A procedure assistant for astronauts in a functional programming architecture, with step previewing and spoken correction of dialogue moves

Gregory Aist[1], Manny Rayner[1], John Dowding[1], Beth Ann Hockey[1], Susana Early[2], and Jim Hieronymus[3]

[1]Research Institute for Advanced Computer Science
[2]Foothill/DeAnza College
[3]NASA Ames Research Center
M/S T35B-1, Moffett Field CA 94035
{aist, mrayner, jdowding, bahockey, jimh}@riacs.edu; searly@mail.arc.nasa.gov

Abstract

We present a demonstration of a prototype system aimed at providing support with procedural tasks for astronauts on board the International Space Station. Current functionality includes navigation within the procedure, previewing steps, requesting a list of images or a particular image, recording voice notes and spoken alarms, setting parameters such as audio volume. Dialogue capabilities include handling spoken corrections for an entire dialogue move, reestablishing context in response to a user request, responding to user barge-in, and help on demand. The current system has been partially reimplemented for better efficiency and in response to feedback from astronauts and astronaut training personnel. Added features include visual and spoken step previewing, and spoken correction of dialogue moves. The intention is to introduce the system into astronaut training as a prelude to flight on board the International Space Station.

1 Introduction

Astronauts on board the International Space Station engage in a wide variety of tasks on orbit including medical procedures, extra vehicular activity (E V A), scientific payloads, and station repair and maintenance. These human space flight activities require extensive and thorough procedures. These procedures are written down in the form of a number of steps and, with various notes, cautions, and warnings interspersed throughout the procedure. Each step may have one or more sub steps. Procedures also include branch points, call-outs to other procedures, and instructions to communicate with mission control. Since December 2001, the RIALIST group has been developing a spoken dialogue system for providing assistance with space station procedures. Aist and Hockey (2002) and Aist et al. (2002) described the first version of the system, which operated on a simplified (and invented) procedure for unpacking and operating a digital camera and included speech input and speech output only. Aist et al. (2003) described a second version of the system with an XML-based display, and that included support for not only procedures, but also voice notes and recorded alarms, and parameter settings such as increasing and decreasing volume. In this paper, we describe the third version of the system, with a reimplemented architecture based on a functional specification of the domain-specific aspects of the system combined with an event-driven generic architectural framework. We also describe two new features: previewing of steps, and spoken correction of dialogue moves.

2 System Description

The March 2003 version of the Intelligent Procedure Assistant is shown in Figure 1, just after loading a procedure. The March 2003 version provides the following functions:

Loading a procedure by specifying its name, for example, "Load water procedure."

Sequential navigation through individual steps, for example, "Next step" or "Previous step."

Navigation to arbitrary steps, for example, "Go to step two point one."

Setting system parameters, such as "Increase volume" or "Decrease volume."

Handling annotations, such as voice notes or alarms ("Record a voice note"), or pictures ("Show the small waste water bag.").

Previewing steps; for example, "Read step three".

Issuing spoken corrections (of entire commands), for example, "I meant go to step two."

We will discuss previewing steps and issuing spoken corrections in turn.

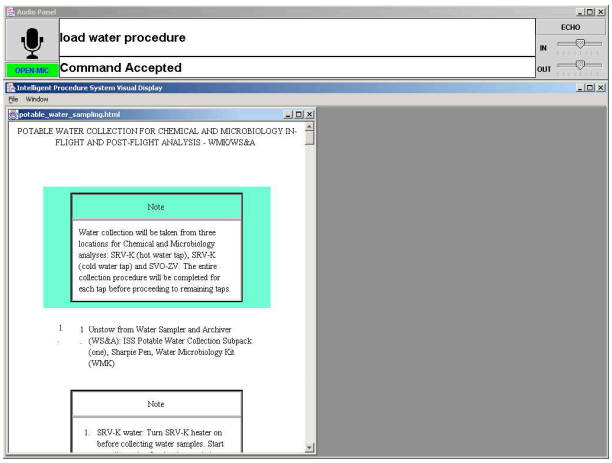

Figure 1. Loading a procedure.

2.1 Previewing steps (Reading mode)

Besides acting on the current step, astronauts indicated that they would like a spoken preview of the next step. Currently this functionality is implemented as displaying a second procedure window in the upper right corner of the screen. Furthermore, steps are prefixed with a spoken indication of previewing, for example, "Reading mode. Note before step two…" To transition back into normal (execution) mode, the user may say "Stop reading." Figure 2 shows the resulting display for the reading mode.

