Three NASA Application Domains for Integrated Planning, Scheduling and Execution

David Kortenkamp, Debra Schreckenghost and R. Peter Bonasso
Texas Robotics and Automation Center (TRACLabs)
Metrica Inc.
NASA Johnson Space Center – ER2
Houston, TX 77058
kortenkamp@jsc.nasa.gov

Abstract

This paper describes three application domains for integrating planning, scheduling and execution at NASA Johnson Space Center. The three domains are: advanced life support systems; traded control of robotic manipulators and free flying space robots. The challenges of each domain will be given and initial progress will be described. For each domain we have applied the same layered control architecture, called 3T, which will also be described in this paper.

Introduction

NASA has many applications, both robotic and non-robotic, that require integration of planning, scheduling and execution. This paper describes three on-going projects at NASA Johnson Space Center. Each project uses a layered control architecture also developed at NASA Johnson Space Center. First, we will introduce this architecture. Then we will look at its application to three projects: control of advanced life support systems; traded control of space manipulators; and control of a free flying space robot. For each application we will look at the significant research issues inherent in the application and how our architecture addresses those issues. Most of this paper is devoted to control of advanced life support systems as this domain is most closely allied with the workshop topics. Robotic examples are given to show how a properly constructed architecture can be used on a wide variety of robotics and non-robotics projects.

3T architecture

Over the last several years, researchers at NASA Johnson Space Center have developed an autonomous robot control architecture that separates the general robot intelligence problem into three interacting layers or tiers (and is thus known as 3T, see Figure 1):

- A set of robot specific situated skills that represent the architecture's connection with the world. The term situated skills (Slack 1992) is intended to denote a capability that, if placed in the proper context, will achieve or maintain a particular state in the world. For example, grasping, object tracking, and local navigation. The skills are maintained by a *skill manager*.

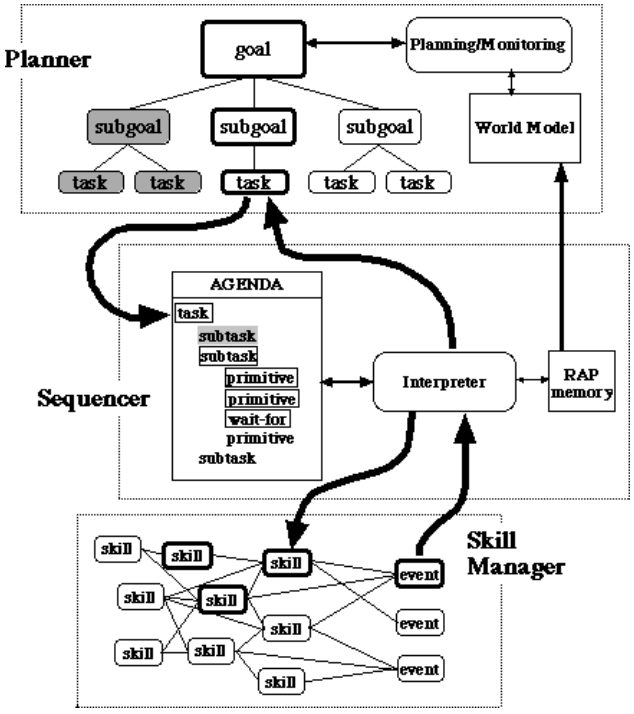

Figure 1: A layered architecture for intelligent control.

- A sequencing capability that can differentially activate the situated skills in order to direct changes in the state of the world and accomplish specific tasks. For example, exiting a room might be orchestrated through the use of reactive skills for door tracking, local navigation, grasping, and pulling. In each of these phases of operation, the skills of the reactive

level are connected to function as what might be called a "Brooksian" robot (Brooks 1986) – a collection of networked state machines. We are using the Reactive Action Packages (RAPs) system (Firby 1989) for this portion of the architecture.

- A deliberative planning capability that reasons in depth about goals, resources and timing constraints. We are using a state-based non-linear hierarchical planner known as AP (Elsaesser & MacMillan 1991). AP is a multi-agent planner which can reason about metric time for scheduling, monitor the execution of its plans, and replan accordingly.

