68th International Astronautical Congress (IAC), Adelaide, Australia, 25-29 September 2017.

IAC-17-B3.7.6

Next-Generation Human Extravehicular Spaceflight Operations Support Systems Development

Matthew J. Miller[a]*, Cameron W. Pittman[b], Karen M. Feigh[a]

[a] *Department of Aerospace Engineering, Georgia Institute of Technology, 270 Ferst Drive, Atlanta, Georgia, USA, 30332, mmiller@gatech.edu; karen.feigh@gatech.edu*
[b] *Software Engineer, Chicago, IL, USA, cameron.w.pittman@gmail.com*
* Corresponding Author

Abstract

This paper presents the research, design, and development efforts aimed at constructing next generation software that supports human extravehicular activity (EVA), commonly known as a spacewalk. EVA operations today rely on an extensive team of Earth-based flight controllers who actively monitor and direct EVA progress while maintaining crew and vehicle safety. However, future deep-space mission destinations will impose round-trip communication delays. In the case of Mars, round-trip delays range from 8 to 40 minutes. As a result, astronauts will need to rely on local decision support systems (DSS) to make tactical decisions during execution without Earth-based support personnel. This paper first presents the content, structure and form of existing EVA support systems that were leveraged as source material for prototype development. Then detailed descriptions of the engineering specifications and design features are provided of the two prototypes, (Baseline and Advanced), that were built. Finally, the phases software development and corresponding architectures are discussed. As a result of this effort, this study was able to harness the rapid prototyping capabilities and broad platform support of modern web technologies to iterate on designs and successfully deploy a high fidelity system to controlled laboratory simulations and NASA analog research sites. The results of this work provide an empirically derived set of design solutions to guide the future of EVA operational support systems for future human spaceflight missions.
Keywords: Decision Support System, Extravehicular Activity

1. Introduction and Motivation

NASA's Journey to Mars outlines a vision that includes sending humans to an asteroid by 2025 and to Mars in the 2030s. While the development of operational concepts and capabilities needed to send humans to Mars is under-way [1]-[4], additional attention to the specific support systems utilized by crew must also be made. One critical consideration in extending humans into deep-space is the unavoidable communication delays that will occur between Mars and Earth. Latencies ranging from 4 to 22 minute one-way light time (OWLT) are expected. As a consequence of these latencies, no longer will ground support personnel be able to support astronauts as they execute tasks or troubleshoot disturbances through immediate, or real-time, communications. In other words, the ability of mission control to impact crew actions on a moment-by-moment basis will become impaired, leaving the crew to cope locally with the tactical execution components of operations. One proposed solution to overcome these challenges is to provide crew technology-based support to help them cope with these disturbances [5]-[8]. But what might those support systems look like and what sort of capabilities will they provide?

This paper begins to address these problems by providing an example of one way to overcome the gamut of challenges that future operations will impose on the spaceflight community. Specifically, this paper discusses the design and development campaigns of next generation software that supports human extravehicular activity (EVA), commonly known as a spacewalk. EVA operations will play a critical role in future operations, enabling both the execution of scientific and exploration objectives. However, EVA operations rely on an extensive team of Earth-based flight controllers who actively monitor and direct EVA progress while maintaining crew and vehicle safety [9]. As a consequence of future concepts of operations, the allocation of information, responsibilities and ultimate authority over EVA execution will need to shift towards an architecture that enables the crew to ensure mission success, while also ensuring crew and vehicle safety [10].

The remainder of this paper is structured into 6 remaining sections. Section 2 provides a description of the present-day EVA work domain artefacts to provide overall context and describe the source material that was leveraged for prototype development. Section 3 summarizes the prototype development process and the details pertaining to the prototype designs. Section 4 describes the underlying code logic and development performed for the Advanced DSS. Section 5 describes the evolution of the prototypes from a software architecture and engineering perspective and highlights the implementation of various web-based technologies. Section 6 and 7 describe some key aspects of these development campaigns and provide some concluding remarks.

2. Brief Review of Present-day EVA Work Artefacts

Currently, teams of ground support personnel actively engage in the acquisition, synthesis, and processing of EVA data pertaining to both vehicle/spacesuit telemetry and timeline execution [9], [11]. The following sections describe these elements in detail and highlight existing EVA work artefacts. The aim here is to 1) provide a descriptions of work artefact from the existing EVA work domain to establish a common reference point, and 2) orient prototype development efforts towards standards and expectations within the domain we hope to support in a future context. Figure 1 shows a representative EVA ground support console that includes both life support system and timeline elements.

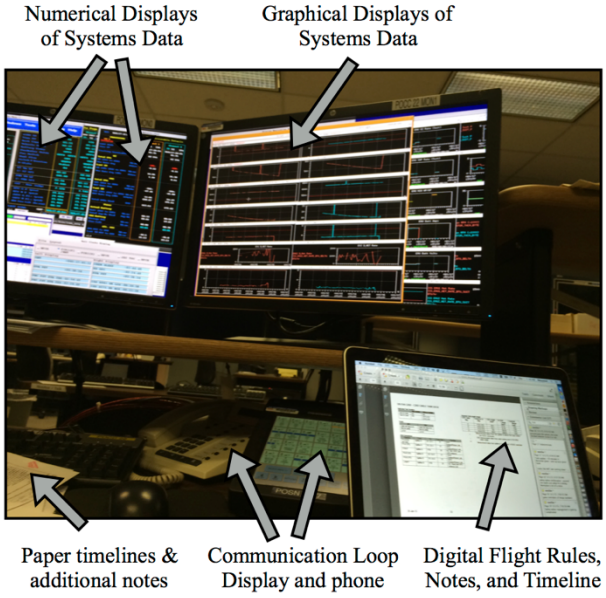

Fig. 1. EVA ground support console components.

2.1. Life Support System Elements

In order to maintain awareness and insight into the EVA spacesuit, telemetry is relayed to ground support work stations. Over 60+ variables per spacesuit are examined continually throughout EVA execution by trained personnel, in addition to a host of data pulled from the spacecraft airlock. The raw data is presented in two main views, numeric and graphic form, that entail a variety of subsystem and consumable variables. Initial developments in ground support EVA data systems were focused on providing flight controllers with a detailed view of the raw data provided by spacesuit subsystems. For a brief history of spacesuit tracking systems for Shuttle and ISS EVA, see Ref: [12]. While hard engineering limits do exist with 'automated' alerting systems on a majority of the data captured in the life support system displays, the bulk of insight gleaned from the data are obtained in more subtle ways. For example, flight controllers manually monitor trend changes over time as a way to forecast potential issues that have yet to occur. Cross-correlations are also manually made between numeric/trend values and alerting variables as a means to confirm and diagnose issues, and generate appropriate resolutions.

While these data systems at the helm of trained flight controllers were sufficient for supporting Shuttle and ISS EVA operations, handling this volume of raw data directly by astronauts will require the life support systems themselves to perform many of the higher level synthesis functions currently provided by ground support. For a more comprehensive discussion of the type of demands and capabilities future systems will need to manage this data, see Ref: [10].

2.2. Timeline Elements

In addition to digital console displays, EVA work domain artefacts include a variety of paper-based products. One key artefact is the EVA timeline (in its various forms) as shown in Figures A1 and A2 in the Appendix A. EVA is one of the most scripted, planned, and coordinated events that astronauts perform [9], [13]. As a result, a detailed set of procedures that can range from anywhere from 20 to 80 pages are developed throughout the planning and training process so that nearly every minute of the 7+ hours of EVA is specified. Two views of timeline information exist: 1) a summary timeline view and 2) a detailed procedure view. Figure A2 shows an annotated example of an ISS EVA summary timeline that conveys 6.5 hours of EVA activities. Each activity in the summary view has detailed procedures associated with them in the detailed procedure documentation. This volume of timeline detail and structure dates back to Apollo EVA missions. For a detailed examination of Apollo EVA timelines, see Ref: [14], [15].

If we ever hope to have productive crew in deep-space, the systems the crew utilize locally will need to support this high volume of timeline content in a way that is not detrimental to safety or productivity. Furthermore, crew will need to cope with the demands associated with life support system information. The remaining sections begin to explore how future crew might one-day manage both timeline and life support system information with the help of a decision support system to conduct future operations.

3. Decision Support System (DSS) Prototype Designs

This paper builds on prior works surrounding the development of future EVA operations [9], [10], [16]. Central to this research effort was to identify the constraints that shape EVA execution and understand the likely shifts and potential design solutions that could support these shift in the the EVA work domain for future operations. A set of software design requirements were generated from an in-depth examination of present-day EVA flight controllers and the cognitive demands they strive to overcome throughout EVA execution. A subset of assumptions and requirements are summarized below from this prior work to motivate the remainder of this paper.

Key assumptions:

- The *intravehicular (IV) operator* has been identified as a critical component of future operations, serving as a communication relay and field marshal of EVA execution.
- The work functions: *Timeline Management* and *Life Support System Management* are high priority functions that will require software support of some kind in future EVA operations.

  Critical DSS Requirements
  - The DSS shall integrate forecasted consumable affordances with as-performed and projected timeline tasks (Timeline Management)
  - The DSS shall associate LSS constraints with remaining timeline tasks (Life Support System Management)
- This research focuses specifically on the moment-by-moment *execution* of EVA, rather than aspects related to EVA planning or hardware development.

