Billman, D., Schreckenghost, D. & Miri, P. (2014) Assessment of Alternative Interfaces for Manual Commanding of Spacecraft Systems: Compatibility with Flexible Allocation Policies. In Proceedings of the Human Factors and Ergonomics Society Annual Meeting (Vol. 58, No. 1, pp. 365-369). SAGE Publications.

Assessment of Alternative Interfaces for Manual Commanding of Spacecraft Systems: Compatibility with Flexible Allocation Policies

Dorrit Billman, San Jose State University @ NASA Ames Research Center
Debra Schreckenghost, TRACLabs and
Pardis Miri, University of California, Santa Cruz

Astronauts will be responsible for executing a much larger body of procedures as human exploration moves further from Earth and Mission Control. Efficient, reliable methods for executing these procedures, including manual, automated, and mixed execution will be important. We evaluated a new procedure system that integrates step-by-step instruction with the means for execution. While the system allows automation, the critical first step, investigated here, is effectiveness supporting manual execution. We compared manual execution using the new system to a system analogous to the manual-only system currently in use on the International Space Station; we assessed whether manual performance with the new system would be as good or better than with the legacy system. This lays the foundation for integrating automated execution into the flow of procedures designed for humans. In our formative study, we found speed and accuracy of procedure execution was better using the new, integrated interface over the legacy design.

INTRODUCTION

We report an initial study of methods for controlling vehicle systems like life support on the International Space Station (ISS). Much of this work is proceduralized, and carried out by following the sequence of actions specified in a procedure, such as verifying sensor values and sending commands to equipment. Currently, almost all component actions are done through computerized commands not visual inspection or manually twisting a valve, but humans initiate each action in the procedure. While low-level components may operate automatically once initiated, execution of a procedure is primarily done by a person reading, assessing, and initiating each action in the procedure. Many procedures are executed by Mission Control, but much of crewmembers' work on the ISS also relies on procedure execution, e.g., science work. For the ISS, procedure guidance for astronauts relies PDF or styled XML files of written instructions. This simple form of electronic vs paper procedure seems beneficial here, as noted elsewhere (aviation Boorman, 2000; though with potential complications Mosier, Palmer, & Degani, 1992).

Substantially increasing efficiency of proceduralized work and independence from Mission Control will be critical for long-distance manned space flight (Ambrose, Wilcox, Reed, Matthies, Lavery, & Korsmeyer, 2010). Different mixtures of ground- versus crew-based control, and manual versus automatic execution will be needed. Communication lags imply less reliance on ground-based support than for current, near-earth operations. Missions will have smaller crews, operating more complex, less-tested systems, with longer retention intervals since training. Thus methods that allow crewmembers to carry out more work, more efficiently will be important. A spectrum of methods from efficient manual execution, through mixed-initiative, and automated execution will be needed.

Prior research has approached the topic of human and automated procedure execution from two starting points. Some researchers began with a situation relying on paper documents (as for checklists or procedures) and have asked whether and how these might be made dynamic or how more support might be provided through automation (Boorman, 2000; Carvalho, dos Santos, Gomes, Borges, & Guerlain, 2008; Hutchins, 1996; Mosier et al., 1992). Others have started with a focus on procedure automation, and recognized that supporting human involvement and oversight is important for overall effectiveness of the human-machine system (Dalal & Frank, 2010; Kortencamp et al., 2008; Morelli, Bouleau, Chinchilla, & Noguero, 2010). Our broader research agenda develops systems for mixed manual and automated execution, organized around procedures developed for humans (Schreckenghost, et al., 2008). The system used in the experiment reported here allows for direct manipulation of the procedure interface to perform procedures manually and for flexible allocation of procedure actions to manual or automated execution. Such support for flexible allocation is a key driver of the design; here we report on manual execution.