The architecture works as follows, the deliberative layer takes a high-level goal and synthesizes it into a partially ordered list of operators. Each of these operators corresponds to one or more RAPs in the sequencing layer. The RAP interpreter (sequencing layer) decomposes the selected RAP into other RAPs and finally activates a specific set of skills in the reactive layer. These skills include appropriate event monitors which notify the sequencing layer of the occurrence of certain world conditions. The activated skills will move the state of the world in a direction that should cause the desired events. The sequencing layer will terminate the actions, or replace them with new actions when the monitoring events are triggered, when a timeout occurs, or when a new message is received from the deliberative layer indicating a change of plan. For a more detailed description of our architecture see (Bonasso *et al.* 1997).

Advanced Life Support Systems

Life support systems are the technologies that manage conditions in a closed environment to guarantee the environment is hospitable for life, including humans and/or plants. Maintaining conditions for humans requires air regeneration, water recycling, solid waste incineration, thermal and humidity control, and food production. Maintaining conditions for plants requires air regeneration, water recycling, biomass incineration, thermal and humidity control, and nutrient delivery. When both humans and plants co-exist, issues of resource management arise. Water and product gases (oxygen, carbon dioxide, hydrogen, methane) must be produced, collected, allocated, and transferred among different systems and reservoirs. Crops must be selected, planted, and harvested to provide food for the crew and to convert carbon dioxide into oxygen. When resource limitations arise (as is inevitable on a remote site with limited re-supply), resources must be allocated to meet the highest priority objectives.

Issues

An example scenario illustrates the complexity of managing life support. Oxygen produced by plants can be used in a number of ways; it can be (1) consumed by humans, (2) used to burn solid waste, and (3) dissolved in water as a source of nutrients for plants and for bacteria in the bioreactor of the water recovery system. Since re-supply is so costly, it is important to minimize the use of bottled oxygen by using oxygen produced by the plants. The rate of oxygen produced by plants is affected by the type and maturity of the crop and the lighting profile. Thus, the strategic resource management issues related to oxygen include:

- the effect on oxygen production of changing the mix of crop types
- when to plant and harvest crops to maintain stable, consistent oxygen production
- when and how much oxygen to accumulate for incineration (incineration uses oxygen at a much higher rate than humans or other life support systems)
- what oxygen concentration setpoint to maintain in habitats (for such large reservoirs, a slight reduction in setpoint can enable storing oxygen for other activities for extended periods of time).
- what light profile is acceptable for plant growth without interfering with other activities needing light

Another set of issues affect oxygen control tactics used to implement standard operating procedures and alarm handling. These issues include:

- maintenance of oxygen concentration in habitats within a safe deadband (too high is a fire hazard; too low is harmful to humans)
- prevention of unsafe gas pressures in oxygen reservoirs (provide methods for venting or transferring to alternative reservoirs)
- redundant or alternative oxygen instrumentation in case of hardware maintenance or failures

Finally, a third set of issues affect the activation and deactivation of hardware controllers that concentrate, store, transfer, and inject oxygen. These issues include:

- adjustment of control parameters to maintain setpoint (e.g., PID, feed forward controllers)
- recovery from failure to accomplish control (react when controller activation/deactivation times out)

This example makes it clear that managing life support for a remote facility involves a complex set of tasks. Appropriate allocation and coordination of these tasks among humans, robots, and life support systems is important for efficient and effective operations. In the past, such coordination has been a manual task. Indeed, the operation of the traditional control software used for life support is manually intensive, requiring eyes-on, vigilance monitoring and frequent manual adjustment of low level control parameters. Traditional software also does not support real-time changes to operations well. The need for flexibility and reactivity in operations combined with the need to reduce crew workload and the high cost of failure to manage resources effectively suggests automating low level control tasks and assisting humans in the task of strategic planning and resource management. A layered control architecture enables effective allocation of tasks between humans and automation. Upper layers of the architecture operate semi-autonomously, supporting mixed initiative interaction with crew to manage resources. Lower layers operate autonomously, safely effecting control strategies determined at upper layers.

