Zoomed*

Figure 94 presents the in-zoomed view of **SSRMS Powering**. **SSRMS Powering** requires **RWS Setup** to be **completed** and **CEU** to be **initialized(verified).** Although **SSRMS Powering** happens after **RWS Setup**, **CEU** could be uninitialized after **RWS Setup** by accident or system failure. Thus, it is necessary for the simulation program to include **CEU** in **initialized(verified)** as a condition for **SSRMS Powering**.

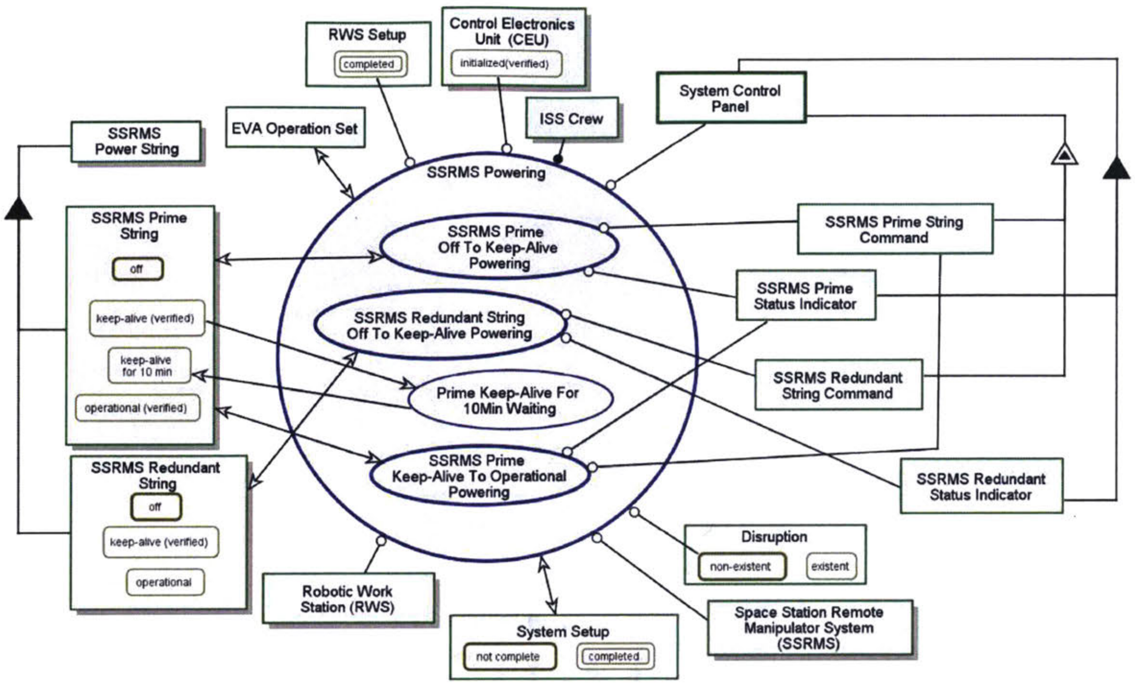

RWS Setup
completed
Control Electronics Unit (CEU)
initialized(verified)
ISS Crew
System Control Panel
SSRMS Power String
EVA Operation Set
SSRMS Powering
SSRMS Prime String
off
keep-alive (verified)
keep-alive for 10 min
operational (verified)
SSRMS Prime Off To Keep-Alive Powering
SSRMS Redundant String Off To Keep-Alive Powering
Prime Keep-Alive For 10Min Waiting
SSRMS Prime Keep-Alive To Operational Powering
SSRMS Prime String Command
SSRMS Prime Status Indicator
SSRMS Redundant String Command
SSRMS Redundant Status Indicator
SSRMS Redundant String
off
keep-alive (verified)
operational
Disruption
non-existent
existent
Robotic Work Station (RWS)
System Setup
not complete
completed
Space Station Remote Manipulator System (SSRMS)

*Figure 94: OPD of SSRMS Powering In-Zoomed*

**SSRMS Power String** is decomposed to **SSRMS Prime String** and **SSRMS Redundant String**. These are two identical power and data cables for redundancy. The **SSRMS Prime String Command** and **SSRMS Redundant String Command** are capabilities in the **System Control Panel** used to change the state of the **SSRMS Prime String** and **SSRMS Redundant String** respectively.

Status of the **SSRMS Power String** state are provided by **SSRMS Prime Status Indicator** and **SSRMS Redundant Status Indication** in the **System Control Panel**. All the subprocesses beside **SSRMS Redundant String Off To Keep-Alive Powering** affect the state of **SSRMS Prime String**.

The OPD also shows that the **SSRMS Prime String** need to be in keep-alive for 10mins before it can be switched to operational. This is to ensure the system is sufficiently warm up. Thus, the simulation needs to be programed to prevent the crew from changing **SSRMS Prime String** to **operational** prematurely. The similar restriction can be imposed in the program for SSRMS Redundant String but is not required for this experiment due to its sequence in the procedure.

### *1.2.2.1 In-Zoomed Processes in SSRMS Powering*

Figure 94 to Figure 96 are the in-zoomed of the subprocesses in **SSRMS Powering**. In **SSRMS Prime Off To Keep-Alive Powering** in-zoomed, **Activating** changes the state of **SSRMS Prime String** from **off** to **keep-alive** and **Verifying** changes the state from **keep-alive** to **keep-alive(verified)**. The similar concept applies to **SSRMS Redundant String Off To Keep-Alive Powering** and **SSRMS Prime keep-Alive To Operational Powering**.

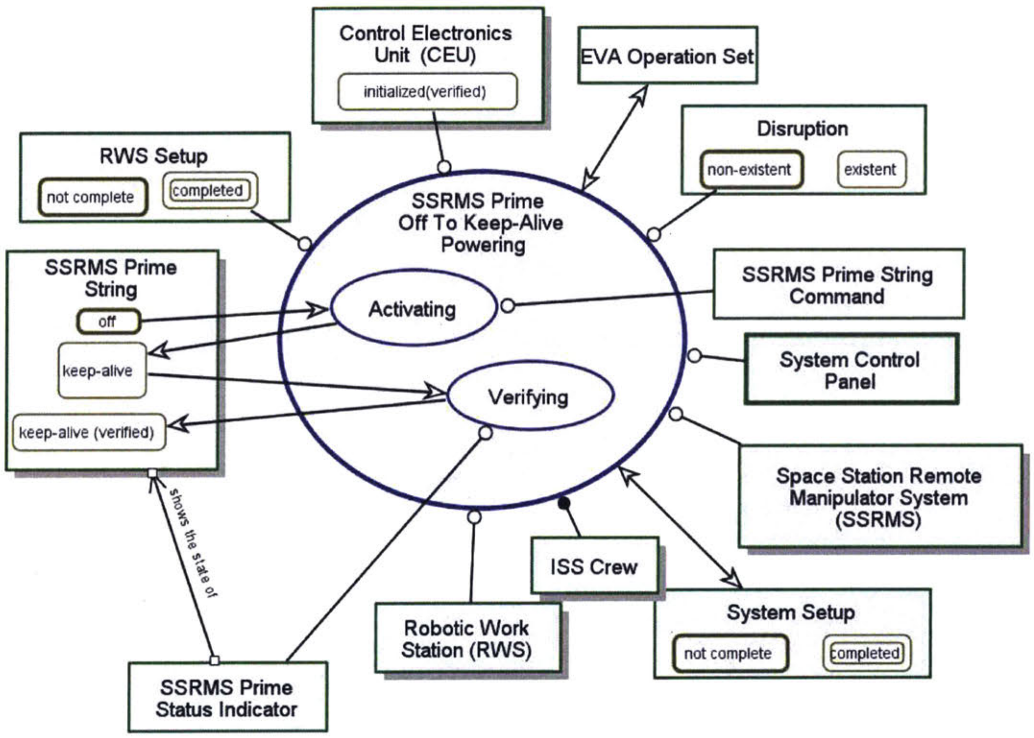


*Figure 95: OPD of SSRMS Prime Off To Keep-Alive Powering*

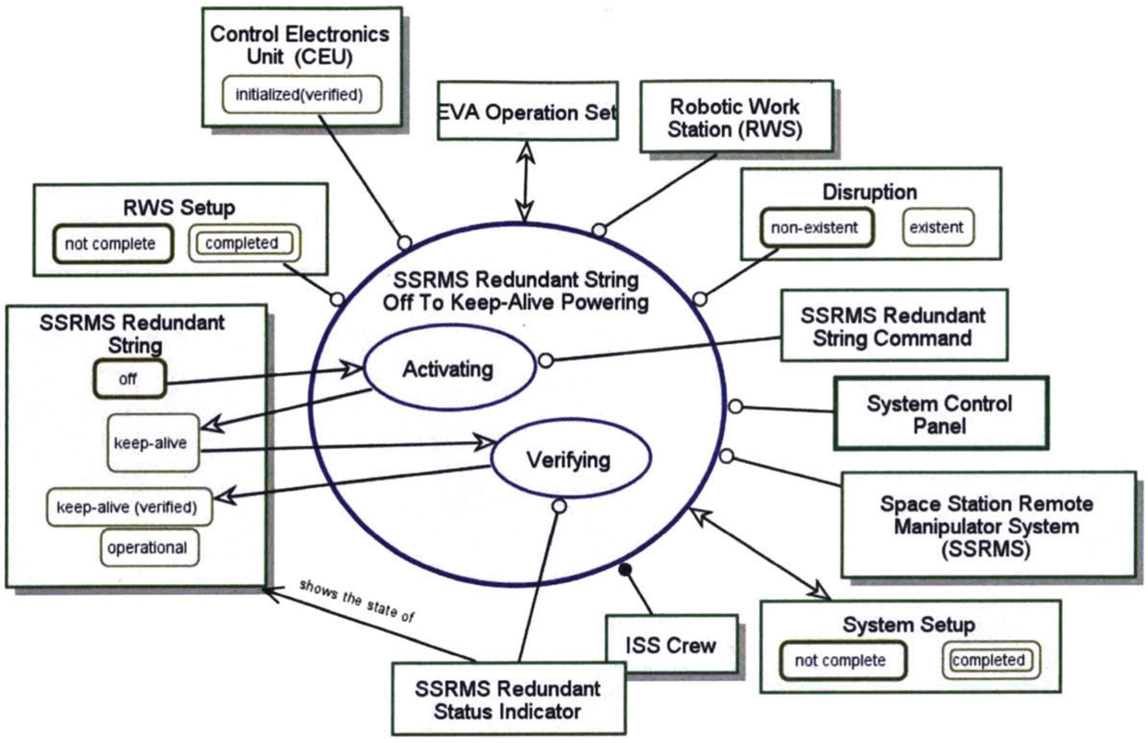


*Figure 96: OPD of SSRMS Redundant String Off to Keep-Alive Powering*

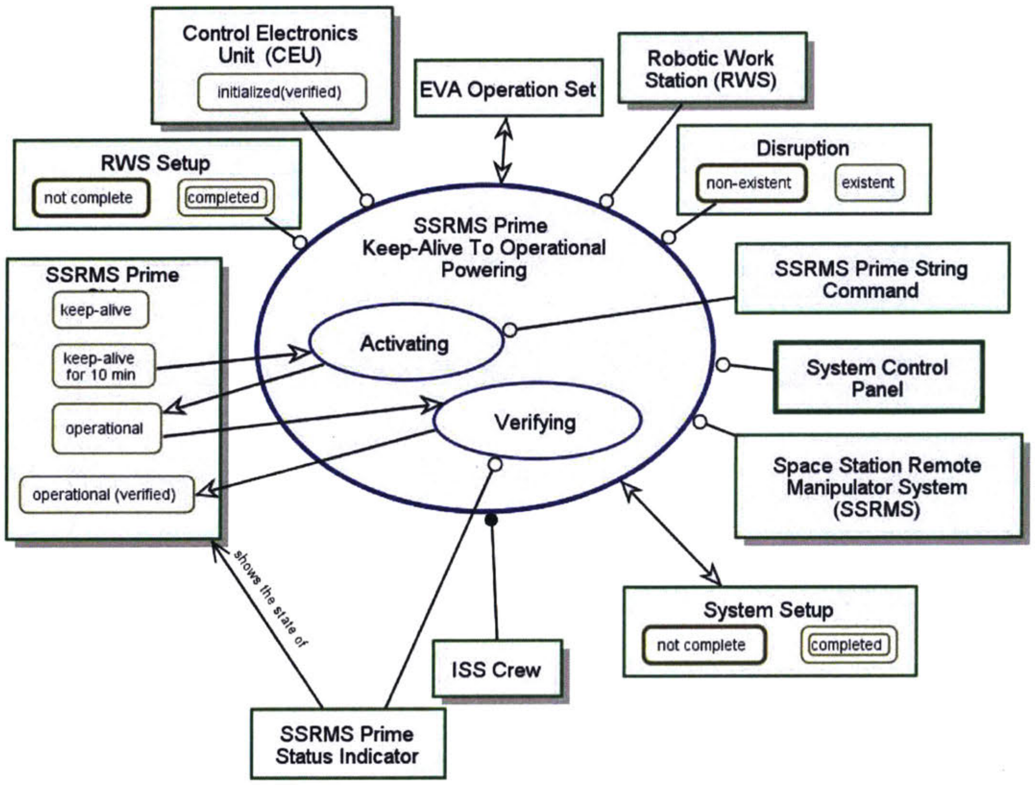


*Figure 97: SSRMS Prime Keep-Alive To Operational Powering*

The OPDs define the procedure as follows:

[1] Set Prime String to Keep-Alive

[2] Verify Prime String – Keep-Alive

[3] Set Redundant String to Keep-Alive

[4] Verify Redundant String – Keep-Alive

[5] Set Prime String Keep-Alive to Operational [Caution: Keep-Alive for at least 10 mins]

[6] Verify Prime String – Operational

### 1.2.3 *Video System Powering*

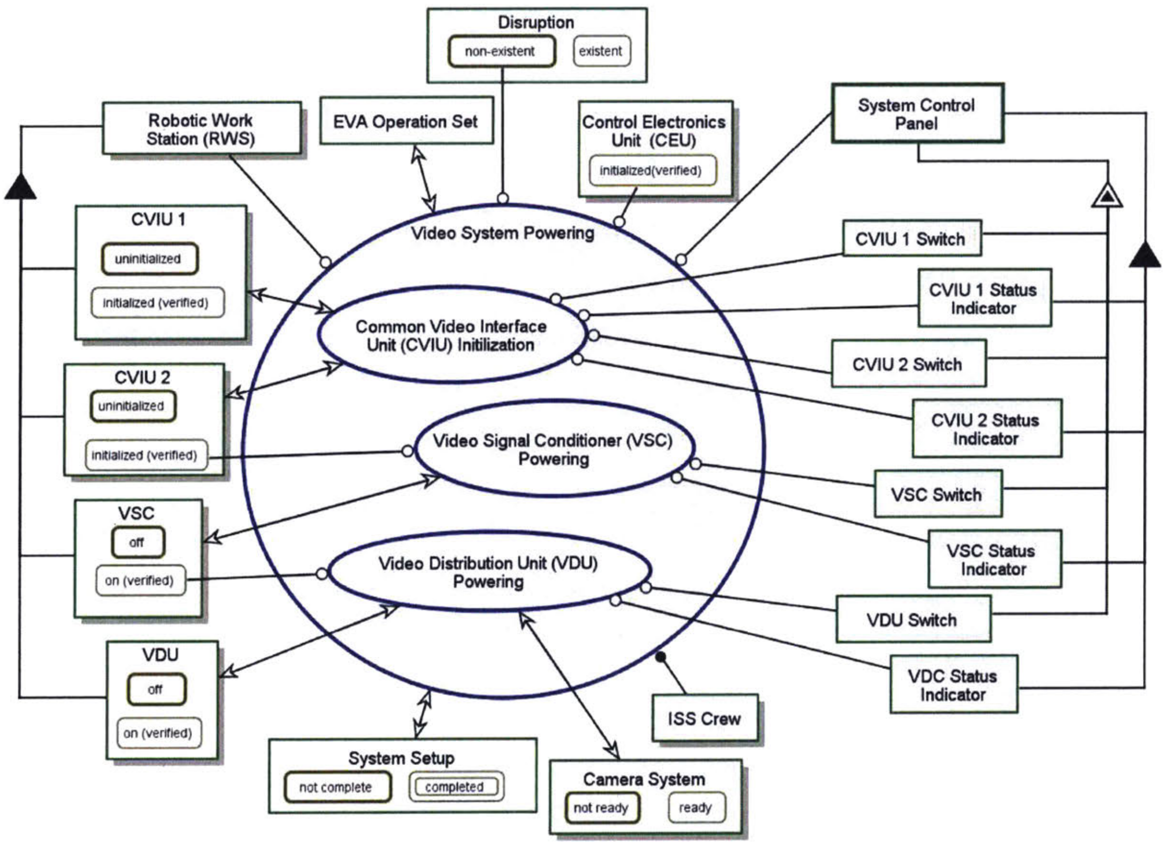


*Figure 98: OPD of Video System Powering In-Zoomed*

**Video System Powering** needs **CEU** to be **initialized(verified)**, hence, the condition has to be programmed in the simulation. **Common Video Interface Unit (CVIU) Initialization** affects **CVIU 1** and **CVIU 2** states and requires switches and status indicator of the systems. There are two CVIU units and both are mandatory to be initialized to support the video system. CVIU 2 in **initialized(verified)** state is the requisite for Video Signal Conditioner (VSC) Powering and VSC in **on(verified)** state is the precondition for Video Distribution Unit (VDU) Powering. These subprocesses require a switch and system indicator. CVIU 1, CVIU 2, VSC and VDU are subsystems of the RWS.

