Adjustable Autonomy with NASA Procedures

Debra Schreckenghost, R. Peter Bonasso, David Kortenkamp
TRACLabs, 1012 Hercules, Houston, TX, USA 77058
ghost@ieee.org, r.p.bonasso@nasa.gov, korten@traclabs.com

Scott Bell, Tod Milam, Carroll Thronesbery
S&K, 201 Flint Ridge Plaza, Suite 102, Webster, TX 77598
scott@traclabs.com, tmilam@traclabs.com, carroll.g.thronesbery@nasa.gov

Abstract

NASA's Exploration missions will require more effective use of human resources in space and on Earth. The use of automation to perform routine or hazardous tasks can free humans for other tasks. But increased use of automation must be done without compromising human safety or introducing unnecessary risk. At JSC we have developed technology for incrementally increasing the level of system autonomy. Our approach is based on the electronic procedures used for crewed operations. We provide assistive software for humans to adjust whether a procedure step is executed manually or automatically. This permits a gradual shift to more automated operations. We have evaluated our adjustable autonomy technology in two NASA domains – procedures for the International Space Station and procedures for remote supervision of robots. In this paper we describe our approach to adjustable autonomy, compare it to other approaches, and summarize the results of using it in NASA applications.

1. Introduction

NASA's Exploration program poses challenging new missions, such as a permanent lunar outpost and manned exploration of Mars. Fulfilling such missions will require more effective use of human resources in space and on Earth. The use of automation to perform routine or hazardous tasks typically performed by humans has potential to make more effective use of humans in space operations. But increased use of automation must be done without compromising human safety or introducing unnecessary risk.

At JSC we have developed technology for increasing the level of system autonomy as a person gains experience with and trust in automated operations. Our approach is to provide support software for humans to adjust whether a step in a procedure is executed manually or automatically. This differs from other approaches to adjustable autonomy in that the designation of whether a step is manual or automated is under human control. This adjustment is constrained by what can be automated (based on available instrumentation) and what should be automated (based on flight rules and operational protocols). We support three levels of autonomy (LOA): manual execution by a person, automatic execution with human consent, and automatic execution when preconditions are met. These levels can be adjusted at any time prior to the execution of the procedure. These adjustments do not require modification and consequent recertification of the procedure because the procedure includes the knowledge needed for both manual and automated execution. For manual execution, knowledge is captured to support tracking what instruction a person is performing, prompting a person to take action, and annotating the presentation of instructions customized to both the user and the situation. For automated execution, knowledge is captured about the necessary pre-conditions for a step, the commands to be dispatched, and the expected effects of these commands. We use the Procedure Representation Language (PRL) to represent this knowledge [11].

We have applied our adjustable autonomy technology in two NASA domains – procedures for the International Space Station (ISS) and procedures for remote supervision of the Centaur robot [17]. We have evaluated our technology on procedures for the ISS power distribution system with a high fidelity simulation. We also have evaluated our technology on procedures for the supervisor of a Centaur robot operating from the JSC Cockpit with a 5-10 second time delay. In this paper we describe our approach to

adjustable autonomy, compare it to other approaches to adjustable autonomy, and summarize the results of using it in NASA applications.

2. Approach

For manned space flight, procedures define the instructions for how to manage spacecraft systems. This includes nominal configuration and control of systems, as well as diagnosis and fault mitigation. Procedures for the International Space Station (ISS) encode knowledge in the procedure using the eXtensible Markup Language (XML). It is expected that the Constellation spacecraft will use a similar approach. For the ISS, this knowledge defines what information should be presented to operations personnel, and how it should be presented to comply with procedure standards. All actions in these procedures are manually executed using system displays separate from the procedure display.

We have defined an approach for using procedural knowledge to provide adjustable autonomy for spacecraft systems. To ensure that we capture the knowledge needed to automate actions, we have extended the XML schema for ISS procedures to support capturing this knowledge when procedures are authored. Once this knowledge is represented in XML, a Level of Autonomy (LOA) is assigned to the actions in a procedure. The LOA setting designates whether a person should take the action or the action should be performed automatically. Adjustable autonomy is supported by providing a means to change these LOA settings during operations.

