Proceeding of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space: i-SAIRAS 2003, NARA, Japan, May 19-23, 2003

Autonomous Robotics and Ground Operations

Régent L'Archevêque, Erick Dupuis
Canadian Space Agency
6767 route de l'Aéroport, St-Hubert, Canada, J3Y 8Y9
erick.dupuis@space.gc.ca, regent.larcheveque@space.gc.ca

Keywords Teleoperation, Autonomy, Ground Control, Planetary Exploration, On-Orbit Servicing, International Space Station, SSRMS, SPDM.

Abstract

The main objective of the Autonomous Robotics and Ground Operations (ARGO) project is to generate a software toolbox to enable ground control of space robots with varying levels of autonomy and to provide a framework for the integration of the space operations process from planning to post-flight analysis.

This paper provides an overview of the ARGO toolbox with a particular emphasis on the AArVARC reactive inference engine for on-board autonomy. Experimental results of various implementations of AArVARC are presented.

1. Introduction

Most space robotic operations in the foreseeable future will require ground control. Depending on the application, the level of autonomy will wildly vary from case to case. On the International Space Station, operations of the Mobile Servicing System will eventually be conducted partially from the ground to free-up astronauts to conduct more science experiments. In this case, the autonomy requirements are relatively low since the environment is known and static, and the communication delays are relatively short.

In contrast, for a rover or manipulator on a Mars exploration mission, the operator will be located on Earth and will interact with the robot only once or twice every day. The communication delays in this case will be on the order of 20-40 minutes. Such an application will therefore require a very high level of autonomy: the robot being sent command scripts for up to 12 hours of operations at a time.

In the case of Space Vehicles performing servicing of satellites in low Earth Orbit, the operator will still be located on Earth. Typical tasks of such a system would involve capture, refueling or maintenance of a satellite. Although the delays are only on the order of a few seconds, the level of autonomy will still be relatively high since the environment in which the robot operates changes dynamically.

Over the last decade, many approaches have been implemented to circumvent these problems and allow controlling space-based robots from ground stations. In 1993 DLR demonstrated ground control capabilities with the ROTEX [1] flight experiment utilising various modes of control. In 1998, NASDA's ETS-VII satellite was successfully controlled from a ground station at NASDA's Tsukuba Space Center in Japan [2]. ETS-VII was also used to demonstrate ground control using DLR's MARCO architecture [3] and ESA's DREAMS architecture [4]. Using a similar approach, the Canadian Space Agency (CSA) and MacDonald Dettwiler Space and Advanced Robotics (MD Robotics) developed an architecture for remote operation of space-based assets [5] and tested it in an

operational environment by remotely controlling a robotic excavator in northern Canada. Given the rise in interest of remotely controlling the ISS infrastructure from ground-based stations, the CSA, DLR and MD Robotics have started to tailor their ground segment software specifically for MSS operations [6].

More recently, much attention was dedicated to the incorporation of higher levels of autonomy into ground control architectures. This has led to the inclusion of Goal Decomposition Hierarchies [7] in the CSA's ROSA architecture [8], the development of ESA's MUROCO [9] framework onto DREAMS and JPL's CLARAty[10].

2. The ARGO Project

2.1 Project Objectives

The first main objective of the ARGO project is to generate a software toolbox to enable ground control of space robots with varying levels of autonomy. Applications shall cover the full spectrum of autonomy from supervisory control such as might be expected for ISS robotics to more autonomous operations such as would be encountered in planetary exploration missions. It shall also cover the full range of space robotic applications from orbital manipulators to planetary exploration rovers. To facilitate the re-use of software tools from one application to the next, the design is modular and portable to the maximum extent possible. Hence ARGO is not so much an architecture as a set of tools that can be connected together depending on the application.

The second main objective is to seamlessly integrate the space operations process from planning through verification, execution up to post-flight analysis. The intent of ARGO is to achieve a reduction of the costs associated with space operations by streamlining the planning process. Typical issues to be addressed are efficient and fast mission planning, issues associated with operations verification for autonomous missions, operation monitoring and diagnosis tools.