CVIU 1 Switch, CVIU 2 Switch, VSC Switch and VDU Switch are features of the System Control Panel. The System Control Panel also comprises of CVIU 1 Status Indicator, CVIU 2 Status Indicator, VSC Status Indicator and VDU Status Indicator.

#### *1.2.3.1 In-Zoomed of Video System Powering*

Figure 99 to Figure 101 are models of the subprocesses of Video System Powering. Using the CVIU 1 Switch, the subprocess of Initializing can be triggered to change the CVIU 1 state from **uninitialized** to **initialized.** Verifying requires CVIU 1 Status Indicator and it changes the CVIU 1 state to **initialized (verified)**. CVIU 1 **initialized (verified)** is a pre-condition for CVIU 2 initializing subprocess. The remaining subprocesses are similar to CVIU 1.

VSC Powering and VDU Powering subprocess are similar to CVIU Initialization. Of noteworthy is Verifying in VDU Powering in-zoomed changes the Camera System from **not ready** to **ready**. That means, after this step of the procedure, the cameras are ready for use.

The OPDs define the procedure as follows:

[1] Set CVIU 1 – Initialized
[2] Verify CVIU 1 – Initialized
[3] Set CVIU 2 - Initialized
[4] Verify CVIU 2 – Initialized
[5] Set VSC to On

[6] Verify VSC – On

[7] Set VDU to On

[8] Verify VDU – On

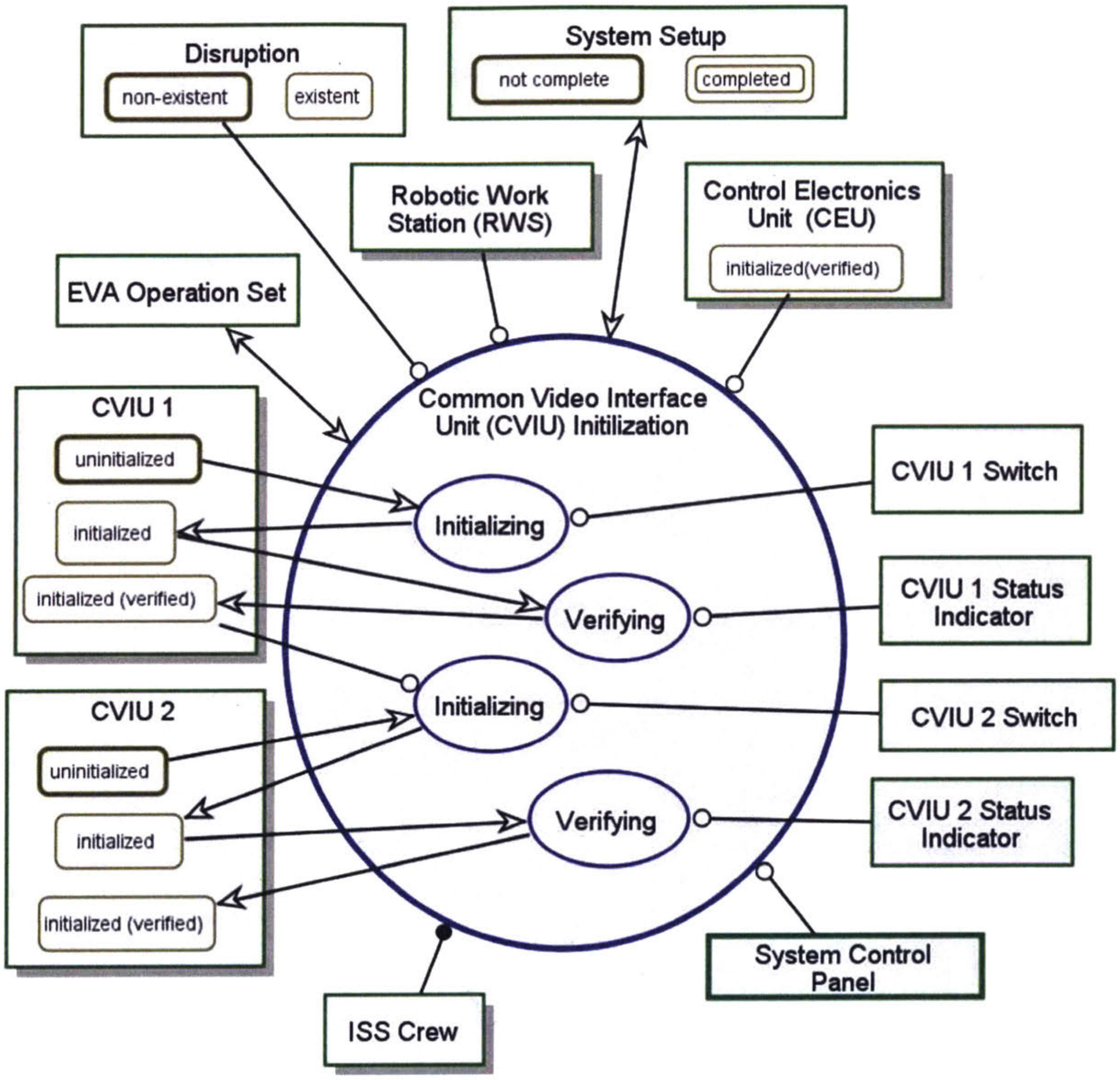


*Figure 99: OPD of Common Video Interface Unit (CVIU) Initialization*

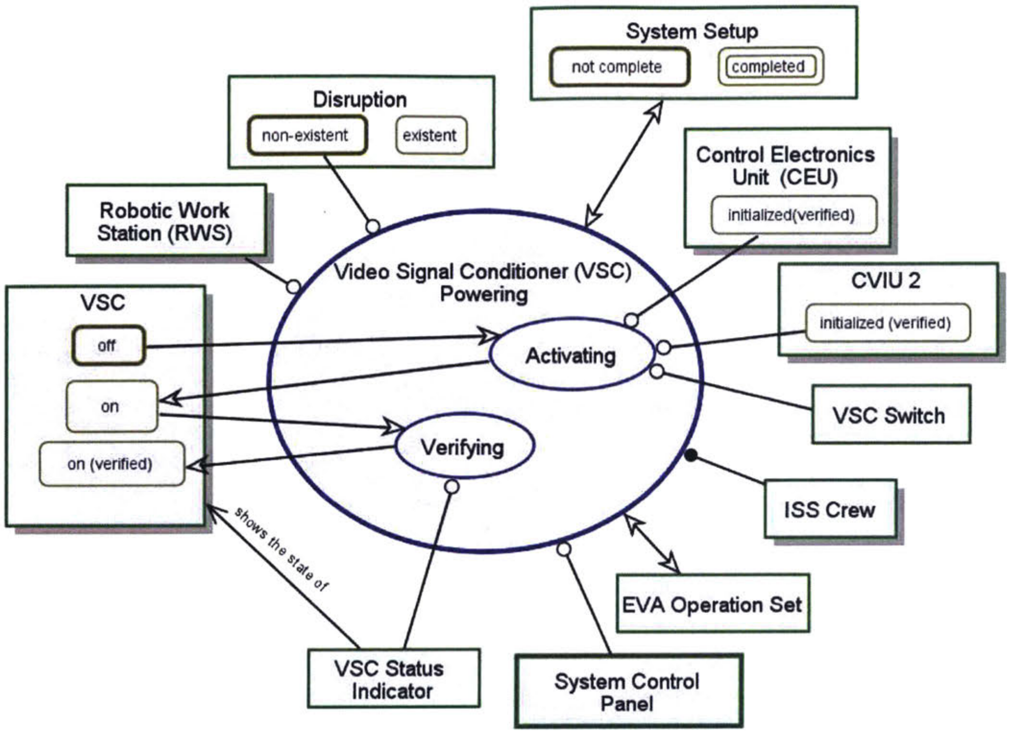


*Figure 100: OPD of Video Signal Conditioner (VSC) Power*

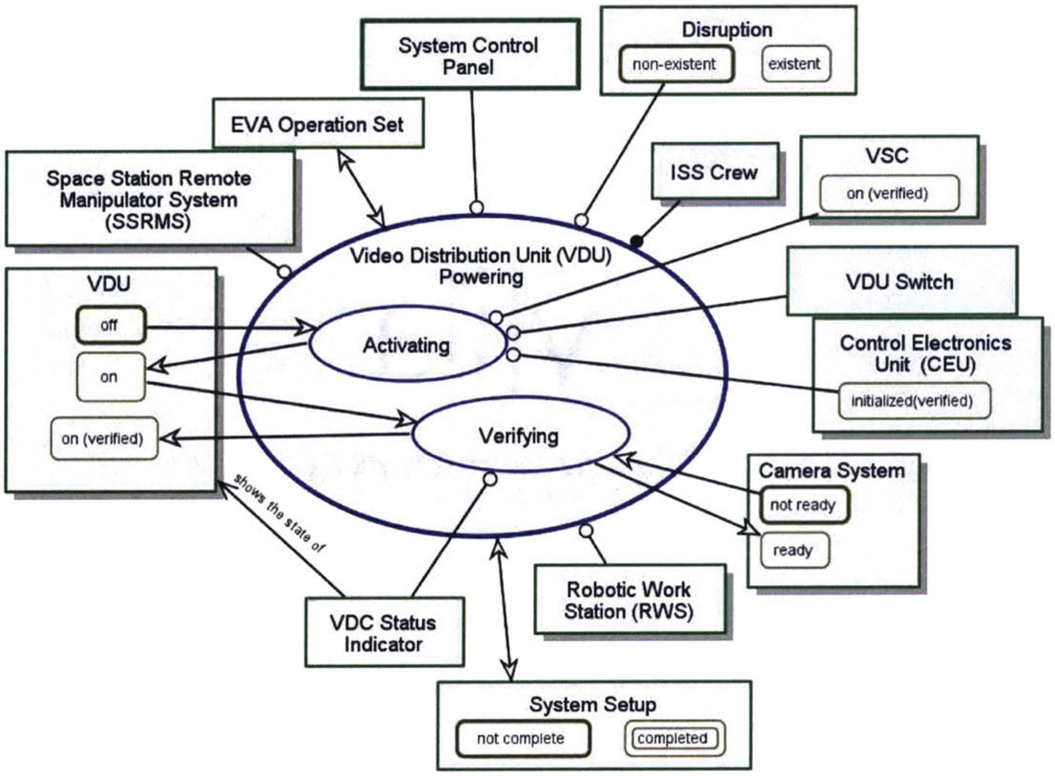


*Figure 101: OPD of Video Distribution Unit Powering In-Zoomed*

### *1.2.4 Monitor Setting In-Zoomed*

Monitor Setting is the fourth subprocess in Systems Readying (See Figure 32). Figure 102 is the in-zoomed of Monitor Setting. Setting Display using the Keyboard affects the Camera Display Set. Camera Field of View Adjusting affects the Camera Zoom and Camera Pan. Camera Zoom and Camera Pan are features of the Camera Displays. Finishing the adjustment of camera changes the System Setup from **not complete** to **completed**. This change is to symbolized that the ISS Crew can proceed to perform their robotics operations.

The procedures based on the OPDs for video system setup are as follows:

[1] Set Display to Camera X

[2] Adjust Camera view as required

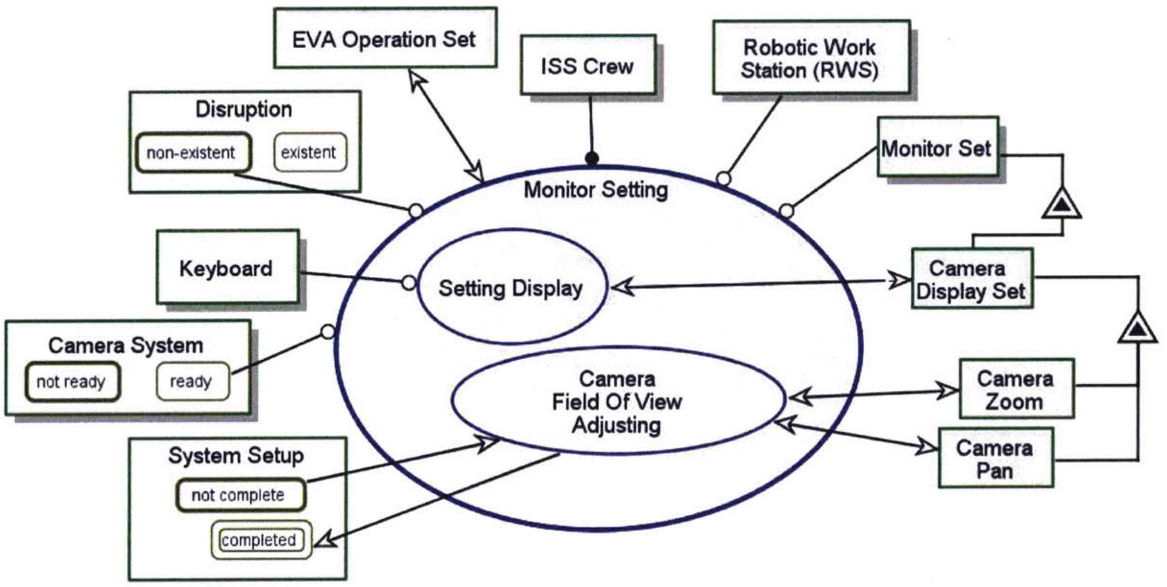


*Figure 102: OPD of Monitor Setting In-Zoomed*

### *1.2.5 Unsafing In-Zoomed*

The last subprocess in System Readying is Unsafing with in-zoomed shown in Figure 103. It requires System Setup state to be **completed.** Deactivating changes the state of Safing from **on** to **off**, and Verifying changes the state to **off(verified)**.

The Safing Setting Indicator shows the Safing's state and is a form of On-Screen System Indicators. The switch to engage and disengage safing is 'S' on the Keyboard.