To help facilitate the DSS development process, two DSS prototypes were developed: Baseline DSS and Advanced DSS. The Baseline DSS mimicked present-day support capabilities to help illustrate the departures that will likely occur from present-day tools and operations. The Advanced DSS design explored how novel solutions might be applied to reshape the work that goes into supporting EVA operations as an IV operator. The Advanced DSS consisted of a novel timeline management tool that was developed and integrated with life support system data, known as Marvin.[*] The digitization of the EVA timeline was a major focus of the Advanced DSS because the current method of timeline management relies wholly on paper-based products and human integration across multiple data sources, neither of which will be sustainable for future operations.

3.1. Baseline DSS Prototype Design

The Baseline DSS is comprised of three main focus areas: life support system displays, timeline artefacts, and communication systems. Figure 2 shows each prototype component in the form of their respective focus areas which are described in the subsequent sections. The defining characteristic of the Baseline DSS is the extensive use modern-day work artefacts, reconfigured to be used by a single IV operator to support EVA operations.

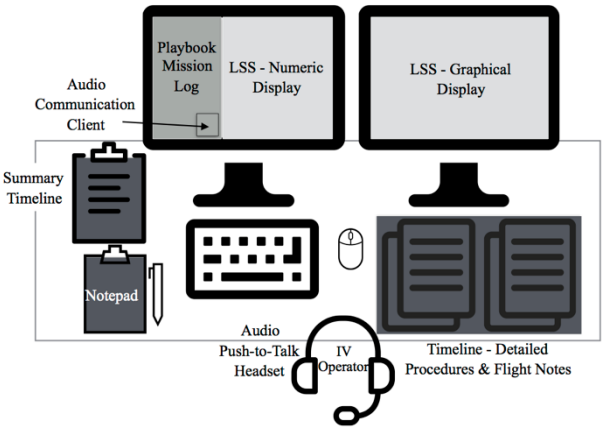

Fig. 2. Baseline DSS prototype. Shaded regions signify unique design focus areas.

3.1.1. Baseline DSS Life Support System Displays

The life support system (LSS) focus area is divided into two digital display types: 1) numerical displays and 2) graphical displays. The data included in these displays reflect the full volume of spacesuit data that present-day flight controllers manage. The interface formatting for the LSS displays also mimics present-day flight controller console displays.

The numerical display, as shown in Figure B1, arranges the data into spacesuit subsystem and consumable detail windows based on the current ISS spacesuit telemetry. Channel 1 and 2 represent EV 1 and EV 2, respectively into two columns within each window. Six consumable variables per crew member are provided in the consumables summary console. Any one of these consumable variables can impose a limiting constraint on the time available to perform an EVA. Additionally, the most limiting consumable for each and between crew members can fluctuate throughout the execution of an EVA. The most limiting consumable is shown in yellow

[*] Marvin is not an acronym as typically seen with NASA programs but rather is in reference to Marvin the Paranoid Android from Douglas Adam's science fiction series: The Hitchhiker's Guide to the Galaxy. – "There's only one life-form as intelligent as me within thirty parsecs of here and that's me." - Marvin

for each crew member throughout the EVA and are currently estimated based on sensor data and measurements made before and during the EVA in conjunction with estimated usage-rates to calculate a projected estimate of time remaining.

In addition to numeric raw sensor data and time estimates of EVA telemetry variables, a subset of the data (16 variables for each EV crew) is also shown in graphical form as shown in Figure B1. EVA telemetry can present in multiple trends formats including variables that increase and/or decrease, have various levels of sensor noise, are linear and/or exponential. Not only do the trend variables exhibit global trends, they also can exhibit local deviations such as step changes or slope changes that can have important implications for system performance even if the instantaneous values remain within acceptable engineering limits.

The LSS displays have limited interactive capability. The numeric windows can be rearranged and the zoom can be controlled to adjust the font sizes. Otherwise, the displays are passive and only meant to be monitored during EVA execution much like how flight controllers currently observe the variety of data during ISS EVA operations. Colour changes signify alerting events in the numeric display, and red line limits signify operational/engineering limits in the graphical display. In total, the LSS console displays represent a key portion of ISS spacesuit telemetry monitoring. While other auxiliary console displays are leveraged during ISS EVA operations, the LSS displays represent the core data acquisition system used for spacesuit telemetry [17]-[19].

3.1.2. Baseline DSS Timeline Artefacts

EVA timelines contain a variety of content that typically require months to years to develop. The overall intent of a timeline is to convey as much detail as necessary to ensure all task objectives are completed safely and to maintain coordination amongst the entire flight team. The Baseline DSS incorporated similar levels of content and detail exhibited in present-day ISS EVA operations timeline artefacts. As a result, all timeline products in the Baseline DSS were static, paper-based artefacts that maintain the structural format of ISS EVA operations. Additionally, Apollo EVA and NASA analog timeline content and objectives were incorporated into the Baseline DSS timelines to add flight relevant objectives pertinent to future planetary exploration objectives. Future operations are envisioned to contain more exploratory tasks which represent a departure from the engineering tasks primarily performed during ISS EVA operations. Figure B2 (see appendix) shows the resultant Baseline DSS summary and detailed procedure timeline products, respectively.

The summary timeline shown in Figure B2 conveys the sequence of activities to be performed for the entire period of the EVA. The summary timeline is intended to provide the highest level summary of activities to be completed in the desired sequence. The summary timeline content developed for the Baseline DSS was modelled after the 7-hour PET EVAs demonstrated during Apollo 17, which had the crew perform overhead activities to start and end the EVA with periods of translation and station activities throughout execution [14]. The EV1 and EV2 timelines are shown side-by-side to provide overviews of when each crew member is scheduled to perform particular activities. In this case, the two EV crew perform each activity group in-sync with each other. However, as shown in the detailed procedures in Figure B2, each EV crew performed different tasks within each activity group.

To provide a higher degree of timeline resolution, the detailed procedures document provided all necessary details required to achieve each EVA objective. The document is divided into three columns that correspond to each EVA flight team member [IV (LEFT), EV1 (Middle), EV2 (Right)]. In the Baseline DSS, an IV operator performs the stated tasks in the far-left column, while the EV crew perform the actions listed in the remaining columns. All primary tasks are underlined, all sub-tasks are numbered, and all procedures are indicated with checkboxes. Time duration estimates are provided where appropriate in the form of (Hours:Minutes) next to their respective task or subtask description. Additional features of the timeline include action boxes (in grey) to be filled out by the IV operator. Notes, Cautions, and Warning messages were inserted where appropriate throughout the document for reference.

To assist with the expected tasks of the IV operator, a Flight Notes calculation sheet was provided. This sheet supports three main categories of IV action: 1) minutes behind and timeline margin calculations, 2) recording numerical telemetry events, and 3) recording graphical telemetry events. The content of these specific actions is explained in Section 6. In general, these calculations and recording responsibilities are representative of the actions to be expected by an IV operator in a hypothetical future EVA. While such a structured sheet does not exist in reality, it reflects the *ad hoc* calculations performed by the fully certified EVA flight controllers now.

In total, the timeline products incorporated three paper artefacts to support timeline management. These materials assume an EVA timeline exists and its content remains static throughout EVA execution. The summary and detailed procedure views maintain the same form and structure as present-day ISS EVAs and the content was reflective of engineering and planetary exploration objectives taken from a blend of ISS and Apollo EVA activities and objectives.

3.1.3. Baseline DSS Communication System

Two forms of communication were supported in the Baseline DSS prototype: audio and text communication. An open source Voice over IP (VoIP) audio software known as Mumble/Murmur was utilized (See the Mumble website for more details[†]). Push-to-talk audio communication was provided between the EV and IV operators only. Text communication was provided between the IV operator with Earth-based support that included a simulated 5-minute one-way light time delay. The Playbook Mission Log text client shown in Figure 3 was provided on loan from the NASA Ames Playbook software team [20].

No streaming video footage was included in this prototype design. While video footage is typically utilized during present-day EVA operations, video footage is not required, per the flight rules, for present-day EVA operations.

Fig. 3. Playbook Mission Log text client.

3.2. Advanced DSS Prototype Design

The Advanced DSS is also comprised of three main focus areas: life support system displays, timeline displays, and communication systems. The focus areas themselves are dedicated to the same content included in the Baseline DSS design. However, instead of being isolated, the life support system and timeline focus areas are integrated in a key ways as described below. Figure 4 shows each prototype component in the form of its respective focus areas; each focus area is described in detail in the subsequent sections. The defining characteristic of the Advanced DSS is the transition to a digital timeline management tool with automatic calculations to alleviate the burden on the IV operator to manually fulfil those requirements.