2.2 ARGO Modules & Architecture

The following are technologies that will be addressed to meet the objectives of the ARGO project:

- Generic Ground Control Station (GCS);
- Local and distributed autonomy (AArVARC);
- Safety System (ground and on-orbit safety);
- Distributed and shared resources;
- Event Logging and monitoring tools;
- Operation Planning Tools;
- Operation Analysis Tools;

The GCS constitutes a core element in the *Space Operations Process*. It is the central element enabling an Earth-based operator to remotely control a space-based robot. Currently, robotic space operation planning, verification and monitoring tools form heterogeneous systems and are hardly compatible with each other. The GCS is a good mechanism to integrate the *Space Operations Process* and to link the different technologies into a single system. A GCS requirements document has been initiated and has proven the importance of this user interface to link the systems (Figure 1).

Another important component of ARGO is the local autonomy software. Over the last two years the CSA has developed such a capability through the Amorphous Architecture for Variable Autonomy Robot Control (AArVARC) reactive engine. Further details are given in the next section of the current paper.

Increasing the autonomy of space-based systems and removing the operator from the local site has important implications on the overall safety of the system. The ARGO project will therefore investigate the how using video analysis, posture estimation, health and telemetry monitoring can effectively play the role of a local operator to ensure system safety.

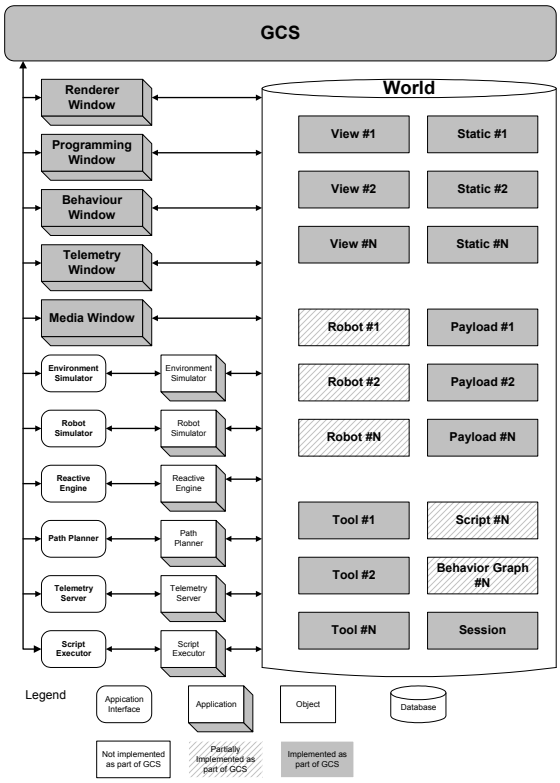

Figure 1: ARGO Ground Control Station

To support collaboration between systems and resources sharing, CSA has been developing the *ARGO Resource API*. This tool allows control distribution and resources sharing. It provides standard command and telemetry interfaces to command and control any devices or software. By applying a layer over existing software, the software obtains additional capabilities such as resource client arbitration, resource accessibility and abstraction of the communication links. Up to know, prototypes have been developed using Java Remote Method Invocation (RMI). This allows AArVARC engines, for example, to invoke methods located physically on another system. This approach makes the communication process between the resource server and the resource client transparent.

Another important aspect of ground control and autonomy in the context of space operations is to provide a log of the events happening during operations for monitoring and post-flight analysis. To this end, the *ARGO Log Event API* has been developed. This API provides functionalities such as event logging, browsing capability and log management. Each *ARGO Resource* includes an instance of *ARGO Log* and is in charge of its own event logging. The GCS implements an *ARGO Resource Client* to display chronologically the events.

To enhance support to the *Space Operations Process*, the ARGO architecture includes planner and analysis tools. Based on certain rules, it is possible for an operator to write a plan, validate it, to execute it and to correct it if it is required (Figure 2). The *ARGO Analysis Modules* analyse the on-line telemetry and the resource logs to display sequence of tasks or telemetry value progresses.