The OPD defines the procedure as follows:

[1] Set Safing – Off

[2] Verify Safing – Off

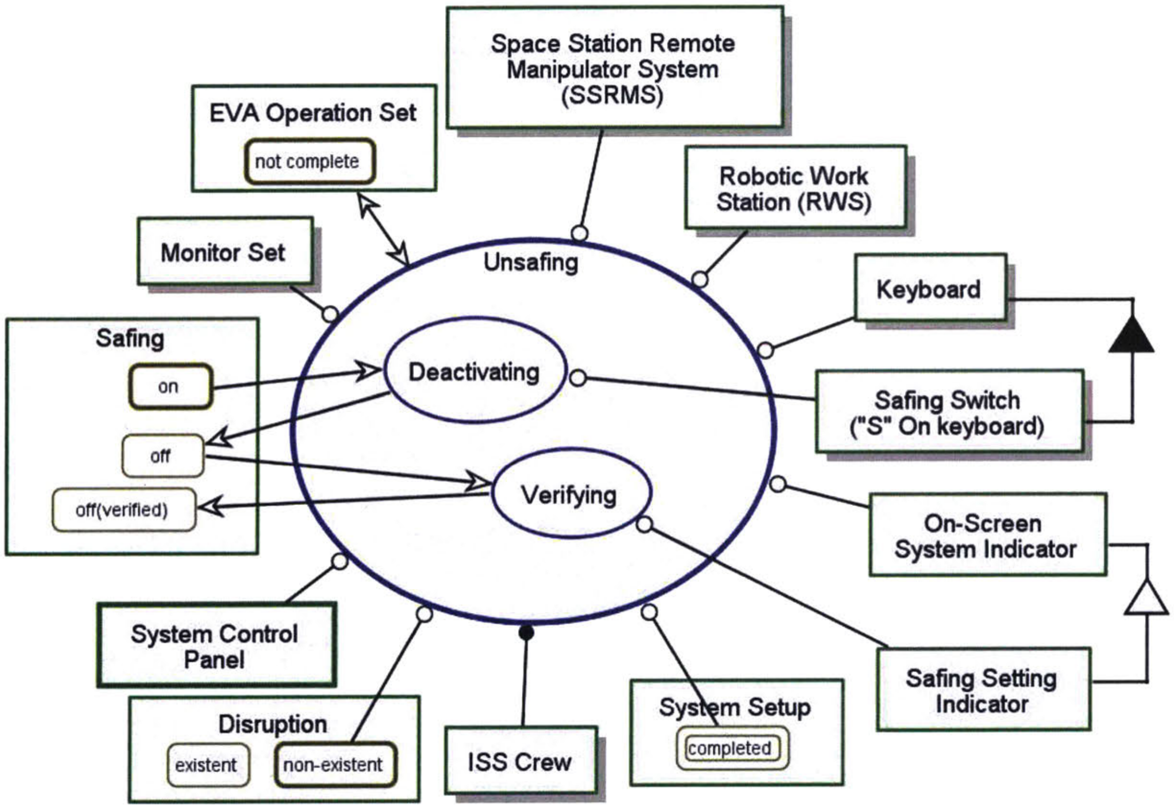


Figure 103: OPD of Unsafing In-Zoomed

## 1.3 EVA Operations Task 1 Executing (Continue)

Section 4.3.1.2 discussed the OPD of EVA Operations Task 1 Executing in zoomed. This section covers OPDs that are excluded in the earlier section.

### *1.3.1 Speed Selecting and Brake Releasing In-Zoomed*

Figure 104 and Figure 105 show the **Speed Selecting** and **Brake Releasing** In-Zoomed respectively. **Activating** changes the **Speed Mode** to **Vernier** i.e slow that is a necessary for an EVA operation and is triggered using the **Speed Mode Switch**, which is key 'v' on the **Keyboard**. **Verifying** changes the **Speed Mode** from **Vernier** to **Vernier(verified).**

Similarly, for **Brake Releasing**, **Deactivating** changes the **Brake** to **off** and is triggered by the **Brake Switch** that is key 'b' on the **Keyboard**. **Safing** in **off(verified)** state is the pre-condition for **Deactivating.** The ISS Crew needs to verify the state of the **Brake** and the subprocess changes the **Brake's** state to **off(verified)**.

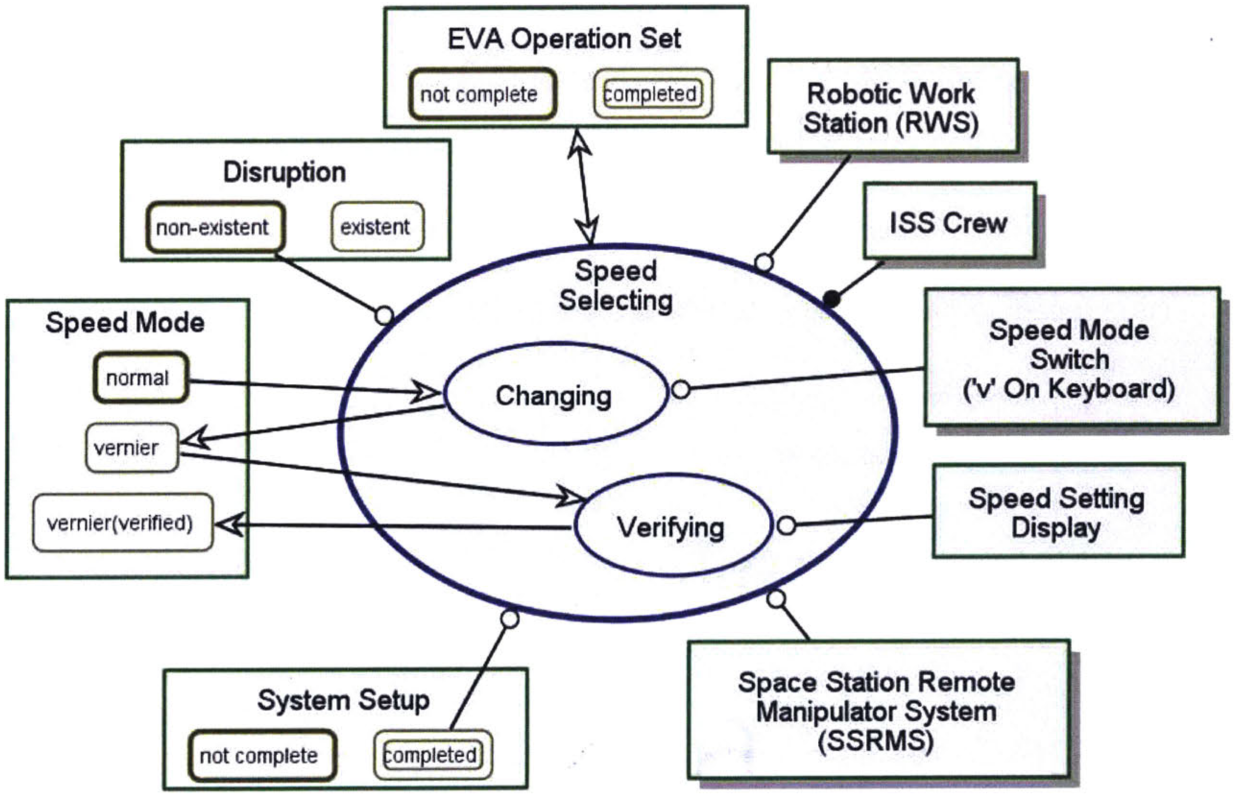


*Figure 104: OPD of Speed Selecting In-Zoomed*

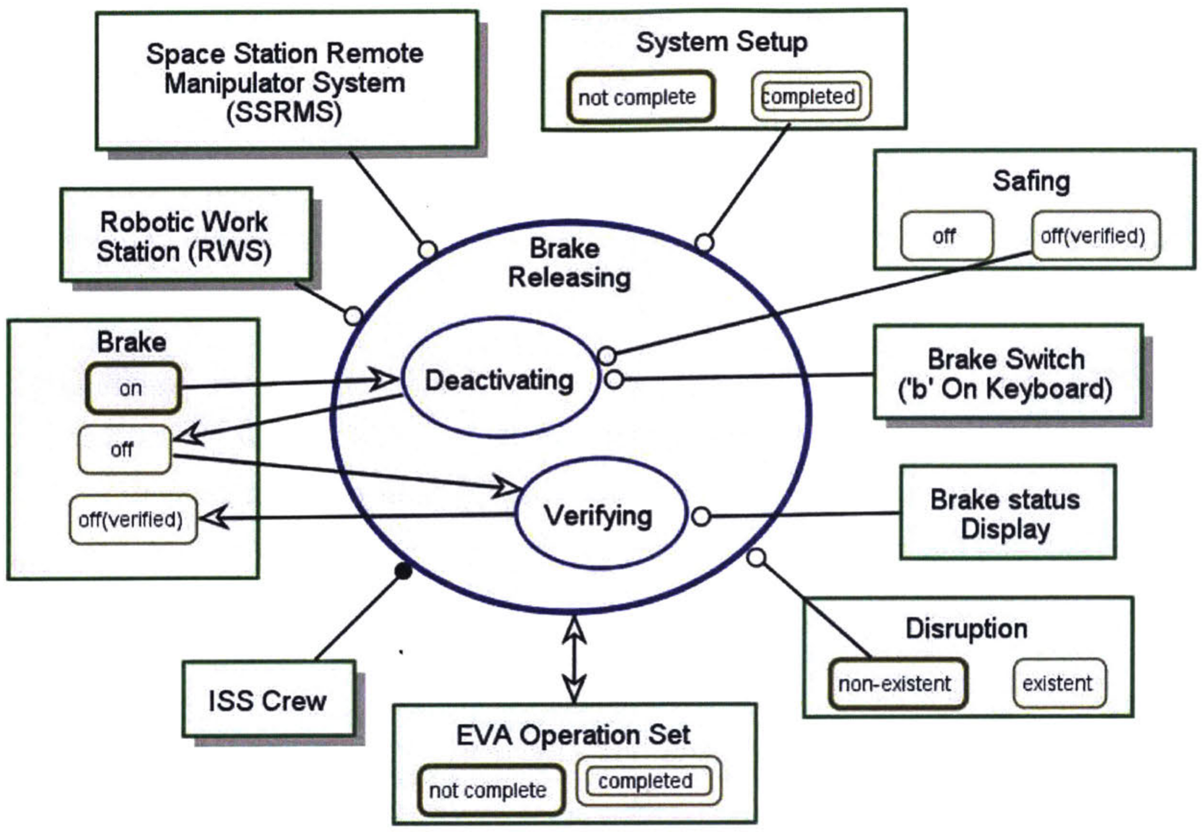


*Figure 105: OPD of Brake Releasing In-Zoomed*

The procedure based on these OPD are as follows:

[1] Set Speed – Vernier

[2] Verify Speed – Vernier

[3] Set Brake – Off

[4] Verify Brake – Off

### 1.3.2 Braking In-Zoomed

Figure 106 is the in-zoomed OPD for **Braking**. **Activating** the **Brake** from **off(verified)** to **on** requires **Safing** to be **off(verified)** because if **Safing** is **on/on(verified)**, the **Brake** will be **on**. **Brake** can be activated and deactivated using the **Brake Switch**. **Verifying** changes the **Brake** from **on** to **on(verified).**

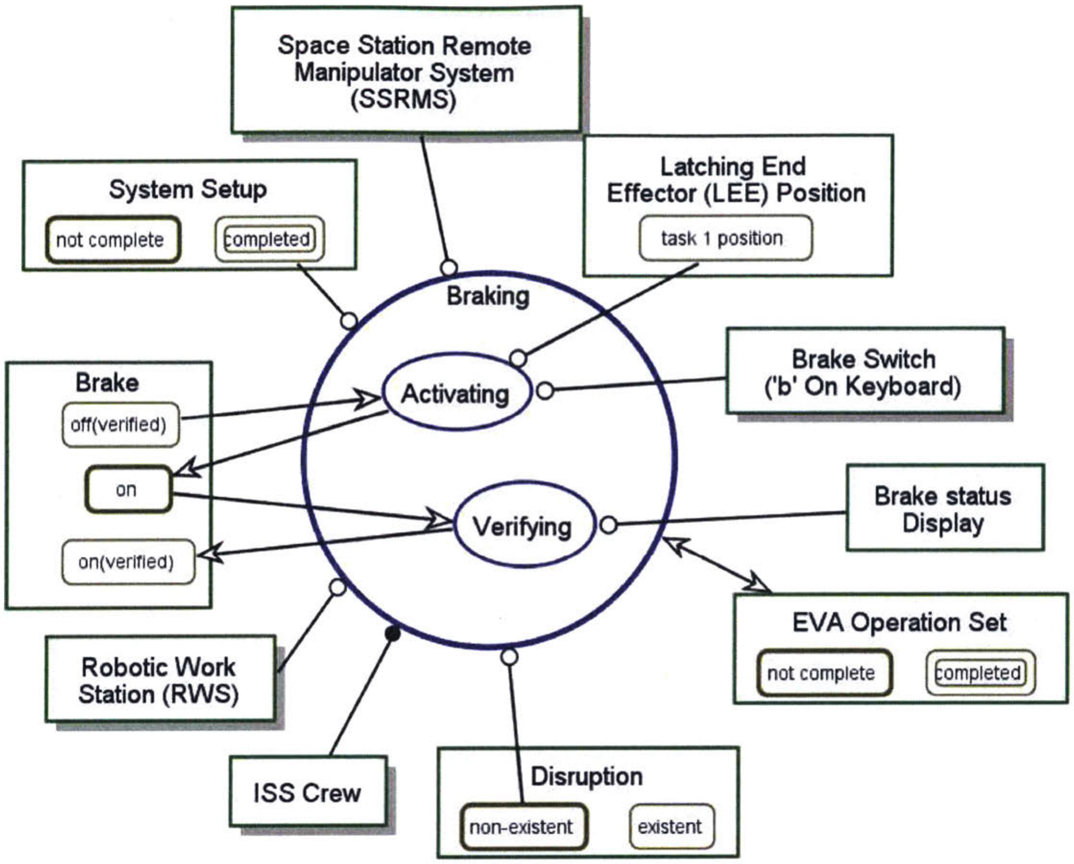


*Figure 106: Braking In-Zoomed*

## 1.4 EVA Operations Task 2 and 3 Executing

The OPDs for EVA Operations Task 2 and 3 Executing are similar to EVA Operations Task 1 Executing and will not be elaborated.

## 1.5 System Shutting Down In-Zoomed

Figure 107 presents the **System Shutting Down** In-Zoomed OPD. It comprises of four subprocesses, **Safing** and one subprocess for each subsystem - Video System, SSRMS and RWS.

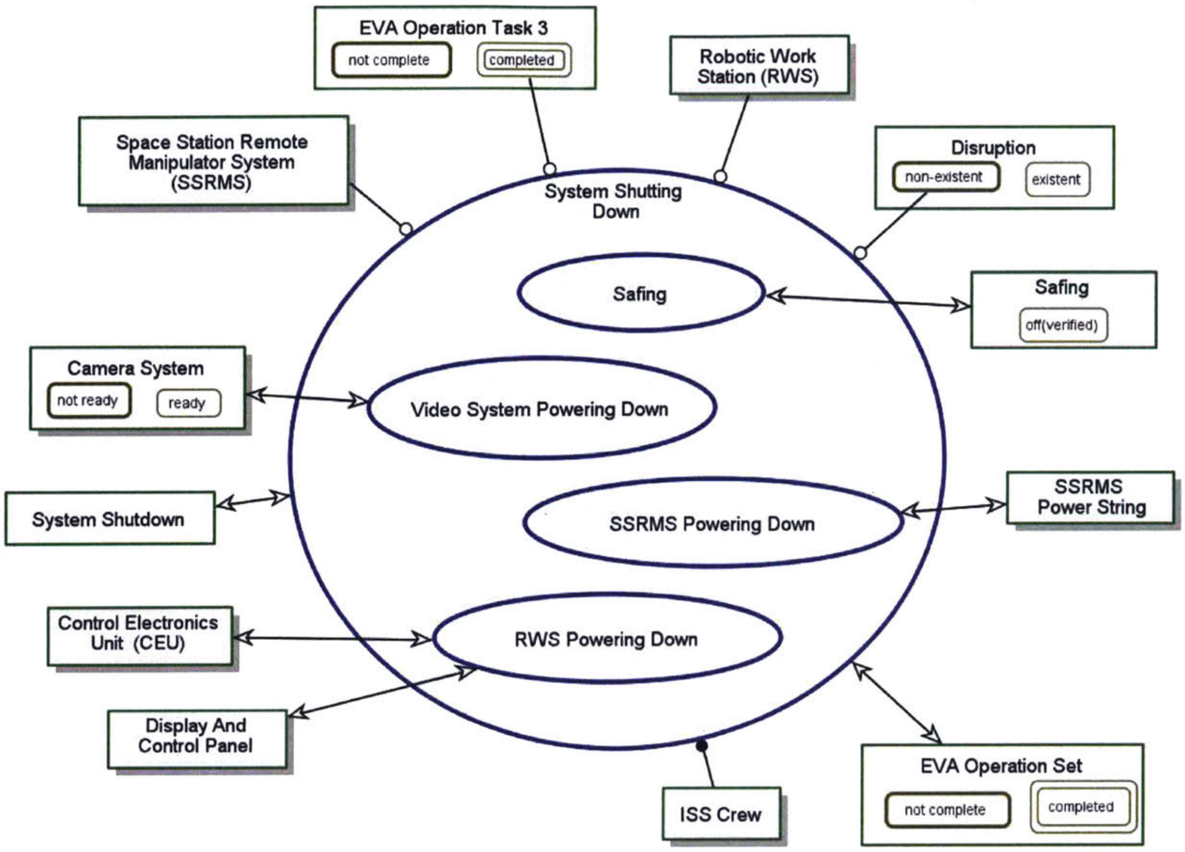


Figure 107: OPD of System Shutting Down In-Zoomed

### 1.5.1 Safing In-Zoom

Figure 108 shows the subprocess of Safing the MSS. It is a required step to safe the system at shut down. The state of Safing changes from **off(verified)** to **on** and subsequently **on(verified)** after going through the steps of Activating and Verifying.