As methods for flexible allocation to manual and automated execution are developed, we need to ensure that manual operations are well supported and mission performance does not deteriorate relative to present methods. This ensures that, whatever progress is made in automating procedures, a procedure can be executed manually in the new system at least as well as in current, all-manual operation. In turn, this ensures that execution systems do not increase automation yet decrease overall human-system performance. Therefore, our initial evaluation compared manual procedure execution using a legacy versus novel interaction design. Each of the two interfaces commanded the same simulation of an ISS life support device for removing carbon dioxide (CO_2).

The Legacy Interface mirrored current ISS manual commanding while the Integrated Interface was designed to support mixed-execution as well as manual commanding. Our study focused on comparing manual execution but exploration of mixed-allocation operations is reported in (Schreckenghost, Milam, & Billman, 2014). We draw on prior evaluation methods for interaction design for space systems (Billman et al., 2011).

Supporting both speed and accuracy are important requirements for operations software. Errors when commanding spacecraft systems can lead to unsafe states and accidents. Software and interface design that slows operator control produces inefficient operations, reducing the ability to accomplish other critical tasks or to accomplish primary mission objectives such as science experiments. While efficient execution of a specific step may not be critical, slowed procedure execution can produce cumulative effects that impact mission safety and success.

Procedures for ISS astronauts are information structures (currently used as PDF or styled XML documents) typically designed by engineers or scientists that spell out how to do tasks such as operating ISS life-support systems or conducting science experiments. Our study concerns operating the system for removing CO2 from cabin air; procedure steps are built from actions such as verifying sensor values (e.g., blower is on) and issuing system commands ((e.g., open the vent valve). The highly specific nature of procedures allows us to compare carrying out the identical procedure (e.g., the same sequence of verifying and commanding) through alternative interfaces.

STUDY METHOD

Design

Interface Condition (Legacy versus Integrated) was a within-subjects factor, with order counterbalanced between subjects. There were 10 execution Trials using four procedures, in each condition. Several automated execution trials followed. Here we report data from the first 3 trials executing a simple form of a start-up procedure and the 4th trial executing a complex form of the start-up procedure with 4 more steps added. (A few exploratory trials using the automation features were included as time permitted and are not part of this design.) Completion time was the primary dependent variable and errors were also measured.

Participants

All 11 users were graduate students, ten in aeronautics engineering departments; two were also professional pilots. Astronauts frequently have these backgrounds.

Materials

The Simulated System Controlled by Procedures. Both Legacy and Integrated interaction designs allowed execution of the identical procedures, which operated a simulation of the carbon dioxide removal system (the CDRS). Prior research developed the Procedure Representation Language (PRL) (Kortencamp et al., 2008) which includes the machine-executable commands and the sensor data needed to carry out the text instructions for operation. TRACLabs' PRIDE procedure system uses PRL to generate an interface that integrates text instructions with controls and displays for their execution, our Integrated Interface.

The Two Interfaces. The *Legacy Interface* emulated the current method used on the ISS. The *Legacy* interface (Fig. 1) separates procedure specification from procedure execution. Procedure specification is provided in a PDF document (for each procedure) that describes what to do for every procedure sub-step. This specification provides a navigation path for each step though linked windows to a target window. Once at the target window, the user can inspect and verify a value or press a button to issue a command, for each instruction in the procedure. The same display and command windows are used for multiple procedures, in multiple sequences. Executing a particular procedure typically requires navigation through multiple display/control windows, as well as shifting focus between each instruction and the windows needed to carry out that instruction. This requires multiple changes in focus, produces multiple open windows as a residual, and adds window management to the process of procedure execution.