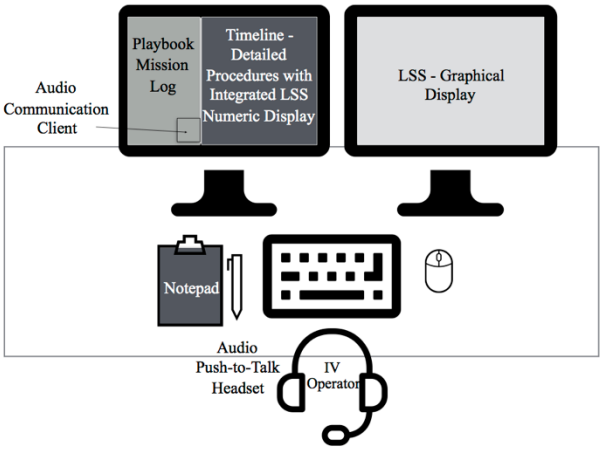

Fig. 4. Advanced DSS prototype. Shaded regions signify unique design focus areas.

3.2.1. Advanced DSS Timeline Modifications

The majority of the modifications made for the Advanced DSS addressed shortcomings to the timeline tracking. As such, these modifications are described in detail in Section 4.

3.2.2. Advanced DSS Life Support System Modifications

The LSS displays remain the same in terms of content. Future spacesuit systems are currently under development and their specific capabilities remain undefined. Therefore, existing ISS spacesuit and its telemetry data were maintained between the Baseline and Advanced DSS (Marvin) configuration, with some subtle user interface modifications.

The numeric console display is not readily visible in Marvin. Instead, an alerting feature indicates communication dropouts using an alert symbol. When an alert was triggered, an alert symbol appeared on the header to prompt user interaction by 'clicking' on the button to acknowledge the issue. The highest priority value (e.g. the most limiting consumable) was pulled from the consumable summary console and included in the header (top inch of the display) of Marvin for easy reference. Figure B3 shows these components within Marvin. The graphical displays remained the same between the two configurations as shown in Figure B1.

In summary, the Advanced DSS synthesizes LSS data in new ways. The LSS numeric display design was modified to show a subset of console variables in the

[†] For more details on the audio communication client see: https://www.mumble.com/

header view and introduces a new alerting feature that is triggered when anomalies arise. The telemetry graphical display remained consistent between the two DSS configurations. These slight modifications represent only a first step towards addressing the many potential avenues that exists to layer and present subsystem information and to include advanced analytics to support LSS data synthesis and processing. However, these efforts are beyond the scope of this work.

3.2.3. Advanced DSS Communication System Modifications

No changes were made to the audio or text communication systems described in Section 3.1.3.

4. Structuring and Defining an EVA Timeline

The major development effort within the Advanced DSS design was dedicated to digitizing the EVA timeline. But to successfully do that first requires an examination of how an EVA timeline can realistically evolve throughout EVA execution. Additionally, to make the timeline amenable to software development, a formalized a set of standardized rules must be developed. Figure 5 shows a generic summary timeline progress described in two categories that relate to the *tasks* being performed and *time* when they are performed.

As a result of knowing both the temporal and task distributions, the timeline can provide a reference point to support coordination amongst all EVA operators. It is the integration of these two components that enables a reference point to be made between actual execution and what was scripted. In an ideal world, every single planned task would be performed exactly when it was scheduled. However, EVA execution rarely goes as planned and timeline progress can evolve in multiple ways. Tasks may be added/dropped, reordered or modified while the timing of events may deviate from schedule (e.g. crew perform tasks faster or slower than expected or tasks are modified, thereby resulting in temporal shifts). Both types of timeline progress deviations must be expected in future missions; therefore, a digital timeline tool should be capable of supporting this variation.

Furthermore, consideration must be given to the sheer volume of tasks that are scripted in an EVA timeline. Every moment, of every EVA, is scripted and continually assessed to ensure adequate and safe progress is being made. Therefore, if a timeline is to be digitized, tasks within timelines must be decomposed into appropriate units that reflect this variety of task detail. These intrinsic units of action descriptions were derived from existing EVA timelines and are shown in Figure A2 where the generic term, Task, is decomposed into a hierarchy consisting of 4 levels (Activity, Task, Subtask, Procedure).

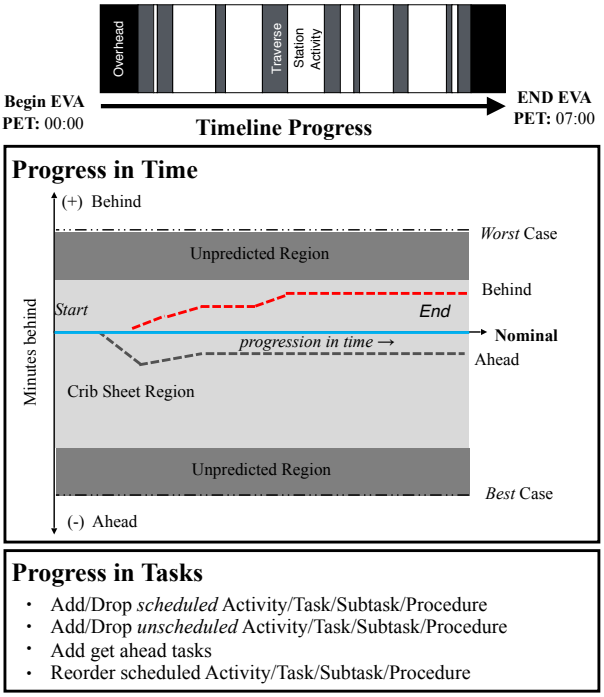

Fig. 5. EVA Timeline execution evolution tendencies.

From this perspective, the EVA timeline structure was generalized to a hierarchical data structure, known as called a "tree." Marvin needs to represent sequenced, multi-level actions within actions. As such, Marvin's timeline tree consists of an ordered and four level deep nested series of EVA actions as shown in Figure 6. Based on this current design, every moment of the EVA is represented within one of the four levels of Marvin's timeline tree.

A single action within a level of the Timeline is called a Step. We defined four hierarchical Steps of a Timeline: Activity, Task, Subtask, Procedure. Activities represent the highest level of the Timeline. They may account for hours of EVA time and identify distinct phases (e.g. translations, station operations, overhead activity, etc.) Each Activity may include zero or more Task children. Tasks describe checkpoints within Activities, usually measured on the order of tens of minutes. For instance, a station operation Activity may include Tasks that represent phases of scientific experiments. Within translation activities, Tasks may represent reaching traversal waypoints.

Tasks may include zero or more Subtask children. Subtasks begin to describe minute-by-minute, step-by-step aspects of a timeline. For instance, the Task of operating a science apparatus may include Subtasks for configuring individual components. Finally, each Subtask may include zero or more procedures, where a

Procedure represents a quantum of action by the crew member (e.g. tightening a bolt, turning a lever, actuating a telescopic arm, etc.).

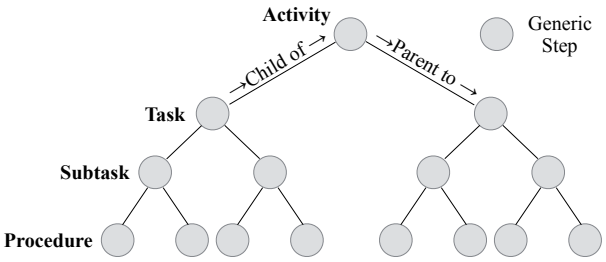

Fig. 6. EVA timeline tree structure.

Note that children are always optional. The level of detail included in a timeline is a decision for the EVA design team - Marvin's timeline calculations work the same regardless of the volume of content. However, Marvin's ability to report accurate timeline progress increases with greater timeline granularity.

Each Step must include a short description and may optionally include a planned duration, or PET, required to complete the Step (measured in seconds). Marvin also tracks each crew member individually, and as such each step must also identify the crew member channel responsible for it (with each crew member's data occupying a channel). In total, there are five possible states for any Step at any given instance during execution, all of which are determined on the fly every second during EVA progress:

Step States:

- **Fresh**: incomplete and has never run.
- **Active**: is currently running and is not complete.
- **Complete**: has run and has been completed.
- **Paused**: has run and has been stopped without being completed.
- **Skipped**: has never run but is marked complete. Presently, Marvin treats this situation the same as a complete step.

4.1. Step-level Ephemeral Timing Data

Marvin differentiates timeline descriptions, which may be shared across EVAs, from timing data collected during a mission. Marvin takes the view that timing data reflects the execution status of a single EVA. It follows that Marvin must also differentiate planned and elapsed durations. While planned durations are metadata that may be shared across missions, Marvin calculates elapsed durations on the fly during EVA execution. Marvin tracks three separate time-related values for every step:

- **Planned durations**, which may be set explicitly in timeline metadata or determined implicitly.
- **Elapsed durations**, which are calculated as either PET_{end} - PET_{start} or PET_{now} - PET_{start} depending on the Step's state of completion.
- **Projected durations** which only apply to active steps. This projection estimates the time to complete an active step based on how much time has elapsed and how much time was planned. This calculation provides the most insight when applied against parents with many children, or the Timeline as a whole, because timing calculations against high level Steps are effectively aggregates of the planned and elapsed durations of lower level children.

4.2. Timeline Linearization

Trees are inherently nonlinear, which poses a problem for calculations against time, which flows linearly. Calculating time values for a parent with children necessitates knowledge of the state of all of its children. Timeline progress metrics similarly suffer from this characteristic. Deciding if a parent is active requires knowledge of the state of every deep child (i.e. every generation of children to the bottom of the tree) such that $Parent_{is\ active} = child_{1\ is\ active} \vee child_{2\ is\ active} \vee \ldots child_{n\ is\ active}$. If any child Step is active within a parent, Marvin treats the parent as active as well.