The OPD defines the procedure as follows:

[1] Set Safing – On

[2] Verify Safing – On

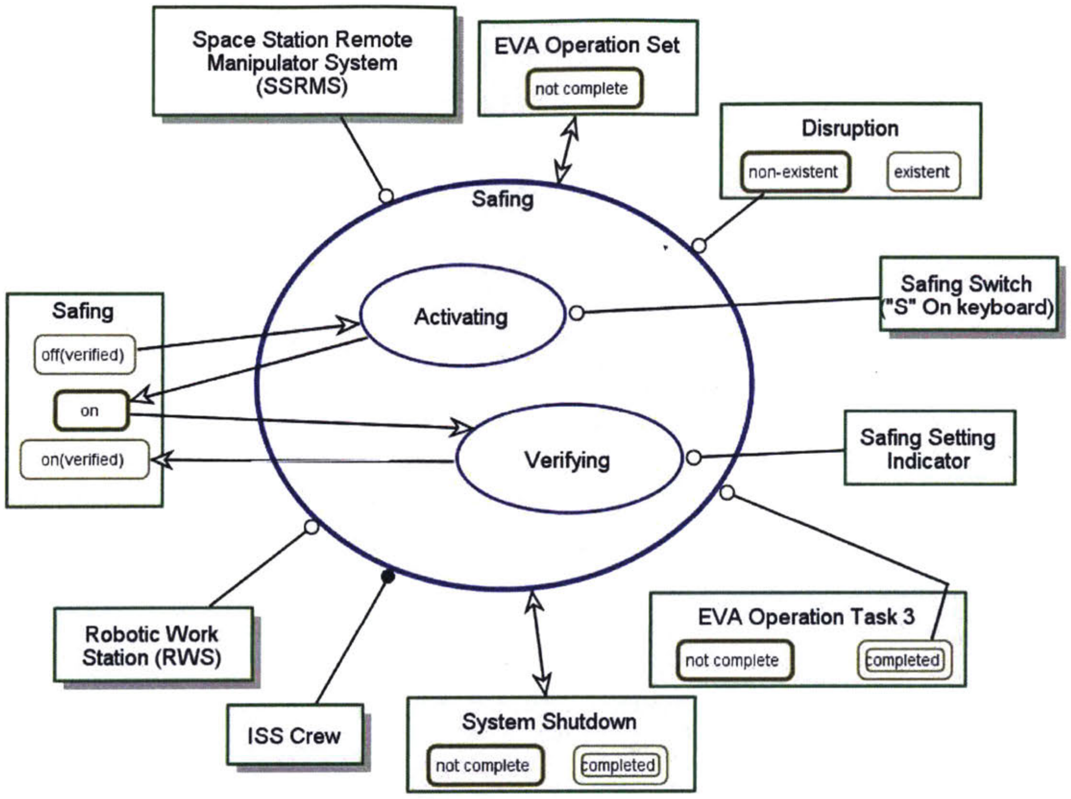


*Figure 108: OPD for Safing In-Zoom in Shut Down*

### *1.5.2 Video System Powering Down In-Zoom*

Figure 109 is the OPD of the **Video System Powering Down** in-zoomed. **CVIU 1 Switch** is required to triggered the **CVIU 1 Uninitializing**. Beside changing **CVIU 1** to **uninitialized**, **CVIU 1 Uninitializing** will also change the state of **CVIU 2** to **uninitialized**, **VSC** to **off** and **VDU** to **off**. Thus, the dependent systems i.e. **CVIU 2, VSC** and **VDU** states are changed in the simulator when **CVIU 1 Uninitializing** is triggered. Also, **CVIU 1 Uninitializing** changes **Camera System** from **ready** to **not ready** i.e. the program must deactivate all cameras display if **CVIU 1** is **uninitialized**.

**Verifying** requires **VSC Status Indicator**, **CVIU 1 Status Indicator**, **CVIU 2 Status Indicator** and **VDU Status Indicator** and it changes the state of **CVIU 1**, **CVIU 2**, **VSC** and **VDU.**

The OPDs define the procedure as follows:

[1] Set CVIU 1 – Uninitialized

[2] Verify CVIU 1 – Uninitialized

[3] Verify CVIU 2 – Uninitialized

[4] Verify VSC – Off

[5] Verify VDU – Off

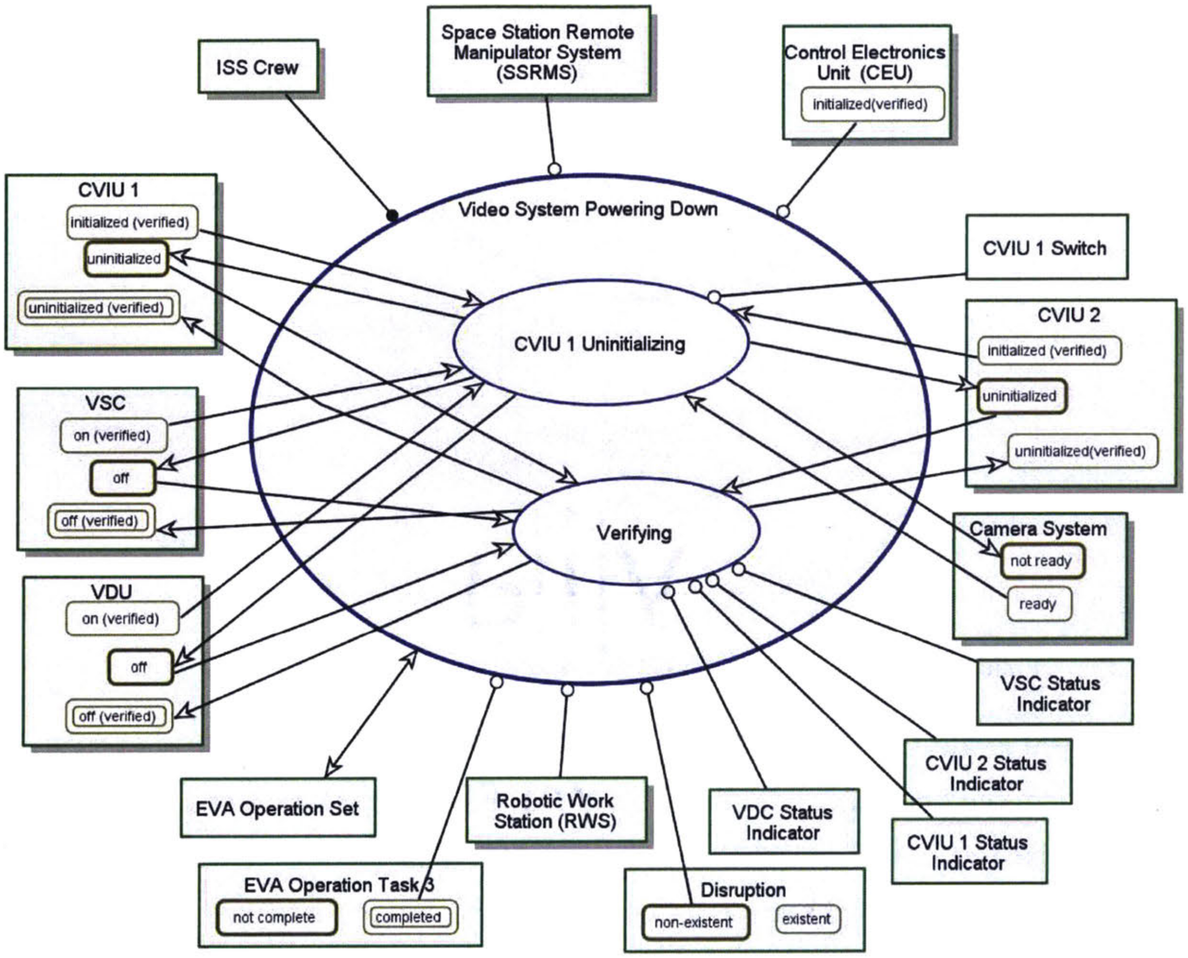


*Figure 109:OPD of Video System Powering Down In-Zoomed*

### *1.5.3 SSRMS Powering Down In-Zoomed*

The OPD of **SSRMS Powering Down** in-zoomed is shown in Figure 110. The shutdown is similar to the setup discussed in para 1.1. The **SSRMS Redundant String Powering** down will change the state of **SSRMS Redundant String** to **off**. The same concept applies to **SSRMS Prime String Powering Down**.

The OPD defines the procedure for **SSRMS Powering Down** as follows:

[1] Set SSRMS Redundant String – Off

[2] Verify Redundant String - Off

[3] Set SSRMS Prime String – Off

[4] Verify Prime String - Off

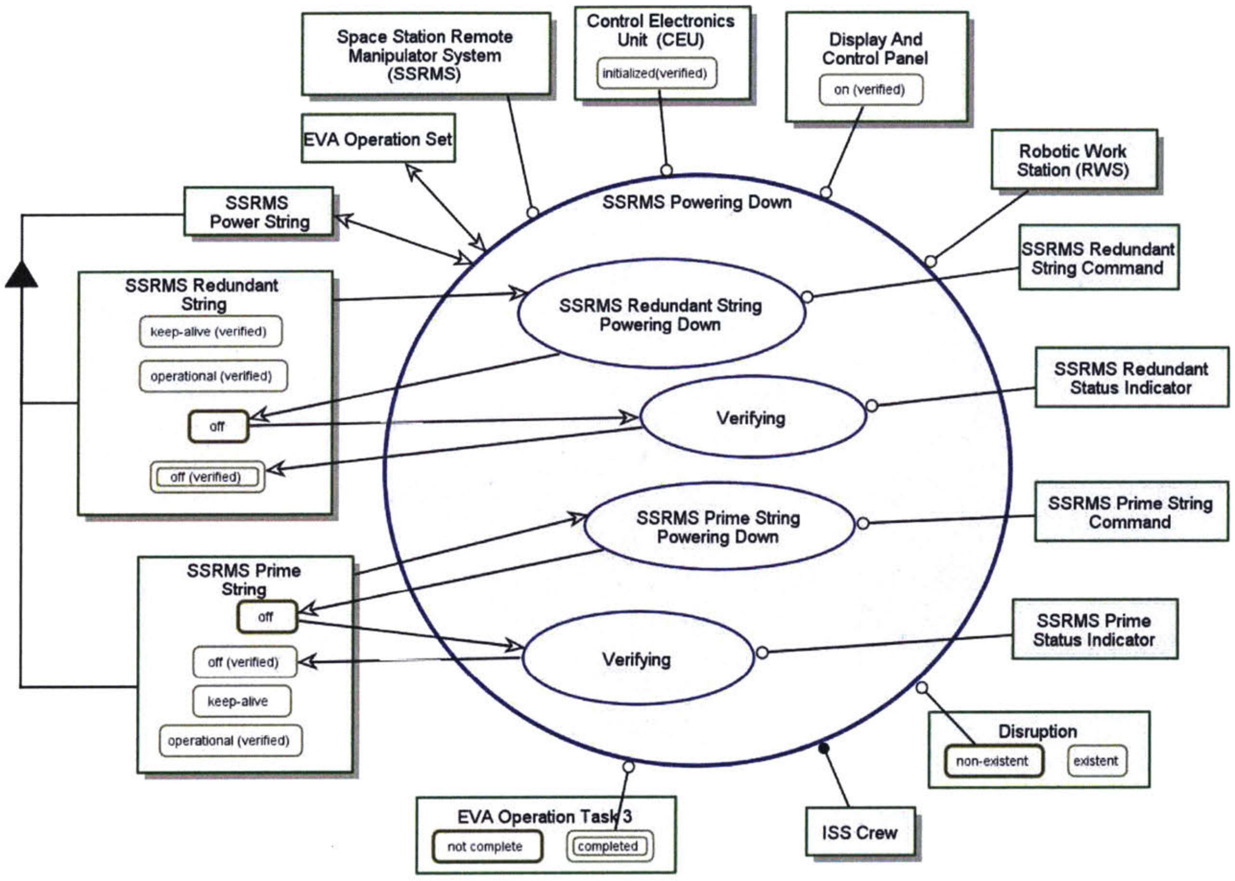


*Figure 110: OPD of SSRMS Powering Down In-Zoomed*

### *1.5.4 RWS Powering Down In-Zoomed*

Figure 111 shows the **RWS Powering Down** in-zoomed OPD. **CEU Powering Down** changes **CEU** from **initialized(verified)** to **uninitialized** and simultaneously, it sets **Comm** to **disabled** automatically. The **CEU Status Indicator** and **Comm Status Indicator** are needed in **Verifying**. **DCP Powering Down** requires **DCP Power Switch** and changes **Display And Control Panel** to off. The last subprocess (**Verifying**) in **RWS Powering Down** in-zoomed changes **System Shutdown** to

**completed** and **EVA Operation Set** to **completed**. This last step marks the completion of the simulation tasks.

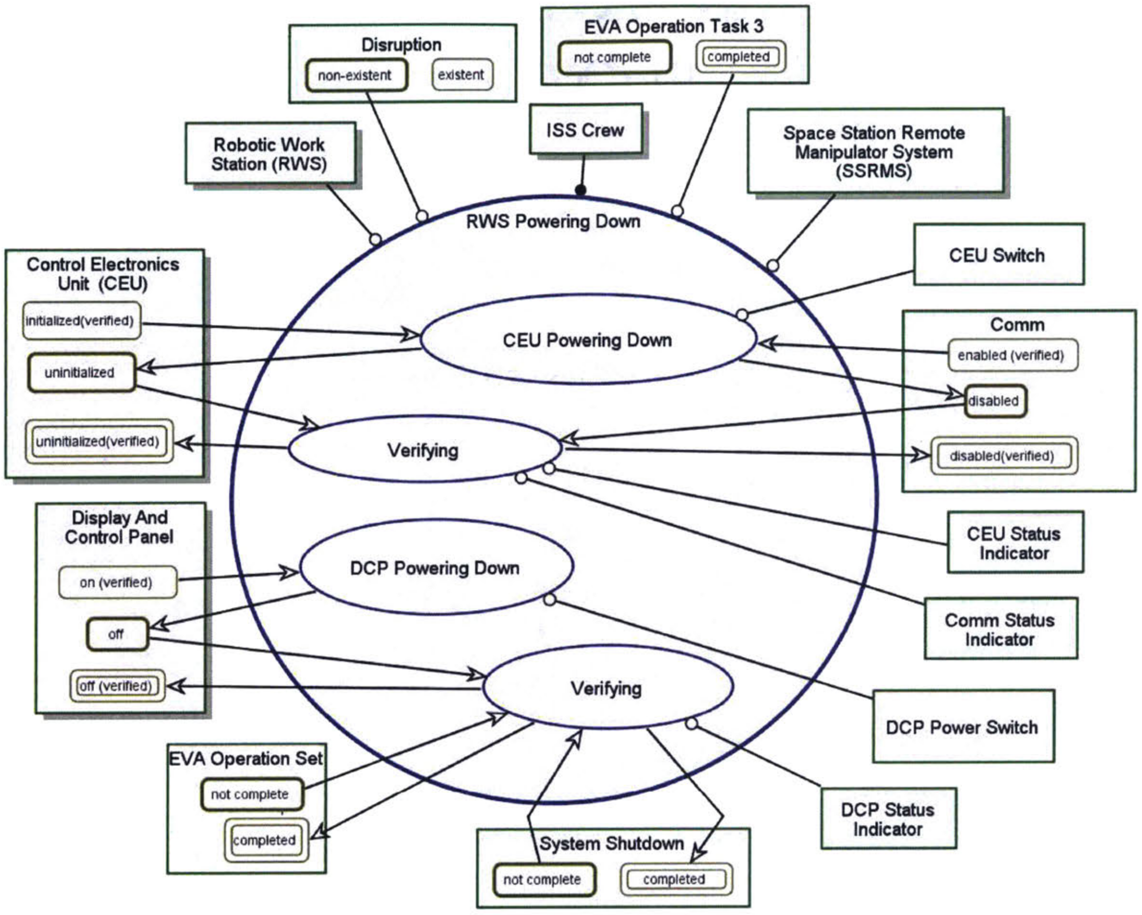


*Figure 111: OPD of RWS Powering Down In-Zoomed*

The procedure based on the OPD for **RWS Powering Down** are as follows:

[1] Set CEU – Uninitialized

[2] Verify CEU – Uninitialized

[3] Verify Comm – Disabled

[4] Set DCP – Off

[5] Verify DCP - Off

## 1.6 Comm Recovering

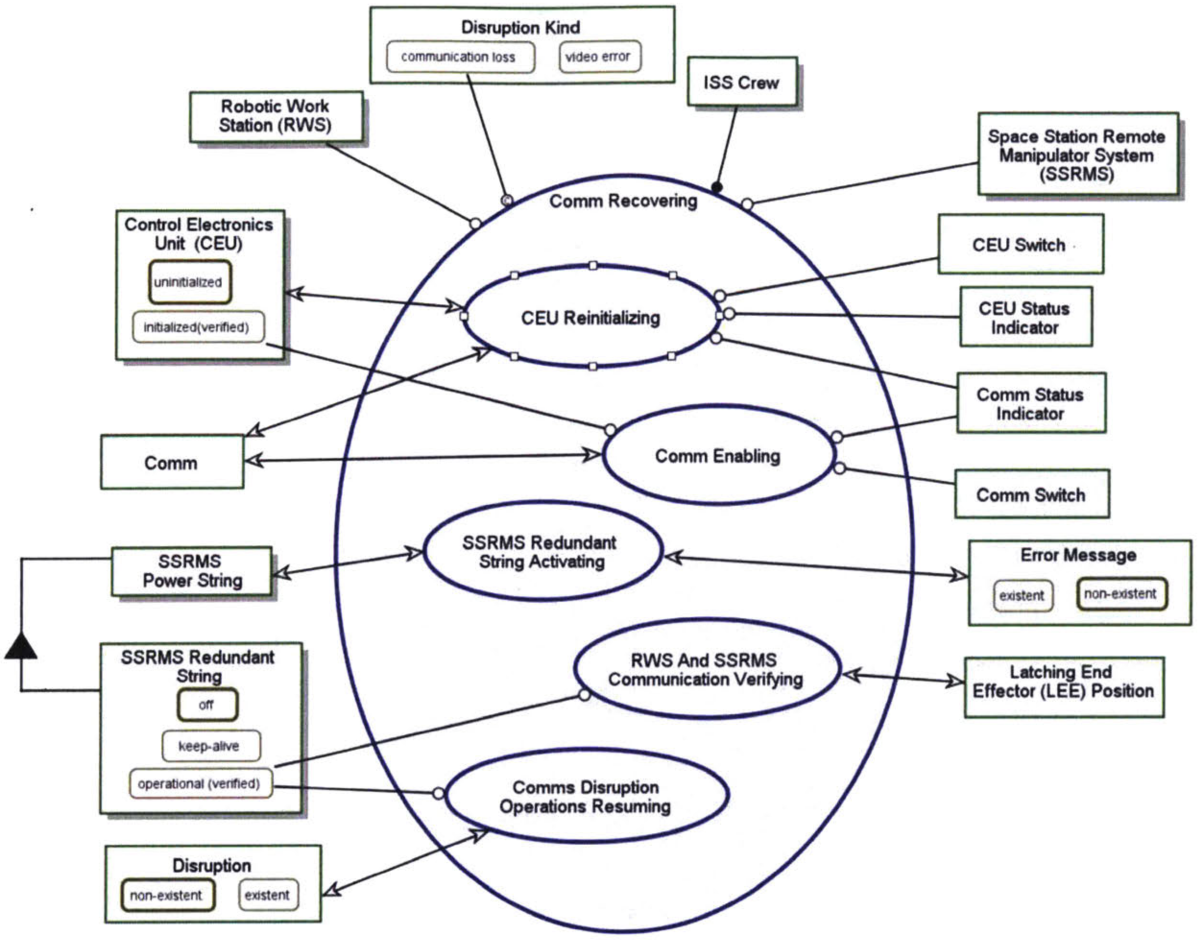


*Figure 112: OPD of Comm Recovering In-Zoomed*

The OPD in Figure 112 shows the subprocess that the **ISS Crew** needs to perform if **Disruption Kind** is **communication loss**. This failure meant that the RWS is unable to communicate with the ISS and SSRMS systems. **CEU Reinitializing, Comm Enabling,** and **SSRMS Redundant String Activating** affect the state of some systems that will be explained later. **RWS And SSRMS Communication Verifying** and **Comms Disruption Operations Resuming** requires **SSRMS Redundant String** to be **operational(verified)**.

### *1.6.1 CEU Reinitializing In-Zoomed*

In **CEU Reinitializing**, the **ISS Crew** re-cycle the **CEU** by setting **CEU** to **uninitialized**. This subprocess also changes the state of **Comm** to **disabled** as mentioned in para 1.5.4. Then, the **ISS Crew** has to initialize **CEU**, similar to **the RWS Powering** discussed in setup (see para 4.3.1.1.1.1).

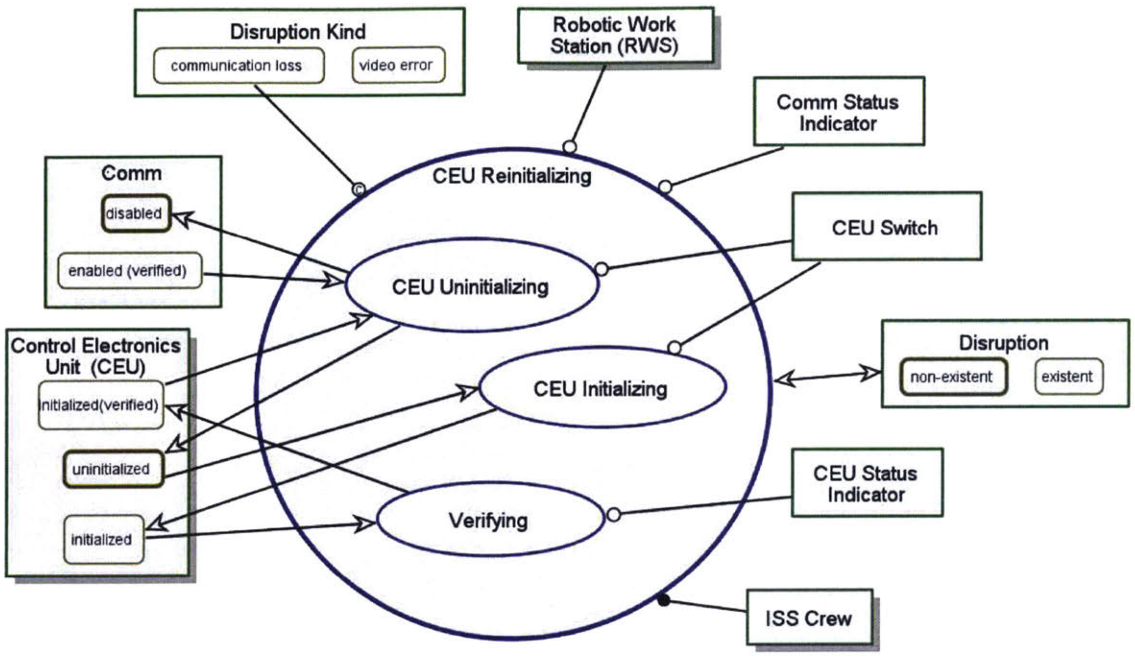


*Figure 113: OPD of CEU Reinitializing In-Zoomed*

The OPD defines the procedure for **CEU Reinitializing** as follows:

[1] Set CEU – Uninitialized

[2] Verify CEU – Uninitialized

[3] Set CEU – Ininitialized

[4] Verify CEU – Ininitialized

### *1.6.2 Comm Enabling In-Zoomed*

The Comm Enabling OPD in Figure 114 is similar to that drawn for system setup (Figure 91). Thus, it will not be elaborated in this section.

The OPD defines the procedure for **Comm Enabling** as follows:

[1] Set Comm – Enable

[2] Verify Comm – Enabled

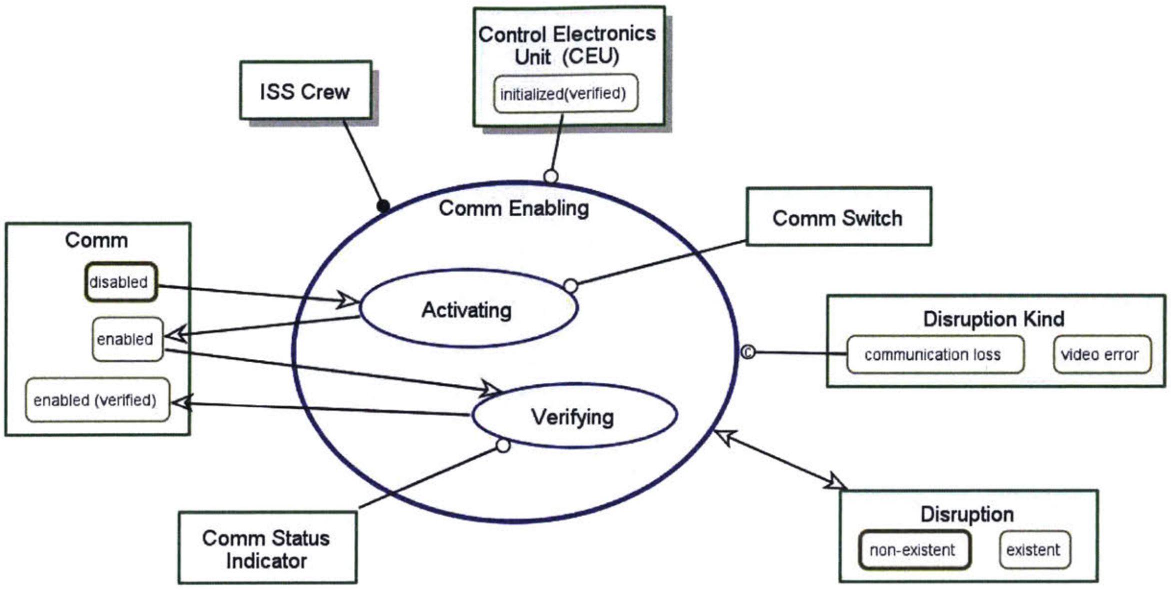


*Figure 114: OPD of Comm Enabling In-Zooming*

### *1.6.3 SSRMS Redundant String Activating In-Zoomed*

Figure 115 is the OPD for **SSRMS Redundant String Activating** In-Zoomed. As the power and data for the **SSRMS Power String** has redundancy, this subprocess requires the ISS Crew to turn **off** the **SSRMS Prime String** and set **SSRMS Redundant String** to **operational**. The simulator should be designed such that the **Error Message** will change to **non-existent** at the completion of **SSRMS Redundant String Keep-Alive To Operational Powering** as illustrated in the OPD. Nonetheless, Disruption Handling is still existent until the **ISS Crew** finishes the **Comm Recovering** procedures.

The OPD defines the procedures **SSRMS Redundant String Activating** as follows:

[1] Set SSRMS Prime String – Off
[2] Verify SSRMS Prime String – Off
[3] Set SSRMS Redundant String – Operational
[4] Verify SSRMS Redundant String – Operational

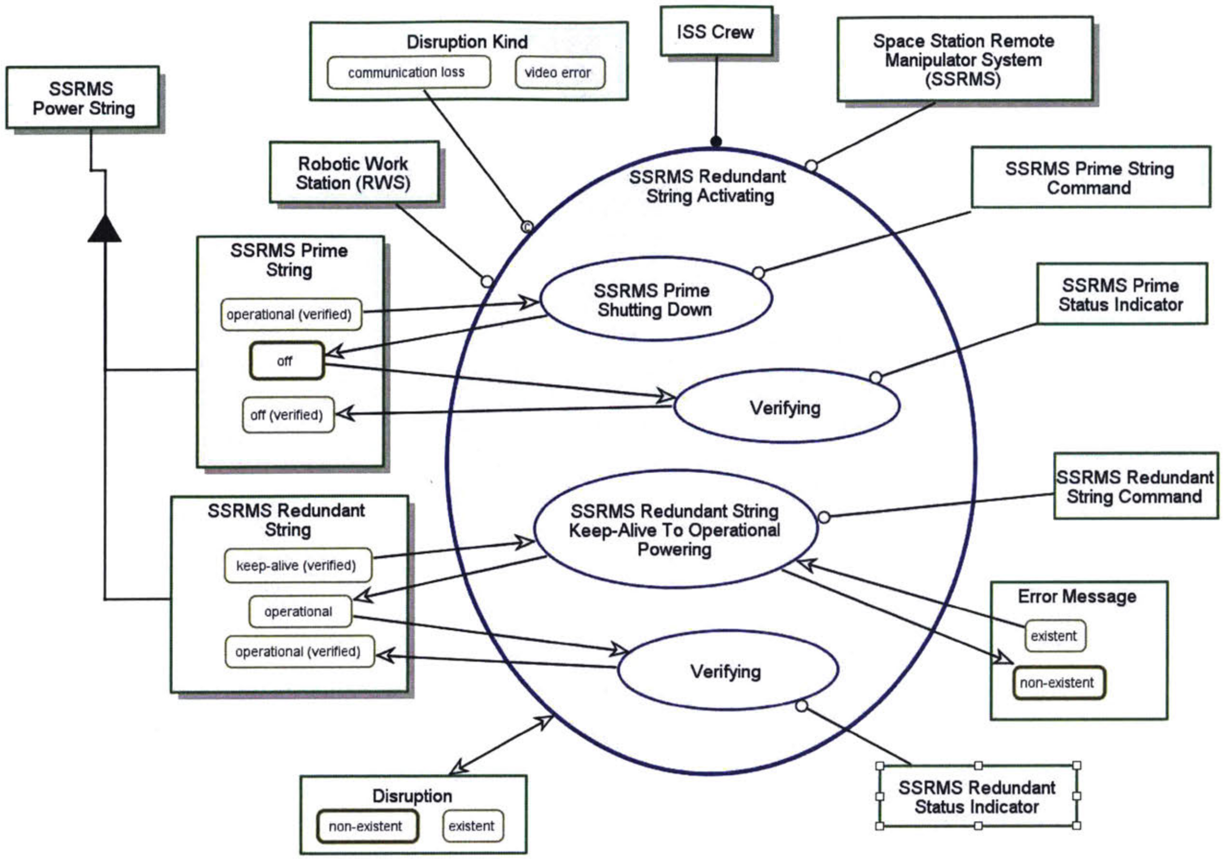
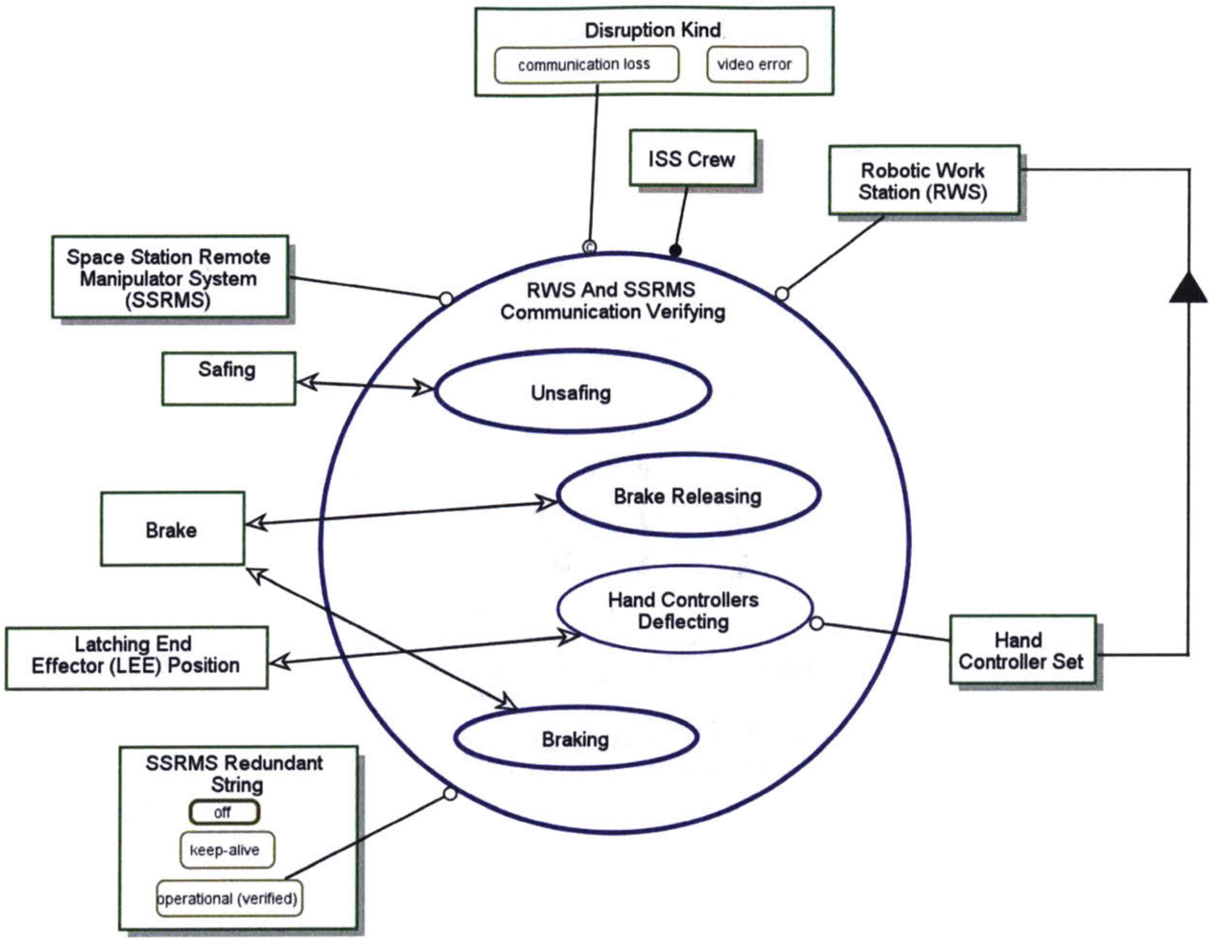