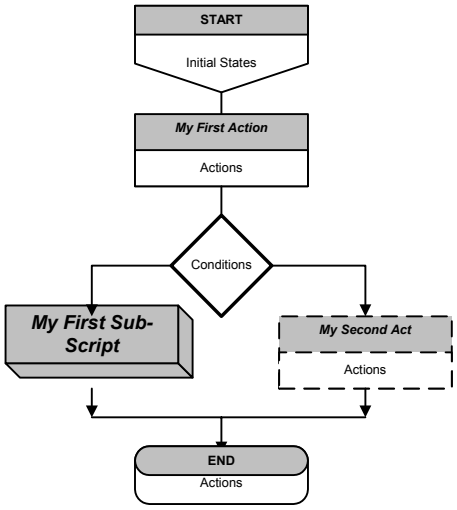

Figure 2: ARGO Script Programming

3. AArVARC

To support variable levels of autonomy, the CSA has developed the *Amorphous Architecture for Variable Autonomy Robot Control* (AArVARC) [1]. This reactive engine based on the Hierarchical Task Network (HTN) concept allows the implementation of local and distributed autonomy. It is event-driven and provides a capability for immediate reaction based on external information such as sensor data, time events and operator-entered commands.

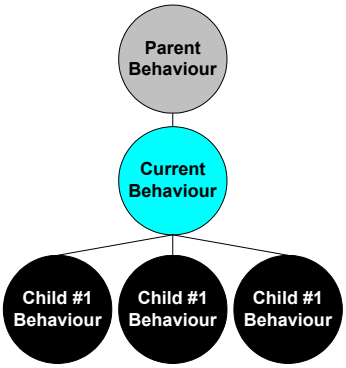

Figure 3: AArVARC Behaviour Relationship

High-level commands are entered as standard behaviours that specify the actions to be taken according to the events and the environment changes. Each behaviour may implement completely a complex task or may invoke, spawn or request existing behaviours to achieve its goal. Using this approach, it is possible to drastically reduce the information transferred between the operator and the remote system and to increase the local autonomy of remote systems by providing sets of rules to execute in order to achieve mission success and safety. Typically, the AArVARC engine receives high-level commands from the operator, invokes the proper behaviour and decomposes it into lower level behaviours (Figure 3). These, in turn, can get decomposed into yet lower-level behaviours. Throughout execution, any behaviour reports its status to its parent and, eventually, to the operator. Malfunctions may be handled by the behaviours themselves or may be reported to the operator for human intervention.

As shown in Figure 4, a *Command Interface* is used to send command and allows the user to control the operations to be performed by the AArVARC engine.

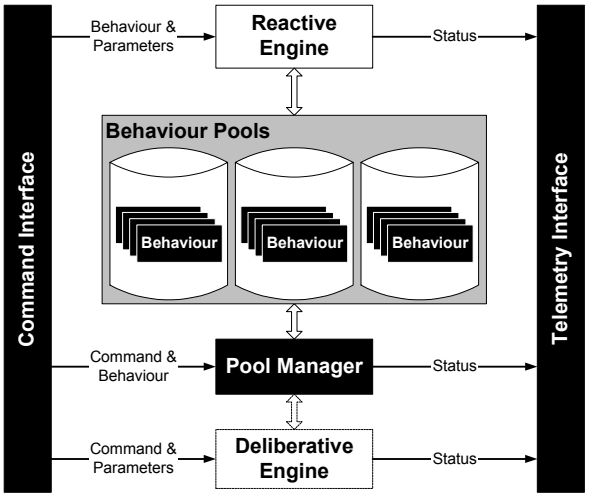

Figure 4: AArVARC Architecture

By using the *Telemetry Interface* the user can get the details status of an AArVARC engine. This module provides:

- state of AArVARC engine execution(play, pause, stop);
- state of each running behaviour (see below);
- value of the AArVARC engine options;
- events history (through the log event capability);

To share and to avoid duplication of behaviours, AArVARC supports the concept of Behaviour Pools. The *Reactive Engine* accesses these containers in order to execute, spawn, pause or stop behaviours. This core module has the important task to decompose high-level behaviours into lower-level behaviours. It is also notified by the behaviours if error or malfunctions occurred.