In the remainder of this section we summarize the changes to PRL to capture automation information and describe how we combine this information with LOA settings to support adjustable autonomy.

2.1. Procedure Representation Language

At JSC, we are investigating how to extend the XML used to represent current procedures to include knowledge needed to automate the execution of procedure steps and instructions [12]. The revised XML is called the Procedure Representation Language (PRL). A PRL procedure consists of one or more steps. Each step includes one or more instructions, grouped into blocks. Steps can be linked serially (e.g., go to step 2 after step 1) or conditionally (e.g., if switch is on go to step 2 else go to step 3). Instructions within a block can be sequenced using a variety of control flow constructs including (1) if-then, (2) for-each, and (3) repeat-while. Instructions define operational actions including: (1) command a spacecraft system, (2) verify a telemetry value, (3) execute another procedure, and (4) get information from a person. See Kortenkamp [11] for a description of PRL.

The following extensions in PRL are needed to support automatic execution of a step or instruction:

- *Linking procedure actions to spacecraft telemetry and commands*

Telemetry and Commands are referenced in procedures today by specifying (1) a navigation path to a system display page that shows the item, and (2) a text string describing the action to be taken on that screen. Automatic execution of these actions requires linking the action description to a mechanism for retrieving the required telemetry or dispatching the required command. For ISS, this means identifying the telemetry or command identifier (called a PUI, Program Unique Identifier) and specifying any command arguments needed. These arguments can be assigned by a person or can be derived from telemetry. We have extended PRL instructions to include a DataReference tag that identifies the command or telemetry needed to perform an action in an instruction.

- *Identifying the conditions that must hold to move between steps*

Procedures are constructed as sequences of steps, where each step corresponds to one or more actions. The order in which steps are sequenced in the procedure can result (1) from state dependencies between steps (e.g., close the isolation valve before opening the flow valve) or (2) from the need to make these sequences easier for humans to learn and perform. The knowledge necessary to distinguish between these two cases is needed when automating the procedure steps but is not captured in procedures today. Instead, such knowledge is passed on to the operations personnel during training. PRL has added the AutomationData tag that defines pre-conditions and post-conditions on actions. When the appropriate action cannot be determined at authoring time, the BranchingCondition tag defines how to select the appropriate action when the procedure is executed.

Once the procedure encodes this additional knowledge about command and data referencing (via the DataReference tag) and conditions on actions (via the AutomationData and BranchingCondition tags), it is possible to derive software for taking these actions automatically from the procedure.

We have used PRL to support our approach of adjustable autonomy through procedures. When authoring the procedure, we identify whether a step can be automated and, if so, whether a step should be automated. Physical constraints on the automation of procedures include (1) inadequate instrumentation to automate a command or detect the effects of a command, and (2) limited response times not possible

with the procedure infrastructure (e.g., millisecond response times are best implemented in flight software). Mission constraints (e.g., flight rules) on the automation of procedures include (1) actions always requiring human approval before execution, and (2) actions best suited to human decision making.

These constraints can be added to the PRL for each instruction using the executionMode attribute. Possible values for the executionMode include (1) Manual – the action should only be performed by a person, (2) Automated – the action should only be performed automatically by software, and (3) Mixed – the action can be performed either by a person or by software.

Manual actions are accomplished via PRL InputInstructions, which provide the user with a statement of what needs to be done and a means for confirming when it has been done. Automated actions are accomplished via a variety of PRL instruction types such as a CommandInstruction that sends a command to the spacecraft or a VerifyInstruction that monitors for a telemetry value. The instruction type used depends upon the action to be taken. Kortenkamp [11] describes the actions available in PRL. When an action is designated as Mixed, it is necessary to provide two methods for accomplishing it – one manual and one automated.