SSRMS Power String
Disruption Kind
communication loss
video error
ISS Crew
Space Station Remote Manipulator System (SSRMS)
Robotic Work Station (RWS)
SSRMS Redundant String Activating
SSRMS Prime String Command
SSRMS Prime String
operational (verified)
off
off (verified)
SSRMS Prime Shutting Down
SSRMS Prime Status Indicator
Verifying
SSRMS Redundant String Command
SSRMS Redundant String
keep-alive (verified)
operational
operational (verified)
SSRMS Redundant String Keep-Alive To Operational Powering
Error Message
existent
non-existent
Verifying
Disruption
non-existent
existent
SSRMS Redundant Status Indicator

*Figure 115: SSRMS Redundant String Activating In-Zoomed*

### *1.6.4 RWS and SSRMS Communication Verifying*

The **RWS and SSRMS Communication Verifying** is designed for the **ISS Crew** to test that the **SSRMS** is able to receive input from the **RWS** using the **Hand Controller Set** – THC and RHC (see para 2.1.4). **Hand Controllers Deflecting** is the subprocess of moving the **Hand Controller Set** that will affect the **LEE Position** as reflected in Figure 116. If **LEE Position** changes, it validates that the error is resolved. The **Unsafing, Brake Releasing and Braking** subprocesses are discussed in the earlier sections and will not be covered here. See para 1.2.5, 1.3 and 1.3.2 respectively.



Disruption Kind
communication loss
video error
ISS Crew
Robotic Work Station (RWS)
Space Station Remote Manipulator System (SSRMS)
RWS And SSRMS Communication Verifying
Safing
Unsafing
Brake
Brake Releasing
Hand Controllers Deflecting
Hand Controller Set
Latching End Effector (LEE) Position
Braking
SSRMS Redundant String
off
keep-alive
operational (verified)

*Figure 116: OPD of RWS And SSRMS Communication Verifying*

The OPDs defines the procedures for **RWS** and **SSRMS Communication Verifying, Unsafing, Brake Releasing and Braking** as follows:

[1] Set Safing – Off

[2] Verify Safing – Off

[3] Set Brake – Off

[4] Verify Brake – Off

[5] Deflect Hand Controllers and ensure SSRMS movement

[6] Set Brake – On

[7] Verify Brake – On

### *1.6.5 Comms Disruption Operations Resuming*

This part of the OPD is specifically included due to the scenario that **Disruption** happen during **SSRMS relocating**. Because 'safing' (refer to 4.3.2.1) resets some of the setting, in particularly **SSRMS Mode** back to **manual**, the **ISS Crew** needs to re-setup the 'Autosequence'. The four subprocesses **Autosequence Setting, Speed Selecting, Brake Releasing** and **Autosequence Executing** are similar to those covered in para 4.3.1.3 to 4.3.1.3.1.

The OPDs define the procedure as follows:

[1] Select – Autosequence
[2] Select SSRMS Mode – Joint Angle
[3] Input Joint Angles
[4] Verify – Joint Angles
[5] Load Joint Angles
[6] Set Speed – Vernier
[7] Verify Speed Vernier
[8] Set Brake – Off
[9] Verify Brake – Off
[10] Confirm Autosequence
[11] Verify Autosequence - Commencing

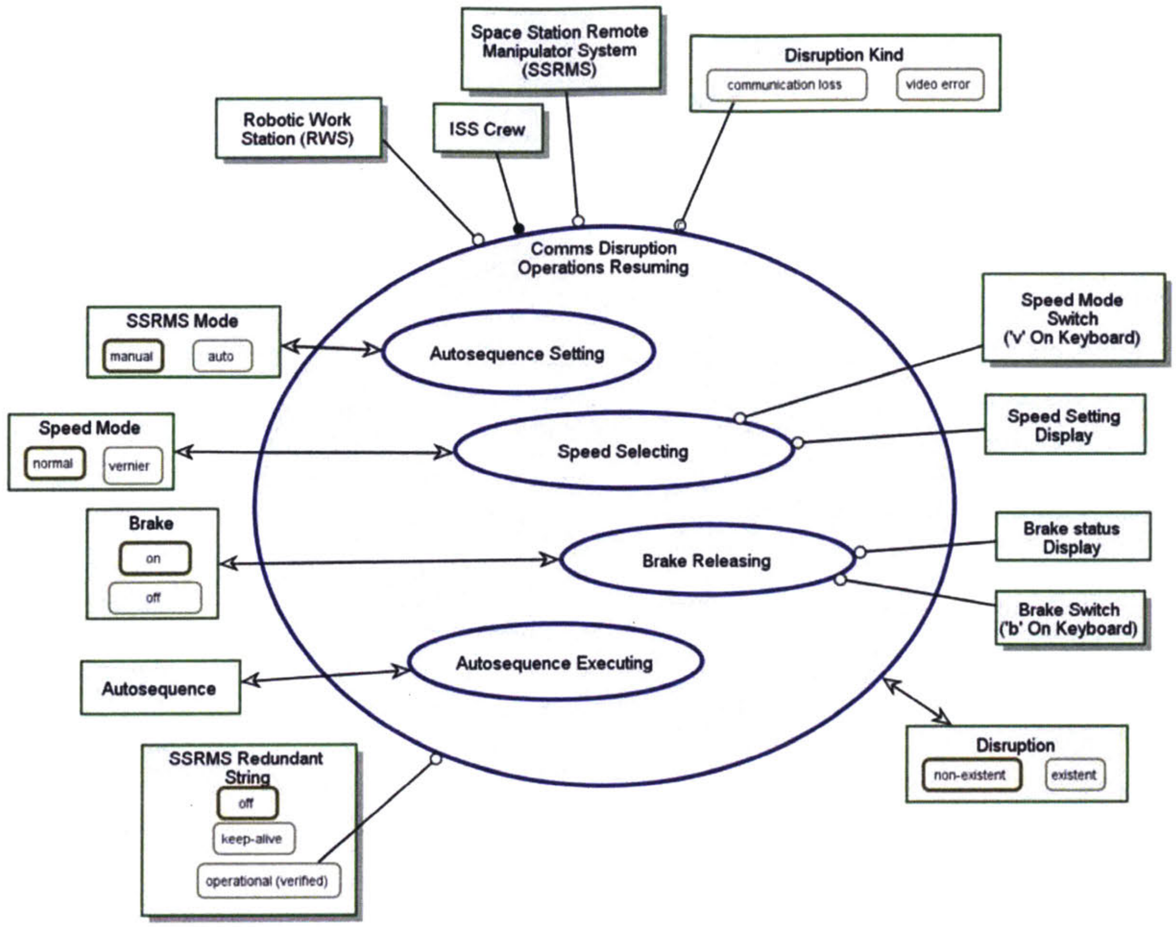


*Figure 117: OPD of Comms Disruption Operations Resuming*

## 1.7 Video Recovering

Video Recovering procedure is carried out if the Disruption Kind is a **video error**. When **video error** happens, the camera views will black out to simulate issue with the video systems. Figure 118 shows the five subprocesses – **CVIU Re-Powering, VSC Re-Powering, VDU Re-Powering** and **Monitor Resetting** – that need to be carried out for **video error.** These subprocesses resemble **Video System Powering** in para 1.2.3. Therefore, only the differences will be discussed in the subsequent sections.

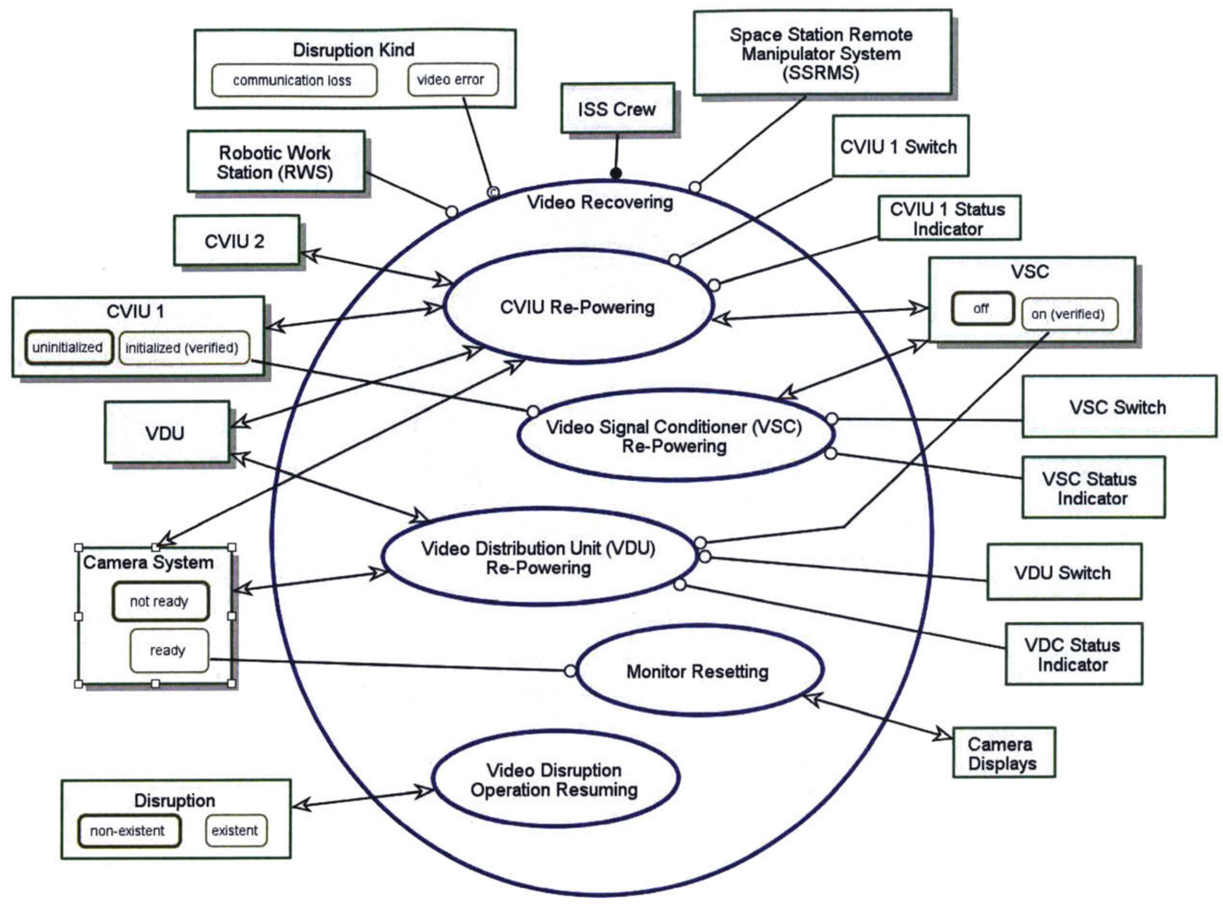


Figure 118: Video Recovering In-Zoomed

### 1.7.1 CVIU Re-Powering In-Zoomed

In this scenario, the recovery procedure calls for the re-powering of the video systems i.e. **CVIU 1, CVIU 2, VSC** and **VDU**. The key difference in the subprocesses between Figure 119 and Figure 99 is the **CVIU 1 Uninitializing**. **CVIU 1 Uninitializing** causes **CVIU 1** to become **uninitialized** and concurrently **uninitialized** **CVIU 2** and set **VSC** and **VDU** to **off.** This is also discussed in **Video System Powering Down** in para 1.5.2. The remaining subprocesses are covered in para 1.2.3.1 under **Video System Powering**.

The OPD defines the procedures as follows:

[1] Uninitialized CVIU 1

[2] Initialize CVIU 1

[3] Verify CVIU 1 – Initialized

[4] Initialized CVIU 2

[5] Verify CVIU 2 – Initialized

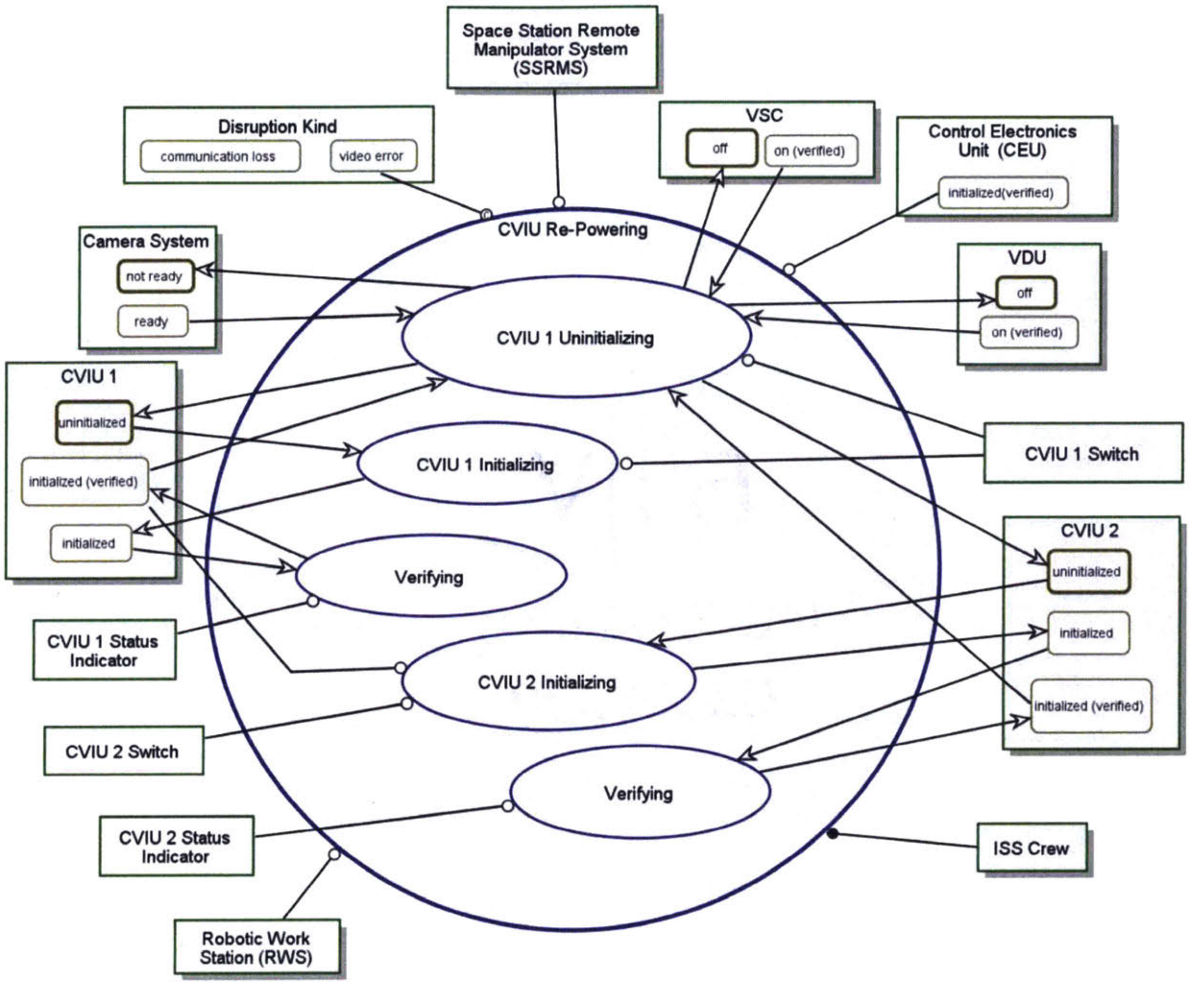


*Figure 119: OPD of CVIU Re-Powering In-Zoomed*

### *1.7.2 Video Signal Conditional (VSC) Re-Powering In-Zoomed*

Figure 120 shows the **VSC Re-Powering** in-zoomed OPD. **Activating** sets **VSC** from **off** to **on** and **Verifying** changes **VSC** to **on(verified)**. **CVIU 2 initialized(verified)** is the pre-condition for **Activating** and it is triggered by the **VSC Switch.**