The *Pool Manager* also has an access to the pools. In fact, this module provides the tools to perform operations on a the content of a behaviour pool. It provides methods to modify, add or remove behaviour on-line. This is useful to correct and upgrade an improper behaviour during the system operation. This interface allows the implementation of the *Deliberative Engine*. This module corresponds to the Artificial Intelligence of the AArVARC engine.

Based on *Behaviour Pools*, environment changes and events, the *Deliberative Engine* may change the content of *Behaviour Pools*.

To implement an AArVARC engine, the user must create a pool of behaviours. To have an efficient pool, the user may be concerned by the modularity of the behaviours. A behaviour tree is firstly drafted to get an overall picture of the behaviours to be implemented.

As shown in Figure 5, a behaviour can be in any one of six states. The first state to be executed is *Initialization*. This state is executed once and initialises the behaviour. Connection to devices, memory allocation and pre-processing computations are normally performed in this state. The *Paused* and *Run* states deal with the execution of the behaviour under nominal conditions. If a "stop" command is issued at any time during execution or if the behaviour completes its actions successfully, then the behaviour falls into the *Completed* state. Any anomalies detected by the behaviour during execution bring it to the *Error Recovery* state. If the behaviour is not able to handle the problem then it falls into the *Failed* state. This marks the end of the behaviour. The parent must take the relay to recover.

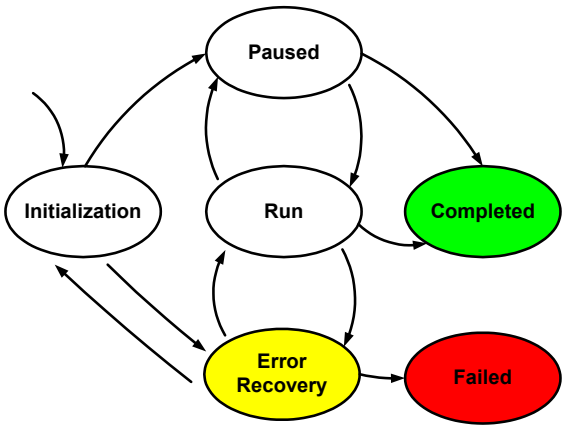

Figure 5: AArVARC Behaviour States

The process described above is applied to the whole behaviour tree. Each child reports to a single parent and the parent is notified of each behaviour state transition.

Up to now, two releases have been implemented and tested. AArVARC I uses the graphical programming environment: Matlab/Simulink's Stateflow toolbox. This allows operators to use a high level of abstraction when implementing behaviours in the HTN.

By using a finite state machine representation, it is easy to represent complex logical structure. This release does not implement the *Pool Manager* capability. It does not support on-line Behaviour Pools operation. MSS Ground Control and Satellite Servicing applications AArVARC engines have been implemented and tested using this prototype release.

Since May 2002, a second version, AArVARC II, has been under development. This Java based engine implements all modules proposed in the AArVARC Architecture (Figure 4).

4. Implementation & Applications

To validate, verify and demonstrate the adaptability of ARGO to different levels of autonomy, CSA has selected three applications. As described earlier, the main goal of ARGO is to develop generic tools and a generic architecture in order to cover the whole spectrum of system autonomy.

The first application was a satellite-servicing scenario. An AArVARC engine was implemented on MD Robotics' Reusable Space Vehicle Payload Handling Simulator (RPHS), a dual arm robotic test-bed used to validate various technologies for satellite servicing operations, shown on Figure 6.

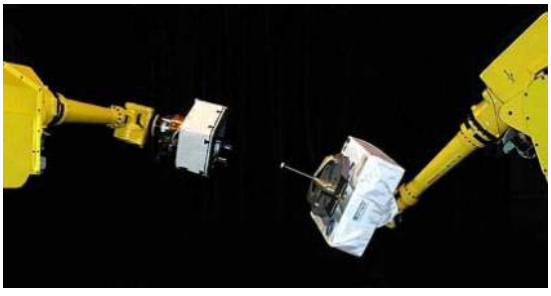

Figure 6: MD Robotics' Reusable Space Vehicle Payload Handling Simulator (RPHS)

The RPHS facility is located in a dark room environment to faithfully emulate the lighting environment of Low-Earth Orbit to validate various artificial vision technologies.