Two potential solutions were examined to simplify the hierarchical timeline math. In the first solution, the timeline was flattened into a linear path based on Kahn's Algorithm for topological sorting[‡], and in the second the bottom layer of the timeline was directly pruned. While it may be mathematically possible to work with Kahn's Algorithm, we found that it translated the same hierarchical issues to a linear model without simplifying calculations or simplifying Marvin's software architecture thereby making the math more difficult to reason about intuitively. As a result, the timeline was pruned to the bottom layer which created a linear timeline made of ministeps ("mini" in the sense that they include some of but not all of the information contained within Steps in the timeline). The following rules [R] and postulates [P] were developed and used in the transformation and generation of the linear timeline:

R1. Planned EVA duration = sum of ministep durations
R2. (Simplified) Child duration = parent duration / number of children under parent
R3. (Detailed) Child duration = (parent duration - manually allocated child duration) / number of children without manually allocated duration under parent

[‡] For more details on topological sorting, see: https://en.wikipedia.org/wiki/Topological_sorting

R4. Linear Timeline (LT) = list in order of steps without children

P1. Every instant of the EVA must be represented in the Timeline.
P2. Time waterfalls from parents to children evenly to the lowest level of the tree, such that the sum of the times (planned or elapsed) of a parent's children equals the time of the parent.
   a. A child may have a manually set planned duration, in which case time flows from the parent to the child's siblings evenly.
P3. Only the lowest children in the tree may be used for time calculations.

Rule 1 (R1) and Postulate 1 (P1) clarify that upon completing one step, the next step begins without downtime. R2 and P2 dictate that every parent distributes its planned time to its children. R2 states that the planned duration of a parent should spread evenly amongst its children. R3 clarifies how to account for children with manually set durations. If those children have children, they redistribute their time to the next generation down. As such, every planned moment of the timeline sinks to the bottom of the tree. R4 and P3 cull the lowest level of the tree where the lowest level is a step without children, regardless of the type of step, and places these steps in a separate, continuous Linear Timeline that represents the entire mission in chronological sequence. Marvin tracks the connection between Timeline and Linear Timeline, such that it can identify which steps in the Linear Timeline correspond to a step at any level in the Timeline.

4.3. Timeline Calculations

To perform time calculations against a parent, Marvin identifies its deep children in the Linear Timeline by traversing from the original parent in the Timeline down through deep children until it finds Steps that do not have children.

The following general formulas apply to both individual Steps and the whole Timeline. Note that Marvin does not make a distinction between calculations running against a single Step and the entire Timeline. Calculations against the whole Timeline are, in effect, calculations against a parent Step with every other Step as a deep child, making every Ministep in the Linear Timeline a child.

It is worth noting that Marvin creates a Linear Timeline for each channel of the Timeline (i.e. every EVA crew Timeline is linearized independently). Though there is usually strong correlation between EV crew Timelines and their execution objectives, Marvin takes a naïve stance and assumes independent execution. As such, many overall timeline calculations (especially any involving PET) involve performing the same calculations against Linear Timelines for both crewmembers (e.g. channels) and then returning the most conservative result. For instance, if PET Remaining for EV 1 is 01:50 while EV 2 is 01:55, the overall PET Remaining reported is the greater of the two, 01:55.

The following calculations, shown in Table 1, are made throughout EVA execution to populate the Marvin header for IV consumption.

Table 1. Loop calculations.

Planned Duration	Sum planned duration for each ministep from 0 to total ministeps [planned durations n]
PET Remaining	(remaining time for *running* ministeps) + (planned time for *fresh* ministeps) + (projected time remaining for *paused* ministeps)
Projected Duration	(duration of *completed* ministeps) + (planned duration of *fresh* ministeps) + (planned duration OR elapsed duration, whichever is larger, for *active* ministeps and *paused* steps); elapsed duration of *completed* and *skipped* steps, planned duration of *active* steps that are under time, elapsed duration of *active* steps over time, planned duration of *fresh* steps. Note that project durations will grow as actual step durations exceed planned durations.
Is a Step Active?	$Step_{is\ active} = child_{1\ is\ active} \vee child_{2\ is\ active} \vee \ldots child_{n\ is\ active}$
Time Behind	(Planned EVA Duration) - (Projected EVA Duration); Note that Time Behind is only ever used in the context of the entire Timeline.
Timeline Margin	(PET Remaining for channel) - (Limiting Consumable remaining for channel); Note that with the current iteration of Marvin, Timeline Margin represents the one and only metric on the timeline view that integrates telemetry. Once again, the same calculation is performed against each crewmember with the most conservative value returned.
Time Until Start	(sum of PET remaining for incomplete, e.g. active, fresh, or paused, preceding ministeps); Note that for each Activity, this variable calculates the projected time to complete all preceding Activities, which provides an estimate of when that Activity should start.

Timeline calculations on the loop are performed once every second.

4.4. Resultant DSS Support Features

As a result of the formalized constructs pertaining to describing the EVA timeline, a more streamlined and potentially desirable DSS prototype was developed. Figure 7 shows the schematic that depicts how the Baseline DSS supports an IV operator required to calculate Timeline Margin and Minutes Behind/Ahead. While the Baseline DSS does provide all necessary components to perform such calculations, the information remains dispersed among a variety of

locations and the IV operator is responsible retrieving and integrating this data.

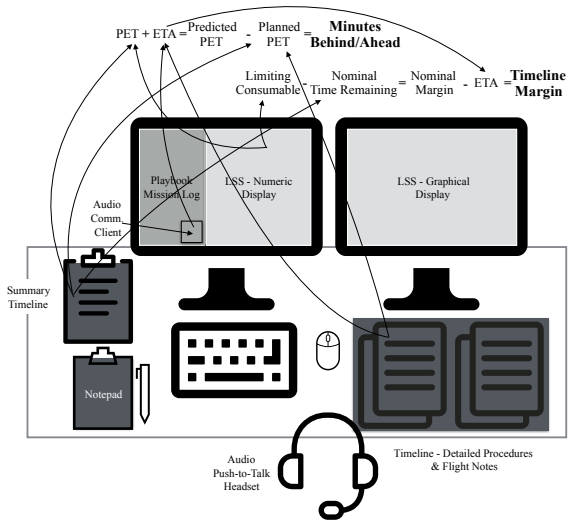

Fig. 7. Baseline DSS integration capability.

In direct contrast, the Advanced DSS facilitates the same calculations as a by-product of the IV operator tracking timeline progress. The IV operator interacts with Marvin by 'checking-off' timeline steps upon verbal verification of step completion from the EV crew. Figure 8 shows the resultant pathway to generate Timeline Margin and Minutes Behind/Ahead.

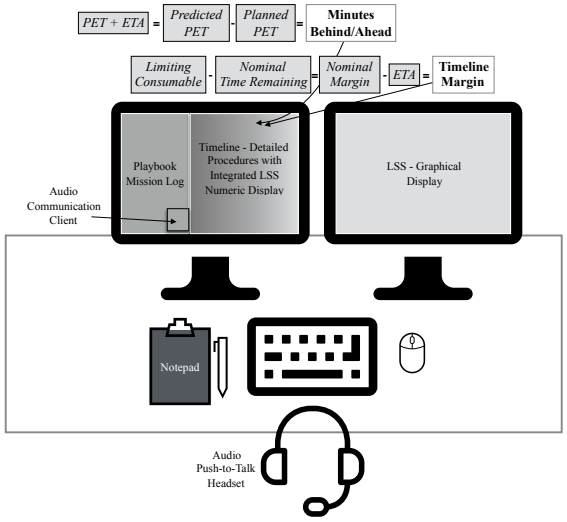

Fig. 8. Baseline DSS integration capability.

Given the ill-defined nature surrounding future EVA operations, these two DSS prototypes represent an incremental development pathway towards novel software support. While these design features and capabilities may appear straightforward, there exists a non-trivial amount of effort required to construct these prototype designs, as highlighted in the subsequent section.

5. Prototype Software Development Evolution

To this point, we have highlighted the design features of two DSS prototypes (Baseline and Advanced). However, three software prototypes were actually developed throughout this project. The first two prototypes correspond to the Baseline and Advanced DSS prototypes respectively, while the third prototype combined the other two in a more stable architecture. This section highlights each phase of prototype development.

Each prototype consists of three primary components - a *front-end application*, a *loop* and a *database*. A front-end application (i.e. front-end) is a website that loads in a web browser with an interface that users can see and click (i.e. a user interface (UI)). Each open instance of the website is generally referred to as a *client*. The loop and database operate in the background or on a server. A loop performs timeline calculations, while a database stores any information related to timelines and telemetry data in a medium that simplifies the loop's work. Browsers build websites by running HTML, CSS and JavaScript files[§]. Web developers almost universally use libraries of code called frameworks that simplify web development by adding the ability to define templates into which data can be injected and rendered. We used a mix of the Polymer and React frameworks throughout development. Databases are software capable of storing arbitrary data in a structured way, allowing the execution of search queries to retrieve specific stored data and the ability to share data to clients over a network.