The OPD defines the procedure for **VSC Re-Powering** as follows:

[1] Set VSC – On

[2] Verify VSC - On

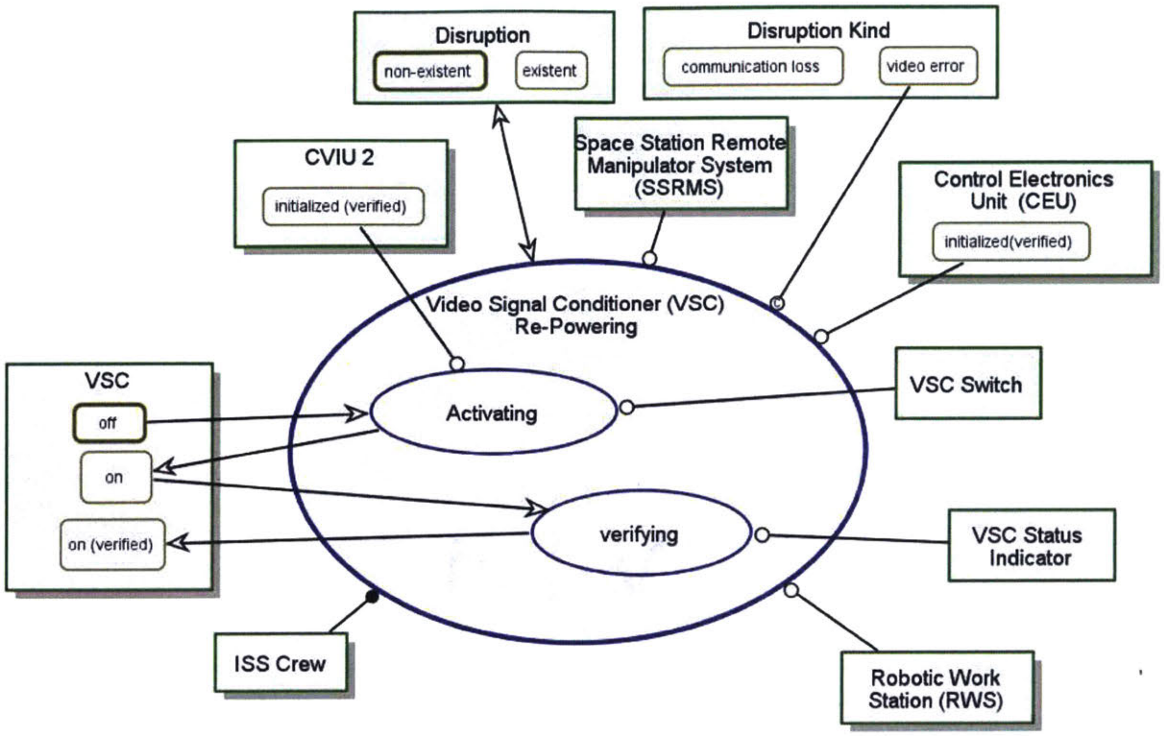


*Figure 120: OPD of Video Signal Conditioner (VSC) Re-Powering In-Zoomed*

### *1.7.3 Video Distribution Unit (VDU) Re-Powering In-Zoomed*

The noteworthy difference in **VDU Re-Powering** in Figure 121 compared to Figure 101 **VDU Powering** is the **Error Message**. Figure 121 depicts that the design requirement is for **Error Message** to change from **existent** to **non-existent** after **VDU** is set to **on**. Although the error is rectified, **Disruption** state remain as **existent** because the **ISS Crew** needs to re-setup the monitor and 'Autosequence'. This OPD demonstrates the value of OPM in designing procedure and system requirement in comparison to HTA. HTA would not be able to define when the **Error Message** should appear and how can it be eliminated.

The OPD defines the procedure for **VDU Re-Powering** as follows:

[1] Set VDU – On

[2] Verify VDU – On

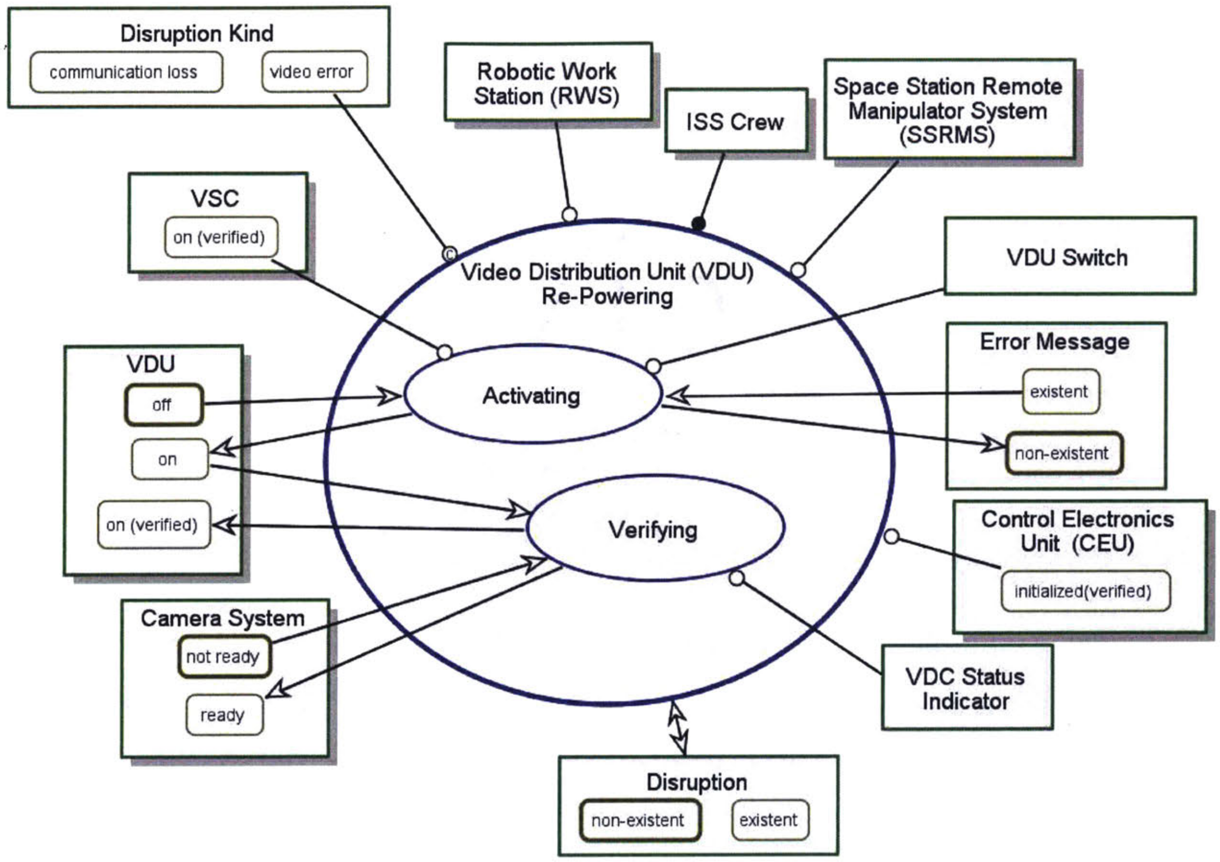


*Figure 121: OPD of Video Distribution Unit (VDU) Re-Powering*

### *1.7.4 Monitor Resetting In-Zoomed*

**Monitor Resetting** in Figure 122 is identical to **Monitor Setting** in para 1.2.4, hence, it will not be discussed in this section.

The OPD defines the procedure for **Monitor Resetting** as follows:

[1] Set Display - Camera X

[2] Adjust Camera view as required

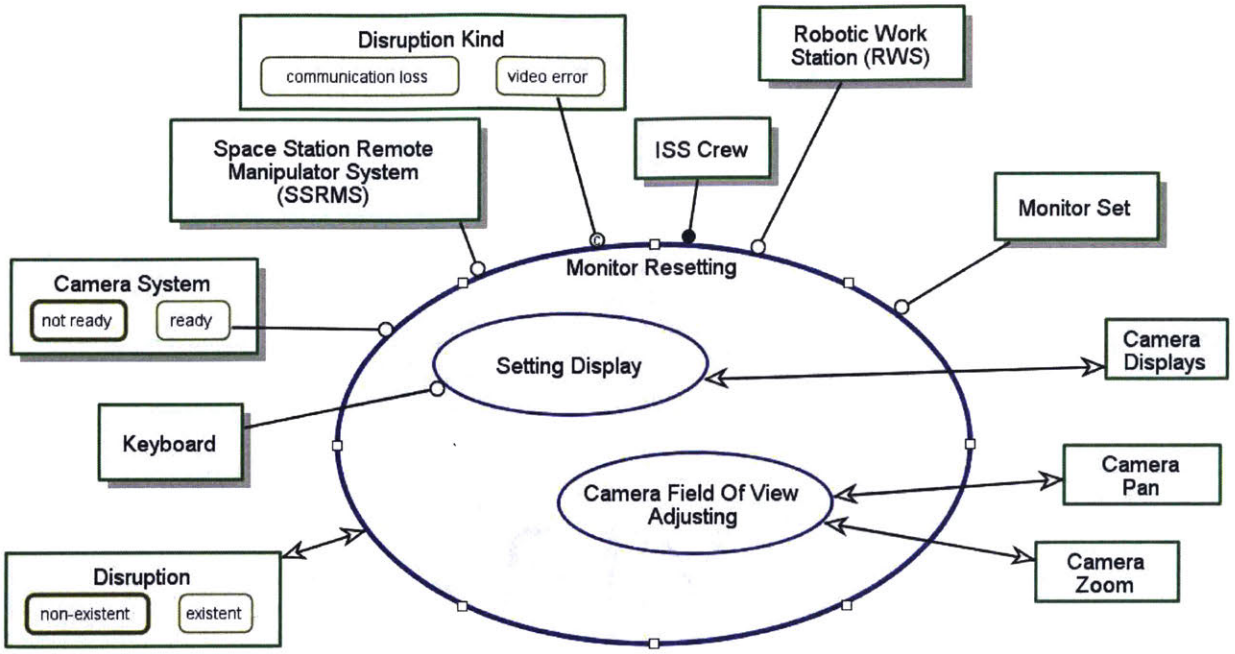


*Figure 122: OPD of Monitor Resetting In-Zoomed*

### 1.7.5 *Video Disruption Operation Resuming In-Zoomed*

As explained in para 1.6.5 **Comms Disruption Operations Resuming**, **ISS Crew** also has to re-setup 'Autosequence', resulting in the **Video Disruption Operation Resuming** subprocess. The nuance is that in **Video Disruption Operation Resuming**, there is an additional subprocess – **Unsafing** as shown in Figure 123. This is because in **Comm Recovering**, the **Safing** is set to **off(verified)** during **RWS And SSRMS Communication verifying**. The remaining four subprocesses **Autosequence Setting, Speed Selecting, Brake Releasing** and **Autosequence Executing** are similar to those covered in para 4.3.1.3 to 4.3.1.3.1.

The OPD defines the procedure as follows:

[1] Set Safing – Off

[2] Verify Safing - Off

[3] Select – Autosequence

[4] Select SSRMS Mode – Joint Angle

[5] Input Joint Angles

[6] Verify – Joint Angles

[7] Load Joint Angles

[8] Set Speed – Vernier

[9] Verify Speed Vernier

[10] Set Brake – Off

[11] Verify Brake – Off

[12] Confirm Autosequence

[13] Verify Autosequence - Commencing

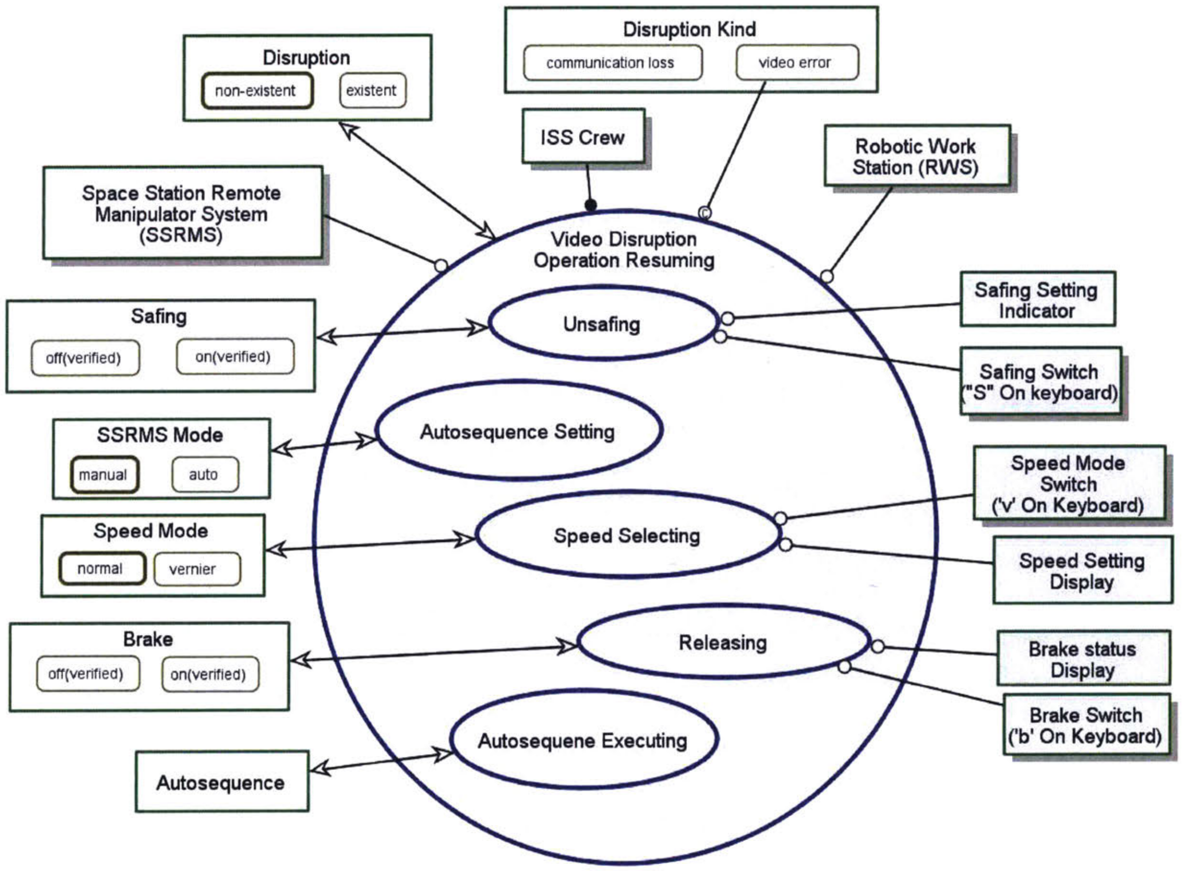


Figure 123: OPD of Video Disruption Operation Resuming In-Zoomed

# APPENDIX C – NOMINAL AND OFF-NOMINAL PROCEDURES

## ***Space Station Remote Manipulator System (SSRMS) Nominal Procedure***

### System Set Up

1. Robotic Workstation Powerup

   Set DCP – On
   Verify DCP – On

   Set CEU - Initialized
   Verify CEU - Initialized

   Set Comm – Enabled
   Verify Comm – Enabled

   Download WHS firmware [Note: Wait 30 sec for download to complete.]
   Verify WHS Firmware – Downloaded

   Set FDIR – Enabled
   Verify FDIR – Enabled

2. SSRMS Powerup

   Set SSRMS Prime String – Keep-Alive
   Verify SSRMS Prime String – Keep-Alive
   Record Keep Alive Start Time

   Set SSRMS Redundant String – Keep-Alive
   Verify SSRMS Redundant – Keep-Alive

   Set SSRMS Prime String – Operational
   Verify SSRMS Prime – Operational
   [Caution: Switching to Operational before 2 min in Keep-Alive can damage the equipment.]