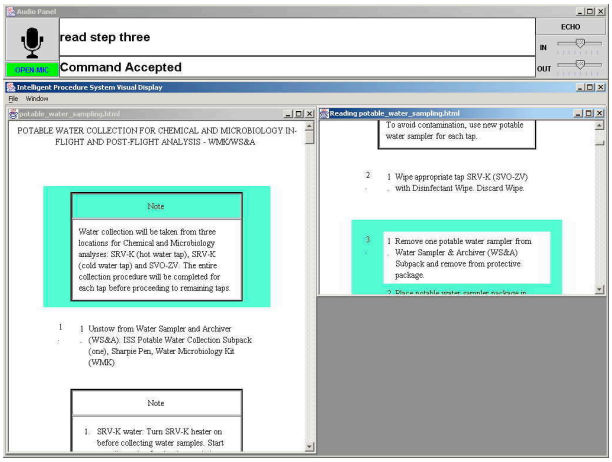

Figure 2. Preview mode, step three.

2.2 Issuing spoken corrections

In the March 2003 version of the Checklist system, the user may issue a spoken correction in the case of an incorrectly given command, or in the case of a speech recognition error (e.g. "read me step three" → "repeat step three"). The dialogue history is represented as a list of the prior dialogue states. Currently we model a correction as a change in the information state, a rollback of the previous action plan, and then an application of the new action plan. Figure 3 shows the display after issuing a correction, "I meant the wash cloth". Reading mode has been exited, and a picture of the washcloth is displayed.

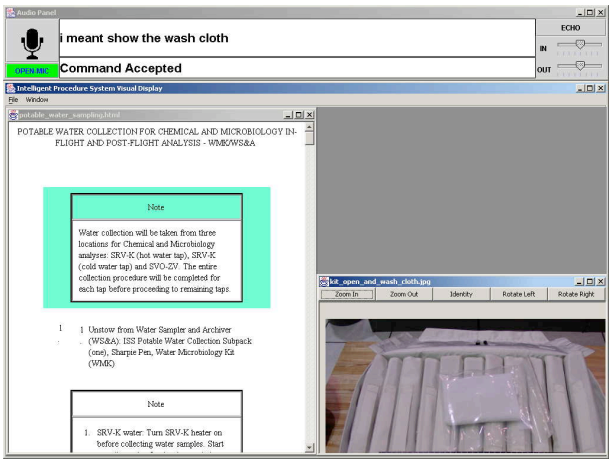

Figure 3. A subsequent correction, resulting in a return to execution mode, and the implementation of the other command.

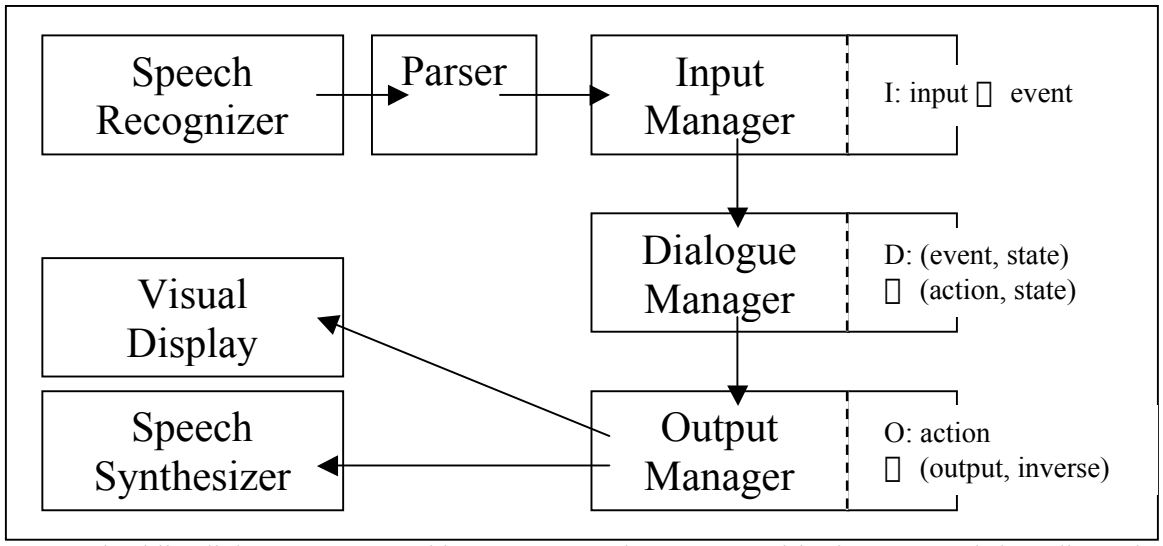

Figure 4. Checklist dialogue system architecture.