Requirements for an integrated planning, scheduling and control architecture

The effectiveness of an integrated planning, scheduling and control architecture for advanced life support will depend on to what degree the following requirements are satisfied:

- Interactive planning and scheduling. Given the uncertainties associated with long-missions it would be impossible to automatically generate a plan, pass it to the scheduler, and be done with it in one iteration. The architecture must support the ability of move easily between planning and scheduling in an iterative process, allowing the user to provide significant feedback, if desired.
- User-definable abstraction levels for planning and scheduling. The decisions under consideration may be short-term in the order of minutes, to long-term in the order of years. The architecture must be able to move between various time and granularity scales presenting the correct level of information to the user at different decision situations.
- Flexibility. The user should be able to revise implemented solutions, as well as generate new solutions. The architecture should allow the user to play "what-if" games, evaluating different hypothetical scenarios while, at the same time, the architecture is executing the current scenario.
- The architecture must present a common view of the system to the user. Even though the architecture may be built of many different modules, the user's view should be centered on the kinds of functions the user desires, the problems they want to solve, and the kinds of information the want to see.
- The architecture must explicitly recognize multiple performance criteria. The user must be presented with information that depict the dependencies and compromises among these often conflicting criteria.
- The architecture must be open to facilitate its integration with other systems.

Our motivation is to produce a architecture that satisfies the above requirements. The system will allow the crew to build plans and schedules, play "what-if" games with those plans and schedules, then have the architecture execute them while presenting a consistent view of the state of the system to the use. Such an architecture will greatly increase the effectiveness of the crew in advanced life support environments and greatly increase the scientific results of advanced life support experiments.

Current status

We applied the 3T control architecture during a 90 day manned test in September 1997 of advanced life support systems for the Lunar/Mars Life Support Technical Program (LMLSTP) at NASA Johnson Space Center (JSC). 3T was used to control the transfer of product gases (oxygen and carbon dioxide) between multiple gas reservoirs, including a plant growth chamber, storage tanks, the crew habitation module, and an airlock from which the solid waste incinerator draws air and vents effluent. For this application, the planning tier is essential to manage complex system reconfiguration and changes in control strategy required to maintain these multiple gas reservoirs at required levels during a variety of activities including seed germination, plant growth, harvest, and incineration. Even in this constrained application, we have identified a need for more advanced scheduling techniques to provide a finer time granularity in the plan (a detailed schedule) and more exact control of start and stop times for activities.

The 3T control architecture also has been selected for controlling computer-controlled machines (robotic and regenerative life support) in the BIOPlex facility to be completed at NASA JSC in 2000. The BIOPlex facility will be a ground-based, manned test facility for advanced life support technology destined for use in lunar and planetary bases, and planetary travel (such as Mars Transhab Project). It consists of five connected

modules - two plant growth chambers, a crew habitation module, a life support module, and laboratory. Regenerative life support systems include water recovery, air revitalization, solid waste management, and thermal/atmospheric control. Plant support systems include nutrient delivery, gas management, and thermal/humidity control. Robotic systems include transport, manipulation, and sensor/video scanning. Controlling these heterogeneous systems to maintain food supplies and water and gas reservoirs, while minimizing solid waste reservoirs (inedible biomass and fecal matter) poses a challenging set of problems for planning and scheduling. The planner must balance conflicting system needs and account for cross system coupling, at time scales varying from hours to months. In this facility, human and robots will jointly execute tasks and must coordinate their efforts. A common/shared schedule for both crew and computer-controlled machines is needed to guarantee such coordination. This schedule must be sufficiently flexible to adapt to crew preferences while stable and robust for computer control. An integrated planning, scheduling, and control architecture that includes both fine time grain scheduling and optimization as well as long term crop planning will be required for BIOPlex. A more complete description of this on-going work is available in (Schreckenghost *et al.* 1998).

Traded Control for Robot Manipulators

Effective traded control requires a robot system that can both perform routine operations autonomously yet give control to a human to perform specialized or difficult operations. The advantage of a traded control system is that the unique (and expensive in space situations) capabilities of a human can be brought to bear when needed most and not during tedious, repetitive and routine operations. However, a problem with traded control is that the robot does not know what the state of the world or of the task will be when the human finishes his or her portion of the job. This can make it difficult and dangerous for the robot to resume autonomous operation. As an extreme example, the human may forget some crucial aspect of their portion of the task (such as securing a fastener) that the robot is expecting to be accomplished. Less drastic, but probably more commonplace would be subtle side effects of the human's performance such as putting a tool in a slightly different orientation than is expected by the robot. In either case, the problem is that the robot's model of the world and of the task are not consistent with the real state of the world and of the task.

Effective traded control also requires that the robot system know when a human should be performing a task and when it can safely perform a task. Ideally, the robot would plan and schedule both its own activities and those required of the human with whom it will be trading control. In this case, the robot will proceed autonomously until reaching the point where human intervention is required; the robot will then inform the human and safely wait until the human is ready to accept control. The robot system should also recognize situations which are off-nominal and stop to ask for human assistance even if human assistance was not originally required for this step of the plan. The robot system will then need to account for the human intervention and replan its task.