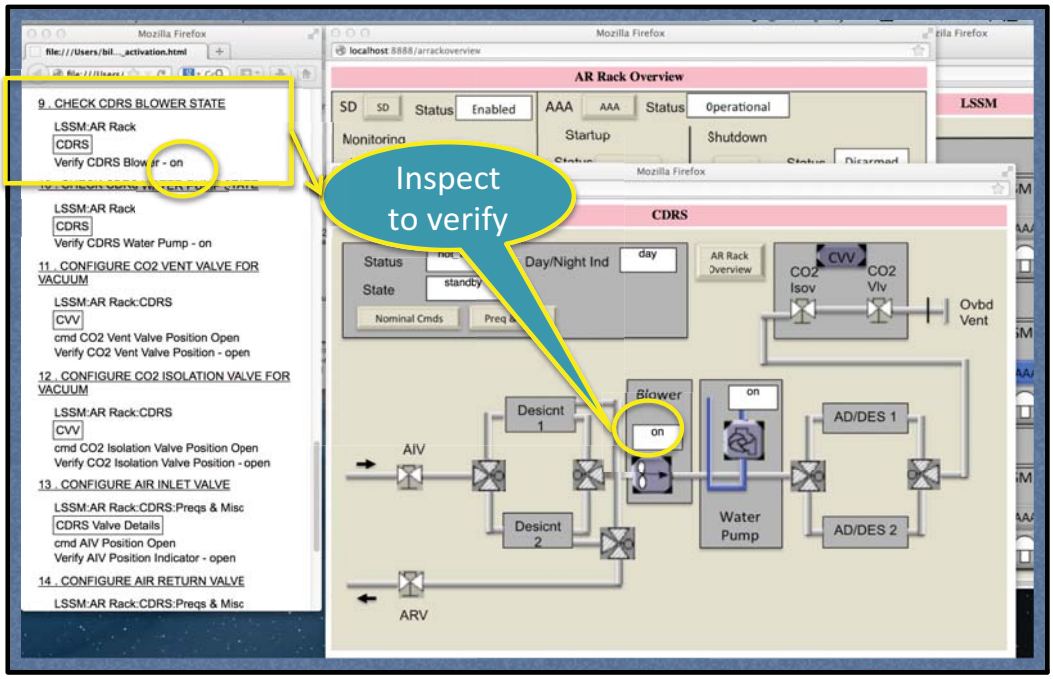

Figure 1. The Legacy Interface closely emulating software currently in use on the ISS for controlling the CDRS. Top window in this layout is the most graphical of all with windows used. The display shows the navigation path for Steps 9 and 10, with the top "CDRS" window open for verifying that the Blower (Step 9) and the Water Pump (Step 10) are turned on.