Lastly, each prototype runs a loop to periodically update timeline calculations against each step and the timeline, or to periodically generate telemetry that is a composition of other telemetry data.

5.1. Phase 1 - making a Baseline DSS

The Baseline DSS uses a linear data flow designed to mimic real data recorders, in which raw electrical signals would be received by some hardware, converted into structured data and passed along to some kind of storage and display logic. The Baseline DSS utilized a simplified scheme, as shown in Figure 9.

[§] HTML controls the content and structure of a website, CSS controls styles, and JavaScript can change content, structure and style on the fly.

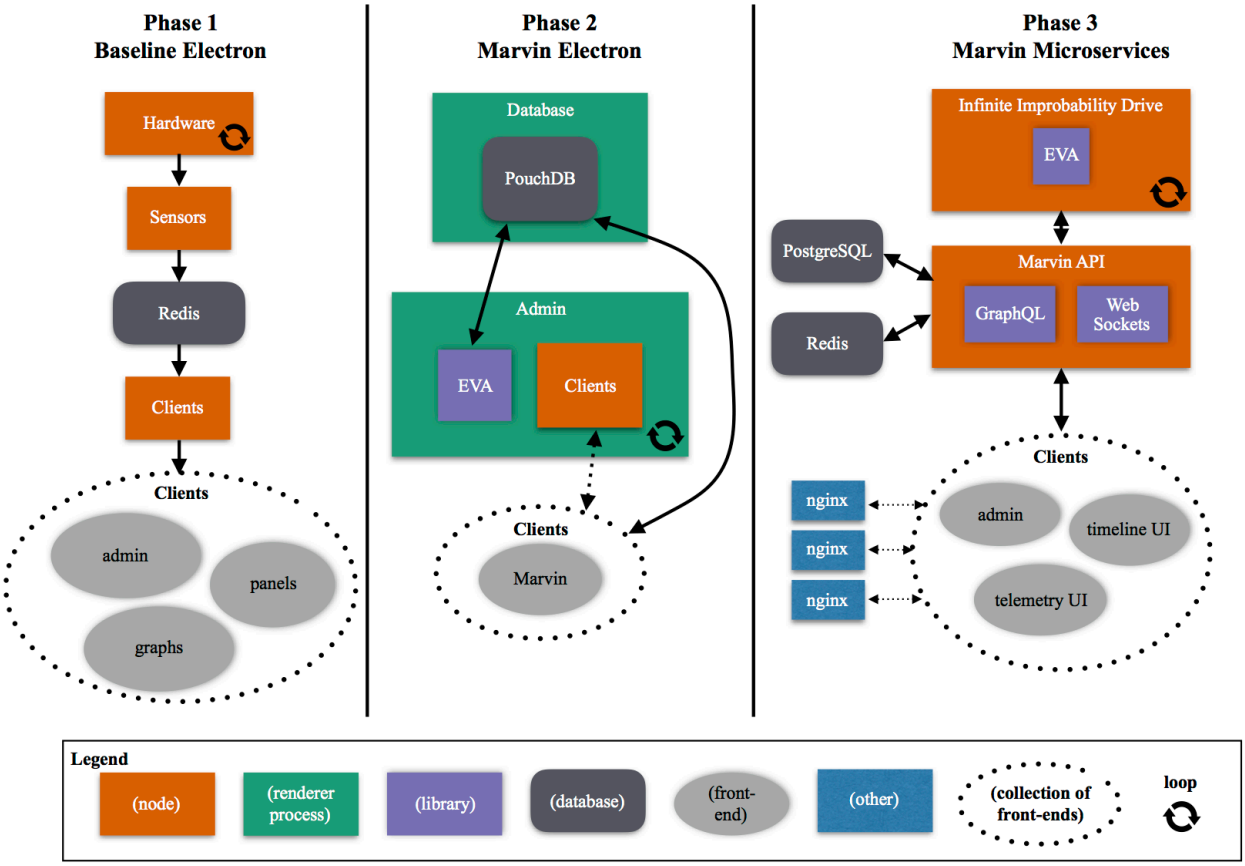

Fig. 9. DSS prototype phases of development architectures.

The Hardware layer acts like the data receiver, except rather than recording analog sensor signals, it parses simulated telemetry from a CSV file. The Sensors layer receives simulated "raw" hardware signals while structuring the information into distinct channels for each sensor. The Sensors layer publishes the parsed telemetry to a Redis database running in publish/subscribe mode,[**] which stores the telemetry and notifies the final layer, a Client server, that new data is ready. The Client layer serves HTML, CSS and JavaScript files to browsers. It also is responsible for disseminating telemetry to web browsers in a data format that browsers can easily receive, parse and display. To do so, it uses web sockets.[††]

To run the Baseline DSS, the researcher had to download the source code, install dependencies and run shell scripts to start and stop the servers on a local machine. Any computer on the same internet network could access the Baseline DSS using a custom I.P. address.

5.2. *Phase 2 - making an Advanced DSS*

During Phase 2, the deployment architecture was switched to enable the Advanced DSS to incorporate new timeline management capabilities and to be operated without the need for installing and interacting with source code. The Electron [‡‡] webview framework was implemented to build a desktop application that operated as a stand-alone software application without the need for source-code installation. This application was called Marvin. Electron packages a full browser (in this case, Chromium) along with the HTML, CSS and JavaScript files needed to run a website. The application UI is actually a website in a custom browser window that renders the files that the developer packaged inside the application.

Marvin contains the database, the loop, front-end code and a server to allow research subjects to access the timeline UI from both the local application and a browser on the same network. Marvin runs a PouchDB[§§] database to allow for storage and retrieval of timeline and

[**] In publish/subscribe mode, clients can create a subscription that guarantees Redis will let the clients know when a specific type of data has changed.

[††] For more information on web sockets, see: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

[‡‡] For more information on Electron, see: https://electron.atom.io/

[§§] For more information on Electron, see: https://pouchdb.com/

telemetry data. The database runs inside a separate browser window running in the background (i.e. hidden from the user's view). PouchDB was configured in a simplified way that enabled clients to receive data directly from the database. However, allowing clients to have unfettered access to a database means there are no checks to prevent potentially destructive actions, leading to potential stability issues and data loss. This issue is addressed in phase 3 development. PouchDB was chosen because it allows the same publish/subscribe model as Redis. As such, clients listen for changes to the database and then update their UIs when new data arrive. In effect, updating the database simultaneously broadcasts data to all clients, keeping their UIs in sync with the latest information.

The Advanced DSS was comprised of two subcomponents within the loop. The first subcomponent runs at one hertz to merge the timeline and ephemeral timing data into a larger structure to be stored and immediately broadcasted. The second calculates instantaneous values, such as limiting consumable, from telemetry data every 20 seconds. While this version of Marvin works well as a prototype, it suffers from its reliance on web browser windows. Browsers inherently limit both the processing power and memory available to any application running in them (for many reasons well outside the scope of this paper). More importantly, browser windows are designed to run applications for only a few minutes at a time. They tend to crash for reasons outside the developer's control without any automated restart mechanism.

5.3. Phase 3 - Advanced DSS on the path to production

The third phase of development involved transforming Marvin into an architecture that could run reliably for any number of users with minimal monitoring and maintenance requirements. These efforts were made to support the NASA Extreme Environment Mission Operations project (NEEMO 22) conducted in June 2017 in the Aquarius Habitat in Key Largo, Florida. Within this environment, four crew members lived and worked within the Aquarius habitat for 10 days, performing EVA on the surrounding coral reef. The IV crew member within Aquarius and Mission Control used Marvin for four mission days to track timeline progress during EVA. The IV workstation is shown in Figure 10.

Fig. 10. NEEMO 22 IV workstation with Marvin software (shown in the far right monitor).

To enable high reliability and performance, Marvin was re-architected as a cluster of microservices[***] to take advantage of modern tools designed to run software in the cloud. One key technology, known as Docker,[†††] enabled the easy creation, distribution and operations of file systems as if they were single files. Imagine running many sandboxed computers inside a single host computer and being able to easily distribute those sub-computers so that other hosts can run them. In a distributable form, these computers are called images. When they are running, the images are called containers.

By breaking up Marvin's architecture into separate microservices (e.g. Docker images), the ability to monitor and automatically restart applications was now possible. For instance, if the loop were to throw an unexpected error, an Electron application, as developed in phase 2, would crash and render Marvin useless. In this new architecture, the loop automatically restarts and the other components continue running completely unaware that anything happened. More importantly, users would never know anything happened (and neither would the researcher unless they searched the logs). This capability provides a high degree of fault tolerance. Furthermore, by moving Marvin's core functionality outside a web browser, Marvin components now have access to the entirety of the host machine's CPU and memory resources, thereby boosting performance.

In this configuration, Marvin consists of many microservices - a loop, two databases, an API and three separate servers (one for the timeline UI from the previous prototype, one for the telemetry UI from the Baseline DSS and a new admin site for uploading new timeline and telemetry files). Docker Compose was implemented to orchestrate the microservices. While this development effort resulted in quite a bit of new code development, much of the original user interface and

[***] A technical (but understandable) introduction to microservices: https://www.nginx.com/blog/introduction-to-microservices/

[†††] For more information on Docker, see https://www.docker.com/

timeline logic code developed in Phase 1 and 2 was reused. Phase 3 software development involved the departure from tightly coupled software components with well-defined, rigid flows of information, to one where a "switchboard" enabled independent components to easily mix and match information without affecting one another. The switchboard itself consisted of one of the new databases and the new API. Specifically, a PostgreSQL database was implemented which uses a relational schema to store the entirety of telemetry and timeline data, while promoting a fast and efficient way to query varied information.