Using 3T to resolve traded control issues

Our solution to the problems outlined in previous subsection consists of three parts. First, we use the multi-agent planner in our architecture's top tier to plan for a traded control task – assigning particular parts of the task to the robot and others to the human operator based on the capabilities of the robot and human agents. Second, we enhance our architecture's sequencing and monitoring capabilities to allow the robot to "mentally" check off steps of the task as they are being performed by humans. In essence the robot is watching and thinking, but not doing. Third, we use our architecture's reactive ability to recover from off-nominal situations to overcome small differences between the real world and the robot's model of the world; this typically also has to be done even when the robot is in complete control.

An application to the shuttle remote manipulation system

We have had several applications of our architecture in traded control situations. One of the most interesting was implementing a procedure tracking system for the space shuttle Remote Manipulator System (RMS). The system, called 3TPT, is designed to track the expected steps of the crew as they carry out RMS operations, detecting malfunctions in the RMS system from failures or improper configurations as well as improper or incomplete procedures by the crew. 3TPT, was employed this past fall to track the RMS checkout procedures on a space shuttle mission. It successfully carried out its task because the reactive nature of the architecture allowed it to stay synchronized with the procedures even in the face of intermittent loss of telemetry and unexpected crew actions. While this particular test on the space shuttle mission did not require traded control (because the actual RMS does not allow for any autonomous operation), extensive tests were performed against an RMS simulation that did

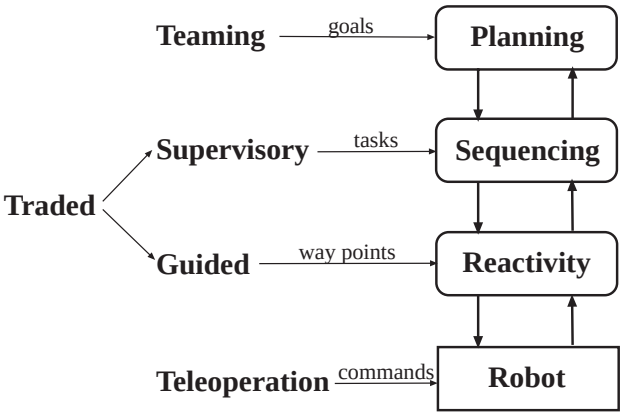

Figure 2: The user can interact at any tier of the 3T architecture

allow for autonomous activity. In these latter experiments, the user could select either autonomous or teleoperating for each task. In teleoperation mode, the system prompted the user to perform each step and verified that it was performed properly. A more detailed description of the 3TPT system is available in (Bonasso, Kortenkamp, & Whitney 1996). Note that the planning layer of 3T was not used in the 3TPT implementation, but was used in other traded control robot situations.

Current status

We are working on a new project with a single manipulator designed to perform space station maintenance tasks. This manipulator will be able to perform some tasks autonomously, some teleoperated and for some tasks it will need to rely on an on-site human. The deliberative tier of 3T is used to plan the entire task, assigning subtasks either to humans or robots. The sequencer then carries out those subtasks, monitoring the human when they are in charge. The system allows the human supervisor to take control at any time, with the sequencer responding appropriately. The goal is to allow for user interaction at any tier of the 3T architecture as shown in Figure 2.

A Free-flying Space Robot

AERCam (Autonomous Extravehicular Robotic Camera) is designed to provide astronauts and ground control camera views of the space shuttle and station. The first generation of AERCam, called AERCam Sprint, flew on a shuttle mission in December 1997. AERCam Sprint was teleoperated by an astronaut inside the space shuttle. The only autonomy it had was the ability to automatically stop its rotation when