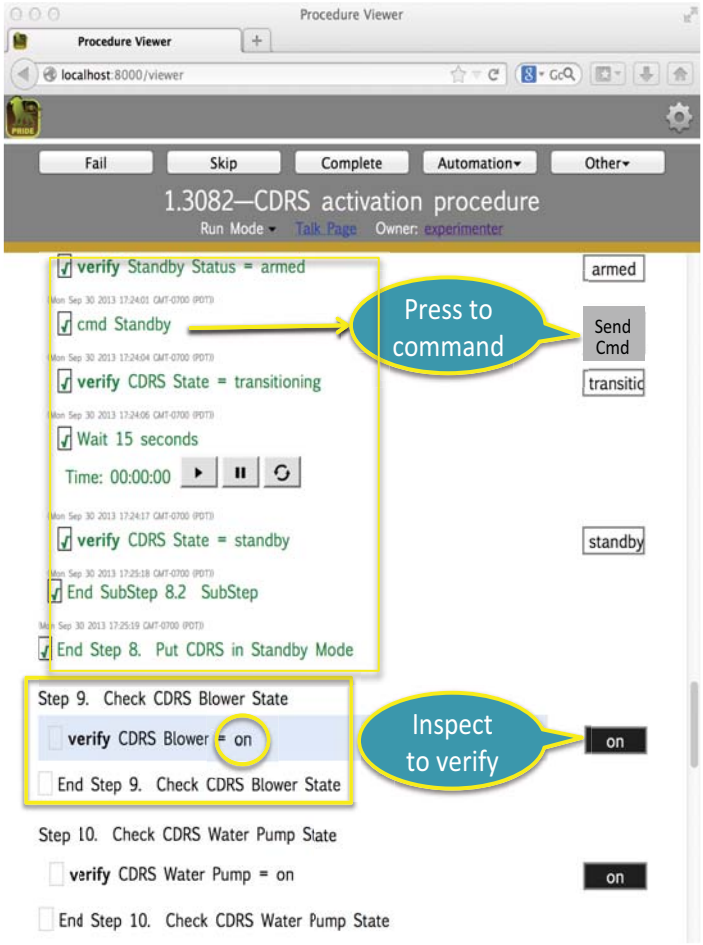

Figure 2. The Integrated interface provides the procedure and the means for verifying and commanding within one window.

The *Integrated Interface* (Fig. 2) differs from *Legacy* in several ways. Most critically, it places the procedure instructions in the same window as the controls and displays needed to carry out the instruction; this eliminates the need for navigation and change in focus. Specifically, it provides the single display element (e.g., a command button or value to verify) needed for each procedure sub-step on the same line as the instruction. Instruction types are primarily commands or verifications but also include timed waits, data recording, and decisions about conditional tasks. In addition, the interface provides feedback to the user when an action has been taken. After pressing a button to command a change or to verify a value, the color of the button changes, the instruction text turns green, and the instruction line is checked. Finally, the interface provides some meta-controls akin to "reset", which were occasionally used.

CDRS Procedures Used and Training Materials. A procedure has a high-level goal and is made up of steps. Steps are made up of actions. The results from the first four trials, using two of the four procedures, a Simple (Sim) and a Complex (Cpx) CDRS Activation Procedure, are reported here. Table 1 shows step titles. The Simple Activation Procedure had 15 steps and the Complex Activation Procedure had 19, with one to 13 actions per step (not including the window navigation for the Legacy interface). For example, Step 9 consists of one verification action, for status of the water pump, shown in Figure 1 & 2. Step 7 has two steps, commanding and verifying the water pump is turned on, shown in Figure 1.

Table 1- Steps in the Procedure		
Activation steps: 2 Versions		
Steps in CDRS Activation	**Sim**	**Cpx**
Verify power to CO2 & Air	1	1
Check CO2 Valve Prerequisite	2	2
Check CDRS Air Valve Prereq	3	3
Verify Day/Night configuration	4	4
Verify Power Available	5	5
Enable CDRS Blower	6	6
Enable CDRS Water Pump	7	7
Ensure CDRS is inactive	NA	8
Put CDRS in Standby Mode	8	9
Check CDRS Blower State	9	10
Check CDRS Water Pump State	10	11
Record CO2 level in the cabin	NA	12
Configure CO2 Vent Valve	11	13
Configure CO2 Isolation Valve	12	14
Configure Air Inlet Valve	13	15
Configure Air Return Valve	14	16
Configure CDRS Single Bed Op	NA	17
Configure CDRS Dual Bed Op	15	18

Training materials had an introduction to the CDRS and its operation; training on the Legacy Interface; and training on the Integrated Interface. Each component included verbal comprehension and inference questions.

Experiment-Running Procedure

In Phase I, users trained on the CDRS, reading training material as a slide presentation, and then answering questions with feedback. In Phase 2 they trained on their first system (Legacy or Integrated) by reading instructions and answering questions about how to accomplish different goals. Then they executed a series of 10 trials, running 4 procedures; the first 4 trials are reported here. In Phase 3 they trained on the second system and then used that system to do the identical sequence of trials as in Phase 2. In Phase 4 they used the Integrated Interface to execute parts of these procedures automatically. In Phase 5 they took part in a structured interview and filled out a questionnaire about the systems. This took about four hours.

RESULTS

For orientation, in the Legacy Interface correct execution of the Simple Activation procedure required using 7 different windows for displays and controls, and 10 focus changes among these windows; the Complex procedure required the same 7 display and control windows plus twice recording values from another window in a spread sheet, and required at least 13 focus changes among windows. While participants could learn efficient ways of leaving windows open and navigating among them, the focus changes are an intrinsic part of the Legacy design.