The nominal scenario involved locating, tracking and capturing a moving satellite mock-up using a vision system. Transitions between phases of the operation were triggered by sensory events. Anomalies were injected in the scenario to verify the ability of AArVARC to deal with off-nominal situations. In one case, the lights were turned off to emulate blinding of the vision system and in another case, the telemetry server was turned off thus emulating the loss of the link between the low-level controller and the intelligent agent.

In all cases, AArVARC performed admirably. When encountering anomalous conditions, the engine executed a pre-determined set of rules to complete the operation successfully. As a testimony to its robustness, the set of behaviours implemented using the Hierarchical Task Network approach encountered a set of untested conditions during the final demonstration of the project and, nevertheless, succeeded in completing the capture of the satellite mock-up.

The second application selected to demonstrate the power of AArVARC was a simulation of MSS ground control using the MSS Operations and Training Simulator (MOTS), a high-fidelity dynamic and graphical simulator of the MSS used for astronaut training (Figure 7).

The architecture was set-up such that the intelligent agent, in this case AArVARC, was located in the ground station and the operator monitored and approved all transitions between script elements.

The AArVARC engine successfully completed the step-off and stow procedure that was executed during Canadarm-2's maiden flight (Flight 6A). This procedure involved many of the most complex operations to be expected of the Canadarm-2 such as the capture of a grapple fixture and a change of base from one grapple fixture to another.

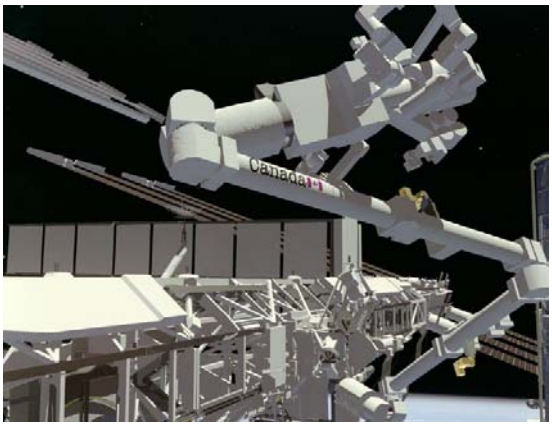

Figure 7: MSS Operation & Training Simulator

The third application used the CSA's Mobile Robotic Testbed (MRT) in a Mars exploration-like scenario (Figure 8). This application implemented a rudimentary autonomous navigation capability on a laboratory mobile robot. The capabilities that were implemented include the ability for a mobile robot to localize itself, to plan a path to a final destination, to detect obstacles and react to anomalies in the environment.

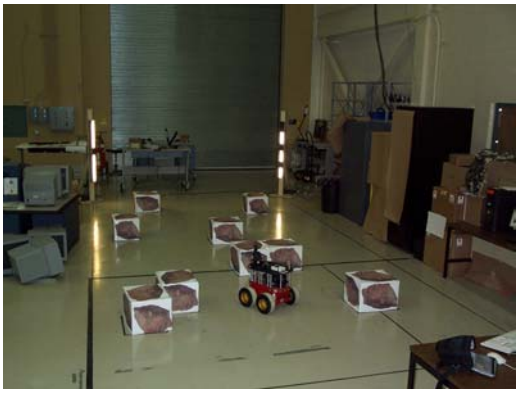

Figure 8. The MRT robot traversing the terrain.

The autonomy uses a set of 11 behaviours. This application provided the most stringent requirements for AArVARC and prompted the development of the second generation of the software. As shown in Figure

9, the highest-level behaviour, named *GoToLocation*, spawns in sequence 4 Behaviours. One of these four behaviours, *FollowTrajectory,* invokes combinations of sequential and parallel lower-level behaviours.