2.2. Adjustable Autonomy

Our approach to adjustable autonomy is to provide assistive software that guides the performance of a procedure written in PRL based on the Level of Autonomy (LOA) settings defined for the procedure, its steps, and its instructions. As described in the previous section, constraints on the LOA settings are specified when the procedure is authored. The LOA settings can then be adjusted within these constraints when the procedure is executed. We have implemented three LOA settings in our initial approach:

- Manual: the action associated with the current instruction is performed by a person using systems outside the adjustable autonomy software,
- Automated: the action associated with the current instruction is performed automatically by the adjustable autonomy software, including telemetry monitoring and command dispatching, and
- Consent: the action associated with the current instruction is performed by the adjustable autonomy software after receiving human approval

The current *instruction* is identified by the adjustable autonomy software as the instruction in the current step that has met the instruction pre-conditions and start-conditions of the instruction but has not yet met the instruction end-conditions. Likewise the current *step* is identified as the step that has met the step pre-conditions and start-conditions of the step but has not yet met the step end-conditions.

The LOA can be assigned for a procedure, a step, an instruction, or a combination of settings at these different levels of the procedure hierarchy. Currently we do not support setting a LOA for the procedure block. If no LOA setting has been specified for a procedure element, the LOA setting at the level above it is used (e.g., if no LOA for an instruction, the LOA setting for the step containing it is used). If no LOA setting has been specified at any level above an element, the LOA setting is considered undefined.

The LOA settings are represented using XML, based on an XML schema. These settings are linked to procedures, steps, and instructions using the same identifiers defined in the PRL procedure. The following example defines a CONSENT LOA setting for step_10 in procedure 3209:

```
<procedure xmlns="http://www.traclabs.com/....."
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.traclabs.com/..... ./LOA.xsd" id="3209">
    <step id="step_10">
        <loa loa="CONSENT" />
    </step> ……
```

We have developed a prototype to evaluate our approach to adjustable autonomy. This prototype provides the following capabilities:

- Procedure Executive: steps through the procedure and takes action based on the LOA setting. Actions taken for automated instructions include monitoring for a value in telemetry and dispatching a command to a spacecraft system. Actions taken for manual instructions include asking the user for information needed to determine the next action to take (e.g., do you smell smoke) and asking the user to respond when a manual task is done.
- Procedure Support Software: tracks the state of procedure execution for display to the user and routes queries from the Procedure Executive to the designated users.
- Procedure Display: provides a user interface for monitoring the execution of automated instructions and for interacting with the Procedure Executive for manual or consent instructions.
- LOA Server: distributes LOA settings to the Procedure Executive and the Procedure Display, including updates to settings made from the display just prior to executing the procedure.

Figure 1 shows the adjustable autonomy architecture.

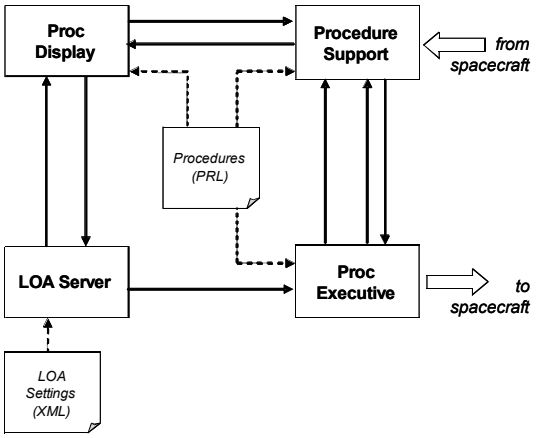

Figure 1. Adjustable Autonomy Architecture

These LOA values are served to the Procedure Executive and the Procedure Display by a LOA server. The LOA server is initialized from an XML file with default settings for procedures. As described previously, if a LOA setting is requested and nothing has been specified for it, the server attempts to derive a LOA setting by using the setting at the level above the requested element (e.g., step LOA for an unspecified instruction LOA or the procedure LOA for an unspecified step LOA). If it is not possible to derive a setting using this approach, the empty string is returned indicating an undefined LOA setting.