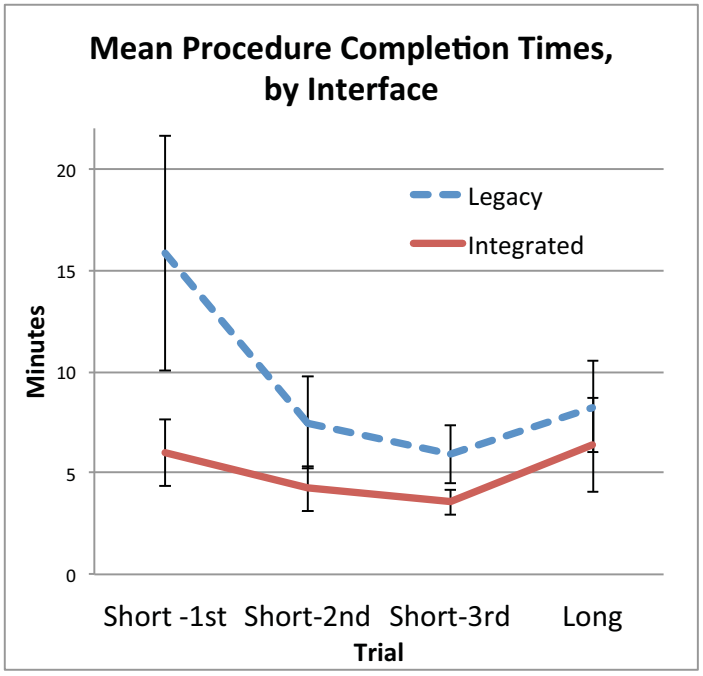

Figure 3. Mean completion times (+- StDev) for four trials of the short (simple) or long (complex) activation procedure, using the Legacy or Integrated Interface.

Completing the four procedures was almost twice as fast in the Integrated versus Legacy Condition at 5.0 (SD=1.3) versus 9.4 (SD=2.4) minutes. The Integrated Condition was faster for every user. The difference was greatest for the first trial, but persisted. See Figure 3. A mixed model ANOVA found effects of Interface, $F(1,9)=55.9$, $p<.01$, Trial $F(3,27)=32.5$, $p<.01$, Interface X Trial $F(3,27)=19.0$, $p<.01$, and the interaction of Order and Software $F(1,9)=13.3$, $p<.01$; the interaction reflects the greater difficulty in using the interface that is presented first. Concerning learning over trials, performance on the first trial with the Legacy Interface was particularly difficult, as users sought both to find needed elements within a window and to manage window layout. Speed-up in the Integrated Condition may be due to reduced time reading and understanding the procedure.

Each trial was scored for errors, as deviation from the correct sequence of device commands. Thus, this scoring criterion ignored deviations from the written procedure, such as verification steps, that did not alter the command sequence. It did, however, score a deviation from the command sequence as an error, whether or not the outcome of the deviation was an unintended state in the device; for example, repeating a command often did not adversely affect the device but was still a commanding error. Critically, this scoring rule can be applied in both conditions. First, commands are logged in both conditions, though no active response to a 'verify' step is required in the Legacy Condition. Second, the correct command sequence is identical for both procedures and both conditions, as the added steps in the Complex Procedure did not include commands. command sequence is identical for both procedures and both conditions, as the added steps in the Complex Procedure did not include commands.