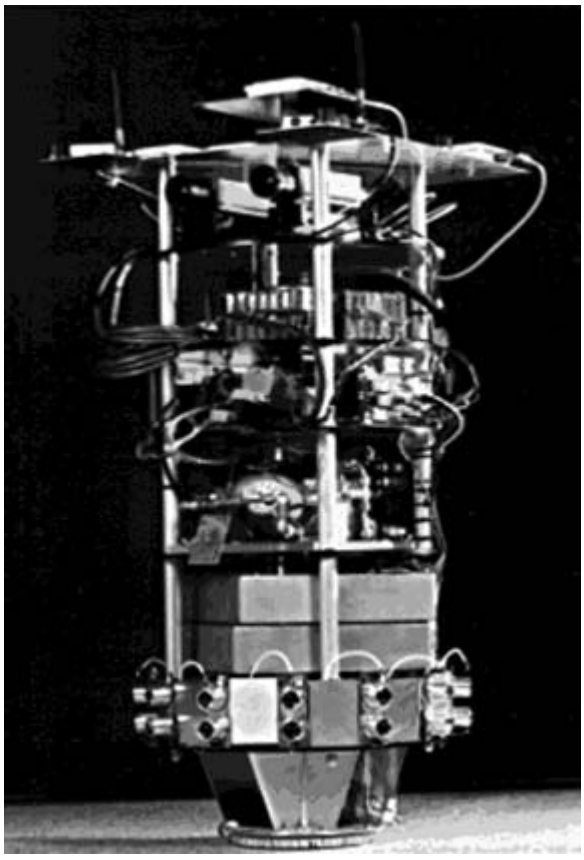

Figure 3: The AERCam air bearing table robot

commanded to do so. The next generation of AERCam, called AERCam II, is currently in development at NASA Johnson Space Center. This robot will have additional autonomous functionality and will be controlled by the bottom two layers of 3T.

A prototype of AERCam II was implemented on an air bearing table (ABT) at NASA Johnson Space Center. The robot hardware is show in Figure 3. The robot consists of many different hardware and software systems, including GPS, infrared proximity detectors, an inertial measurement unit, stereo cameras and processing, voice recognition, and path planning. The 3T skill manager and sequencer serves to integrate all of these different functions.

Issues

The issues involved in using 3T for AERCam are different than those for other systems. AERCam is a very small robot – space and computation are at a premium. Thus, the focus was on creating efficient implementations of the skill manager and the sequencer. Another issue for AERCam was integrating outside processes,

like path planning or user interfaces, into the sequencing tier. A protocol was established for communicating between the sequencer and other processes via a TCP/IP-based interprocess communication (IPC) system.

While the AERCam robot has very few true planning and scheduling issues, it is presented in this paper as an example of an implemented robot that will fly in space and will be autonomous. Future generations of AERCam will need to plan an entire day's worth of inspection tasks, balancing resources (e.g., fuel) and time. These plans will need contingencies for dealing with loss of communication or navigation difficulties. There may be multiple AERCam's asked to coordinate inspection of large structures. For more details on the AERCam project see (Kortenkamp *et al.* 1998).

Conclusions

Each of the domains presented in this paper have their unique problems. However, they also have problems that cut across domains. One issue that arises continually in all of our domains is the user interaction with the control system. This is often overlooked both in research and in designing architectures. It is important to remember that these systems will not run in isolation; they will be monitored and commanded by people who will need easy access to the current state of the system and its control parameters. This means more than just designing a graphical interface, it involves deciding what information (and at what level of abstraction) should be presented to the user and how it should be presented. It also means integrating the user's goals with the system's goals to provide efficient and effective control.

References

Bonasso, R. P.; Firby, R. J.; Gat, E.; Kortenkamp, D.; Miller, D.; and Slack, M. 1997. Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence* 9(2).

Bonasso, R. P.; Kortenkamp, D.; and Whitney, T. 1996. Using a robot control architecture to automate space shuttle operations. In *Proceedings of the 1997 Innovative Applications of Artificial Intelligence Conference.*

Brooks, R. A. 1986. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation* 2(1).

Elsaesser, C., and MacMillan, R. 1991. Representation and algorithms for multiagent adversarial planning. Technical Report MTR-91W000207, The MITRE Corporation.

Firby, R. J. 1989. *Adaptive Execution in Complex Dynamic Worlds.* Ph.D. Dissertation, Yale University.

Kortenkamp, D.; MacMahon, M.; Ryan, D.; Bonasso, R. P.; and Moreland, L. 1998. Applying a layered control architecture to a free-flying space camera. In *IEEE Symposium on Intelligence in Automation and Robotics.*

Schreckenghost, D.; Bonasso, R. P.; Kortenkamp, D.; and Ryan, D. 1998. Three tier architecture for controlling space life support systems. In *IEEE Symposium on Intelligence in Automation and Robotics.*

Slack, M. G. 1992. Sequencing formally defined reactions for robotic activity: Integrating raps and gapps. In *Proceedings of SPIE's Conference on Sensor Fusion.*