The LOA settings can be retrieved from the server by requesting all LOA settings for a procedure or by requesting a single setting for either a step or instruction within a procedure. The Procedure Display takes changes to a LOA setting made by the user and sends them as updates to the server. If the setting changes while the server is running, the change is passed to the Procedure Executive. The user then can save the updated settings to file if the changes should persist after the software is shutdown.

For Manual steps and instructions, the adjustable autonomy software cues the person executing the procedure when it is time to take action. It prompts the user to indicate when the actions are complete, and modifies the appearance of the Procedure Display to reflect the state of execution (e.g., completed). It also provides a means for the user to exit the procedure by cancelling a step or instruction.

For Automated steps and instructions, the adjustable autonomy software checks the conditions in the AutomationData, dispatches any commands within the step, and monitors any telemetry identified within the step. It informs the user when it begins to execute the step or instruction, and when it completes the step or instruction by changing the appearance of the step or instruction on the Procedure Display.

For steps and instructions requiring Consent, the adjustable autonomy software first requests the user for permission to proceed with automated execution of the step or instruction. If permission is granted, the software behaves as if the step or instruction were designated as Automated. If permission is denied, the software stops executing the procedure and exits.

The ability to adjust the LOA settings without having to re-author the procedure is central to our operations concept. We accomplish this by authoring the procedure to include sufficient information to execute the procedure instructions both manually and automatically, when permitted by LOA constraints (i.e., Mixed mode). We use the LOA settings during execution to select which of these methods to use.

There are a number of advantages to this approach. The procedure contains the full range of manual to automated capability early on, even if the operational use of the procedure remains primarily manual, as in crewed operations today. This supports the gradual adoption of automation by permitting portions of procedures to be incrementally automated as users gain knowledge about how they want to use automation and as additional instrumentation becomes available. It also permits adjusting normally automated steps to be manual in special circumstances where a more active user involvement is desired. When changing aspects of the procedure to accommodate operational changes, both the manual and automated methods can be changed and certified at one time. Finally, the knowledge captured to automate instructions can improve crew situation awareness (e.g., knowledge of preconditions for an instruction is informative for a person performing the instruction). Such knowledge is typically captured as part of mission training today.

3. Evaluation

We have applied our adjustable autonomy technology in two NASA domains – procedures for the International Space Station (ISS) and procedures for remote supervision of the Centaur robot. We have evaluated our technology on procedures for the ISS power distribution system with a high fidelity simulation. We also have evaluated our technology on procedures for the supervisor of a Centaur robot operating from the JSC Cockpit.

3.1. International Space Station Procedures

We have developed a prototype of our adjustable autonomy architecture for evaluation with ISS procedures. We represented ISS procedures for the power distribution portion of the Electrical Power

System (EPS) in PRL. We executed these procedures using our adjustable autonomy prototype integrated with a high fidelity simulation of Station (ISS in a Box) that models flight hardware and software. We used this prototype to evaluate the use of adjustable autonomy for responding when a power distribution switch trips unexpectedly. In this situation, any system downstream of the tripped switch loses power. The immediate response is to mitigate fault effects and diagnose the problem using malfunction procedures. The longer term response is to reconfigure or repair the system to fix the problem. We implemented both malfunction and checklist procedures for ISS EPS in PRL.

We used this prototype to evaluate the following hypotheses about our approach to adjustable autonomy:

- Hypothesis 1: Procedure steps and instructions can be encoded in PRL with the information necessary to be executed either manually or automatically
- Hypothesis 2: The LOA settings for these PRL procedures can be adjusted just prior to execution without modifying the PRL for the procedure.

For this evaluation, we used the Reactive Action Package System (RAPS) [6] as the Procedure Executive. Because RAPS does not interpret PRL directly, we developed a translator that uses the semantics of PRL to produce RAP code from a PRL file. We also have translated these ISS procedures into the PLEXIL language [19] in a similar fashion.