No commanding errors occurred in the four trials of the Integrated Condition producing 11 of 11 "successful" users. In

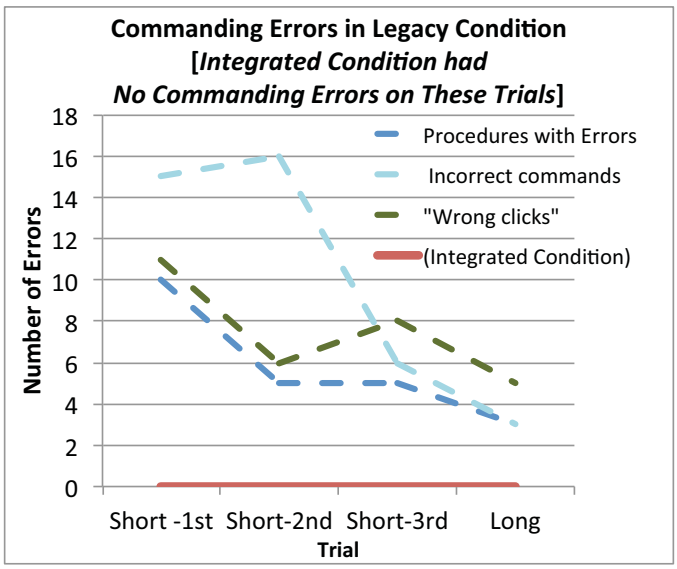

Figure 4. Numbers of commanding errors for the four trials using the Legacy or Integrated Interface.

contrast only 1 of 11 users was completely correct on the command sequence for all four trials in the Legacy Condition. These proportions of success differ significantly, $p<.0001$, Fisher Exact Test.

The Legacy Condition had 40 total errors, or 5.7% using a denominator of 704 (11 users*16 commands*4 trials). Errors in the Legacy Condition dropped over trials, but persisted into Trial 4. Figure 4 shows the number of procedures that had commanding errors, the number of command errors, and the number of "wrong clicks". "Wrong clicks" are recorded in the Legacy Condition if the user clicks on an window region not intended for use; while these are errors we cannot tell whether the user was attempting to command or to navigate windows.

Deviations from the correct sequence were scored by type. There were 11 errors where a command was missed; 12 where an extraneous command was inserted, 4 where order of an adjacent pair was switched, 8 where a command was repeated and a group of 5 "omitted" commands. "Omitted" commands could not be executed because prior user error(s) prevented execution of the omitted command. Informally, some errors are more serious than others. Missing a command can produce a later situation where the user could not complete execution; however, repeating a command, perhaps due to uncertainty whether the button had been successfully pressed, would not produce bad outcomes, in the context of the device used.

All users rated the Integrated system higher for aiding efficiency, 8 of 11 higher for accuracy, and 9 of 11 users preferred it. Providing a more visual representation was mentioned as a positive aspect of the Legacy interface.

CONCLUSIONS

Command errors and speed of execution provide converging evidence of the benefit from the Integrated Interface relative to the Legacy Interface design, for both accuracy and efficiency. Though we had a small number of users, condition effects for time and number of successful

users were large enough to be significant. Mean completion time was reduced by approximately half, and no command errors were observed for the Integrated Condition on these trials. This suggests the new approach works better than current designs for manual operation, while allowing integration with flexible automation. Analyses of the additional trials are in progress and the pattern of faster time and reduced errors for the Integrated Condition persists.

For manual execution, the integration of instructions with the means to execute them is the most fundamental design change in the Integrated interface, with provision of clear and specific feedback also fundamental. We believe these are key contributors to improved performance. Our future work will explore interface options within the integrated interface concept, and will investigate tasks that require flexible allocation between mixtures of manual and automatic execution.

Due to the challenging demands and increased workload of long-distance manned space exploration, increased automation will almost certainly play a key role. *Flexible allocation* of work allows an operator (i.e., the astronaut) to dynamically assign units of work to automation or to execute them manually, in light of current circumstances. We expect that a productive design approach for integrating manual and automatic operations is to align the units of automation with the units of work in manual operations, for example, procedures designed for humans (Schreckenghost et al., 2008). When the units of work are meaningful, shifts in control between manual and automatic execution are likely more understandable for the operator, and in turn the operator has a more informed basis for allocating work. Initially a procedure might be executed manually, as the operator assesses whether it produces the expected effects. Based on such assessment, more work units may be allocated to automatic execution. Conversely, an operator may shift to manual execution if the equipment is replaced or has degraded functions, to monitor for departures from expected behavior. Flexible allocation requires methods that support effective manual operation as well as allowing for automation.