Furthermore, an API was implemented to simplify, organize and speed up non-trivial queries from multiple sources, many of which included values calculated on the fly based on current timeline and telemetry data. In the Marvin ecosystem, all information flows through the API. The loop sends updated timeline and telemetry calculations to the API, and the API is responsible for providing new data to clients. Clients may also arbitrarily query the API for information, which occurs most often when a user loads or refreshes the client.

To support NEEMO 22 operations, Marvin was deployed on a small, portable computer designed to run commercial video displays that was outfitted with an Intel i7 CPU and 32 GB of memory. The server (named the Heart of Gold) utilized Docker Compose to download and run the latest images for our microservices. The Heart of Gold existed locally onsite at NEEMO 22 and connected directly to the local intranet to avoid potentially limited Internet connections. Marvin experienced 100% uptime during NEEMO 22 with minimal perceived performance issues from the user's perspective.

6. Discussion

6.1. Bridging the gap between the existing and future domains

> "These tools permit us to transform difficult tasks into ones that can be done by pattern matching, by the manipulation of simple physical systems, or by mental simulations of manipulations of simple physical systems. These tools are useful precisely because the cognitive processes required to manipulate them are not the computational processes accomplished by their manipulation. The computational constraints of the problem have been built into the physical structures of the tools [21, p. 171]."

Hutchins' quote epitomizes the translation we aimed to enable in this prototype development effort. The paper examined how two DSS designs could support EVA operations and how specific design features could integrate the intrinsic work domain computations within the 'software' structure. The hierarchical task structure already embodied within the EVA paper products and the realistic volume of life support system detail provided the necessary context to consider when attempting to execute EVA successfully.

This effort leads to a richer and more complete description of the assumptions being made (and purposefully omitted) with respect to new technologies and the environment within which they are desired. By fabricating these components, we promote constructive criticism to its short comings as a means to hopefully yield a more appropriate and realistic design solution for future EVA operations. The perspective of extensibility is key to this effort and making strides towards not only addressing and overcoming key challenges within the domain of interest, but also establish the necessary conditions from which new designs and software capabilities that can leveraged. As a result, this work represents a first step towards informing the necessary computational support dedicated to supporting future EVA operations.

6.2. Leveraging modern-day technologies for software development

The current state of software development encourages and rewards the use of open-source tools. Very few software projects require 100% novel, custom components - the vast majority of software development consists of piecing together solutions from freely available components. This approach let us quickly iterate to minimum viable products (MVPs) during the different phases of development. Each MVP was essentially a core of proprietary algorithms surrounded by shells of open-source libraries and frameworks.

While this approach enabled the first two phases of development, phase 3 presented a new set of demands beyond simply meeting the needs of EVA work domain. To support the NEEMO simulation, Marvin needed to become a production-ready system, which means that the software could support a large number of concurrent users without noticeable downtime or bugs. Fortunately phase 3 developments could also leverage established development practices to meet this production-ready state even with a short development cycle and minimal testing. To be clear, the result of phase 3 was still very much a prototype architecture, but it did successfully blend the capabilities of both phase 1 and 2 development into a production-ready framework that performed as expected at the NEEMO deployment.

The three phases of development collectively were found useful to establish the necessary demands of the work the software was intended to support. Software development requires a clear understanding of the rules of the system to be develop as emphasized in Section 4.

During the second phase of development, the EVA timeline was decomposed to a level of abstraction that encompassed the levels of detail in the current EVA work domain while being amenable to translation into a well-defined data structure. This translation is paramount to ensuring software development that meets the needs of its users but also is extensible to cope with potential future needs.

6.3. Providing effective and meaningful support

Central to this study was to examine how to best synthesize timeline and life support system data in a meaningful way that was conducive to support EVA operations. Figures 7 and 8 demonstrate feasible ways in which EVA operational information can be produced. In one instance, the astronaut is relied upon to make the bulk of data association. In the other case, the software synthesizes the same information as a by product of the crew utilizing the software. These prototype designs represent potential hypotheses as to how to best provide enable future astronauts to perform EVA.

While the laboratory and NASA analog testing campaigns are topic of future work and outside the scope of this paper, a few general trends were observed. The users of the Advanced DSS produced more consistent estimates of timeline margin and minutes behind with less variability as compared to when they used the Baseline DSS. Additionally, the ability to produce these estimates 'on-demand' was markedly improved when using the Advanced DSS. As a result, the users of the Advanced DSS were capable of providing an estimate at a moment's notice and were not expending unnecessary time and effort to produce an estimate.

The aim of this development effort was not to develop software capable of producing decisions for the astronauts. Instead, the software, by virtue of the astronaut interacting with it, produced mission critical information in a manner and format conducive for future astronauts to make one of potentially many decisions. Their decisions however will be shaped by the underlying constraints they face. In this case, EVA operations are inherently influenced by the constraints imposed by both timeline execution and life support system performances. Therefore, if future crews are to be expected to be productive in future EVA, their ability to synthesize and react to those constraints will play a key component in their future success.

7. Conclusions

This study demonstrates the design, development and testing campaign related to how to support future EVA operations. By leveraging the requirements derived in prior work (see Refs: [10], [22]), two prototype designs were hypothesized and developed into functional software tools. A Baseline DSS was developed to reflect existing work domain artefacts, re-imaged for use in future EVA operations. An Advanced DSS was developed and incorporated novel design components to establish a first-of-a-kind digital management system for future EVA operations. For the first time, EVA work domain artefacts were designed in software that reflected both the content, form, and demands associated with EVA execution.

Specifically, life support system consumable constraints were directly linked to timeline tracking information to yield necessary EVA execution metrics such as timeline margin and minute behind/ahead estimates. Additionally, not only are these mission critical elements explicitly incorporated into the prototype design, they are generated in a way that likely reflects the work practices of future intravehicular (IV) operators. This study evaluated the resultant prototypes in flight-like scenarios as means to quantify and establish objective measures of desired performance of future EVA operations.

Acknowledgements

This research was sponsored by the NASA Space Technology Research Fellowship - Grant # NNX13AL32H. The authors would like to thank the members of the NASA community and Cognitive Engineering Center for their continued support and guidance in the completion of this work. The opinions expressed in this study are of that authors alone and are not reflective of NASA.

Appendix A (Existing EVA Timeline Artefacts)

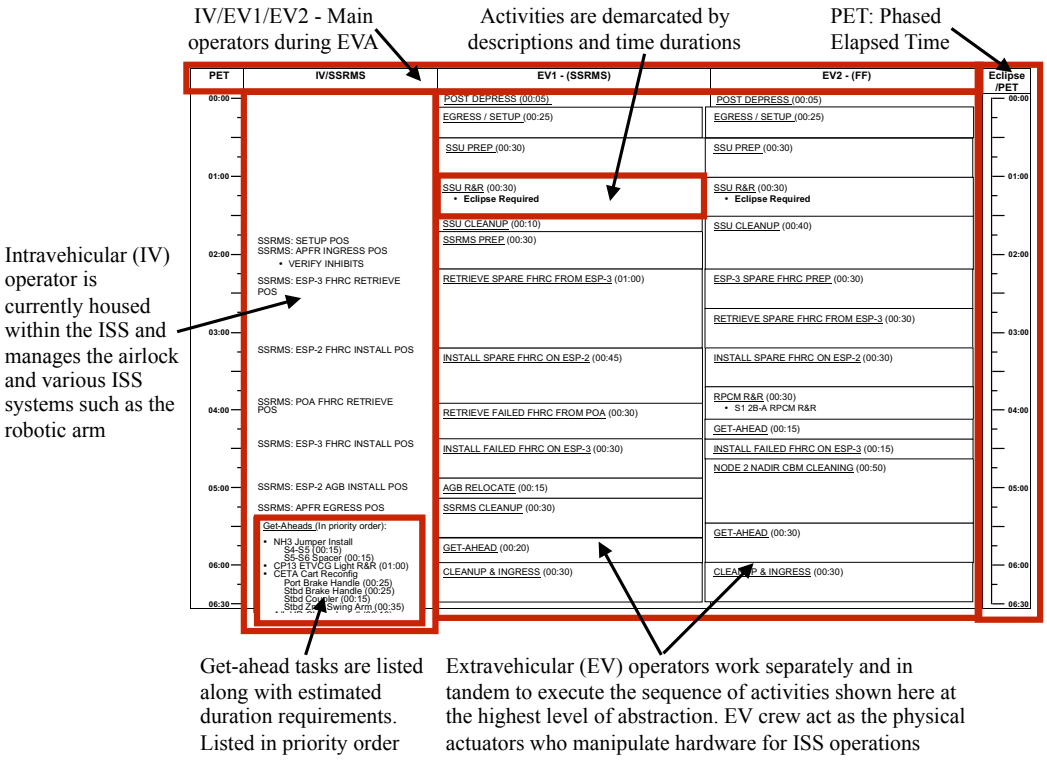

Figure A1. EVA summary timeline example with annotations.