The LOA translation for our RAPs executive has both an autonomous and a manual method of execution for every RAP that engages the system under control (known as a primitive). For each RAP primitive, our implementation uses a LOA setting in memory to select the appropriate method. Since each PRL instruction has a corresponding RAP primitive, it is natural for the RAP executive to use the LOA provided by the LOA server to select the appropriate method. With the PRL semantics, we have expanded our approach by adding a consent method for each primitive that takes priority over the other two methods. If the LOA is Consent, the consent method is executed first. If consent is not obtained, the primitive fails. If consent is obtained then the automated method is executed.

To evaluate Hypothesis 1 we encoded two ISS procedures using PRL: (1) power switch trip malfunction procedure, and (2) power switch reconfiguration checklist procedure. Each of these procedures includes methods for some steps to be executed either manually or automatically. The malfunction procedure performs initial diagnosis of the cause of the trip and attempts to re-close the switch, if it is safe to do so. The checklist procedure reconfigures the portion of the power distribution system affected by the trip to restore normal operations.

We successfully represented both ISS EPS procedures using PRL. These representations included the information needed for both manual and automated execution of many of the steps in these procedures. Encoding methods for automatic execution required capturing information about the conditions to be checked before taking action (i.e., pre-conditions and start conditions) and the conditions indicating an action is complete (i.e., post-conditions and end conditions). As anticipated, we found that these conditions often are not expressed in current procedures, but are part of flight training. Incorporating adjustable autonomy into flight operations will require capturing such information during procedure authoring.

To evaluate Hypothesis 2, we injected the power switch trip fault into the simulation and executed the power switch malfunction procedure to diagnose this fault and mitigate its impacts. We evaluated this case twice – first with all steps performed manually as done for Station today and second with some steps executed automatically. Manual steps are performed by a person using the ISS PCS displays connected to the ISS in a Box simulation. For manual steps, our software bookmarks the current step and prompts the person to indicate when the current step is complete. It also marks which steps have been completed successfully or aborted. Automatic steps are executed as soon as they become the current step by automatically sending Station commands to the ISS simulation. Our software also monitors available telemetry for evidence that the command had the intended effect. When these effects are observed, the step is marked complete. Consent steps require a person to approve their execution before commands are dispatched to the ISS simulation.

For the power switch trip malfunction procedure, our adjustable autonomy software successfully performed the PRL procedure with different LOA settings: (1) Case 1 – LOA set to Manual for all steps, and (2) Case 2 – LOA set to Automated for eight of sixteen steps. Situated condition checking specified in the PRL procedures was used to identify the correct diagnostic actions to take, and the mitigation commands were performed correctly and in a timely manner. Anecdotal data from this experiment indicates that the procedure can be executed more quickly in case 2 where many of the telemetry checking steps are performed automatically. This improved performance time results from eliminating the need to manually navigate to the appropriate PCS display before telemetry can be observed or commands can be dispatched. This approach also reduces the potential for human error when executing this procedure.

3.2. Centaur Cockpit Procedures

We have developed a prototype of our adjustable autonomy architecture for evaluation with Centaur Cockpit procedures. Centaur is a humanoid robot developed at JSC. It combines highly dexterous manipulation with stereo vision and a mobile base. The Cockpit is a facility at JSC for supervision of robots like Centaur. It provides multiple computers with configurable software for situation awareness and remote commanding of robots. We developed Cockpit procedures for the sample retrieval task performed by Centaur during the Desert Research and Technology Study (RATS) in 2006 [9]. We integrated our adjustable autonomy prototype with both a simulation of Centaur and the actual robot. We used this prototype to evaluate the following hypothesis:

- Hypothesis 1: Procedures for adjustable autonomy can be used to aid humans in performing procedures that interleave human actions with the actions of a remote robot

To evaluate this hypothesis, we encoded 16 procedures using PRL. The primitive procedures all dispatch automatically a single command to Centaur. These primitive procedures control Centaur by issuing task-level commands to control software (called the Central Commander) running onboard the robot. The composite procedures combine manual steps with these robot primitives to accomplish more complex tasks. Manual tasks include verifying robot availability and providing seed coordinates for the vision system tracking objects in the robot's environment. For example, the user is requested to verify that the robot base is available for commanding. The execution of the procedure is paused until this action is performed. Once it is performed, commands to move the robot base are automatically dispatched.