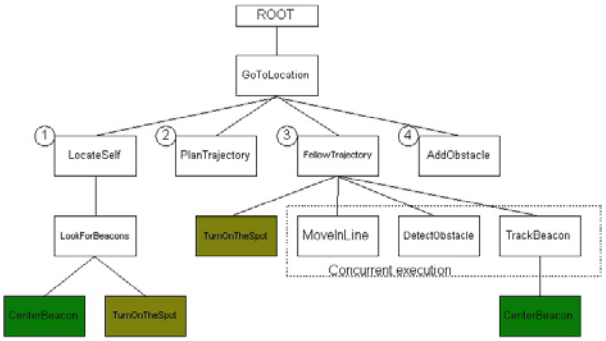

Figure 9. MRT Behaviours Tree

5. Conclusion

To address the need of most space robotic missions, the Autonomous Robotics and Ground Operations (ARGO) aims at generating a software toolbox to enable ground control of space robots with varying levels of autonomy. The toolbox can be reconfigured to handle space robotic applications ranging from orbital manipulators to planetary rovers.

The second main objective of the project is to seamlessly integrate the space operations process with the intent of achieving a reduction of the costs associated with space operations by streamlining the planning process.

Much of the recent development has focused on the AArVARC reactive inference engine for on-board autonomy. AArVARC is event-driven and based on the Hierarchical Task Network (HTN) concept. It is provides a capability for immediate reaction based on external information such as sensor data, time events and operator-entered commands.

Three different implementations have demonstrated the versatility of the AArVARC reactive engine. A scenario using the MSS Operations and Training Simulator demonstrated its usefulness for ISS robotic operations where the need for autonomy is low. Implementations on testbeds to emulate satellite servicing and planetary exploration have demonstrated its usefulness in scenarios where more autonomy is required.

Reference

[1] Hirzinger, G., Brunner, B., Dietrich, J. and Heindl, J., "Sensor-Based Space Robotics - ROTEX and Its Telerobotic Features", IEEE Trans. on Robotics and Automation, Vol.9, No.5, pp.649-663, October 1993.

[2] Oda, M., "Space Robot Experiments on NASDA's ETS-VII Satellite - Preliminary Overview of the Experiment Results", Proc. 1999 IEEE Conf. on Robotics and Automation (ICRA 99), Detroit, USA, pp.1390-1395, 1999.

[3] Brunner, B., Landzettel, K., Schreiber, G., Steinmetz, B.M. and Hirzinger, G., "A Universal Task-Level Ground Control and Programming System for Space Robot Applications - the MARCO Concept and its Applications to the ETS-VII Project", Proc. Fifth International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS 99), ESTEC, Noordwijk, The Netherlands, pp.217-224, June 1-3 1999.

[4] Galardini, G.M., Kapellos, K., Maesen, E., Visentin, G. and Didot, F., "Vision and Interactive Autonomy Bi-Lateral Experiments on the Japanese Satellite ETS-VII", Proc. Fifth International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS 99), ESTEC, Noordwijk, The Netherlands, pp.217-224, June 1-3 1999.

[5] Dupuis, E., Gillett, R., Boulanger, P., Edwards, E., and Lipsett, M., "Interactive, Intelligent Remote Operations: Application to Space Robotics", SPIE

Telemanipulator and Telepresence Technologies VI, Boston, Massachusetts, USA, September 1999.

[6] Dupuis, E., Gillett, R., L'Archevêque, R. and Richmond, J., "Ground Control of International Space Station Robots", Journal of Machine Intelligence and Robotic Control, Special Issue on Space Robotics, Vol. 3, No. 3, pp. 91-98, September 2001.

[7] Hartman, L., "Reactive goal decomposition hierarchies for on-board autonomy", 53rd International Astronautical Congress, October 2002

[8] Dupuis, E. and Gillett, R., "Remote Operation with Supervised Autonomy", 7th ESA Workshop on Avanced Space Technologies in Robotics and Automation (ASTRA 2002), Noordwijk, The Netherlands, November 2002.

[9] Galardini, D and Kapellos, K., "A Study on a Formal Approach to MUlti RObot COoperation", 7th ESA Workshop on Avanced Space Technologies in Robotics and Automation (ASTRA 2002), Noordwijk, The Netherlands, November 2002.

[10] Volpe, R., Nesnas, I., Estlin, T., Mutz, D., Petras, R. and Das, H., "The CLARAty Architecture for Robotic Autonomy", Proceedings of the 2001 IEEE Aerospace Conference, Big Sky, Montana, March 10-17, 2001