In addition to investigating flexible allocation for procedure execution, an important research topic will be exploring a broader set of conditions, such as off nominal operations and situations that do not exactly match those of an existing procedure. For example, users might integrate steps from multiple procedures or follow a procedure but adapt a step to change the value of a variable to be verified or which command should be executed. Means for coordinating a procedure step with broader information about system state may prove valuable. With respect to execution of procedures through a mix of automated and manual methods, we are exploring how users carry out mixed sequences and how software supports users doing this.

Designing for flexible work allocation in procedures can provide effective manual system operation, and our design approach is promising. This formative study provides a rich set of performance data, which will guide development of future assessment methods and design.

Acknowledgements

Thanks to Tod Milam, Scott Bell, Mary Beth Hudson, and Kevin Kusy at TRACLabs for developing software for this evaluation. This work is supported by the National Space Biomedical Research Institute through NASA NCC 9-58.

References.

Ambrose, R., Wilcox,B., Reed,B., Matthies,L., Lavery, D. & Korsmeyer, D. Robotics. (2010). Tele-Robotics and Autonomous Systems Roadmap, Technology Area 04. http://www.nasa.gov/pdf/501622main_TA04-ID_rev6b_NRC_wTASR.pdf

Billman, D., Arsintescu, L., Feary, M., Lee, J. . C., Smith, A., & Tiwary, R. (2011). Benefits of Matching Domain Structure for Planning Software: The Right Stuff. In *Proceedings of the 29th Conference of the SIGCHI conference on Human factors in computing systems (CHI)*. Vancouver.

Boorman, D. J. (2000). Reducing Flight Crew Errors and Minimizing New Error Modes with Electronic Checklists. In *Proceedings of the International Conference on Human-Computer Interaction in Aeronautics, Cépaudès Editions*. Toulouse, France.

Carvalho, P. V. R., dos Santos, I. L., Gomes, J. O., Borges, M. R. S., & Guerlain, S. (2008). Human factors approach for evaluation and redesign of human–system interfaces of a nuclear power plant simulator. *Displays*, *29*, 273–284.

Dalal, K., & Frank, J. (2010). Bridging the Gap between Human and Automated Procedure Execution. Presented at the Aerospace Conference, 2010 IEEE, IEEE.

Hutchins, E. (1996). *The Integrated Mode Management Interface* (Final Report for grant NCC 2-591). Ames Research Center.

Kortencamp, D., Bonasso, R. P., Schreckengost, D., Dalal, K., Verma, V., & Wang, L. (2008). A Procedure Representation Language for Human Spaceflight Operations. Presented at the *9th International Symposium on Artificial Intelligence, Robotics & Automation in Space (i-SAIRAS-08)*.

Morelli, G., Bouleau, F., Chinchilla, R., & Noguero, J. (2010). Unleashing the full power of today's technologies for flight procedures automation.Morelli, G., Bouleau, F., Chinchilla, R., & Noguero, J. (2010). Presented at the : *Delivering on the Dream*, Huntsville, Alabama: American Institute of Aeronautics and Astronautics.

Mosier, K. L., Palmer, E., & Degani, A. (1992). Electronic Checklists: Implications for Decision Making. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*.

Schreckenghost, D., Milam, T., & Billman, D. (2014). Human performance with procedure automation to manage spacecraft systems. *Proceedings of IEEE Aerospace*, Big Sky, MT.

Schreckenghost, D., Bonasso, R. P., Kortencamp, D., Bell, S., Milam, C., & Thronesbery, C. (2008). Adjustable Autonomy with NASA Procedures. Presented at the *The 9th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS),* Pasadena,CA.