At the time of publication, we have evaluated five procedures for commanding the Centaur robot from the JSC Cockpit, including commands to both the upper and lower body of Centaur. Using our adjustable autonomy prototype, we demonstrated the ability to assist the Cockpit Supervisor in commanding Centaur for portions of the sample retrieval task. This assistance includes (1) prompting the Cockpit Supervisor when manual actions are needed, (2) collecting information from the user needed to construct robot commands, and (3) tracking the completion of robot commands and human actions to improve human situation awareness.

4. Related Work

Adjustable autonomy was introduced for supervisory control of robotic systems [14]. Since then, techniques for adjustable autonomy have proliferated for robotics [7, 8, 10, 18] as well as a variety of other fields including multi-agent systems [1, 4, 15], process control [13], and vehicle system control [3, 5, 16, 20]. Our implementation of Level of Autonomy is derived from the work by Bonasso et al. [2] to develop adjustable autonomy for supervising the Shuttle remote manipulator. He defines manual and automated control methods for each primitive task (where possible) and adjusts the LOA by selecting one of these methods for each primitive task. We have extended this approach by implementing additional levels derived from Parasuraman et al. [14] levels of autonomy (i.e., Consent = Level 5, computer suggests an action and executes that suggestion if the human approves).

Our procedure-based approach to adjustable autonomy is consistent with other approaches to adjustable autonomy for robots [7, 8, 10, 18] in assigning tasks among a heterogeneous team of humans and robots. Similar to Fong et al. [7], we focus on one-to-one interaction between a person and a robot, while other approaches have focused on one-to-many interaction between a person and a team of robots [8, 10, 18]. Heger and Singh [10] define four levels of autonomy, including the ability for a robot to ask for human assistance that is not addressed in our approach, to coordinate a human-robot team performing assembly tasks. They use Markov models to automatically determine the task transitions for each level of autonomy. Fong et al. [7] assume autonomous robot operation that can be adjusted when a robot encounters a problem or a person needs robotic assistance. This adjustment can be initiated by either the human or the robot reasoning about a model of the skills of team members. Using a policy-based approach, Sierhuis et al. [18] adjust the tasks performed by the human-robot team by constraining task complexity, execution autonomy, and obligation for human involvement. Our procedure-based approach is unique in that it assists not only the performance of robotic tasks at different levels of autonomy, but also human tasks. Assisting manual tasks is typically outside scope for other approaches.

Adjustable autonomy for coordinating multi-agent systems has focused on strategies for dynamically adjusting the allocation of tasks among distributed software agents and humans. Scerri, et al. [15] uses Markov decision processes to model and reason about transfer of control during the execution of tasks. Adjustments consist of designating which agent performs a task and are made to improve performance. Barber et al. [1] associate an agent's decision-making interaction style with its level of autonomy. Adjusting

the level of autonomy corresponds to changing the agent's interaction among the following levels: (1) command-driven – similar to our Manual, (2) consensus – involving both human and agent like Consent but giving the agent the ability to influence decisions, and (3) locally autonomous/master – similar to our Automated. This range corresponds to an organization adjustment that affects the degree to which the agent controls decision making within its organization. In our procedure-based approach, humans determine task allocations within the constraints on allowable transitions. This is closer to the policy-based approach of Bradshaw et al. [4] that defines categories of actions constraints (e.g., permitted actions, obligated actions) and adjusts these policies based on changes in situation and capabilities. Because of the close tie to human operations, our procedure-based approach to adjustable autonomy has only been used for human-system coordination, while many of the agent-based approaches have been used for system-system coordination as well. Another difference is that our procedure-based approach provides centralized coordination of agents, while other multi-agent approaches [1, 15] use distributed coordination of agents.