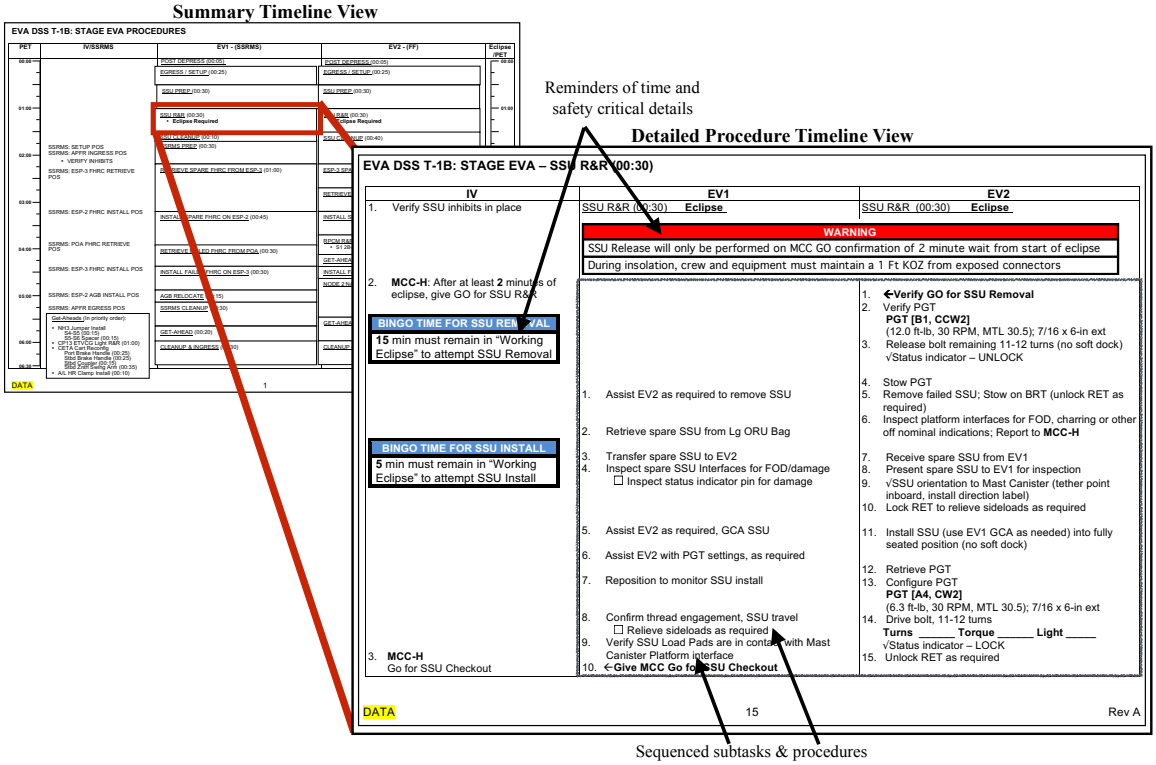

Figure A2: EVA summary and detailed procedure timeline example with annotations.

Appendix B (Prototype Design Screenshots)

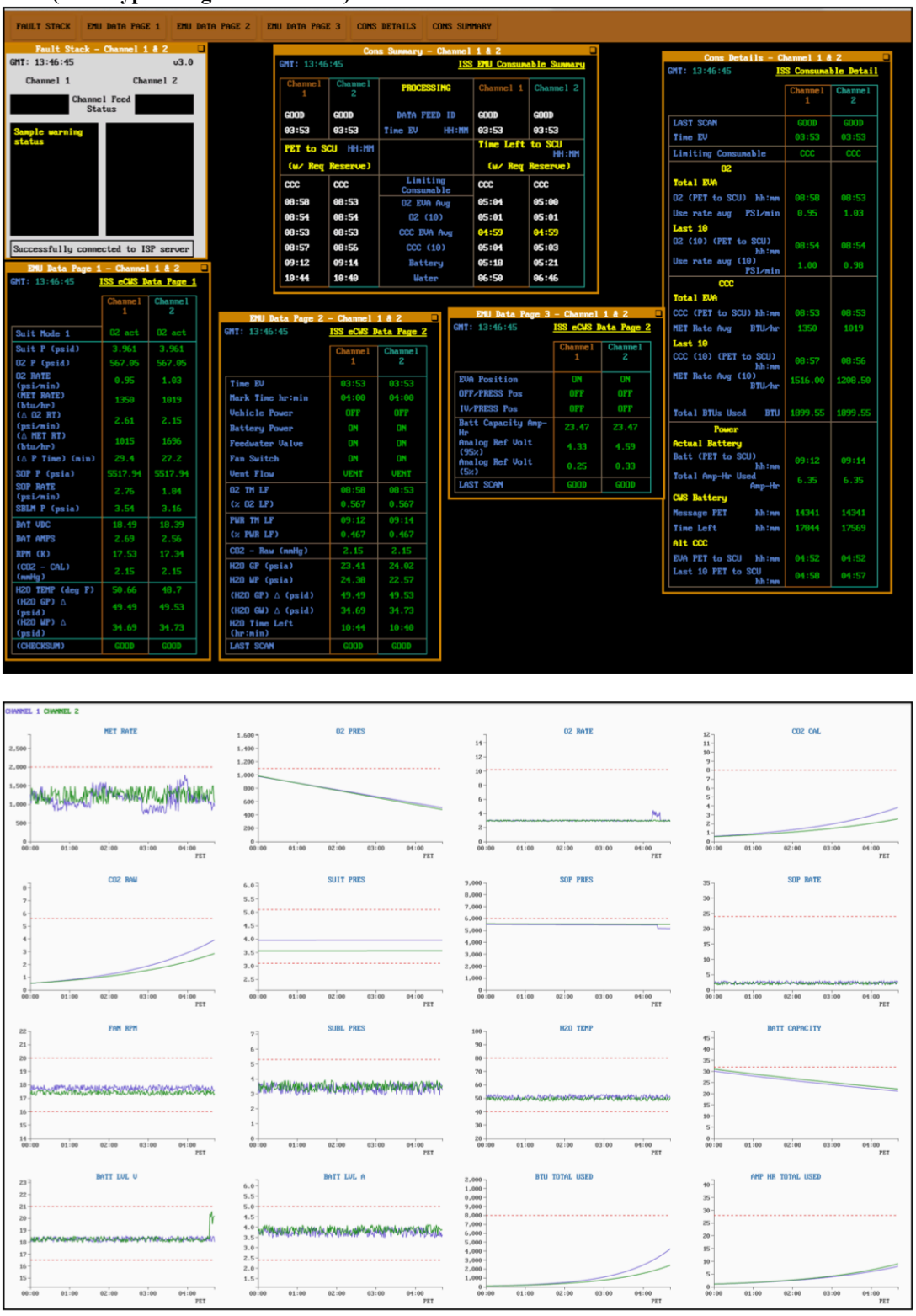

Figure B1. Baseline DSS numeric (TOP) and graphical (BOTTOM) displays of life support system data. The graphical display was also used in the Advanced DSS prototype.

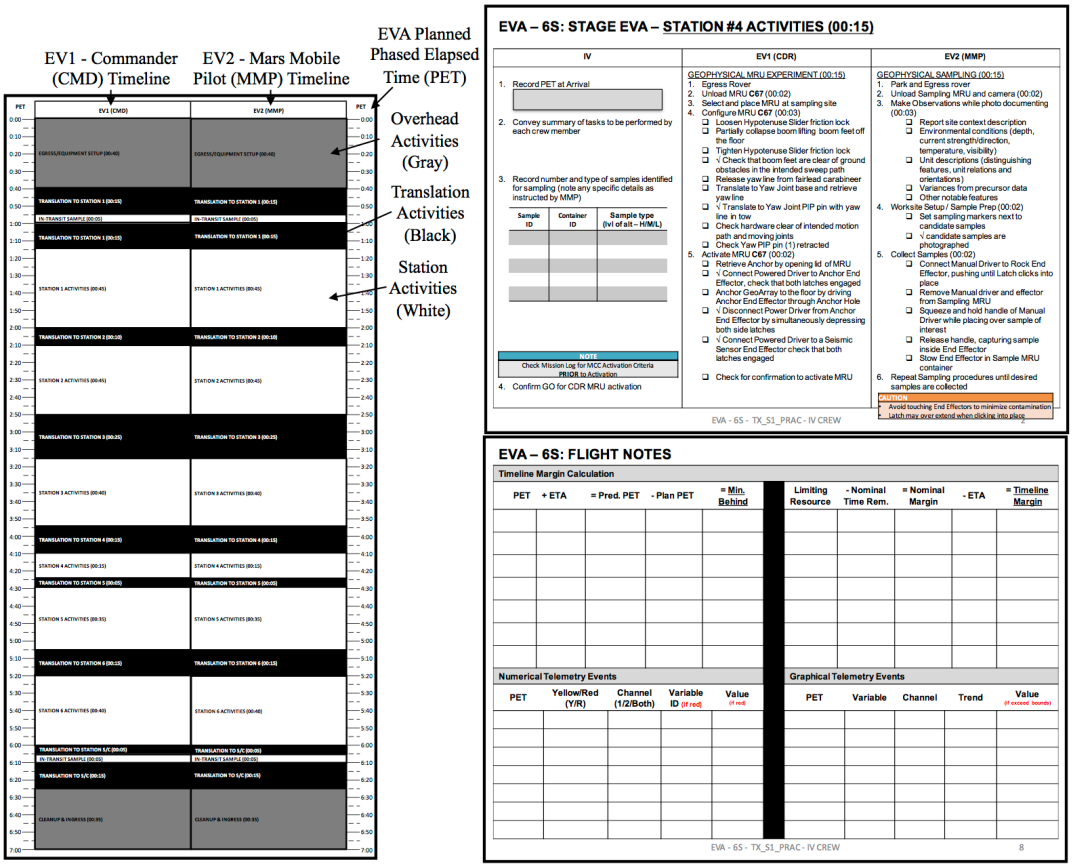

Figure B2. Baseline DSS prototype timeline artefacts, summary (LEFT), detailed procedures (TOP RIGHT) and flight notes (BOTTOM RIGHT).