3. Video Components Powerup

   Set CVIU 1 – Initialized [Note: Wait 25 seconds to complete.]
   Verify CVIU 1 – initialized

   Set CVIU 2 - Initialized [Note: Wait 20 seconds to complete.]
   Verify CVIU 2 – Initialized

Set VSC – On
Verify VSC – On

Set VDU – On
Verify VDU – On

4. Operation Prep

    Set Monitor 1 – Camera 01
    Set Monitor 2 – Camera 02
    Set Monitor 3 – Camera 03

    Set Safing – Off
    Verify Safing – Off

EVA Operations Task 1

1. Setup Autosequence

    Select Autosequence
    Select Joint Angles (JA)
    Enter JA -125.2, 127.2, -153.2, 25.9, 125.2, 5.0
    Verify - Joint Angles
    Load Autosequence

2. Set SSRMS Speed

    Set speed – Vernier
    Verify Speed – Vernier

3. Release brake

    Set Brake – Off
    Verify Brake – Off

4. Initialize Autosequence

    Confirm Autosequence
    Verify Autosequence – Commencing

5. Engage Brake when Autosequence Finished
    Set Brake - On
    Verify Brake – On

6. Proceed with repair

    Inform Astronaut to Proceed

<u>EVA Operations Task 2 and 3</u>

1. Configure Camera

    Set Monitor 1 – Camera 75
    Set Monitor 2 – Camera 90
    Set Monitor 3 – Camera 82

2. Setup Autosequence

    Select Autosequence
    Select Joint Angles (JA)
    Enter JA -65.4, 32.4, -85.1, -123.0, 11.5, 183.3
    Verify - Joint Angles
    Load Autosequence

3. Set SSRMS Speed

    Set speed – Vernier
    Verify Speed – Vernier

4. Release brake

    Set Brake – Off
    Verify Brake – Off

5. Initialize Autosequence

    Confirm Autosequence
    Verify Autosequence – Commencing

6. Engage Brake when Autosequence Finished
    Set Brake - On
    Verify Brake – On

7. Proceed with repair

    Inform Astronaut to Proceed

System Shutdown

1. Video Components shutdown

    Set CVIU 1 – Uninitialized
    Verify CVIU 1 – Uninitialized
    Verify CVIU 2 – Uninitialized
    Verify VSC – Off
    Verify VDU - Off

2. SSRMS Shutdown

    Set SSRMS Prime String – Off
    Verify SSRMS Prime – Off

    Set SSRMS Redundant String – Off
    Verify SSRMS Redundant – Off

3. Robotic Workstation Shutdown

    Set CEU – Off
    Verify CEU – Off

    Set DCP – Off
    Verify DCP – Off

## ***Space Station Remote Manipulator System (SSRMS) Failure Recovery***

Identify Failure

1. Identify Failure

    Set Safing – On
    Verify Safing – On
    Check error message (Comm → Step A and Video → Step 4)

Failure Recovery

*A. Communication Recovery Procedure*

1. Comm Recovery

    Set CEU – Uninitialized
    Set CEU – Initialized
    Verify CEU – Initialized

Set Comm – Enable
Verify Comm – Enabled

2. Switch to SSRMS Redundant String (if Error Message Persist)

   Set SSRMS Prime String – Off
   Verify SSRMS Prime – Keep-Alive

   Set SSRMS Redundant String – Operational
   Verify SSRMS Redundant – Operational

3. Test Comm Between RWS and SSRMS

   Set Safing - Off
   Verify Safing – Off

   Set Brake – Off
   Verify Brake – Off
   Deflect THC and RHC to ensure SSRMS moves

   Set Brake – On
   Verify Brake – On

B. *<u>Video Recovery Procedure</u>*

4. Video Error Recovery

   Set CVIU 1 - Off
   Set CVIU 1 - Initialized [Note: Wait 25 seconds to complete.]
   Verify CVIU 1 – Initialized

   Set CVIU 2 - Initialized [Note: Wait 20 seconds to complete.]
   Verify CVIU 2 - Initialized

   Set VSC – On
   Verify VSC – On

   Set VDU – On
   Verify VDU – On

5. Re-Configure Camera as required
   Set Monitor 1 to Camera 30
   Set Monitor 2 to Camera 75
   Set Monitor 3 to Camera 82

6. Remove Safing
   Set Safing – Off
   Verify Safing – Off

Resume operations

[Note: Autosequence would need to be re-set after safing has been turn on if error occurs during Autosequence.]

1. Setup Autosequence

   Select Autosequence
   Select Joint Angles (JA)
   Enter JA -125.2, 127.2, -153.2, 25.9, 125.2, 5.0
   Verify - Joint Angles
   Load Autosequence

2. Set SSRMS Speed

   Set speed – Vernier
   Verify Speed – Vernier

3. Release brake

   Set Brake – Off
   Verify Brake – Off

4. Initialize Autosequence

   Confirm Autosequence
   Verify Autosequence – Commencing

## REFERENCES

[1] "A Brief History of Space Exploration | The Aerospace Corporation." [Online]. Available: http://www.aerospace.org/education/stem-outreach/space-primer/a-brief-history-of-space-exploration/. [Accessed: 20-Jul-2016].

[2] N. Administrator, "July 20, 1969: One Giant Leap For Mankind," *NASA*, 19-Feb-2015. [Online]. Available: http://www.nasa.gov/mission_pages/apollo/apollo11.html. [Accessed: 20-Jul-2016].

[3] J. Wilson, "Journey to Mars Overview," *NASA*, 06-Mar-2015. [Online]. Available: http://www.nasa.gov/content/journey-to-mars-overview. [Accessed: 20-Jul-2016].

[4] N. A. Stanton, "Hierarchical task analysis: Developments, applications, and extensions," *Appl. Ergon.*, vol. 37, no. 1, pp. 55–79, Jan. 2006.

[5] D. P. Jenkins, N. Stanton, G. H. Walker, P. M. Salmon, and M. S. Young, "Creating interoperability between the Hierarchical Task Analysis and the Cognitive Work Analysis Tools," *Hum. Factors Integr. Def. Technol. Cent.*, Mar. 2006.

[6] N. Stanton, P. M. Salmon, and L. A. Rafferty, *Human Factors Methods: A Practical Guide for Engineering and Design*. Ashgate Publishing, Ltd., 2013.

[7] R. E. Forman, "Objective Performance Metrics for Improved Space Telerobotics Training," Massachusetts Institute of Technology, 2011.

[8] R. R. C. Galvan, "Effects of fatigue on simulated space telerobotics performance: a preliminary study analysis," Massachusetts Institute of Technology, 2012.

[9] C. Lowenthal, A. M. Liu, A. Natapoff, and C. M. Oman, "EFFECT OF SLEEPINESS ON PERFORMANCE AND WORKLOAD DURING SPACE ROBOTICS TASKS," in *SLEEP*, 2012, vol. 35, pp. A79–A79.

[10] "SSRMS Image." [Online]. Available: http://spaceflight.nasa.gov/gallery/images/station/crew-21/html/s129e009180.html. [Accessed: 11-Jul-2016].

[11] "Power_Data_Grapple_Fixture_on_station.jpg (2403×2041)." [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/8/8d/Power_Data_Grapple_Fixture_on_station.jpg. [Accessed: 11-Jul-2016].

[12] G. Gibbs and S. Sachdev, "Canada and the International Space Station program: Overview and status," *Acta Astronaut.*, vol. 51, no. 1–9, pp. 591–600, Jul. 2002.

[13] "168115main_05_shireman_012607.jpg (640×480)." [Online]. Available: http://www.nasa.gov/images/content/168115main_05_shireman_012607.jpg. [Accessed: 11-Jul-2016].

[14] "Space Station User's Guide | SpaceRef." [Online]. Available: http://www.spaceref.com/iss/elements/mss.html. [Accessed: 08-Mar-2016].

[15] N. J. Currie and B. Peacock, "International Space Station Robotic Systems Operations - a Human Factors Perspective," *Proc. Hum. Factors Ergon. Soc. Annu. Meet.*, vol. 46, no. 1, pp. 26–30, Sep. 2002.

[16] "Dynamic Onboard Ubiquitous Graphics (DOUG) Software Application." [Online]. Available: https://software.nasa.gov/software/MSC-23586-1. [Accessed: 01-Aug-2016].

[17] "Final Report of the International Space Station Independent Safety Task Force - 170368main_IIST_ Final Report.pdf." [Online]. Available:

http://www.nasa.gov/pdf/170368main_IIST_%20Final%20Report.pdf. [Accessed: 08-Mar-2016].
[18] K. Holden, N. Ezer, and G. Vos, “Evidence Report: Risk of Inadequate Human-Computer Interaction,” 01-Jan-2013.
[19] R. Rembala and S. Aziz, “Increasing the utilization of the ISS Mobile Servicing System through ground control,” *Acta Astronaut.*, vol. 61, no. 7–8, pp. 691–698, Oct. 2007.
[20] T. Tran, “NASA - Canadarm2 and the Mobile Servicing System.” [Online]. Available: http://www.nasa.gov/mission_pages/station/structure/elements/mss.html. [Accessed: 08-Mar-2016].
[21] Man Vehicle Laboratory, “Robotics Experiment - Fly-to and Grapple Task Training Version 1.0,” Fall-2010.
[22] Man Vehicle Laboratory, “Robotics Experiment - Track and Capture Task Training Version 1.0,” Fall-2010.
[23] A. Degani and E. L. Wiener, “On the design of flight-deck procedures,” Jun. 1994.
[24] D. Boorman, “Reducing Flight Crew Errors and Minimizing New Error Modes with Electronic Checklists,” 2000.
[25] M. R. Endsley, “Level of automation effects on performance, situation awareness and workload in a dynamic control task,” *Ergonomics*, vol. 42, no. 3, pp. 462–492, 1999.
[26] Liu, A. M. and C. M. Oman, “Design and Evaluation of Automated Electronic Checklists for Robotics Operations.” 03-Dec-2014.
[27] “Human Exploration Research Opportunities (HERO), NASA Research Announcement, NNJ14ZSA001N.” NASA Administration, 31-Jul-2014.
[28] A. Degani and E. L. Wiener, “Human factors of flight-deck checklists: The normal checklist,” May 1991.
[29] M. Martignano, M. Wolff, U. Brauer, and P. Kiernan, “Software assisted authoring and viewing of ISS crew procedures,” *Acta Astronaut.*, vol. 61, no. 11–12, pp. 1053–1060, Dec. 2007.
[30] D. Billman, D. Schreckenghost, and P. Miri, “Assessment of Alternative Interfaces for Manual Commanding of Spacecraft Systems Compatibility with Flexible Allocation Policies,” *Proc. Hum. Factors Ergon. Soc. Annu. Meet.*, vol. 58, no. 1, pp. 365–369, Sep. 2014.
[31] D. Kortenkamp, R. P. Bonasso, D. Schreckenghost, K. Dalal, V. Verma, and L. Wang, “A procedure representation language for human spaceflight operations,” in *Proceedings of the 9th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS-08)*, 2008.
[32] D. Schreckenghost, T. Milam, and D. Billman, “Human Performance with Procedure Automation to Manage Spacecraft System,” *Proc. IEEE Aerosp. Big Sky MT*, 2014.
[33] J. Annett and N. A. Stanton, *Task Analysis*. London: Taylor & Francis, 2000.
[34] F. TAYLOR, *Principles of Scientific Management*. Harper and Row: New York, 1911.
[35] F. B. Gilbreth and R. T. Kent, *Motion study*. Constable London, 1911.

[36] D. Diaper and N. Stanton, *The Handbook of Task Analysis for Human-Computer Interaction*. CRC Press, 2003.
[37] B. Kirwan and L. K. Ainsworth, *A Guide To Task Analysis: The Task Analysis Working Group*. London: Taylor & Francis, 1992.
[38] A. Crystal and B. Ellington, "Task analysis and human-computer interaction: approaches, techniques, and levels of analysis," *AMCIS 2004 Proc.*, Dec. 2004.
[39] J. Annett and K. D. Duncan, "TASK ANALYSIS AND TRAINING DESIGN.," 1967.
[40] E. Piso, "Task analysis for process-control tasks: The method of Annett et al. applied," *J. Occup. Psychol.*, vol. 54, no. 4, pp. 247–254, 1981.
[41] J. Annett, K. D. Duncan, R. B. Stammers, and M. J. Gray, *Task analysis. Department of Employment Training Information Paper 6*. HMSO, London, 1971.
[42] B. Kirwan, *A Guide To Practical Human Reliability Assessment*. London: Taylor & Francis, 1994.
[43] J. M. Schraagen, S. F. Chipman, and V. L. Shalin, *Cognitive Task Analysis*. Psychology Press, 2000.
[44] P. M. Salmon, D. P. Jenkins, N. A. Stanton, and G. H. Walker, "Hierarchical task analysis vs. cognitive work analysis: comparison of theory, methodology and contribution to system design," in *Theoretical Issues in Ergonomics Science*, 2010, vol. 11, pp. 504–531.
[45] K. J. Vicente, *Cognitive Work Analysis: Toward Safe, Productive, and Healthy Computer-Based Work*. Lawrence Erlbaum Associates, 1999.
[46] G. Lintern, "Tutorial: Work Domain Analysis." Cognitive Systems Design, 2013.
[47] T. Xiao, P. M. Sanderson, M. Mooij, and S. Fothergill, "Work domain analysis for assessing simulated worlds for ATC studies," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 2008, vol. 52, pp. 277–281.
[48] C. A. Miller and K. J. Vicente, "Comparison of Display Requirements Generated Via Hierarchical Task and Abstraction-Decomposition Space Analysis Techniques," *Int. J. Cogn. Ergon.*, vol. 5, no. 3, pp. 335–355, Sep. 2001.
[49] A. Sharon and D. Dori, "A Project -Product Model-Based Approach to Planning Work Breakdown Structures of Complex System Projects," *IEEE Syst. J.*, vol. 9, no. 2, pp. 366–376, Jun. 2015.
[50] Y. Mordecai and D. Dori, "Model-Based Operational-Functional Unified Specification for Mission Systems," *2016 Annu. IEEE Syst. Conf. SysCon*, pp. 1–8, Apr. 2016.
[51] Y. Mordecai and D. Dori, "6.5.1 I5: A Model-Based Framework for Architecting System-of-Systems Interoperability, Interconnectivity, Interfacing, Integration, and Interaction," *INCOSE Int. Symp.*, vol. 23, no. 1, pp. 1234–1255, Jun. 2013.
[52] Y. Mordecai and D. Dori, "Conceptual Modeling of System-Based Decision-Making," in *24th Annual INCOSE International Symposium*, Las Vegas, NV, USA, 2014.
[53] Y. Mordecai and D. Dori, "Agile modeling of an evolving ballistic missile defense system with Object-Process Methodology," in *Systems Conference (SysCon), 2015 9th Annual IEEE International*, 2015, pp. 839–846.
[54] Y. Mordecai and D. Dori, "Model-based protocol engineering: Specifying Kerberos with object-process methodology," in *Electrical & Electronics Engineers in Israel (IEEEI), 2014 IEEE 28th Convention of*, 2014, pp. 1–5.

[55] J. Somekh, G. Haimovich, A. Guterman, D. Dori, and M. Choder, "Conceptual Modeling of mRNA Decay Provokes New Hypotheses," *PLoS ONE*, vol. 9, no. 9, p. e107085, Sep. 2014.
[56] D. Dori, *Object-Process Methodolgy - A Holistic Systems Paradigm*. Springer-Verlag Berlin Heidelberg, 2002.
[57] D. Dori, *Model-Based Systems Engineering with OPM and SysML*. New York: Springer, 2016.
[58] "ISO/PAS 19450:2015 Automation system and integration -- Object-Process Methdology." 15-Dec-2015.
[59] D. Dori and S. Thipphayathetthana, "Model-based guidelines for user-centric satellite control software development," *Int. J. Satell. Commun. Netw.*, vol. 34, no. 2, pp. 295–319, 2016.
[60] E. Crawley, B. Cameron, and D. Selva, *System Architecture: Strategy and Product Development for Complex System*. Pearson Higher Education, Inc., 2016.
[61] D. Dori, C. Linchevski, R. Manor, and O. M. Opm, "OPCAT–An Object-Process CASE Tool for OPM-Based Conceptual Modelling," in *1st International Conference on Modelling and Management of Engineering Processes*, 2010, pp. 1–30.
[62] "OPCAT Version 3.0: Getting Started Guide." 2007.
[63] J. Nielsen, *Usability Engineering*. Academic Press, Inc., 1994.
[64] M. R. Endsley, "Situation awareness global assessment technique (SAGAT)," in *Aerospace and Electronics Conference, 1988. NAECON 1988., Proceedings of the IEEE 1988 National*, 1988, pp. 789–795 vol.3.