5. Conclusions

We have defined an approach for providing adjustable autonomy using electronic flight procedures. We have developed software for humans to adjust whether a step in a procedure is performed manually or automatically, based on the LOA setting. The LOA is set by a person and constrained by flight rules and the available instrumentation. We initially implemented three LOA settings: manual, consent, and automated. These settings were determined by interviewing flight controllers that execute procedures on current spacecraft systems. We have evaluated our approach in two domains: ISS power procedures and procedures for supervising robots.

Using ISS procedures in a flight-like environment, we demonstrated the ability to define and execute a procedure in PRL with steps and instructions that can be performed either by a person or automatically. By changing the LOA settings just prior to execution, we used the same procedure to successfully perform operational tasks both manually and automatically. Because adjusting the level of procedure autonomy did not require modifying the procedure content, this approach does not require costly recertification of procedures when increasing the Level of Autonomy for a spacecraft system.

Using Cockpit procedures remotely with the Centaur robot, we demonstrated the ability to define electronic procedures for joint human-robot tasks. These procedures encode knowledge required to both guide a person through manual tasks and to command a robot to perform its tasks automatically. By providing the ability to require human consent before dispatching a robot command, we can shift a person's attention to an ongoing task to confirm that previous actions were successful and that the robot is ready to proceed with the next action. This supports both safe operations and improves situation awareness of robot autonomy.

There are some key differences between the adjustable autonomy prototype for Centaur and the adjustable autonomy prototype for ISS. Procedures for complex Centaur tasks are built up from one or two step "primitive" procedures. These primitives are defined to be reusable in multiple "composite" procedures. Procedures for complex ISS tasks, however, are not built from such reusable primitive procedures. While some ISS procedures do link to other procedures, there is much less hierarchy in ISS procedures than in Centaur Cockpit procedures and similar steps are repeated instead of encapsulated in a reusable primitive. A second difference between the Centaur Cockpit procedures and the ISS procedures relates to the confirmation of command effects before sending subsequent commands. For the ISS, it is common practice to confirm that a command has had the intended effect before dispatching the next command. For Centaur, we are investigating an approach where multiple commands are dispatched and queued for execution without such confirmation of effects. This approach is intended to reduce command latency over time delay [17]. Finally, the prototype for Centaur Cockpit procedures uses executive software that natively executes PRL instead of translating PRL to another executive language, as done in the prototype for ISS procedures. This PRL Executive was developed at JSC.

Based on our evaluation, we conclude that automation of electronic procedures can be used to implement adjustable autonomy for mission tasks normally performed by humans.

6. Acknowledgments

We would like to acknowledge Lui Wang/JSC and Jeremy Frank/ARC for sponsoring the development of the adjustable autonomy software. We acknowledge the contributions of Vandi Verma/ARC and Mike Dalal/QSS in developing and deploying the ISS adjustable autonomy prototype. We acknowledge the contributions of Robert Burridge/TRACLabs, Kim

Hambuchen/JSC, and Tam Ngo/JSC in developing and deploying the Centaur adjustable autonomy prototype.

7. References

[1] Barber, K., A. Goel, C. Martin, "Dynamic Adaptive Autonomy in Multi-Agent Systems", *Journal of Experimental and Theoretical Articial Intelligence.* 1999.

[2] Bonasso, R.P., D. Kortenkamp and T. Whitney, "Using a Robot Control Architecture to Automate Space Shuttle Operations", *IAAI* 1997.

[3] Bonasso, R.P., D. Kortenkamp, and C. Thronesbery, Intelligent Control of a Water Recovery System: Three Years In The Trenches, in Artificial Intelligence. 2003. p. 19-44.