3 Architecture, or, How to write a dialogue system in three easy steps

There are three main sections to the dialogue handling code: the input manager, dialogue manager, and the output manager (Figure 4). These are similar divisions to those proposed in Allen et al. (2000). Here, we also adopt a further division of the code into application-specific code and generic code. Application-specific code computes the following function for each component, as a compilation step:

Input manager: Input → Event
Dialogue manager: (Event, State)
→ (Action, State)
Output manager: Action → (Output, Inverse)

The Output and Inverse computed by the Input manager are the multimodal output plans and their multimodal inverses, respectively. The multimodal inverses are used when applying a correction – in conjunction with a return to a previous state on the history list.

The generic code is an interpretation (or execution) step; the input manager's code collects incoming events and dispatches the events to the dialogue manager. The dialogue manager's code collects the incoming events, retrieves the previous state, applies the application-specific function, saves the new state, and then dispatches the new action. The output manager takes the action, applies the application-specific function to compute the output and its inverse, and then dispatches the output plan one action at a time. Each action is represented as an OAA solvable, and dispatched sequentially to be handled by the appropriate agent such as the text-to-speech agent.

The entire dialogue manager is side-effectfree. (With the minor exception of loading a procedure file, which causes a change in the "last accessed" time of the file.) In a more typical dialogue system architecture such as that shown in Figure 5, the side effects are represented separately. The integration of side effects into the output plan has positive benefits for robustness, since they will be represented in one place (and thus modified at the same time when programming changes are made).

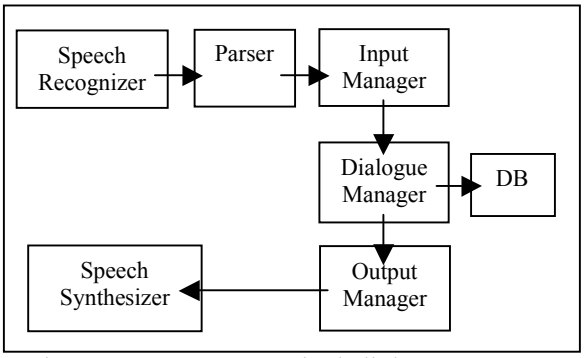

Figure 5. A more typical dialogue system architecture, with the side effects executed separately from the spoken output.

4 Related Research and Future Work

Rudnicky, Reed, and Thayer (1996) describe a system for supporting vehicle maintenance with speech interfaces. Schreckenghost et al. (2003) describe a scenario involving similar tasks (life

support / maintenance related) but with the computer in more control of the actual task. S & K Electronics (n.d.) mention a procedure development environment for rapidly developing and verifying on-orbit procedures (http://sk-web.sk-tech.com/proj.html).

Possible future work includes adding procedures involving inventory management and robot arm assistance, automating dialogue system construction from XML procedures, integrating with telemetry to monitor execution of the procedure and develop error recovery options, improving naturalness of the speech output, modeling dialogue to include dialogue moves and expected user responses, and improving speech recognition to be robust to ISS noise.

References

G. Aist. J. Dowding, B. A. Hockey, and J. Hieronymus. 2002. An intelligent procedure assistant for astronaut training and support. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, refereed demonstration track.

G. Aist and B. A. Hockey. 2002. Generating Training and Assistive Dialogues for Astronauts from International Space Station Technical Documentation. ITS 2002 Workshop on Integrating Technical and Training Documentation. Presented along with system demonstration.

G. Aist, J. Dowding, B. A. Hockey, M. Rayner, J. Hieronymus, D. Bohus, B. Boven, N. Blaylock, E. Campana, S. Early, G. Gorrell, and S. Phan. 2003. European Association for Computational Linguistics (EACL) 2003 meeting, Software Demonstration, Budapest, Hungary, April 2003.

J. Allen, D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent. 2000. An architecture for a generic dialogue shell. *Natural Language Engineering*, Special issue on Best Practice in Spoken Language Dialogue Systems Engineering, pp. 323-340.

A. Rudnicky, S. Reed, and E. H. Thayer. 1996. SpeechWear: A mobile speech system. http://www.speech.cs.cmu.edu/air/papers/speechwear.ps

D. Schreckenghost, C. Thronesbery, P. Bonasso, D. Kortenkamp and C. Martin, Intelligent Control of Life Support for Space Missions, in *IEEE Intelligent Systems Magazine*, September/October, 2002.

Portions of the dialogue systems described in this paper were constructed with Rayner, Hockey, and Dowding's Regulus open source toolkit. Interested readers may find the toolkit and supporting documentation online at:

http://sourceforge.net/projects/regulus/ .