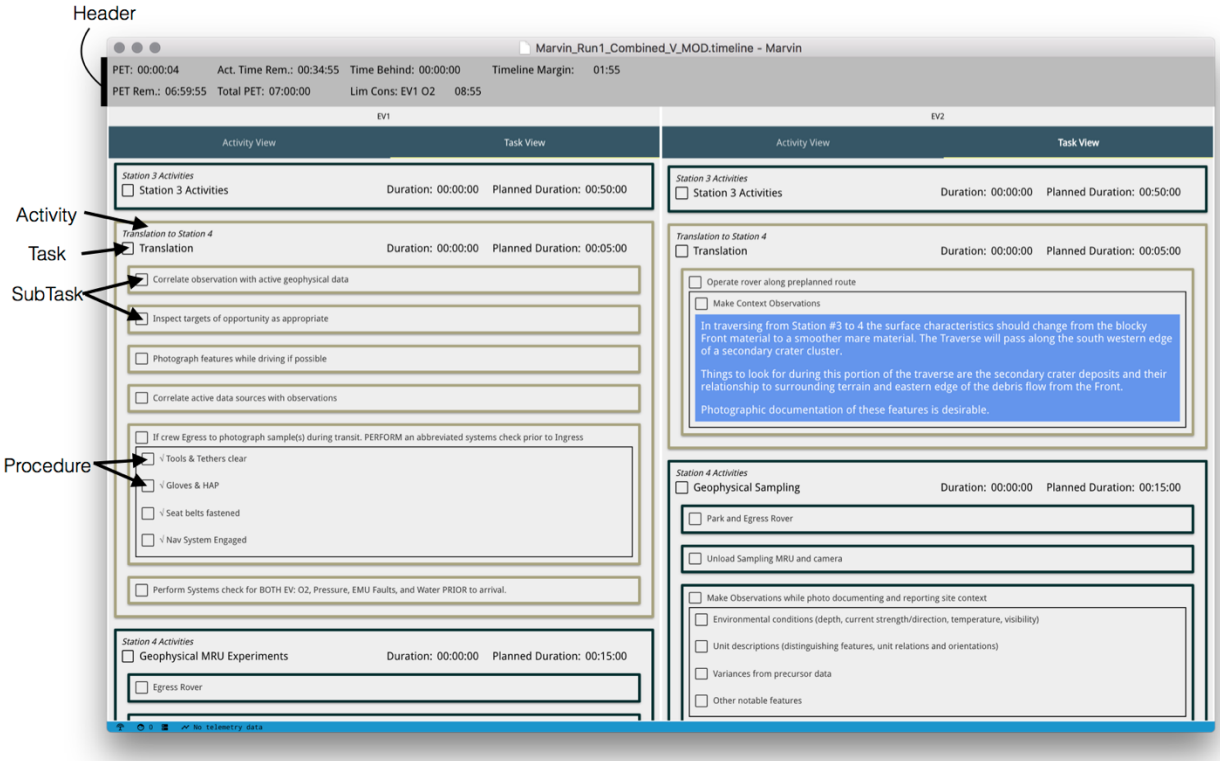

Figure B3. Advanced DSS Prototype display of both timeline and life support system data.

References

[1] B. G. Drake, Ed., "Human Exploration of Mars Design Reference Architecture 5.0," NASA Headquarters, NASA/SP–2009–566, Jul. 2009.

[2] B. G. Drake, Ed., "Human Exploration of Mars Design Reference Architecture 5.0 - Addendum," Mars Architecture Steering Group - NASA Headquarters, NASA/SP–2009–566-ADD, Jul. 2009.

[3] K. H. Beaton, S. P. Chappell, A. F. J. Abercromby, M. J. Miller, S. K. Nawotniak, S. S. Hughes, A. Brady, and D. S. S. Lim, "Extravehicular activity operations concepts under communication latency and bandwidth constraints," presented at the 2017 IEEE Aerospace Conference, Big Sky, MT, 2017, pp. 1–20.

[4] S. P. Chappell, K. Beaton, T. Graff, C. Newton, A. F. J. Abercromby, and M. L. Gernhardt, "Integration of an Earth-Based Science Team during Human Exploration of Mars," presented at the IEEE Aerospace Conference, Big Sky, MT, 2017.

[5] J. J. Marquez, M. Feary, J. Rochlis Zumbado, and D. Billman, "Evidence Report: Risk of Inadequate Design of Human and Automation/Robotic Integration," Jun. 2013.

[6] C. Hudy and B. Woolford, "Space Human Factors Engineering Gap Analysis Project Final Report," NASA/TP–2007–213739, Mar. 2007.

[7] M. Shafto, M. Conroy, R. Doyle, E. Glaessgen, C. Kemp, J. LeMoigne, and L. Wang, Eds., *NASA Space Technology Roadmaps and Priorities*. NASA's Office of the Chief Technologist, 2012.

[8] K. Holden, N. Ezer, and G. Vos, "Evidence Report: Risk of Inadequate Human-Computer Interaction," 2013.

[9] M. J. Miller, K. M. McGuire, and K. M. Feigh, "Information Flow Model of Human Extravehicular Activity," In Proceedings of the IEEE Aerospace Conference, Big Sky, MT, 2015.

[10] M. J. Miller, K. M. McGuire, and K. M. Feigh, "Decision Support System Requirements Definition for Human Extravehicular Activity Based on Cognitive Work Analysis," *Journal of Cognitive Engineering and Decision Making*, vol. 11, no. 2, pp. 136–165, 2017.

[11] B. S. Caldwell, "Information and Communication Technology Needs for Distributed Communication and Coordination During Expedition-Class Spaceflight," *Aviation, Space, and Environmental Medicine*, vol. 71, no. 9, pp. A6–10, Sep. 2000.

[12] C. Guyse, "Extravehicular Activity Flight Control Discipline Moves to a Distributed Workstation Environment," presented at the 9th Computing in Aerospace Conference, Reston, Virigina, 1993.

[13] E. Bell, D. Coan, and D. Oswald, "A Discussion on the Making of an EVA: What it Really Takes to Walk in Space," presented at the SpaceOps 2006 Conference, Reston, Virigina, 2006.

[14] M. J. Miller, A. Claybrook, S. Greenlund, J. J. Marquez, and K. M. Feigh, "Operational Assessment of Apollo Lunar Surface Extravehicular Activity," NASA/TP–2017–219457, Jul. 2017.

[15] M. J. Miller, A. Claybrook, S. Greenlund, and K. M. Feigh, "Operational Assessment of Apollo Lunar Surface Extravehicular Activity Timeline Execution," presented at the AIAA SPACE 2016, 2016.

[16] M. J. Miller, K. M. McGuire, and K. M. Feigh, "Preliminary Work Domain Analysis for Human Extravehicular Activity," presented at the Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 2015, vol. 59, pp. 11–15.

[17] A. M. Luta, "EVA Console Handbook Volume 1 - Standard Console Procedures," EVA, Robotics, and Crew Operations Division, JSC-20597, Feb. 2011.

[18] A. M. Luta, "EVA Console Handbook Volume 3 - ISS EVA Operations," EVA, Robotics, and Crew Systems Operations Division, JSC-20597, Jan. 2010.

[19] A. M. Luta, "EVA Console Handbook Volume 4 - EMU Operations," EVA, Robotics, and Crew Systems Operations Division, JSC-20597, Feb. 2011.

[20] J. J. Marquez, G. Pyrzak, S. Hashemi, S. Ahmed, K. McMillin, J. Medwid, D. Chen, and E. Hurtle, "Supporting Real-Time Operations and Execution through Timeline and Scheduling Aids," presented at the 43rd International Conference on Environmental Systems, 2013, pp. 1–11.

[21] E. Hutchins, *Cognition in the Wild*. Cambridge, MA: MIT Press, 1995.

[22] M. J. Miller, D. A. Coan, A. F. J. Abercromby, and K. M. Feigh, "Design and Development of Support Systems for Future Human Extravehicular Activity," presented at the AIAA SciTech, 2017.