[4] Bradshaw, J. M., P. J. Feltovich, H. Jung, S. Kulkarni, W. Taysom, & A. Uszok, "Dimensions of Adjustable Autonomy and Mixed-Initiative Interaction", in *Proceedings of Agents and Computational Autonomy* 2003: pp. 17-39.

[5] Dorais, G., Bonasso, R. P., Kortenkamp, D., Pell, B., and Schreckenghost, D., "Adjustable Autonomy for Human-centered Autonomous Systems", in *Proceedings of Adjustable Autonomy Workshop. IJCAI.* Aug 1999.

[6] Firby, J.R., *The RAPs Language Manual*, Artificial Intelligence Laboratory, Department of Computer Science, University of Chicago: Chicago, IL. 1995.

[7] Fong, T., C. Kunz, L. M. Hiatt, M. Bugajska, "The HumanRobot Interaction Operating System", in *Proceedings of ACM/IEEE conference on Human-Robot Interaction* 2006.

[8] Goodrich, M., T. McLain, J. Anderson, J. Sun, J. W. Crandall, "Managing Autonomy in Robot Teams: Observations from Four Experiments", in *Proceedings of ACM/IEEE conference on Human-robot interaction* 2007.

[9] Hambuchen, K., Bluethmann, W., Goza, M., Ambrose, R., Rabe, K., & Allan, M., "Supervising Remote Humanoids Across Intermediate Time Delay." In *proceedings of IEEE-RAS Humanoids06*. Dec. 2006.

[10] Heger F., and S. Singh., "Sliding Autonomy for Complex Coordinated Multi-Robot Tasks: Analysis & Experiments", in *Proceedings Robotics: Systems and Science*, Aug 2006.

[11] Kortenkamp, D., R. P. Bonasso and D. Schreckenghost, "Developing and Executing Goal-Based, Adjustably Autonomous Procedures," in *AIAA InfoTech@Aerospace Conference*, 2007.

[12] Kortenkamp, D., K. M. Dalal, R. P. Bonasso, D. Schreckenghost, V. Verma, & L. Wang, "A Procedure Representation Language for Human Spaceflight Operations", In *iSAIRAS* 2008.

[13] Musliner, D. J. and Kurt D. Krebsbach, "Adjustable Autonomy in Procedural Control for Refineries", in *Proceedings of the AAAI Spring Symposium on Adjustable Autonomy*, Mar. 1999.

[14] Parasuraman, R., Sheridan, T. B., & Wickens, C. D., "A Model for Types and Levels of Human Interaction with Automation", *IEEE Transactions on Systems, Man, and Cybernetics. Part A: Systems and Humans*, 30, 286-297.

[15] Scerri, P., D. V. Pynadath, M. Tambe, "Towards Adjustable Autonomy for the Real World", *Journal of Artificial Intelligence Research* 17 (2002) 171-228.

[16] Schreckenghost, D., Malin, J., Thronesbery, C., Watts, G., and Fleming, L., "Adjustable Control Autonomy for Anomaly Response in Space-based Life Support Systems", In *Proceedings of the IJCAI-2001 Workshop on Autonomy, Delegation, and Control*. Seattle, WA. 2001.

[17] Schreckenghost, D, T. Ngo, R. Burridge, L. Wang, and M. Izygon, "Remote Task-level Commanding of Centaur over Time Delay", *Space Technology Applications and International Forum* 2008.

[18] Sierhuis, M. J. Bradshaw, A. Acquisti, R. van Hoof, R. Jeffers, A. Uszok, "Human-Agent Teamwork and Adjustable Autonomy in Practice", in *Proceeding of i-SAIRAS* 2003.

[19] Verma, V., Jonsson, A., Pasareanu, C., Simmons, R., and Tso, K., "Plan Execution Interchange Language (PLEXIL) for Executable Plans and Command Sequences, In *Proceedings of i-SAIRAS* 2005.

[20] Wood, S., "Automated Behavior-Based Interaction Customization for Military Command and Control", *Intelligent User Interface, Workshop on Behavior-based User Interface Customization*. 2004